A Survey on Automated Metro Map Layout Methods

Martin Nöllenburg
Institute of Theoretical Informatics
Karlsruhe Institute of Technology
Germany

***Abstract*—We give a comprehensive overview of the state of the art in automated metro map layout. A large number of algorithms for computing metro maps have been developed over the last 10–15 years by researchers in computer science and related fields. In this survey, we distinguish three subtasks of generating metro map layouts (schematic network layout, label placement, crossing minimization) and give a list of ten design rules that form the basis of the layout algorithms. Our discussion starts with fundamental work on path and road network schematization, covers algorithms for generating traditional schematic metro maps, and also describes recent methods for special-purpose and non-standard maps.**

I. Introduction

Metro maps are schematic diagrams of public transit networks that focus on conveying the network topology rather than its exact geography to passengers. The goal of a good metro map is thus to reduce the visual complexity and to enable its user to quickly and accurately perform route planning and orientation tasks based on the graphical network representation [1], [2]. This insight is often attributed initially to Henry Beck and his famous octilinear London Tube Map of 1933 [3] (although the Berlin system introduced a highly schematic map already in 1931 [4]). Beck's principles have quickly been adapted by many contemporary metro maps [5] and Roberts explores which features exactly make a good and usable metro map [4].

Today, producing a good metro map is still a very time consuming, often computer-assisted but largely manual process requiring a skilled graphic designer or cartographer who carefully decides about routing metro lines, placing station and interchange symbols, and adding station names to the map. Automating this process has been a challenging research problem in computer science and related fields over the last 10–15 years, involving subdisciplines such as graph drawing, information visualization, computer graphics, and computational cartography. In this survey, we describe the past development and the state of the art of algorithms for various network layout problems in the context of schematic metro maps. We aim to identify common principles underlying many of the proposed automated methods, as well as highlighting the differences in layout models and algorithmic techniques. We also refer the reader to a previous survey by Wolff [6] that focuses on three different layout algorithms and does not cover the more recent developments, as well as to a compact section on metro map layout in a book chapter on graph drawing and cartography [7, Chapter 23.5.2] and in a chapter on map generalization and schematization [8, Chapter 10.6].

A. Subtasks in Metro Map Layout

Generating an effective metro map is a rather complex process that involves several individual subtasks, some of which have interdependencies. Rendering tasks, for example, the choice of a color code for the different metro lines, the choice of interchange and station symbols, the choice of a font set etc. are in fact quite prominent and important for the perception of a metro map. However, these choices are usually not at the core of automating metro map layout, but rather made in advance following established local traditions and the taste and experience of the responsible persons. None of these decisions really influences how subsequent computational layout tasks are approached and thus we do not cover them here.

Arguably the most important and most studied task is to generate the geometry of the underlying network layout. It translates readily into the following abstract graph drawing or network visualization problem.

Problem 1 (Metro Map Layout): Let $G = (V, E)$ be a graph geometrically embedded in the plane $\mathbb{R}^2$, in which each vertex $v \in V$ is a point representing a metro station and each edge $e = (u, v) \in E$ is a curve linking its incident vertices and representing a physical rail link between them. Let further $\mathcal{L}$ be a *path cover*, that is, a set of paths in G representing the different metro lines, so that each edge $e \in E$ belongs to at least one path $L \in \mathcal{L}$. (We call G together with $\mathcal{L}$ a *metro graph*.) Find a *schematic* layout of $(G, \mathcal{L})$ that preserves the topology of the input embedding, satisfies a set of layout constraints, and optimizes a set of quality criteria.

The input embedding of G is usually a geographically accurate drawing of the metro network. Formally, it does not need to be plane, that is, two edges may cross without defining an interchange station. But we may still assume that there are no edge crossings by considering a planarization instead with dummy vertices at crossing points. Accordingly, this explicitly preserves all true edge crossings. A path in $\mathcal{L}$ may be a simple path or a cycle; if a metro line splits into multiple branches it can be easily decomposed into a set of simple paths. Obviously, the final layout and its schematic appearance is largely determined by the actual definition of the layout constraints and the quality criteria. Section I-B lists the most common design principles, which are used to define constraints and quality criteria. Each layout algorithm then uses its own instantiation of Problem 1 based on the selected design principles.

The network layout itself defines the skeleton for the metro map, but it needs to be enriched with further information. The next task is metro map labeling, that is, the placement of legible station names in an additional graphical layer on top of the network layout.

Problem 2 (Metro Map Labeling): Given a schematic layout of the underlying graph $G = (V, E)$ and a station name

for each vertex $v \in V$, place each station name close to its vertex position such that no label intersects any other label or feature of G.

Different methods apply additional design-specific constraints on the possible label positions, for example, one may require horizontally aligned labels or allow the use of diagonal labels, or one may allow introducing line breaks in long station names. Obviously, the labeling quality that can be achieved by a solution of Problem 2 depends directly on the geometry of the layout computed when solving Problem 1. Thus many automated layout methods combine the two problems and solve them in an integrated manner, where the graph layout must provide sufficient space for conflict-free label placement.

