MetroSAT: Logic-based computation of metro maps

Samantha Fuchs
Algorithms and Complexity Group
TU Wien
Vienna, Austria
samantha.fuchs@student.tuwien.ac.at

Soeren Nickel
Algorithms and Complexity Group
TU Wien
Vienna, Austria
soeren.nickel@ac.tuwien.ac.at

Martin Nöllenburg
Algorithms and Complexity Group
TU Wien
Vienna, Austria
noellenburg@ac.tuwien.ac.at

***Abstract*—Computing schematic metro maps is often framed as an NP-hard optimization task. A natural way to model such a task is to use rigorous mathematical optimization techniques like Integer Linear Programming (ILP). We present two alternative logic-based models, which obtain high-quality results using Maximum Satisfiability (MaxSAT) and Satisfiability Modulo Theory (SMT), both employing dedicated solvers, to achieve high-quality results in a competitive time.**

I. Introduction

Transit or metro maps are abstract representations of stations and their connections in a public transportation network. Exact geographical relations and positions can be distorted to emphasize network topology and legibility. A common class of such maps are octolinear metro maps, which draw edges in one of the eight octolinear directions and reduce line complexity by avoiding bends and distributing stations equally along them.

Various automated approaches to compute schematic maps of geographically embedded metro networks have been presented. State-of-the-art approaches include least squares optimization [14], force-based graph drawing [6], multi-criteria hill climbing [12], [13] and iterative shortest path search on octolinear grids [2], [3] (for a complete overview, we refer to a recent and extensive survey [15]). However there have also been successful approaches using declarative constraint-based approaches like Integer Linear Programming (ILP) [2], [10]. In spite of being NP-hard optimization problems themselves, they have been shown to reasonably quickly yield high-quality results for instances of moderately sized metro maps using highly optimized state-of-the-art solvers like Gurobi or CPLEX. These methods are highly customizable, can be easily extended [3], [9] and do not need deep algorithmic know-how beyond formal constraint modeling.

This work is inspired by experimental results in other areas of graph drawing, where logic-based methods such as MaxSAT [16] and SMT [8] have been applied and showed significant runtime improvements over ILP approaches. We present a MaxSAT and an SMT[1] model for the formal Metro Map Layout Problem defined by Nöllenburg and Wolff [10].

Formally we are given a planar input graph $G = (V, E)$, with an embedding (assumed to be the geographical positions of the stations connected with straight line segments). Further we are given a line cover $\mathcal{L}$, which is a set of paths (the metro lines), whose union is E. The goal is to find a topology-preserving plane drawing of G, in which every edge is a straight line segment parallel to the four *octolinear* orientations (horizontal, vertical and $\pm 45°$-diagonal).

While our set of constraints is in large parts equivalent to the existing ILP model of Nöllenburg and Wolff [10], the main contribution is the translation of the linear arithmetic constraints into a SAT model in Boolean logic (Section II) and the combination of linear arithmetic constraints and Boolean clauses into an SMT model (Section III). Finally we present a small case study in Section IV and conclude with open problems in Section V.

II. SAT Model

A Boolean *variable* a can be either *true* ($a = 1$) or *false* ($a = 0$) and its *negation* $\neg a$ inverts the truth value of a; a and $\neg a$ are denoted as *literals*. A *clause* $c = (l_1 \vee \cdots \vee l_i)$ is a set of literals and we say c is true ($c = 1$) if any $l \in c$ is true. A Boolean *formula* $\phi = c_1 \wedge \cdots \wedge c_j$ is a set of clauses and we say ϕ is *satisfiable* if we can assign true or false to all variables in ϕ, s.t., all $c \in \phi$ are true[2]. A weighted MaxSAT instance is formula $\psi = c_1 \wedge \cdots \wedge c_j \wedge s_1 \wedge \cdots \wedge s_k$ and a set of integer weights $w_1, \ldots, w_k$, where $s_1, \ldots, s_k$ are called *soft clauses*. The MaxSAT problem asks to find a variable assignement, s.t., $c_1 \wedge \cdots \wedge c_j$ (hard clauses) is true and the sum $\sum_{i=1}^{k} w_i s_i$ is maximized. We will use MaxSAT to model the hard and soft constraints of the metro map layout problem.

