Disintegrating Automated Transit Map Design: An Algorithmic Core for Fast Layout Iteration

Thomas C. van Dijk
TU Eindhoven
Eindhoven, The Netherlands
0000-0001-6553-7317

Soeren Terziadis
TU Eindhoven
Eindhoven, The Netherlands
0000-0001-5161-3841

***Abstract*—Schematic transit maps are ubiquitous and a wide range of approaches for automated layout have been considered. Much of the literature focuses on mathematically defining an optimal drawing, but it is questionable to what extent the computational runtime of these approaches is justified by this "optimality". Instead we orient ourselves around another motivation for automated map design methods: efficiently exploring a wider range of designs than could be done by hand, as well as iterating on these designs. We present a framework around a simple linear program that computes layouts almost instantaneously given certain design decisions. In contrast to existing efficient approaches, we do guarantee optimality (within our more constrained setting) and support exact restrictions on connection directions such as octolinearity. We achieve this speedup by letting a human designer interactively make some of the decisions in a graphical user interface – decisions which would require computationally expensive binary variables to optimize automatically, and for which it would be hard to formalise exactly what "looks best."**

***Index Terms*—Visualization, Geometrical problems and computations, User Interaction and Presentation**

I. Introduction

In the operations research and optimization communities, the goal is often to get as much as possible of a real-world problem into a single mathematical model. That way, all components can be optimized in relation to each other. An airport operator might first decide which planes land at which gate and then, given those decisions, schedule the buses and drivers to shuttle the passengers. From an optimization perspective, it would be better to integrate these two problems: having to schedule buses and drivers may influence where we would want to park the planes [3]. Such an *integrated* optimization problem can give higher-quality solutions, but is often much harder to solve computationally.

In effect, we propose the opposite approach for schematic transit map design. Various highly advanced methods have been proposed that attempt to automate the design problem in an integrated way [12, 16]. Unfortunately the runtime of these methods make them impractical. (Waiting hours or even days might be fine if the resulting design is actually excellent and can be used as-is, but this is rarely the case.)

The 2nd author has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement Grant Agreement No 101034253.

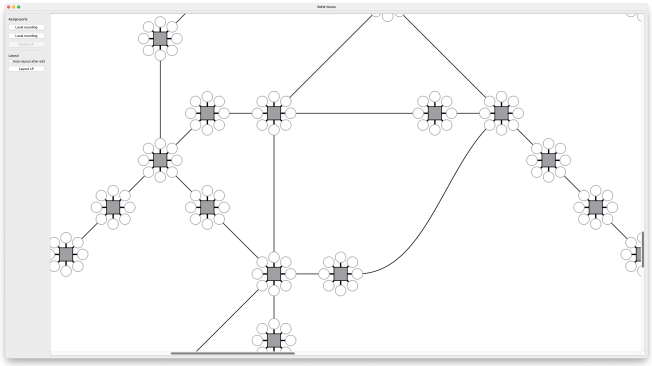


Fig. 1: The system shows inconsistent port assignments using Bézier curves. Here the curve goes from the east port on the left to the south-west port on the right. This allows the design to be in a "work in progress" state.

One response to this has been to still solve the integrated problem, but use heuristic methods. For example, Wang and Chi use nonlinear least squares adjustment [13], a modeling technique where solvers are not guaranteed to find an optimal solution; Bast, Brosi and Storandt use a greedy incremental approach [1] with no known guarantees on quality. Both approaches can be used as the algorithmic core of an interactive system that lets the user change and re-optimize the drawing [4, 6, 14]. However, these methods are hard to reason with, since their solutions are "whatever the algorithm happened to do this time" and are not guaranteed to relate in any particular way to the optimum of a reasonable mathematical model. This leads to a level of inscrutability that hampers usability and might lower trust in the system and adoption in practice.

In contrast, we propose decomposing (or: *dis-integrating*) the process of automated transit map design into a set of well defined subproblems that can be understood on their own terms, such as deciding the direction of individual edges, deciding where the station labels go, and placing the stations on the map based on these decisions. Clearly these steps influence each other – whether a set of edge directions is actually good depends on how well it lets us lay out the stations – and that is what makes integrated approaches slow. In our framework, each component on its own can be optimised efficiently in

practice and can be intuitively tweaked by hand, with changes being propagated through the system. What we potentially lose in the quality of the automated decisions, we gain in the predictability of the system and ease of trying alternatives.