A third task that has been considered from an algorithmic perspective is to compute an optimal routing of the (colored) metro lines along the edges of the underlying graph. Here the optimization goal is to minimize the number of line crossings of partially parallel metro lines, which appear whenever two parallel lines meet and split at opposite sides.

Problem 3 (Metro Line Crossings): Given an embedded metro graph $(G, \mathcal{L})$, draw each metro line $L \in \mathcal{L}$ as a continuous curve along its edge sequence in G such that the total number of metro line crossings is minimum.

This line layout problem can in fact be considered independently of Problems 1 and 2 since it is neither affected by the geometry of the layout (only the network topology matters, which is the same in all metro maps) nor by the label placement. At its combinatorial core, we need to determine for each edge and each incident vertex a corresponding order of all metro lines sharing the edge. All line crossings are fully determined by these orders.

B. Design Principles

Next, we list a collection of design principles and rules on which the existing metro map layout methods are based. We refer to this list when discussing individual layout algorithms. Not every method implements all of these rules. Most of the rules are used to define the constraints and quality measures for Problem 1, but one rule also specifies constraints for Problem 2.

(R1) *Do not change the network topology.* This important rule prohibits structural distortions such as modifying the circular edge orders around vertices or introducing additional edge crossings. In graph theoretical terms the *combinatorial embedding* of the layout is preserved. This rule is respected by almost all methods and thus already included in the definition of Problem 1.

(R2) *Restrict edge orientations.* The vast majority of metro map layout methods uses the octilinear set of orientations, that is, horizontal, vertical, and $\pm 45°$-diagonal orientations. Other orientations such as hexalinear (based on $60°$ angles) are possible, too. Some methods do not use straight-line edges at all and resort to curvilinear edges, for example, based on Bézier splines.

(R3) *Draw each individual metro line as straight/monotone as possible and avoid sharp turns.* For traditional polyline drawings, this implies to use as few bends as possible with preferably obtuse angles. For curvilinear drawings, preferably uniform curvature and few inflection points should be used.

(R4) *Metro lines pass straight through interchanges.* Interchange stations are higher degree vertices, where it is particularly important that metro lines are visually easy to follow without ambiguities. This is supported if no metro line changes its orientation in an interchange.

(R5) *Use large angular resolution.* This rule aims to distribute incident edges evenly around vertices.

(R6) *Minimize geometric distortion and displacement.* Many approaches try to stay as close to the input geometry as possible in order to maintain the user's mental map of the city and the resemblance to geography. Some methods apply this rule only locally, that is, the relative positions of pairs of adjacent vertices should be maintained.

(R7) *Use uniform edge lengths.* Since distances in a metro map are not linked to geographic distances, any edge in the layout ideally has the same length. This often implies that dense parts of the network in the city center are enlarged and peripheral stations move closer together.

(R8) *Keep unrelated features apart.* This rule ensures that there is some minimal clearance between non-incident vertices, edges, and around station labels.

(R9) *Avoid large empty spaces in the map.* This rule asks for a balanced local feature density in the whole map.

(R10) *Use unobtrusive and clearly legible placement of station labels.* The precise interpretation of this rule differs between different layout algorithms. Overlapping labels and occlusions are usually prohibited. Horizontally aligned text is mostly preferred, but horizontal metro lines can also be labeled with diagonal labels. Often all labels of stations between two neighboring interchanges are placed coherently on the same side of the path between them.

II. Network Layout Algorithms

In this section we cover algorithms for Problem 1, the most studied subtask in automated metro map design. As mentioned above, some methods take an integrated approach for the layout and labeling of a metro map. In this case we also describe the labeling procedures. We start with a short discussion of the computational problem complexity before summarizing the various layout algorithms that have been proposed over the last 15 years, grouped by the underlying algorithmic principles.

A. Problem Complexity

It is known that minimizing the number of bends in an octilinear graph (or metro map) layout is an NP-hard optimization problem [9]. This result already applies to a very limited set of design rules, that is, apart from rule (R1), enforcing rule (R2) with the octilinear set of orientations suffices to show the NP-hardness of optimizing rule (R3). This is in contrast to bend minimization in the case of orthogonal graph layout, that is, rule (R2) restricting edges to horizontal and vertical orientations only, which can be solved efficiently using network flow algorithms [10]. As a consequence, no exact polynomial-time algorithms for optimizing metro map layouts involving diagonal edge orientations can be expected and all suggested methods are limited in some way or another, for example, by relaxing constraints, applying heuristics, approximations, and local optimization techniques, using asymptotically slow exact computations or restricting the input graphs.

B. Path-Based Schematization

Work on the schematization of paths can be seen as a precursor to schematizing larger, more complex metro graphs. In fact, one may argue that a metro map can be decomposed into a set of schematized paths, although a set of schematized paths cannot be simply combined into a topologically correct network layout. Neyer [11] studied the $\mathcal{C}$-oriented path simplification problem, where a given polygonal input path P is approximated by a $\mathcal{C}$-oriented path Q, where $\mathcal{C}$ is the set of feasible edge orientations according to rule (R2). Here, the goal was to find a schematized path Q with minimum number of links such that Q stays within ϵ distance to P in the Fréchet metric. The problem was solved by dynamic programming in $O(kn^2 \log n)$ time, where n is the number of vertices of P and k is the number of vertices of Q.