A. Unary Encoding

To encode integers in our SAT model, we use unary encoding as defined by [16]. Integer variables are denoted by a, b and are considered to be bounded in this section. Scalar values are denoted by i, j, k. An integer variable a that is bounded by $l_a \leq a \leq u_a$ is encoded by $u_a - l_a$ different Boolean variables $(a^{l_a+1}, a^{l_a+2}, \cdots, a^{u_a-1}, a^{u_a})$. Note that this requires bounded variables, as we only use a constant set of boolean variables per integer variable. The state $a^i = 1$ in unary encoding is equivalent to the constraint $a \geq i$, hence $a = i$ is encoded by the assignment $(a^{l_a+1} = 1, \cdots, a^i = 1, a^{i+1} = 0, \cdots, a^{u_a} = 0)$. Note that we omit the variable a^{l_a} as it has to be true in every assignment. To ensure this

[1] While this could be called a MaxSMT model, as it also maximizes a given objective function, we will simply use the term SMT model in this paper.

[2] This definition of a Boolean formula is a special case called *conjunctive normal form* (CNF); every Boolean formula can be transformed into CNF.

behaviour, we add the implication clauses $\neg a^i \vee a^{i-1}$ for each unary variable a within its range $l_a + 2 \leq i \leq u_a$.

We can also model inequalities $a \leq b$ between two integer variables a and b as a set of clauses $\neg a^i \vee b^i$ for all $\max\{l_a, l_b\}+1 \leq i \leq \min\{u_a, u_b\}$. The unary encoding of an integer a requires only a linear number (in the smaller range of the two variables) of clauses to be instantiated, instead of a quadratic amount necessary for a naïve encoding (setting one variable to true and all other to false for every allowed value of a). Note that we can easily model equality constraints $a = b$ as $a \leq b \wedge b \leq a$, offsets ($a+g \leq b$) and coefficients ($a = b \cdot g$) by adding or multiplying the offset g to the index of b.

Equations (and inequalities) containing three different integer variables are more complex. With two variables we needed one clause per index in the specific range, but now we need two indices and add clauses for each combinations of them. For $a - b \leq c$ this leads to the following clauses:

$$\neg a^i \vee b^j \vee c^{i-j+1} \qquad \begin{array}{c} \forall l_a < i \leq u_a \\ \forall l_b < j \leq u_b \\ \text{with } l_c < i - j + 1 \leq u_c. \end{array}$$

Again we can restrict the values for each variable using the bounds of the other variables. From $a \leq c+b \leq u_b+u_c$ we get $\neg a^{u_b+u_c+1}$, from $l_a-u_b \leq a-b \leq c$ we get $c^{l_a-u_b}$ and $b^{l_a-u_c}$ respectively. Additionally we disallow each combination of two variables that would exceed the range of the third variable. The constraints for $a+b=c$ can be built in a similar fashion.

In the following sections we use integer variables, inequalities and equations for better readability. In the implementation these are replaced by clauses as defined in this section.

B. Coordinates

In a schematic metro map each vertex v of the input graph G has Cartesian $x(v)$ and $y(v)$ coordinates in the plane. In an optimized drawing we can use an underlying grid[3] and represent these coordinates with integers. To model all needed constraints we use a (partially redundant) set of four variables $x(v)\widehat{=}z_0(v)$, $y(v)\widehat{=}z_2(v)$, $z_1(v)$ and $z_3(v)$ (representing one of the octolinear direction each), from [10] with one modification, namely the coordinates are doubled to ensure integrality. This is necessary, because with our simple implementation of unary encoding we can only model integer values.

