Realtime Linear Cartograms and Metro Maps

Thomas C. van Dijk
Lehrstuhl für Informatik I
Würzburg University
thomas.van.dijk@uni-wuerzburg.de

Dieter Lutz
Lehrstuhl für Informatik I
Würzburg University
dieter.lutz@gmail.com

ABSTRACT

We introduce an efficient algorithm for drawing spatially-informative linear cartograms: transforming a geometric network such that given edge lengths are realised, while distorting edge directions as little as possible. Our algorithm is based on carefully linearised least squares optimisation, forgoing the need for an iterative solver. This is fast and ensures a well-defined result. The classic application of linear cartograms is drawing travel-time maps; we also discuss drawing schematised metro maps.

CCS CONCEPTS

• **Information systems → Geographic information systems**; • **Human-centered computing → Graph drawings**; • **Theory of computation → Continuous optimization**;

KEYWORDS

least squares optimization, metro maps, cartograms

ACM Reference Format:
Thomas C. van Dijk and Dieter Lutz. 2018. Realtime Linear Cartograms and Metro Maps. In *26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18), November 6–9, 2018, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3274895.3274959

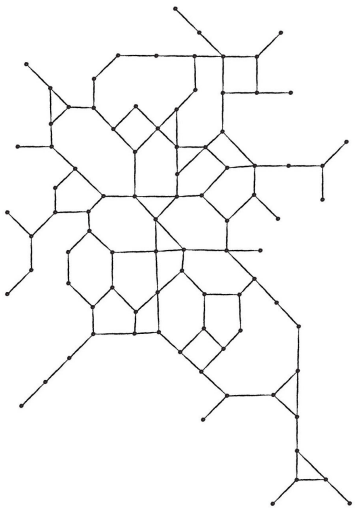


Figure 1: A metro map drawn by our method in a few milliseconds. The painterly rendering hides the fact that the edges are not exactly octilinear.

1 INTRODUCTION

A linear cartogram (or: edge-value cartogram) is a drawing of a network such that every edge is drawn with a prescribed length. In general this is not possible (consider for example a triangle with two short edges and one edge that is much longer), so a compromise between the edges is required.

One particular application of linear cartograms that is relevant to geographic information systems is the so-called travel-time map, in which a geographic network is deformed so that Euclidean distance can be used to read off (approximate) travel time. This goes back to Clark's classic 1977 paper on the topic [2]. Such visualisations can be used as a visual aid for planning public transport journeys or as a visualisation of traffic jams. In this paper we present an algorithm that is fast enough to run interactively on smartphones. We also present a novel application of linear cartograms: drawing metro maps, where stops are conventionally drawn with uniform spacing.

In geographic applications, we are not drawing abstract graphs (even metro maps have a geographic component). Therefore, not only the length of edges is relevant: the original map should be recognisable in order for the cartogram to be useful. This is true even for metro maps: though they are a heavily stylised and visualise the topology rather than the geography of a network, they ideally do not contradict the reader's cognitive map. We say a *spatially-informative linear cartogram* should change the direction of edges as little as possible.

In this paper, we first give an algorithm for spatially-informative linear cartograms (Section 3). On real data, it has runtime in the order of tens of milliseconds. It is based on least-squares optimisation, a well-known mathematical framework for satisfying a set of *soft* constraints. Unfortunately, the natural model has nonlinear constraints. Using standard nonlinear solvers has several drawbacks, so we develop a bespoke linearisation of our problem. Based on this, we cannot only *preserve* edge direction, but can also (approximately) realise arbitrary new edge directions. We show how this can be used to draw metro maps (Section 5).