Merrick and Gudmundsson [12] studied a similar $\mathcal{C}$-oriented path schematization problem, however, using a less strict distance measure, namely the Hausdorff distance between the $\mathcal{C}$-oriented path Q and the vertices of the input path P. The schematized path needs to pass the input vertices in the correct order and at a distance of at most ϵ. According to the authors, this results in fewer zig-zags in comparison to Neyer's model [11]. The proposed algorithm requires $O(|\mathcal{C}|^3 n^2)$ time and minimizes the number of edges of Q.

Dwyer et al. [13] later suggested a heuristic running time improvement of the previous algorithm by Merrick and Gudmundsson [12]. They use least-squares regression to fit $\mathcal{C}$-oriented edge segments to blocks of vertices of the input path P in $O(|\mathcal{C}|n)$ time. A variation of the algorithm runs in $O(|\mathcal{C}|n \log n)$ time and performs more aggressive simplification, which may result in visually more pleasing schematizations. The authors implemented the algorithms and reported that the results are visually similar to the results of Merrick and Gudmundsson [12], but the new algorithm is much easier to implement and does not require manually setting error parameters.

The three path-based methods satisfy rule (R2) and aim to minimize the number of bends (rule (R3)) and the displacement (rule (R6)). But they all suffer from the problem that, if applied to a path decomposition of a more complex metro graph, the resulting layout is not guaranteed to have the correct topology (rule (R1)).

Delling et al. [14] took a slightly different approach when studying the $\mathcal{C}$-oriented schematization of paths for creating route sketches. To implement rule (R6) they did not consider vertex displacement but rather aimed to preserve the *orthogonal order* of the input, that is, the above/below/left/right relations of all vertex pairs. For monotone paths, they gave a polynomial time algorithm minimizing the distortion of edge slopes; for non-monotone paths they proved NP-hardness and presented a heuristic algorithm as well as an approach based on integer linear programming. Both approaches were implemented and experimentally evaluated, but their method is not suitable for graphs other than paths.

C. Discrete Curve Evolution

Barkowsky et al. [15] presented a network schematization algorithm using *discrete curve evolution*, a shape simplification method for step-wise elimination of the least relevant kinks, where relevance is measured by a function taking into account segment lengths and turn angles. Some input vertices are fixed, others are movable or even removable. Their algorithm aims to straighten subsequent pairs of edges (rule (R3)) and preserves the input topology (rule (R1)), but no edge slope restrictions are applied and displacement of interchange stations is forbidden (rule (R6)). Thus the resulting map layouts do not increase the space for dense downtown regions and they generally appear simplified but not really schematized.

D. Force-Based Layout

Hong et al. [16] adapted the popular force-based paradigm for general graph layout algorithms to the special constraints of metro map layouts. Their most refined algorithm is a topology-preserving (rule (R1)) spring embedder that defines, in addition to the standard spring forces that pull edges to a desired length and the repelling forces between non-adjacent vertices, a set of magnetic forces that pulls edges towards the closest octilinear orientation. In an iterative process, vertices are displaced based on the resulting force vectors until the layout stabilizes. This combination of (potentially conflicting) forces models rules (R2), (R7), and (R8) in a multi-criteria fashion. In addition, the authors performed a preprocessing step that contracts maximal sequences of adjacent degree-2 vertices into a single edge of appropriately scaled target length. This results in straight paths between any two interchange stations (rule (R3)). In an independent second step, Hong et al. applied the LabelHints [17] algorithm to place station labels avoiding label–label overlaps, but not label–edge overlaps. Their algorithm is very fast in practice (in the order of a few seconds), but the resulting layouts are not yet comparable in quality to manually designed metro maps. The main reasons are that the restriction of edge slopes is not strictly enforced, edge lengths are not very uniform, and no attempt is made at limiting distortion or displacement (rule (R6)).

A force-based approach has also been applied by Fink et al. [18] for drawing curvilinear metro maps, thus interpreting rule (R2) differently. Starting with a straight-line or octilinear input drawing each line segment of a metro line is replaced by a nearly-straight cubic Bézier curve that shares tangents with both its predecessor and successor curves. Then attracting and repelling forces are applied to vertices and curve tangents without changing the topology (rule (R1)). The aim of their algorithm is to merge as many consecutive Bézier curves on each metro line as possible in order to increase the monotonicity and straightness of each metro line (rules (R3) and (R4)). Further forces implement rules (R5), (R6), (R7), and (R8). Reported running times are in the order of a few minutes for typical instances. For small and medium-sized instances appealing layouts were computed, but longer metro lines in complex networks (such as London) remained rather wiggly and the intended curve merging could not be applied in all situations.

E. Local Optimization

Local optimization algorithms represent a large class of layout schematization algorithms, being based on the quite natural approach of locally improving the positioning and orientation of vertices and edges step-by-step starting from

the initial geographic layout. Such algorithms usually define a set of weighted quality measures in order to implement the selected set of design rules. Then any two layouts can be compared in terms of their layout quality and it remains to implement an algorithm to intelligently search for high-quality maps since exhaustive search in the solution space is prohibitively expensive.