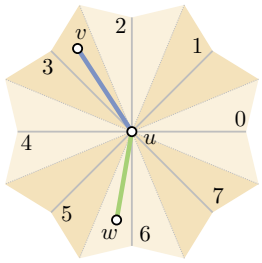


Fig. 1: The closest octolinear direction of (u, v) and (u, w) is 3 and 6, respectively.

$$z_0(v) = 2x(v) \tag{1}$$
$$z_1(v) = x(v) + y(v) \tag{2}$$
$$z_2(v) = 2y(v) \tag{3}$$
$$z_3(v) = y(v) - x(v) \tag{4}$$

[3]Gridsizes in this paper were determined experimentally.

An edge (u, v) has a direction variable $\mathrm{dir}(u, v)$, taking values in $\{0, 1, \ldots, 7\}$ depending on $z_0(u)$, $z_1(u)$, $z_2(u)$, $z_3(u)$ and $z_0(v)$, $z_1(v)$, $z_2(v)$, $z_3(v)$. Valid values for $\mathrm{dir}(u, v)$ are the closest octolinear direction $sec_u(v)$ (Fig. 1) plus/minus a constant offset *dev*, forming the set $S(u, v)$ of admissible directions. We will use $z_i^o = z_{i+2 \mod 4}$ for the coordinate in the orthogonal direction to z_i and we consider all indices for z_i, z_i^o to be modulo 4.

C. Hard Constraints

Following the ILP model, there are three hard constraints any valid schematic map has to fulfill, namely (I) guaranteeing octolinearity and proper spacing between connected stations, (II) preserving the radial order of outgoing connections at every stations from the input and (III) planarity and proper spacing between edges.

Note that parts of these constraints are covered, via the integer coordinate restriction, i.e., two stations on different grid points and two non-crossing edges have (some) minimum distance. We will now illustrate the nuances necessary to adapt the constraints, exemplary for constraint (I) below.

Edge directions and minimum length: Following [9], [10], we introduce for each edge $(u, v) \in E$ a set of Boolean variables $\alpha_i(u, v)$ (per $i \in S(u, v)$) and the following constraints.

$$\bigvee\nolimits_{i \in S(u,v)} \alpha_i(u, v) \tag{5}$$
$$\neg\alpha_i(u, v) \vee \neg\alpha_j(u, v) \qquad \forall i < j \in S(u, v) \tag{6}$$
$$\neg\alpha_i(u, v) \vee (\mathrm{dir}(u, v) = i) \qquad \forall i \in S(u, v) \tag{7}$$
$$\neg\alpha_i(u, v) \vee (\mathrm{dir}(v, u) = i + 4 \bmod 8) \qquad \forall i \in S(u, v) \tag{8}$$
$$\neg\alpha_i(u, v) \vee (z_i^o(u) = z_i^o(v)) \qquad \forall i \in S(u, v) \tag{9}$$
$$\neg\alpha_i(u, v) \vee (z_i(v) + L_{min} \leq z_i(u)) \qquad \forall i \in S(u, v) \tag{10}$$

Equations (5) and (6) enforce exactly one α_i to be true, (7) and (8) ensure that $\alpha_i(u, v)$ is true iff $\mathrm{dir}(u, v) = (\mathrm{dir}(v, u) + 4) \bmod 8 = i$, (9) ensures that (u, v) is drawn in the correct octolinear direction, by forcing their coordinates in orthogonal direction to be equal and (10) gives a minimal distance of L_{min}.

Setting $\alpha_i = 0$ trivially satisfies clauses (6)–(10). For some $j \in S(u, v)$, however, we need to set $\alpha_j = 1$ to satisfy (5); this j corresponds to the actual direction of (u, v) in a solution. While this behaviour is easily expressed in Boolean logic, an ILP is must usually use big-M constraints (see [10]), which is computationally expensive. Constraints (II) and (III) can be similarly transformed from the original ILP model.

D. Optimization

To achieve the desired properties of a high-quality metro map, we encode the minimization of (i) line bends along lines in $\mathcal{L}$, (ii) deviation of the direction $\mathrm{dir}(u, v)$ to the desired original direction $sec_u(v)$ of all edges (u, v) in E and (iii) total length over all edges in E into our MaxSAT instance. For this we will use additional hard clauses and weighted soft clauses. Again we showcase the translation of the constraints as they are used in the ILP model with the example of minimizing the deviation of $\mathrm{dir}(u, v)$. Other contraints are translated similarly.