Related Work

An influential work on mathematical optimisation for schematic transit map design is the mixed integer programming approach by Nöllenburg and Wolff [12]. They represent stations as points and connections as straight-line segments between them. The model minimizes the number of bends, the deviation of connections from their geographic orientation and the overall length of connections. This model can be extended with many additional restrictions (like user-defined fixed positions, orientations and lengths). Unfortunately their approach suffers from long running times: the Vienna subway network was solved optimally within about 30 seconds, but obtaining a solution for medium sized instances like Sydney Metro can require more than 20 minutes.

A recent approach by Bast, Brosi, and Storandt [1] uses an underlying octolinear grid on which stations are placed, and connections are routed. Their model can be solved optimally using integer linear programming (ILP), but this is again very slow and the authors present a heuristic approach that uses repeated shortest path computations to incrementally build a layout. While their heuristic approach can solve usual benchmark instances in seconds and can be extended to more general host graphs [2], it sometimes makes seemingly obvious "mistakes"; the lack of optimality is disappointing and working on a grid can be restrictive. Another line of work uses least-squares optimisation and provides realtime performance [5, 13, 14], but does not guarantee that connections strictly follow octolinear directions. See Wu, Niedermann, Takahashi, and Nöllenburg [15] for a comprehensive overview of different methods.

An important issue for the practical relevance of automated methods is whether practitioners will adopt them. Liu, Marriott, Dwyer, and Tack note that, in general, *trust* in a system plays a large role and that it can be increased by showing partial solutions while the algorithm is still working, and by letting the user interactively explore alternative solutions [9]. The former can be done with most metro map drawing algorithms (especially the slow ones) and has been demonstrated by various authors in informal contexts. However, Liu et al. show that displaying progress during the algorithm can lead users to over-trust the system; in contrast, letting users interact with the solution accurately calibrates their trust.[1] We do note that their study is about an operation research problem, not a design problem: since we do not have an exact mathematical formulation of what the most desirable map is, interactivity for exploration seems even more important in our case.

[1]We would not want to trick designers into using a system that leads to worse designs.

Contribution

We present a framework for designing schematic transit maps that consists of algorithmic components that can be interacted with in a graphical user interface. *Port assignment* decides, for each connection leaving a station, in which octolinear direction it leaves; this fixes all directions unless stations disagree, which we discuss later. *Realisation* positions the stations in the plane compactly while respecting the given port assignment. This is computationally efficient since the direction of all connections is fixed. The framework can be extended to handle station labels, since the orientation of the labels can be included in the "port assignment."

Our port assignment algorithm typically solves most or all of the map, but can get stuck on an *inconsistent* port assignment where the two stations at either end of a connection do not match up. These inconsistencies can either be resolved (Section II) or appropriately visualized (Section IV). Section V includes a brief evaluation of runtime and the the low frequency of inconsistent port assignments.

II. Port Assignment

A crucial decision in the design of a schematic transit map is the following: in which direction do the connections leave the station? Sticking to the typical octolinear style, we consider each station to have eight available *ports*, one for each of the octolinear directions (north, north-east, east, ...). *Port assignment* decides for all stations which ports are used by which connections, under some consistency constraints.

First, at most one connection can be assigned to a port, otherwise edges in the drawing would overlap; this is a hard constraint. Second, an edge leaving from a particular port should arrive at the $180°$ opposite port (west port connects to an east port). If this second constraint is violated (Fig. 2), we call the port assignment *inconsistent*. It follows straightforwardly from the literature that finding a consistent port assignment such that the network can be drawn without edge crossings is NP-hard [12], so that will not be our goal.

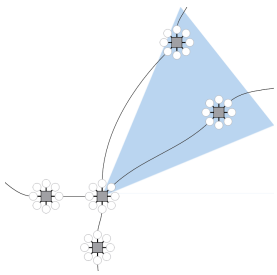


Fig. 2: An inconsistent port assignment

As optimisation objective, we would like the direction of the assigned ports to match the geographic direction between the connected stations. A naive solution would simply snap each connection to the closest port, but this can easily assign more than one edge to a port, which we consider unacceptable.

For a station s, connection e and port p, let $\alpha(s, e, p)$ be the difference in angle between the port and the geographic direction from s to the station at the other end of e. We set the *cost* of making this assignment to $\alpha(s, c, p)^2$. At each station independently, we compute a minimum cost matching between the connections and the ports based on these cost. Note that this way of assigning ports preserves topology around each station (that is, the circular order of edges leaving the station).

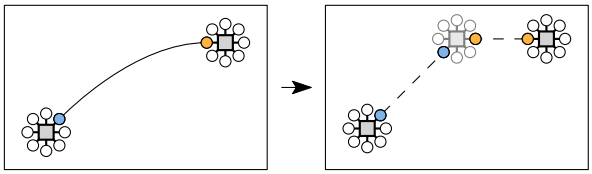


Fig. 3: The inconsistent port assignment between the orange and blue port can be realised with a dummy station.