The first local schematization method was proposed by Avelar and Müller [19] (see also [20]) and demonstrated for the road network of Zurich. They considered topology preservation (rule (R1)), octilinear edge orientations (rule (R2)), and to some extent distances and edge lengths (rules (R7) and (R8)). After a line simplification step on the input graph, all vertices in the layout are iteratively moved to nearby positions that are computed as arithmetic means of the best positions for each of the considered design rules and that do not violate the topology constraint. This process is repeated until a stopping criterion is met. By locally modifying a geographically accurate input layout, displacement is implicitly kept small (rule (R6)). While the topology is indeed preserved and most edges are drawn octilinearly, the algorithm aims to balance potentially conflicting constraints and thus some non-octilinear edge orientations can be observed. Being designed for road maps, certain metro map characteristics related to the shapes of metro lines are also not yet considered. No running times are reported.

Ware et al. [21] presented a similar approach for generating schematic road maps using simulated annealing and implementing more of the design rules. After a line simplification step on the input graph, their algorithm randomly selects a vertex in the layout and moves it to a nearby position. If the modified layout has a better quality measure than the previous layout, the vertex move is performed. With a small probability, solutions with worse quality are also accepted. This process is iterated until a stopping criterion is met. The considered rules are (R1), (R2), and (R6)–(R8), with the highest importance given to the correct topology followed by octilinearity. This is reflected in the weights of the multi-criteria quality function. Metro-map specific constraints on the shape of metro lines are still missing. The reported results for a 750-vertex graph took less than 20s to compute. They also implemented a deterministic gradient-descent version of their previous method, which never performs moves with a negative effect and hence is more susceptible to local optima [22]. Recently, Ware and Richards [23] implemented an *ant colony system* algorithm, which is another local optimization method for the same set of rules and quality measure as used before [21]. Again the basic operation is to randomly select a vertex and locally search for a position that improves the overall quality of the layout. This process is iterated and run in parallel (each process is represented by one *ant*) and vertex positions that yield improvements are marked by increasing a *pheromone* value associated with it. This increases the probability of selecting this position again in future optimization runs performed by other ants. In the end the best found solution is returned. In their experiments, Ware and Richards showed that the new method was both faster and obtained better solutions in comparison to the simulated annealing algorithm of Ware et al. [21].

Stott et al. [24] described a more sophisticated and more tailored local optimization algorithm for metro map layout and labeling. Their layout is based on placing vertices on grid points and like previous methods, for each vertex a set of candidate positions in the vicinity is examined for the position (if any) that most improves the quality value of the entire layout. However, they also considered moving certain groups of vertices and edges as a whole in order to escape some typical local minima situations. Finally, they also apply a similar local search strategy to compute and improve the label positioning. All these local modifications are iterated until some stopping criterion is met. The implemented criteria are based on rules (R1)–(R7), as well as a set of criteria for label placement (rule (R10)). Some criteria are optimized, but without giving guarantees. Others, like preserving relative positions via orthogonal ordering constraints for adjacent vertices and the network topology are strictly enforced. In the resulting layouts one can observe that in many places the layout adheres to the design rules as expected, but in some places their method has difficulties in avoiding local minima and thus there is room for improvements. For example, not all edges are octilinear and congested areas may appear. Reported running times lie in the range of two minutes to two hours, depending on the size of the metro graphs. An empirical study showed that their layouts improved route planning performance over non-schematic and certain official layouts.

An extension of the algorithm by Stott et al. [24] was presented by Chivers and Rodgers [25]. They explored how the local optimization algorithm can be combined with a gesture-based input interface on a tablet computer in order to schematize hand-drawn graph layouts that are not necessarily geographic networks. Some of the previously implemented rules were considered as less important and dropped in their system. In a subsequent study Chivers and Rodgers [26] investigated the effects of different settings for the parameters grid size, displacement limit, and iteration counter in their algorithm.

F. Global Road Map Schematization

Global schematization approaches are different from the previously discussed local techniques as they consider the layout of the entire input graph and not just local displacements of single vertices.

Cabello et al. [27] presented an algorithm for a (road) network schematization problem, in which junction vertices cannot be moved from their initial positions, but edges are schematized as octilinear three-link paths (rule (R2)) without altering the topology (rule (R1)). Their algorithm computes a valid schematic graph layout (or reports failure) by incrementally adding octilinear paths in a suitable order to the layout in $O(n \log n)$ time, where n is the number of edges in the graph. A feasible solution is always found if one exists, otherwise no layout is generated. Thus this is not really an optimization algorithm. We note that the polynomial running time does not contradict the NP-hardness result of Nöllenburg [9] since Cabello et al. [27] did not consider bend minimization.