Relative positions: For each edge (u, v) we introduce an integer variable $\xi(u, v)$, that measures the difference between $\mathrm{dir}(u, v)$ and $sec_u(v)$, the two binary correction variables $\eta_1(u, v)$ and $\eta_2(u, v)$ and the following constraints. Additionally this set of constraints is added for every $1 \leq i \leq dev$.

$$\eta_2(u,v) \vee \xi(u,v)^i \vee (sec_u(v) - \mathrm{dir}(u,v) < i) \quad (11)$$

$$\eta_1(u,v) \vee \xi(u,v)^i \vee (\mathrm{dir}(u,v) - sec_u(v) < i) \quad (12)$$

$$\neg\eta_1(u,v) \vee \xi(u,v)^i \vee (sec_u(v) - \mathrm{dir}(u,v) + 8 < i) \quad (13)$$

$$\neg\eta_2(u,v) \vee \xi(u,v)^i \vee (\mathrm{dir}(u,v) - sec_u(v) + 8 < i) \quad (14)$$

For any $1 \leq i \leq dev$ and any $0 \leq sec_u(v), \mathrm{dir}(u, v) \leq 7$, exactly two of these constraints are satisfied regardless of $\eta_1(u, v), \eta_2(u, v)$ or $\xi(u, v)^i$, if and only if $|sec_u(v) - \mathrm{dir}(u, v)|$ or $8 - |sec_u(v) - \mathrm{dir}(u, v)|$ is strictly smaller than i and the other two constraints can be satisfied using $\eta_1(u, v), \eta_2(u, v)$. If this is not the case, we have to set $\xi(u, v)^i$ to true, which is equivalent to $\xi(u, v) = \min\{|\mathrm{dir}(u, v) - sec_u(v)|, 8 - |\mathrm{dir}(u, v) - sec_u(v)|\}$ (see Section II-A).

Since $sec_u(v)$ is a scalar value, which is known at the time of instance creation, we use that $\mathrm{dir}(u, v) > c$ and $\mathrm{dir}(u, v) < c'$ can be modelled as $\mathrm{dir}(u, v)^{c+1}$ and $\neg\,\mathrm{dir}(u, v)^{c'}$ respectively and arrive at a simplified set of constraints.

$$\eta_2(u,v) \vee \xi(u,v)^i \vee \mathrm{dir}(u,v)^{sec_u(v)-i+1} \quad (15)$$

$$\eta_1(u,v) \vee \xi(u,v)^i \vee \neg\,\mathrm{dir}(u,v)^{sec_u(v)+i} \quad (16)$$

$$\neg\eta_1(u,v) \vee \xi(u,v)^i \vee \mathrm{dir}(u,v)^{sec_u(v)-i+9} \quad (17)$$

$$\neg\eta_2(u,v) \vee \xi(u,v)^i \vee \neg\,\mathrm{dir}(u,v)^{sec_u(v)+i-8} \quad (18)$$

Minimizing the sum of all integer variables $\xi(e)$ for all $e \in E$ minimizes the total number of direction deviations in the final layout. To do this in MaxSAT, we add the following soft clauses with weight f_1 for every edge (u, v).

$$\xi(u,v) < i \qquad \forall 1 \leq i \leq dev \quad (19)$$

In particular this means, we simply add the negated variables $\neg\xi(u, v)^i$ for all edges (u, v) and all $1 \leq i \leq dev$ as soft clauses and set their weight to f_1. In a very similar fashion, we can define corresponding minimization variables θ and λ, with weights f_2 and f_3 to minimize the total number of line bends and the total edge length, respectively.