2 RELATED WORK

The concept of a travel-time map dates back to Clark [2], who draws a single-source time cartogram by starting at a source vertex and building the network outward in a breath-first-search manner. The abstract problem of drawing a set of points such that a given set

SIGSPATIAL '18, November 6–9, 2018, Seattle, WA, USA

ACM ISBN 978-1-4503-5889-7/18/11.
https://doi.org/10.1145/3274895.3274959

of point pairs has given distances is known as *graph realisation*. Saxe has shown that this problem is *NP*-hard in many settings and variants [12]. Still, the problem is well studied from an application perspective, among other reasons for of its application to sensor network localisation [3]. In the field of graph drawing, there is a relation to *multidimensional scaling* (see Gansner et al. [4] and Kaiser et al. [6]). Such approaches are computationally somewhat expensive: they typically require an iterative solver, which we are able to avoid.

As for metro map drawing, we point to Wang and Chi [16], whose algorithm is based on nonlinear least-squares optimisation. Their algorithm uses a course-to-fine approach, whereby they first find a solution to a much simplified version of the network and then later reintroduce the omitted vertices. This helps with convergence and runtime – each of which is problematic in their approach. In contrast, our algorithm is easily fast enough to be run on an entire network at once, and has a well-defined unique optimum. Further metro map drawing algorithms include those of Stott et al. [14] and Nöllenburg and Wolff [10]. The former is based on hill climbing and as such has problems with local minima and is not fast. The latter is an exact algorithm based on mixed-integer linear programming and has runtime approaching an hour on real metro networks that our algorithm draws in the order of milliseconds.

In the literature on linear cartograms, the issue of edge intersections has been largely ignored. Our algorithm can ensure that the a crossing-free input drawing remains crossing-free after transformation by using the *event constraints* of Van Dijk and Haunert [15].

Least-squares optimisation. Here we briefly review the basic framework of least-squares optimisation. We are given a set of variables and a generally overconstrained system of equations. If all constraints are linear, we can represent them using a matrix equation $Ax = b$, where x is the vector of unknowns. Then the least-squares solution x minimises the L_2-norm of $Ax - b$. It is well known that this is achieved by solving $A^TAx = A^Tb$. If the constraints are (over)determined, this has a unique solution that can be readily found using standard linear-algebra software. This is particularly efficient if A and A^TA are sparse [7].

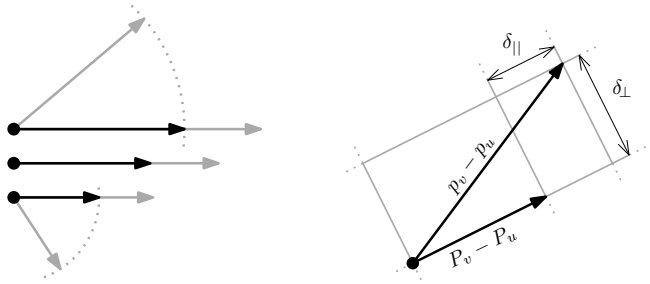
If the system of constraints is not linear, a common approach is to use a sequence of linear approximations in a hill-climbing-like approach. Such algorithms are computationally more expensive, since they have to potentially solve many sub-instances and have to derive these linear approximations. They can also have problems with convergence and can be sensitive to the required *initial solution* used [1], potentially finding a bad local optimum even if the algorithm converges.

3 LINEAR CARTOGRAMS

First we define the objective function that we would actually *like* to optimise. This objective function is nonlinear, so we develop a bespoke linear approximation that performs better than the standard approach based on Jacobian matrices.

3.1 Preliminaries

By convention, we use uppercase for constants and lowercase for variables. Let $G = (V, E)$ be a connected graph, $P_v = (X_v, Y_v)$ for $v \in V$ be the positions in relation to which we want to be spatially informative, and L_e for $e \in E$ be the desired length in the drawing.



(a) Each grey edge is considered equally bad, compared to the corresponding black edge.

(b) Parallel and perpendicular.

Figure 2: The geometry of length, direction, parallel and perpendicular discrepancies.