A conceptually similar incremental approach is the stroke-based schematization algorithm proposed by Li and Dong [28]. Their goal is to create topologically correct (rule (R1)) orthogonal or octilinear (rule (R2)) schematic road maps with bounded distortion (rule (R6)). But unlike the algorithm of Avelar and Müller [19] that locally displaces vertices to create a schematic road map, Li and Dong consider entire roads (i.e., paths that

are called *strokes*) as the basic geometric objects. Schematizing a path as a single object rather than as a set of individual edges often helps to create representations with fewer bends. This approach can be seen as a variation of the metro line design rules (R3) and (R4). Their algorithm first decomposes the input graph into a set of paths, either by road names or by geometric good continuation properties. Each path is schematized using a variation of the Douglas-Peucker line simplification algorithm that takes directional distortion into account and projects path vertices onto an octilinear or orthogonal path approximation. (This step can be replaced by any of the path schematization methods discussed in Section II-B.) Paths are ordered according to type, length, and number of junction vertices and then inserted into the layout in this order. Topological inconsistencies are detected and corrected after each insertion step. Ti and Li [29] extended the stroke-based algorithm by a raster-based preprocessing of the input layout that detects and explicitly enlarges congested areas with high feature density (rule (R9)) by a fish-eye transformation. Since the stroke-based schematization of Li and Dong [28] keeps vertices rather close to their original position, such enlargement is necessary in order to balance the feature density of the resulting layout. They performed a case study with two metro maps that showed the positive effect of the suggested preprocessing step over the original algorithm of Li and Dong. No running times are reported for either of the methods.

G. Mixed-Integer Linear Programming

Linear programming is a global optimization method based on a linear objective function to measure solution quality and a set of linear constraints defining the feasible solution space. Algorithms for solving linear programs find a point in the solution space that yields the best objective value. Linear programs with real-valued variables can be solved in polynomial time, but as soon as variables can also be binary or integers, the problem gets NP-hard. Linear programming with some integer variables is called *mixed-integer linear programming* (MIP). Despite its NP-hardness, MIP is a versatile tool that is frequently used for solving difficult combinatorial optimization problems in practice and sophisticated MIP solvers exist.

Nöllenburg and Wolff [30] designed a metro map layout method based on MIP. They modeled rules (R1), (R2) (with octilinear edge orientations) and (R8) as hard linear constraints that are rigorously enforced in any solution. Rules (R3), (R4), (R6), and (R7) are modeled as a weighted sum forming the linear objective function. Using the commercial MIP solver CPLEX[1] they solved the MIP models and generated corresponding schematic metro maps. Since running times up to several days could observed for finding provably optimal solutions for large networks, a number of improvements were engineered in order to reduce the size of the input graph (contraction of long sequences of degree-2 vertices) and the number of MIP constraints (adding planarity constraints on-demand where needed). Moreover, they included overlapping-free and side-coherent station labeling (rule (R10)) by adding further linear constraints to the initial MIP model. With the above improvements, solutions (potentially with a remaining optimality gap) can be computed in less than a minute for small instances and within minutes up to several hours for more complex instances, in particular if labels are included. In an expert assessment, the maps generated by the MIP method were evaluated and deemed visually superior to the results of Hong et al. [16] and Stott and Rodgers [24]. In some of the design criteria, the MIP-based layout was even preferred over the manually designed official reference layout. This indicates that the method of Nöllenburg and Wolff generates high-quality labeled metro map layouts. owever, solving the MIP is often computationally expensive and there is no guarantee to obtain solutions quickly.

This MIP model was adapted by Milea et al. [31], who included the minimization of the maximum *theoretical planning error* as another optimization goal. This criterion uses the ratio of the travel time along the geometrically shortest path between any two stations in the metro map layout to the actually shortest travel time between the same two stations. The goal of their modification is thus to generate layouts that better support visual route planning. Only preliminary results were reported.

Wu et al. [32] also modified the MIP [30] in order to generate user-specific, travel-route centered metro maps. Their maps aim to place a given travel route horizontally and without bends in the center of the layout. This has the advantage of highlighting the route of interest and making space for large external annotations of the stations along that route. After an initial schematic travel-route centered layout has been computed by solving the modified MIP, interactive editing and constraint modification is possible in order to adapt the layout to personal taste. Finally, a minimum-cost network flow model is used to optimize the placement of large leader-connected labels (also known as boundary labels [33]), for example, thumbnail photographs, next to the central route such that few metro lines are intersected by the leader curves. They reported running times on a desktop PC in the order of a few seconds, which indeed allows to use the method in an interactive setting.

Wu et al. [34] continued to study generating MIP-based metro map layouts that integrate map annotations with large pictographic labels without introducing excessive layout deformations. They implemented a three-step process, which initially computes an unlabeled schematic graph layout using a variation of the MIP of Nöllenburg and Wolff [30]. This layout is then enlarged in order to create enough space to place the labels without overlaps. Label placement is again achieved by a MIP formulation of aesthetic constraints for label positioning and octilinear leader shapes. The labeled layout is finally compacted so that labels are rather tightly contained within their respective faces. This is performed by another MIP that fixes all edge orientation and only shortens edge and leader lengths where possible. Restricting the global solution space by decomposing the task into three independent optimization steps means that globally optimal solutions may be missed, but it also reduces computation times to practically acceptable values between a few seconds and a few minutes depending on layout complexity.