III. SMT Model

Our second model uses a different SAT-based optimization technique called SAT modulo Theory (SMT). Intuitively speaking, an SMT formula Φ also consists of a set of clauses and is true if every single clause is true; the concept of a clause, however, is broadened to allow expressions of integer linear arithmetic in place of literals. A truth assignment for Φ will assign every such expression either true or false and the set of all expressions, which are assigned true can be checked for consistency using a dedicated solver for integer linear arithmetic. If there exists a variable assignment, which is consistent for both the Boolean variables in Φ and the expressions, which are assigned true, Φ is satisfiable. For a more detailed introduction and definition of SMT, we refer to Griggio [7]. Note that we can formulate soft clauses in the exact same fashion as in MaxSAT.

A benefit of an SMT solver over SAT solvers is that we can use data types, like (unbounded) integers directly, instead of using a unary encoding. With these integer variables we can use the hard constraints (e.g. Equations (5)–(10)) as they are, since the theory of linear integer arithmetic supports the use of inequalities and equations in SMT.

Soft constraints are handled in a very similar manner as in our MaxSAT approach. However, we instantiate the inequalities $\xi(u, v) < i$ for every $1 \leq i \leq dev$ (and the corresponding inequalities for θ and λ) directly as weighted soft constraints.

IV. Experimental Results

Both models were implemented and tested. In this section, we first describe some implementation details and optimizations, then compare our experimental run-time results with the existing ILP implementation and finally we showcase the schematic maps created using the MaxSAT and SMT models.

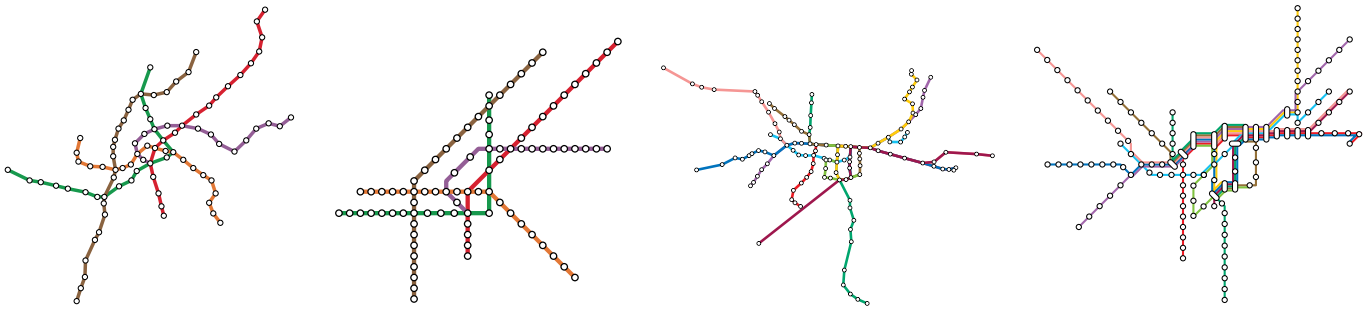
A. Implementation

Our entire implementation uses python 3.7. We used the python package PySAT as a modeling interface to create the MaxSAT model. This package also provides a choice of MaxSAT solvers to find optimal variable assignments for the created formulas; we used the solver RC2.

While SMT solvers are readily available and competitions between existing solvers are held, which can be consulted to find a suitable solver for a given formulation, support for optimizing a given objective expression, while finding a satisfying variable assignment is often lacking. In fact, the concept of optimization is not integrated into the SMT-LIB Standard [1]. However the SMT solver Z3 provides the module νZ [4], [5], which allows the maximization over expressions in the used theory (in our case integer linear arithmetic) as well as support for weighted soft clauses. We used νZ to model and solve our SMT formulation.

For comparison, we used the ILP formulation of Nöllenburg and Wolff [10], implemented using Gurobi 9.5.0 as a solver and the gurobipy package to model the ILP formulation.