Doing this at each station individually can lead to inconsistent port assignments (in Figure 2 two stations prefer the same port), but is very fast: a bipartite matching between eight ports and at most eight connections. We leave the problem of finding a (globally) consistent port assignment open, but note that the rest of the system can handle inconsistently assigned edges in various reasonable ways. For one we can create a consistent assignment by introducing a dummy station on the edge matching the port assignment in both directions (corresponding to a bend in the line). See Figure 3.

III. Realising station locations

Given a port assignment, we use linear programming to obtain station coordinates that minimise the total length of connections. In contrast to (mixed) integer programming, linear programming can be solved efficiently; indeed, our implementation realises even the London tube network within $20\,\mathrm{ms}$ on a commodity laptop from 2024.

We represent the position of a station s with two variables x_s and y_s. Let e be the connection between stations s and t. The linear program enforces that t lies on the ray from s that corresponds to the direction of the port that e is assigned to, and that their distance along that ray is at least 1. (This assumes the port assignment of e is consistent.) Then by additionally minimizing the distance of s and t along the line we minimize the length of all edges in the layout, acting to minimize the overall size of the map.

To illustrate the simplicity of the LP, we state it in full below. For this let $E_\mathrm{r}, E_\mathrm{tr}, E_\mathrm{t}, E_\mathrm{tl}$ be the sets of edges st, where t is to the right, to the top-right, above, or to the top left of s, respectively (according to the given port assignment). Since the distance between the origin and $(1,1)$ is $\sqrt{2}$, we use $\frac{x+y}{\sqrt{2}}$ and $\frac{x-y}{\sqrt{2}}$ to measure diagonal distances.[2] We state the constraints and objective function for E_r and E_tr; the other two cases are symmetric.

$$\left.\begin{aligned} y_s &= y_t \\ x_s &\le x_t - 1 \end{aligned}\right\} \qquad \forall st \in E_\mathrm{r} \qquad (1)$$

$$\left.\begin{aligned} \frac{x_s - y_s}{\sqrt{2}} &= \frac{x_t - y_t}{\sqrt{2}} \\ \frac{x_s + y_s}{\sqrt{2}} &\le \frac{x_t + y_t}{\sqrt{2}} - 1 \end{aligned}\right\} \qquad \forall st \in E_\mathrm{tr} \qquad (2)$$

[2]This leads to algebraic coefficients in the LP model, but there is no technical problem with using a machine-precision approximation of $\sqrt{2}$.

To minimize the total length of all connection in the layout, we set the objective $\min \sum_{st \in E} d(s,t)$ where

$$d(s,t) = x_t - x_s \qquad \text{if } st \in E_\mathrm{r} \qquad (3)$$

$$d(s,t) = \frac{x_s + y_s - x_t - y_t}{\sqrt{2}} \qquad \text{if } st \in E_\mathrm{tr} \qquad (4)$$

Note that no absolute value has to be considered since the order of s and t is known because of the port assignment.

These constraints and objective function are a subset of the ILP formulation proposed by Nöllenburg and Wolff [12]. In particular our LP does not decide the direction of edges and becomes significantly simpler since the edge directions are known at the time the model is constructed.

Moreover we also omit any planarity constraints, which would ensure edges do not intersect; all known formulations require binary variables. It seems like a significant drawback that the model can make realisations with intersecting edges. However, in our experience this rarely happens on real transit networks with a reasonable port assignment. In the spirit of our framework, we accept such "invalid" realisations and invite the user to solve this problem – either by changing the port assignment, or by adding some other constraint to the linear program that will resolve the crossing. We are currently investigating how to do the latter in a way that is intuitive to the user and can still be optimised efficiently.