As variables of our optimisation problem, we take the coordinates of the vertices: x_v and y_v for $v \in V$. For notational convenience, let $p_v = (x_v, y_v)$. Then, in a perfect drawing, directions do not change and we realise the requested lengths, so we have for every edge $e = \{u, v\}$ that

$$p_u + D_e = p_v, \qquad \text{where } D_e = L_e \cdot \frac{P_v - P_u}{\|P_v - P_u\|}. \tag{1}$$

Note that this equation is linear, since D_e is a constant. These constraints are all invariant under translation, so we we arbitrarily fix the position of a single vertex to determine the system.

Our optimisation objective is to find a drawing with only small error in length and direction. Consider (soft) constraints that set $\text{dist}(p_u, p_v) = L_{\{u,v\}}$ and set the direction of $(p_v - p_v)$ equal to the direction of $(P_v - P_u)$ for every edge $\{u, v\} \in E$. Call these the length constraints and the direction constraints respectively. Note that they are not linear in x_v and y_v.

For $e = \{u, v\} \in E$, let $\Delta r_e = \text{dist}(u, v) - L_e$ be the length error (Δr for radius). As we will discuss later, we should not just care about the *absolute* length error. Similarly, let $\Delta\alpha_e$ be the discrepancy of e's direction constraint ($\Delta\alpha$ for angle). We set the relative weight of these soft constraints by factors depending on the requested length L_e. Call these weight functions $W_r(L_e)$ and $W_\alpha(L_e)$ respectively – values to be determined later. Then a least-squares solution to this set of (nonlinear) constraints minimises:

$$\sum_{e \in E} f(\Delta\alpha_e, \Delta r_e, L_e), \text{ where } f(\alpha, r, L) = W_\alpha(L) \cdot \alpha^2 + W_r(L) \cdot r^2. \tag{2}$$

Since we want the weight of length discrepancy to be higher for short edges, we set $W_r(L) = 1/L$. This leaves the choice of weight W_α for the direction discrepancy. Here we decide that a direction error of $\pi/4$ radians is equally as bad as being too long or too short by 50%. Setting this equality and using $W_r(L) = \frac{1}{L}$, we get

$$W_\alpha(e) \cdot \left(\frac{\pi}{4}\right)^2 = \frac{1}{L_e} \cdot \left(\frac{L_e}{2}\right)^2 \quad \Longrightarrow \quad W_\alpha(e) = \frac{4L_e}{\pi^2}. \tag{3}$$

This choice of the weights is illustrated in Figure 2(a).

3.2 Linear Approximation

As noted, we cannot minimise Equation (2) using *linear* least squares technology. Now reconsider Equation (1) for a specific edge – it actually represents two constraints: one for the x coordinate and one for y. There is no directionality to this minimisation.

We can instead express the constraints for a particular edge in any other basis, if the transformation is linear. Since we would like to differently penalise length error and direction error, we rotate the coordinate system such that the edge is aligned with one of the axes: then there is a 'parallel' and a 'perpendicular' discrepancy (rather than a global x and y). For small discrepancies, this is similar to length and direction respectively. We want these to be zero: then Equation (1) would hold.

$$\delta_{||} = (p_v - p_u) \cdot \frac{P_v - P_u}{\|P_v - P_u\|} - L_e \stackrel{!}{=} 0 \tag{4}$$

$$\delta_{\perp} = (p_v - p_u) \cdot \frac{(P_v - P_u)^{\perp}}{\|P_v - P_u\|} \stackrel{!}{=} 0 \tag{5}$$

Note that the dot product is with a constant vector and therefore linear. See Figure 2(b) for the geometry involved.

Now that we have separate constraints for the parallel and perpendicular error, we can weigh them differently: call these weights $W_{||}(L)$ and $W_{\perp}(L)$. Then the linear soft constraints will minimise:

$$\sum_{e \in E} f_{\text{lin}}(\Delta\alpha_e, \Delta r_e, L_e), \qquad \text{where} \tag{6}$$

$$f_{\text{lin}}(\alpha, r, L) = W_{||}(L) \cdot (L - \cos(\alpha)(L + r))^2 \tag{7}$$

$$+ W_{\perp}(L) \cdot (\sin(\alpha)(L + r))^2 \tag{8}$$

Note that the matrix that encodes these constraints is sparse, since every constraint involves only a constant number of variables.