H. Least-Squares Optimization

Wang and Chi [35] modeled the metro map layout problem as a set of weighted, squared energy terms based on variables for vertex positions and edge rotations. This energy function is minimized using the numerical iterative conjugate gradient method. In fact, their approach is described in a focus-and-context sense, where a selected travel route is highlighted and

[1] http://www.ibm.com/software/commerce/optimization/cplex-optimizer/

the remaining edges are providing context information in the background. However, it can also be adapted to generate layouts of the entire metro graph. Their algorithm performs three sequential steps. First, a deformed layout without constraining edge orientations is computed. In this layout, uniformity of edge lengths (rule (R7)), angular resolution (rule (R5)), and vertex displacement (rule (R6)) is optimized in a weighted fashion while edge crossings are avoided heuristically. The second step discretizes all edge directions in the initial layout to become octilinear (rule (R2)) by defining a corresponding energy function. Finally, energy terms for optimal label placement (rule (R10)) in the fixed schematized layout are formulated, taking into account preferential positions, occlusions, spacing, and coherence. Although the least-squares optimization does not provide any formal guarantees (unless a configuration with zero energy is found), the resulting layouts in the reported case studies satisfy most of the implemented constraints, are generally of good visual quality, and can often be computed in less than a second.

III. Station Labeling Algorithms

If automated metro map layout methods consider label placement at all, it is often included as an integrated part of the algorithm due to the mutual interdependencies between graph layout and label placement. Such examples have been discussed in the previous section. Apart from the large body of work on general map labeling algorithms [36], which may also be applied to placing station labels to some extent, we are aware of only a few stand-alone algorithms for metro map labeling (Problem 2) according to rule (R10). The disadvantage of such algorithms is that they cannot modify the graph layout if more space is needed.

Garrido et al. [37] presented algorithms for placing unit-height rectangular or square labels along horizontal and diagonal lines, which appears as a natural problem in labeling octilinear metro maps. The goal is to find the largest possible label scaling factor (that is, the font size) for which all vertices on the given line can be labeled with disjoint horizontally aligned labels. The authors took a primarily theoretical perspective and showed that the problem is (weakly) NP-hard for points on a horizontal line if sliding rectangular labels are used; a pseudo-polynomial time algorithm is given. On the other hand, they presented polynomial-time algorithms for the case of a diagonal line. The algorithms were not implemented.

Wu et al. [38] presented a zone-based algorithm for placing textual and image labels for stations in a metro map. They defined three different zones in the map: the area immediately surrounding the edges and vertices of the network drawing is forbidden for labels, the area obtained by dilating the edges by a fixed width is reserved for the station names, and the remaining area defines the zone reserved for placing images. Using a genetic algorithm, the labeling is optimized in two phases. Initially, the textual labels are placed greedily at the best still available position (straight-line leaders can be used if necessary), where the genetic algorithm determines the best order for the greedy placement. Once textual labels are placed, the same greedy approach is used to place images (preferably at the map boundary) and their leaders. Two case studies are presented, but no running times are given.

IV. Line Layout Algorithms

Finally, an independent combinatorial subtask is to minimize the metro line crossings between parallel metro lines as defined in Problem 3. Since this is of practical interest only in metro networks with many parallel metro lines meeting in complex patterns and since the existing algorithms have not yet been implemented, we keep our description rather brief at this point.

The problem was formally introduced by Benkert et al. [39] who presented a quadratic-time algorithm for optimizing the line orders of a single edge. Argyriou et al. [40] showed that it is NP-hard to minimize line crossings even on a path under the constraint that lines that terminate in some station must be located on the outside of their respective bundle of parallel lines when reaching that station. Recently Fink and Pupyrev [41] showed that Problem 3 is also NP-hard without restricting terminus positions, even on caterpillar graphs. On the positive side, efficient algorithms exist for general planar metro map layouts if all lines terminate at vertices of degree 1 or if the side assignment (left/right) of each terminus is given [40], [42]–[44]. The side assignment of termini can be computed using integer linear programming [42]. If the side assignment is not specified, efficient approximation algorithms for general planar graphs [41] and exact fixed-parameter algorithms for paths [45] are known. Fink and Pupyrev [46] defined an interesting variation of Problem 3, in which the goal is not to minimize individual crossings, but the number of *block crossings* that combine multiple crossings of two bundles of metro lines into a single entity. They presented approximation algorithms and worst-case optimal heuristics for block-crossing minimization.

V. Conclusions

In this survey, we have given a comprehensive overview of the state of the art in automating the design of metro maps, separated into the three different subtasks *layout*, *labeling*, and *line routing*. Schematic map design is a complex task that is usually performed by cartographers, graphic designers, and artists. Hence it cannot be expected that any of the discussed methods with their specific attempt to capture human aesthetic taste in a mathematical formula can create map layouts that are of the same visual quality as those drawn by skilled human map designers. We see two immediate practical purposes of algorithms for creating schematic metro maps. On the one hand, they can be used to produce high-quality base layouts serving as templates and inspirations for graphic designers. On the other hand, they can be used to produce on-demand individual or special-purpose maps, for which it is simply to expensive and impracticable to engage a human designer. As this survey shows, many suitable methods have been developed for both tasks, some of which have also been successfully evaluated with users and domain experts. But no single algorithm stands out that is superior in all respects. Rather one has to carefully decide which of the approaches provides the best fit for a certain task. For future work in this field one of the main challenges, apart from further improving algorithm performance and layout quality especially for large and complex networks (not necessarily public transit networks), is to better include global design principles into the optimization process such as showing symmetries and balancing feature density. Moreover, the existing methods are based on drawing thin curves and

point vertices. Thus they cannot handle thick bundles of parallel metro lines and large symbols for interchange stations well. Further progress on the integration of user-defined constraints in an interactive semi-automatic metro map layout system is also important for creating useful tools for map designers.