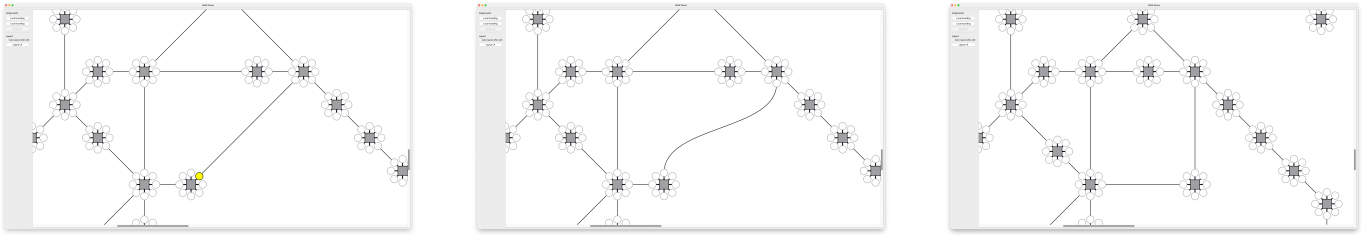
IV. Interaction Loop

Interaction in this system can be categorized with the problem-solving loop, a theoretical framework introduced by Liu, Dwyer, Marriott, Millar, and Haworth, in which users continuously interact with solutions produced by fast solvers [8]. By manually adding or changing constraints and re-optimising, the problem-solving loop enables expert users to include their domain specific knowledge. This may be necessary for achieving good solutions, since this knowledge and expertise may have been abstracted away by mathematical models and system implementations.

An interaction-loop system specifically for schematic map design has the potential to be much more useful than generic graphic design programs, because the latter are not aware of the semantics of a transit map (stations, lines, labels) and do not let the user interact with the map on the level that the designer thinks of it – rather, working on the level of graphical objects such as circles, line segments and text boxes.

Our prototype GUI represents the current state of the network by drawing each station as a box with 8 handles (one per port); connections between stations are visualised as curves starting and ending at the handle corresponding to the port assignment of the connection at that station. If the port assignment on the two endpoints of a connection is not consistent, or the stations are not properly aligned, this curve is a cubic Bézier spline, otherwise a straight-line segment.

If all connections are by straight lines, we call the map *valid*; otherwise we call it invalid. See Figures 1 and 4b, where the Bézier curves draw attention.



(a) The user selects a port handle that is currently in use; then clicks the north port, re-assigning the edge. On the opposite end, the corresponding port is re-assigned automatically (as it was available).

(b) Instant feedback in terms of an "invalid" drawing; no stations have moved. More port edits can be made while in this state, undisturbed by layout changes.

(c) A re-optimised layout, computed almost instantaneously by LP. This prototype does not have "stability" measures to prevent unnecessary changes to the rest of the realisation.

Fig. 4: Interacting with the port assignment; steps from left to right.

The network representation initially uses the automatically computed port assignment. The user can manually switch the port assignment of individual edges: select the port handle the edge is connected to and then select a free port handle to assign it to. By default, this is done in *symmetric* mode, in which the port assignment on the other endpoint is simultaneously changed to keep the port assignment consistent. See Figure 4.

This interaction can naturally be extended to change multiple port assignments at once, which is showcased by a *straightening* mode, in which the user can change the port assignment of a sequence of degree-2 stations all at once.

The second component of the system (the LP solver described in Section III) can be called via a sidebar button. Alternatively the user can tick a checkbox to enable *automatic* mode in which after any manual interaction (like a reassignment of a port), the LP solver is immediately called to obtain a new optimized layout according to this new assignment.

Liu et al. formulated a set of recommendations for problem-solving loop systems. The interaction within our system aligns with a number of these recommendations, such as (i) it [PROVIDES APPROPRIATE VISUAL REPRESENTATIONS OF SOLUTIONS AND CONSTRAINTS] by displaying both the current port assignments and the network layout resulting from the LP-solver, (ii) [SUPPORTS USER MODIFICATION OF THE OPTIMIZATION MODEL] via reassignment of the port assignment and (iii) [ALLOWS USER CONTROLLED RE-OPTIMIZATION] through repeated LP solving. Other recommendations like [ALLOWING DIRECT MANIPULATION OF SOLUTIONS] and [PROVIDING A GALLERY OF SOLUTIONS] can easily be integrated into the system, while others – [SUPPORT FOR COMPARISON OF SOLUTIONS] and [GENERATION OF DIVERSE SOLUTIONS] – require further work. One recommendation in particular, [PROVIDING FEEDBACK ON THE SOLVING PROCESS] is less relevant in our system since solutions are effectively obtained instantaneously.

Another important concept in interactive systems is stability between solutions. Stability has been considered in a variety of visualization settings, e.g. for time-varying data [10], moving objects [17]. In particular Wang and Peng [14] reduce the disorienting effect of re-optimization by "stiffening" the network further away from the edit. This stiffness could be introduced into our realisation component by adding a term to the objective function minimizing the L_1-distance of any station to its previous position (weighted by its network distance to the edit). Alternatively, one could restrict station movement, scaled as a function of distance to the edit. However, such measures could reduce the interpretability of what the system is optimising and thus hurt user trust.

V. Evaluation

We have a prototype written in Python, using `scipy` to compute matchings and `ortools` with `GLOP` to solve linear programs. Even on London, the port assignment is solved in a millisecond and the LP is solved in approximately $20\,\mathrm{ms}$. (Note that we do not suppress degree 2 stations.)