Further we used the following optimizations for all three formulations. If crossings were present in the input, we planarized it, i.e., we removed the crossing segments, placed a dummy vertex at the crossing point, which is connected to the endpoints of the crossing segments and removed the vertex again, before rendering. Additionally, we used the well-established *DEG-2* heuristic. If an input graph contains paths of k degree-2 vertices between two vertices of degree not equal to two (either interchanges or terminals), we replace the entire path by a 3-edge path (modeling two bend points) whose middle edge has minimum length $k - 1$ to reserve sufficient space for re-inserting the degree-2 vertices later. When re-inserting, we distribute all stations on these compressed paths equally over their entire length in the final schematization. Note that the last two steps emphasize equal distance between stations over integer coordinates of such stations.



(a) Geographical input layout of Vienna metro network.

(b) Schematic MaxSAT and SMT layouts for Vienna.

(c) Geographical input layout of Karlsruhe tram network.

(d) Schematic MaxSAT and SMT layouts for Karlsruhe.

Fig. 2: Side by side view of the metro networks with stations at their geographical input locations (a), (c) and the their respective schematization (b), (d). Both the MaxSAT and SMT model produced the same layout.

TABLE I: Model parameters and running (wall clock) times in seconds. Times marked with * exceeded the limit of 10 hours.

	Vienna			Karlsruhe		
$f =$	$(3,2,1)$	$(6,4,2)$	$(5,2,1)$	$(3,2,1)$	$(6,4,2)$	$(5,2,1)$
SAT	1,172	531	102	9,895	5,904	20,176
SMT	5,803	5,232	347	10,027	9,882	6,801
ILP	272	220	4,035	*36,000	*36,000	*36,000

B. Runtime Comparison

For our experiments we used a computation cluster outfitted with a Intel Xeon E5-2640 v4, 2.40GHz 10-core processor (note that all solvers used only a single thread). We used two standard benchmarks, the metro networks of Vienna (90 vertices and 96 edges) and Karlsruhe (127 vertices and 135 edges). The sizes of the underlying grids for the SAT formulation were determined experimentally and are $X \times X$ and $Y \times Y$, respectively; recall that the SMT formulation does not require a bounded grid. All MaxSAT and SMT instances had a maximum available memory of 8GB, while ILP instances were run with 16GB and 100GB for Vienna and Karlsruhe, respectively. Three weight vectors (f_1, f_2, f_3) were used, namely $F_1 = (3,2,1)$, $F_2 = (6,4,2)$ (which has the same relative weighting) and the more pronounced $F_3 = (5,2,1)$. The observed runtimes are presented in Table I. The resulting MaxSAT and SMT layouts for F_1 are shown in Fig. 2. Note that both produced the same layout, however this is not necessarily the case, as multiple layouts can have the same objective function value.

Comparing F_1 and F_2, we see that the SAT and SMT models decrease in solving time when multiplying the weights of F_1 by 2. Using F_3 results in an even shorter time on the Vienna map (significantly so for SMT), but we see diverging behavior for the Karlsruhe network, where SMT is yet again faster than F_1 and F_2, but MaxSAT has a spike in running time. On repeated executions of the solver, we found a solving time of 12387 seconds, indicating a possibly high variance in computation times for these approaches.

Comparing running times to the ILP model, we note that the ILP outperforms our approaches on the smaller Vienna map, but optimal solutions for the larger map were found by MaxSAT and SMT, while the ILP exceeded our set time limit of ten hours. This could indicate a better scalability of our logic-based approach. Here it should also be noted that the ILP solver ended with a reported optimality gap between 13 and 16 percent, but reported intermediate solutions during its runtime. Such functionality is possible, when using MaxSAT and SMT solvers, as they maintain intermediate solutions, however neither neither of our models currently supports this without interrupting the solution process.