3.3 Parameter Values

This leaves the choice of $W_{||}$ and $W_{\perp}$. In order to analyse the difference between our actual nonlinear objective (f) and the linear objective we can efficiently optimise (f_{lin}), let $f_{\text{diff}} = f - f_{\text{lin}}$. Clearly, we want f_{diff} to be small for all reasonable values of α and r, then we optimise what we *want* to optimise. We consider $\alpha \in [-\frac{\pi}{4}; \frac{\pi}{4}]$ and $r \in [-\frac{L}{2}; \frac{L}{2}]$ reasonable and we rarely observe values outside this range in practice. Therefore, we pick $W_{||}$ and $W_{\perp}$ to minimise

$$\int_{-L/2}^{L/2} \int_{-\frac{\pi}{4}}^{\frac{\pi}{4}} f_{\text{diff}}(\alpha, r, L)^2 \, \mathrm{d}\alpha \, \mathrm{d}r. \tag{9}$$

Tedious but standard calculation gives optimal parameter values of $W_{||}(e) \approx 1.0039/L_e$ and $W_{\perp}(e) \approx 0.41305/L_e$.

4 EXPERIMENTAL EVALUATION

We have implemented the described algorithm in C++, using the *Eigen* [5] library for sparse linear algebra. All experiments have been run on a desktop PC with an AMD Phenom™ II X6 1090T CPU at 3.20GHz. Memory usage is not an issue in these experiments, since we use Eigen's sparse representation of the matrices.

This short paper has no room for a proper statistical evaluation of our results; see [8] for more detail. Here we briefly sketch the results of several experiments on the OpenStreetMap street network of Würzburg. In the one experiment we pick a random point of the road network and ask for nearby edges to be lengthened by a factor of 2, representing the travel-time effect of a localised traffic accident. In another, we select a random subset of strokes in the road network and lengthen those in the same way, representing congested traffic. We see that the algorithm is able to effectively realise the requested lengths, with an average relative length error between 1% and 2%.

Table 1: Average runtimes over 100 runs with random edge lengths, with and without intersection constraints

Map	$\|V\|$	$\|E\|$	Runtime (ms)	Runtime* (ms)
Montreal	65	66	2	2
Karlsruhe	126	132	6	4
Sydney	174	183	7	6
London	308	361	24	11

We have also computed focus-and-context metro maps in the style of Wang and Chi [16], where every edge should have length 1, except for a single "focus" metro line with edges of length 2. We see that this task is somewhat harder than the previous one and that the algorithm has a particularly hard time with London, the most complicated metro network tested. Even then, the average length error remains below 15%.

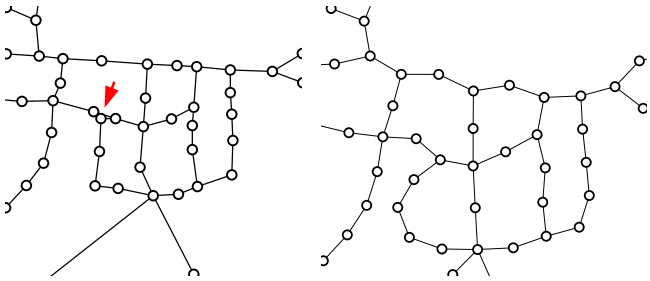
Staying with metro maps, we measure the runtime. Table 1 lists the maps, the number of vertices and edges, the runtime with intersection prevention (using the *event constraints* from [15]) and without (Runtime*) . We see that the runtimes without intersection prevention mostly just depend on the number of edges of the map. The runtime with intersection prevention also depends on the map itself and the likelihood that intersections occur. For example, the runtime for London increases significantly when we prevent intersections, since the London network is dense and complicated.