Due to the locality of our port assignment, inconsistent assignments are possible. As a preliminary evaluation we count the number of inconsistent assignments in standard benchmark instances. They occur in manageable numbers in small instances like Montreal (none), Vienna, Washington and Taipei (each with 1 inconsistency = 1% of assignments) and medium instances like Sydney (2 = 1%), Karlsruhe (4 = 3%) and Lisbon (7 = 9%). Even for large instances like London (12 = 4%) and Tokyo (12 = 5%), it is feasible for the user to inspect the inconsistencies by hand; in our experiments only the highly dense and interconnected Moscow network (23 = 9%) exceeded 20 inconsistent assignments.

VI. Conclusion

Our model is simple and easy to implement and leverages interactivity to speed up layout iteration. While our prototype only supports limited interaction methods (e.g., line straightening), further interactions like manual relocation of stations can easily be included. Extensions of the ILP method [12] also immediately apply to our system, like computing k-linear layouts [11].

We do not support station labels at time of writing, but they could be added to the system by assigning them to a port. Finally the literature has more recommendations for problem-solving loop systems, which would be interesting to consider, in particular maintenance and comparison of multiple solutions.

References

[1] H. Bast, P. Brosi, and S. Storandt. "Metro maps on octilinear grid graphs". In: *Computer Graphics Forum*. Vol. 39. Wiley Online Library. 2020, pp. 357–367.

[2] H. Bast, P. Brosi, and S. Storandt. "Metro maps on flexible base grids". In: *Proceedings of the 17th International Symposium on Spatial and Temporal Databases*. 2021, pp. 12–22.

[3] G. Diepen, J. M. van den Akker, and J. A. Hoogeveen. "Integrated gate and bus assignment at Amsterdam Airport Schiphol". In: *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems* (2009), pp. 338–353.

[4] T. C. van Dijk and T. Janiak. "Algorithmically-assisted interactive metro map design". In: *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*. 2022, pp. 1–4.

[5] "Realtime linear cartograms and metro maps". In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2018, pp. 488–491.

[6] T. Janiak. "Interactive Design of Metro Maps". MA thesis. Universität Würzburg, 2021.

[7] J. Liu, T. Dwyer, K. Marriott, J. Millar, and A Haworth. "Understanding the relationship between interactive optimisation and visual analytics in the context of prostate brachytherapy". In: *IEEE transactions on visualization and computer graphics* 24.1 (2017), pp. 319–329.

[8] J. Liu, T. Dwyer, G. Tack, S. Gratzl, and K. Marriott. "Supporting the problem-solving loop: Designing highly interactive optimisation systems". In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2020), pp. 1764–1774.

[9] J. Liu, K. Marriott, T. Dwyer, and G. Tack. "Increasing user trust in optimisation through feedback and interaction". In: *ACM Transactions on Computer-Human Interaction* 29.5 (2023), pp. 1–34.

[10] S. Nickel, M. Sondag, W. Meulemans, S. Kobourov, J. Peltonen, and M. Nöllenburg. "Multicriteria optimization for dynamic Demers cartograms". In: *IEEE Transactions on Visualization and Computer Graphics* 28.6 (2022), pp. 2376–2387.

[11] M. Nöllenburg and S. Terziadis. "Computing Data-driven Multilinear Metro Maps". In: *The Cartographic Journal* 60.4 (2023), pp. 367–382. DOI: 10.1080/00087041.2024.2304476.

[12] M. Nöllenburg and A. Wolff. "A mixed-integer program for drawing high-quality metro maps". In: *International Symposium on Graph Drawing*. Springer. 2005, pp. 321–333.

[13] Y.-S. Wang and M.-T. Chi. "Focus+Context metro maps". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2528–2535.

[14] Y.-S. Wang and W.-Y. Peng. "Interactive metro map editing". In: *IEEE Transactions on Visualization and Computer Graphics* 22.2 (2015), pp. 1115–1126.

[15] H.-Y. Wu, B. Niedermann, S. Takahashi, and M. Nöllenburg. "A Survey on Computing Schematic Network Maps: The Challenge to Interactivity". In: *Proceedings of the 2nd Schematic Mapping Workshop*. 2019.

[16] 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". In: *Computer Graphics Forum* 39.3 (2020), pp. 619–646. DOI: https://doi.org/10.1111/cgf.14030. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14030. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14030.

[17] J. Wulms, J. Buchmüller, W. Meulemans, K. Verbeek, and B. Speckmann. "Stable visual summaries for trajectory collections". In: *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*. IEEE. 2021, pp. 61–70.