REFERENCES

[1] D. J. Bartram, "Comprehending spatial information: The relative efficiency of different methods of presenting information about bus routes," *J. Applied Psychology*, vol. 65, no. 1, pp. 103–110, 1980.

[2] M. J. Roberts, E. J. Newton, F. D. Lagattolla, S. Hughes, and M. C. Hasler, "Objective versus subjective measures of Paris metro map usability: Investigating traditional octolinear vs. all-curves schematics," *Int. J. Human-Computer Studies*, vol. 71, pp. 363–386, 2013.

[3] K. Garland, *Mr Beck's Underground Map*. Capital Transport Pub, 1994.

[4] M. J. Roberts, *Underground Maps Unravelled*, 2012.

[5] M. Ovenden, *Metro Maps of the World*. Capital Transport Pub, 2003.

[6] A. Wolff, "Drawing subway maps: A survey," *Informatik – Forschung und Entwicklung*, vol. 22, no. 1, pp. 23–44, 2007.

[7] ——, "Graph drawing and cartography," in *Handbook of Graph Drawing and Visualization*, R. Tamassia, Ed. CRC Press, 2013, ch. 23, pp. 697–736.

[8] W. Mackaness and A. Reimer, "Generalisation in the context of schematised maps," in *Abstracting Geographic information in a Data Rich World*, ser. Lecture Notes in Geoinformation and Cartography, Springer, 2014, ch. 10, pp. 299–328.

[9] M. Nöllenburg, "Automated drawing of metro maps," Fakultät für Informatik, Universität Karlsruhe, Tech. Rep. 2005-25, 2005. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000004123

[10] R. Tamassia, "On embedding a graph in the grid with the minimum number of bends," *SIAM J. Comput.*, vol. 16, no. 3, pp. 421–444, 1987.

[11] G. Neyer, "Line simplification with restricted orientations," in *Algorithms and Data Structures (WADS'99)*, ser. LNCS, vol. 1663. Springer, 1999, pp. 13–24.

[12] D. Merrick and J. Gudmundsson, "Path simplification for metro map layout," in *Graph Drawing (GD'06)*, ser. LNCS, vol. 4372. Springer, 2007, pp. 258–269.

[13] T. Dwyer, N. Hurst, and D. Merrick, "A fast and simple heuristic for metro map path simplification," in *Advances in Visual Computing (ISVC'08)*, ser. LNCS, vol. 5359. Springer, 2008, pp. 22–30.

[14] D. Delling, A. Gemsa, M. Nöllenburg, T. Pajor, and I. Rutter, "On d-regular schematization of embedded paths," *Comput. Geom. Theory Appl.*, vol. 47, no. 3A, pp. 381–406, 2014.

[15] T. Barkowsky, L. J. Latecki, and K.-F. Richter, "Schematizing maps: Simplification of geographic shape by discrete curve evolution," in *Spatial Cognition II*, ser. LNCS, vol. 1849. Springer, 2000, pp. 41–53.

[16] S.-H. Hong, D. Merrick, and H. A. D. do Nascimento, "Automatic visualisation of metro maps," *J. Visual Languages and Computing*, vol. 17, no. 3, pp. 203–224, 2006.

[17] H. A. D. do Nascimento and P. Eades, "User hints for map labeling," *J. Visual Languages and Computing*, vol. 19, no. 1, pp. 39–74, 2008.

[18] M. Fink, H. Haverkort, M. Nöllenburg, M. J. Roberts, J. Schuhmann, and A. Wolff, "Drawing metro maps using Bézier curves," in *Graph Drawing (GD'12)*, ser. LNCS, vol. 7704. Springer, 2013, pp. 463–474.

[19] S. Avelar and M. Müller, "Generating topologically correct schematic maps," in *Spatial Data Handling (SDH'00)*, 2000, pp. 4a.28–4a.35.

[20] S. Avelar, "Convergence analysis and quality criteria for an iterative schematization of networks," *Geoinformatica*, vol. 11, pp. 497–513, 2007.

[21] J. M. Ware, S. Anand, G. E. Taylor, and N. Thomas, "Automated production of schematic maps for mobile applications," *Transactions in GIS*, vol. 10, no. 1, pp. 25–42, 2006.

[22] S. Anand, S. Avelar, J. M. Ware, and M. Jackson, "Automated schematic map production using simulated annealing and gradient descent approaches," in *GIS Research Conference UK (GISRUK'07)*, 2007, pp. 414–420.