Comparing our runtime on the Sydney map with Wang and Chi [16], we observe a large improvement from 816 ms to 24 ms on comparable hardware. It should be noted that their algorithm does more that ours, but we cannot tell how much time they spend on each of the steps. The runtime reported by Inoue and Shimizu [13] for computing a time-distance cartogram of a map of Japan with around 100 vertices and edges is about 80 ms, without intersection prevention. Our algorithm takes approximately 4 ms on such a map.

5 METRO MAPS

While "metro map" is not a uniquely defined drawing style, it calls to mind specifically two aspects: octilinearity (or at least: restrictions on the use of angles) and approximately uniform edge length. For an extensive discussion of the design aspects of schematic mapping, see for example the work of Roberts [11].

It is typically part of the design of a metro map that edges have approximately uniform length. This is, for example, *design rule (R5)* of Nöllenburg and Wolff [10]. Wang and Chi [16] use a small set of different edge lengths to create focus on a specific route through the network. Another approach is to use "multi-scale" focus map algorithms to scale parts of the map (see for example Merrick and Gudmundsson [9]), which can be used to enlarge the city centre and compress the suburbs – which is often necessary for a good metro map. A unit-length linear cartogram is a more direct way to achieve



(a) Geographic input network (b) Unit-length output network with improved angular resolution

Figure 3: Crop of the city centre of the Karlsruhe public transport system. The angular resolution has improved and this benefits readability, especially at the indicated spot.

this: it automatically does what needs to be done (using more space where the network is dense) instead of using a roundabout step with a focus map.

Rather than keep the original geographic directions of the edges, we can also optimise the directions to provide a readable map. Figure 3 shows an example of how we can improve the angular resolution and greatly improve the readability of the network while still representing the geography of the network reasonably well. In this specific example, the edge directions were picked to be uniformly spaced ("perfect angular resolution") and rotated to minimise the squared discrepancy between input and output direction.

The above technique already yields an interesting schematised visualisation style, but the most iconic aspect of metro maps is octilinearity. Indeed, Nöllenburg and Wolff [10] have this as their very first design rule *(R1)*. Unfortunately – as they show – this makes many metro map drawing problems NP-hard, since it introduces combinatorics. Their exact algorithm, based on mixed-integer linear programming, takes about 23 minutes of runtime on the transport network of Sydney. In comparison, our runtime on this instance is 6 milliseconds (Table 1). Of course, crucially, we do not actually solve the same problem. However, we argue that – while not *quite* octilinear – our visualisations are nonetheless compelling. Consider Figure 1, where we have computed the vertex positions using the method described below, and then applied an off-the-shelf painterly render style – this drawing is not obviously "wrong."

In an octilinear drawing every vertex has eight available directions (called ports) in 45 degree increments. By assigning each edge to a port and letting our algorithm realise that port's direction we can approximate octilinearity in our drawings. We pick an assignment of edge directions at every vertex, assigning the edges to distinct ports such that the sum of squared discrepancies to the original directions is minimised. This is a discrete least squares problem and we solve it as preprocessing for our main least-squares optimisation. Being a standard assignment problem, it can be readily solved using algorithms for bipartite matching, flow, or linear programming. This take no appreciable time since the instances are independent and small: each assigns up to eight edges to a total of eight ports.

6 CONCLUSION

Using an implementation of our proposed algorithm, we have shown that it is both efficient and effective. With a runtime in the order of milliseconds, it is ideally suited for interactive applications, even on mobile devices. We have furthermore shown that our algorithm is able to realise the requested lengths well in practice.

Drawing metro maps using unit-length linear cartograms merits further research, at least as an efficient preprocessing step that automatically adjusts local scale to account for busy city centres and sparse suburbs and improves angular resolution – these are desirable properties of metro maps. The idea to use painterly rendering techniques to hide underlying geometric imprecision might also find more general map drawing application in other contexts.