V. Conclusion

Our exploratory experiments show that MaxSAT and SMT solvers can be used to competitively compute schematic metro network layouts and are similarly versatile as ILP solvers; additional constraints like minimization or prevention of line bends in interchange stations [11] can easily be added to the models. Various avenues of further research present themselves. First, SMT supports a variety of theories. It is worth investigating if a change in theory (e.g., using rational linear arithmetic, rather than integer linear arithmetic) can speed up computation. Second, a big positive of using logical methods is not only that they produce high-quality layouts, but that usually such high-quality (but not necessarily optimal) intermediate solutions can be found quickly. It has to be verified if this property still holds for MaxSAT and SMT solvers. Third, runtimes of MaxSAT and SMT can be dependent on the specific model formulation, possibly even up to the order in which clauses are added. Best practices of MaxSAT and SMT model creation should be consulted to optimize the formulation of the particular constraints of the metro map layout problem to be as suitable as possible to their respective solvers. And fourth, we expected the runtimes for F_1 and F_2 to be equal, due to their equal relative weighting. We currently lack an explanation, why doubling the weight vector decreases runtime, outside of the previously mentioned variance in runtime.

REFERENCES

[1] C. Barrett, A. Stump, and C. Tinelli. The smt-lib standard: Version 2.0. In *Satisfiability Modulo Theories (SMT 2010)*, volume 13, page 14, 2010.

[2] H. Bast, P. Brosi, and S. Storandt. Metro maps on octilinear grid graphs. *Computer Graphics Forum (CGF)*, 39(3):357–367, 2020.

[3] H. Bast, P. Brosi, and S. Storandt. Metro maps on flexible base grids. In *Spatial and Temporal Databases (SSTD 2021)*, pages 12–22. ACM, 2021.

[4] N. Bjørner and A.-D. Phan. νZ-Maximal Satisfaction with Z3. *Symbolic Computation in Software Science (SCSS 2014)*, 30:1–9, 2014.

[5] N. Bjørner, A.-D. Phan, and L. Fleckenstein. νZ-an optimizing SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2015)*, volume 9035 of *LNCS*, pages 194–199. Springer, 2015.

[6] D. Chivers and P. Rodgers. Octilinear force-directed layout with mental map preservation for schematic diagrams. In *Theory and Application of Diagrams (DIAGRAMS 2014)*, volume 8578 of *LNCS*, pages 1–8. Springer, 2014.

[7] A. Griggio. A Practical Approach to Satisfiability Modulo Linear Integer Arithmetic. *Satisfiability, Boolean Modeling and Computation (JSAT)*, 8(1-2):1–27, 2012.

[8] B. Luteberget and C. Johansen. Drawing with SAT: four methods and A tool for producing railway infrastructure schematics. *Formal Aspects Comput.*, 33(6):829–854, 2021.

[9] S. Nickel and M. Nöllenburg. Towards Data-Driven Multilinear Metro Maps. In *Theory and Application of Diagrams (DIAGRAMS 2020)*, volume 12169 of *LNCS*, pages 153–161. Springer, 2020.

[10] M. Nöllenburg and A. Wolff. Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *IEEE Trans. Vis. Comput. Graph. (TVCG)*, 17(5):626–641, May 2011.

[11] M. J. Roberts. What? s your theory of effective schematic map design? 2014.

[12] J. Stott, P. Rodgers, J. C. Martinez-Ovando, and S. G. Walker. Automatic metro map layout using multicriteria optimization. *IEEE Trans. Vis. Comput. Graph. (TVCG)*, 17(1):101–114, 2010.

[13] J. M. Stott and P. Rodgers. Automatic metro map design techniques. In *International Cartographic Conference (ICC 2005)*, 2005.

[14] T. C. van Dijk and D. Lutz. Realtime linear cartograms and metro maps. In *Advances in Geographic Information Systems (SIGSPATIAL/GIS 2018)*, pages 488–491. ACM, 2018.

[15] H.-Y. Wu, B. Niedermann, S. Takahashi, M. J. Roberts, and M. Nöllenburg. A Survey on Transit Map Layout – from Design, Machine, and Human Perspectives. *Computer Graphics Forum (CGF)*, 39(3):619–646, 2020.

[16] V. Yoghourdjian, T. Dwyer, G. Gange, S. Kieffer, K. Klein, and K. Marriott. High-quality ultra-compact grid layout of grouped networks. *IEEE Trans. Vis. Comput. Graph. (TVCG)*, 22(1):339–348, 2015.