[23] M. Ware and N. Richards, "An ant colony system algorithm for automatically schematizing transport network data sets," in *Evolutionary Computation (CEC'13)*, 2013, pp. 1892–1900.

[24] J. Stott, P. Rodgers, J. C. Martínez-Ovando, and S. G. Walker, "Automatic metro map layout using multicriteria optimization," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 1, pp. 101–114, 2011.

[25] D. Chivers and P. Rodgers, "Gesture-based input for drawing schematics on a mobile device," in *Inform. Visualisation (IV'11)*, 2011, pp. 127–134.

[26] ——, "Exploring local optima in schematic layout," in *Distributed Multimedia Systems (DMS'13)*, 2013, pp. 168–175.

[27] S. Cabello, M. de Berg, M. van Kreveld, "Schematization of networks," *Comput. Geom. Theory Appl.*, vol. 30, no. 3, pp. 223–238, 2005.

[28] Z. Li and W. Dong, "A stroke-based method for automated generation of schematic network maps," *Int. J. Geographical Information Science*, vol. 24, no. 11, pp. 1631–1647, 2010.

[29] P. Ti and Z. Li, "Generation of schematic network maps with automated detection and enlargement of congested areas," *Int. J. Geographical Information Science*, vol. 28, no. 3, pp. 521–540, 2014.

[30] M. Nöllenburg and A. Wolff, "Drawing and labeling high-quality metro maps by mixed-integer programming," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 5, pp. 626–641, 2011.

[31] T. Milea, O. Schrijvers, K. Buchin, and H. Haverkort, "Shortest-paths preserving metro maps," in *Graph Drawing (GD'11)*, ser. LNCS, vol. 7034. Springer, 2012, pp. 445–446.

[32] H.-Y. Wu, S. Takahashi, C.-C. Lin, and H.-C. Yen, "Travel-route-centered metro map layout and annotation," *Computer Graphics Forum*, vol. 31, no. 3, pp. 925–934, 2012.

[33] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff, "Boundary labeling: Models and efficient algorithms for rectangular maps," *Comput. Geom. Theory Appl.*, vol. 36, no. 3, pp. 215–236, 2007.

[34] H.-Y. Wu, S. Takahashi, D. Hirono, M. Arikawa, C.-C. Lin, and H.-C. Yen, "Spatially efficient design of annotated metro maps," *Computer Graphics Forum*, vol. 32, no. 3, pp. 261–270, 2013.

[35] Y.-S. Wang and M.-T. Chi, "Focus+context metro maps," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2528–2535, 2011.

[36] A. Wolff and T. Strijk, "The map labeling bibliography." [Online]. Available: http://i11www.iti.kit.edu/~awolff/map-labeling/bibliography/

[37] M. Á. Garrido, C. Iturriaga, A. Márquez, J. R. Portillo, P. Reyes, and A. Wolff, "Labeling subway lines," in *Algorithms and Computation (ISAAC'01)*, ser. LNCS, vol. 2223. Springer, 2001, pp. 649–659.

[38] H.-Y. Wu, S. Takahashi, C.-C. Lin, and H.-C. Yen, "A zone-based approach for placing annotation labels on metro maps," in *Smart Graphics (SG'11)*, ser. LNCS, vol. 6815. Springer, 2011, pp. 91–102.

[39] M. Benkert, M. Nöllenburg, T. Uno, and A. Wolff, "Minimizing intra-edge crossings in wiring diagrams and public transportation maps," in *Graph Drawing (GD'06)*, ser. LNCS, vol. 4372. Springer, 2007, pp. 270–281.

[40] E. Argyriou, M. A. Bekos, M. Kaufmann, and A. Symvonis, "On metro-lince crossing minimization," *J. Graph Algorithms Appl.*, vol. 14, no. 1, pp. 75–96, 2010.

[41] M. Fink and S. Pupyrev, "Metro-line crossing minimization: Hardness, approximations, and tractable cases," in *Graph Drawing (GD'13)*, ser. LNCS, vol. 8242. Springer, 2013, pp. 328–339.

[42] M. Asquith, J. Gudmundsson, and D. Merrick, "An ILP for the metro-line crossing problem," in *Computing: Australasian Theory Symposium (CATS'08)*, ser. CRPIT, vol. 77, 2008, pp. 49–56.

[43] M. Nöllenburg, "An improved algorithm for the metro-line crossing minimization problem," in *Graph Drawing (GD'09)*, ser. LNCS, vol. 5849. Springer, 2010, pp. 381–392.

[44] S. Pupyrev, L. Nachmanson, S. Bereg, and A. E. Holroyd, "Edge routing with ordered bundles," in *Graph Drawing (GD'11)*, ser. LNCS, vol. 7034. Springer, 2012, pp. 136–147.

[45] Y. Okamoto, Y. Tatsu, and Y. Uno, "Exact and fixed-parameter algorithms for metro-line crossing minimization problems," *CoRR*, vol. abs/1306.3538, 2013.

[46] M. Fink and S. Pupyrev, "Ordering metro lines by block crossings," in *Mathematical Foundations of Computer Science (MFCS'13)*, ser. LNCS, vol. 8087. Springer, 2013, pp. 397–408.