ACKNOWLEDGMENTS

Thomas C. van Dijk is supported by the German Research Foundation (DFG), grant number Di 2161/2-1 in the context of the priority programme *Volunteered Geographic Information*.

REFERENCES

[1] I. Borg and P. Groenen. 1997. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag, Chapter 11.4.
[2] J. W. Clark. 1977. Time-Distance Transformations of Transportation Networks. *Geographical Analysis* 9, 2 (1977), 195–205. https://doi.org/10.1111/j.1538-4632.1977.tb00573.x
[3] Y. Ding, N. Krislock, J. Qian, and H. Wolkowicz. 2010. Sensor Network Localization, Euclidean Distance Matrix completions, and graph realization. *Optimization and Engineering* 11, 1 (2010), 45–66. https://doi.org/10.1007/s11081-008-9072-0
[4] E. R. Gansner, Y. Koren, and S. North. 2005. Graph Drawing by Stress Majorization. In *Graph Drawing*, JÃąnos Pach (Ed.). Lecture Notes in Computer Science, Vol. 3383. Springer Berlin Heidelberg, 239–250. https://doi.org/10.1007/978-3-540-31843-9_25
[5] G. Guennebaud, B. Jacob, et al. 2010. *Eigen v3*. `http://eigen.tuxfamily.org`.
[6] C. Kaiser, F. Walsh, C. J. Q. Farmer, and A. Pozdnoukhov. 2010. User-Centric Time-Distance Representation of Road Networks. In *Geographic Information Science*, S. I. Fabrikant, T. Reichenbacher, M. van Kreveld, and C. Schlieder (Eds.). Lecture Notes in Computer Science, Vol. 6292. Springer Berlin Heidelberg, 85–99. https://doi.org/10.1007/978-3-642-15300-6_7
[7] K. Kraus. 2004. *Photogrammetry – Geometry from Images and Laser Scans*. de Gruyter, Berlin, Germany.
[8] D. Lutz. 2014. Realtime Linear Cartograms using Least-Squares Optimisation (M.Sc. thesis).
[9] D. Merrick and J. Gudmundsson. 2006. Increasing the Readability of Graph Drawings with Centrality-based Scaling. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60 (APVis '06)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 67–76. http://dl.acm.org/citation.cfm?id=1151903.1151914
[10] M. Nollenburg and A. Wolff. 2011. Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *IEEE Transactions on Visualization and Computer Graphics* 17, 5 (May 2011), 626–641. https://doi.org/10.1109/TVCG.2010.81
[11] Maxwell J Roberts. 2014. What's your theory of effective schematic map design? *Schematic Mapping Workshop '14* (2014).
[12] J. B. Saxe. 1979. Embeddability of weighted graphs in k-space is strongly NP-hard. *Proceedings of the 17th Allerton Conference in Communications, Control and Computing* (1979), 480–489.
[13] E. Shimizu and R. Inoue. 2009. A new algorithm for distance cartogram construction. *International Journal of Geographical Information Science* 23, 11 (2009), 1453–1470. https://doi.org/10.1080/13658810802186882 arXiv:http://dx.doi.org/10.1080/13658810802186882
[14] Jonathan Stott, Peter Rodgers, Juan Carlos Martinez-Ovando, and Stephen G Walker. 2011. Automatic metro map layout using multicriteria optimization. *IEEE Transactions on Visualization and Computer Graphics* 17, 1 (2011), 101–114.
[15] T. C. van Dijk and J.-H. Haunert. 2014. Interactive focus maps using least-squares optimization. *International Journal of Geographical Information Science* (2014). https://doi.org/10.1080/13658816.2014.887718 arXiv:http://dx.doi.org/10.1080/13658816.2014.887718
[16] Y.-S. Wang and M.-T. Chi. 2011. Focus+Context Metro Maps. *Visualization and Computer Graphics, IEEE Transactions on* 17, 12 (Dec 2011), 2528–2535. https://doi.org/10.1109/TVCG.2011.205