Travel-Time Maps: Linear Cartograms with Fixed Vertex Locations*

Kevin Buchin[1], Arthur van Goethem[1], Michael Hoffmann[2], Marc van Kreveld[3], and Bettina Speckmann[1]

[1] Technical University Eindhoven, Eindhoven, The Netherlands
[2] ETH Zürich, Zürich, Switzerland
[3] Utrecht University, Utrecht, The Netherlands

Abstract. Linear cartograms visualize travel times between locations, usually by deforming the underlying map such that Euclidean distance corresponds to travel time. We introduce an alternative model, where the map and the locations remain fixed, but edges are drawn as sinusoid curves. Now the travel time over a road corresponds to the length of the curve. Of course the curves might intersect if not placed carefully. We study the corresponding algorithmic problem and show that suitable placements can be computed efficiently. However, the problem of placing as many curves as possible in an ideal, centered position is NP-hard. We introduce three heuristics to optimize the number of centered curves and show how to create animated visualizations.

1 Introduction

Most people depend on maps for navigation. Regular maps, however, do not ensure that time and distance correlate equally across the map. A village just on the other side of a mountain range might be hours away, whereas a city miles away is only five minutes driving. Temporal conditions, such as traffic jams or roads blocks, can make these effects even more pronounced.

To counteract this, visual cues are used to display (relative) travel times, commonly using colors (see Fig. 1 (a)). Size, however, is a better means to visualize numerical attributes [1]. As an alternative to color, length could also be used to show travel time on stretches of road. As a visual variable, length has a better association to quantity than color. "Relative lengths" of stretches are immediately quantified, whereas "relative colors" do not have a clear interpretation. These observations have led to the development of *linear cartograms* [2–5].

Linear cartograms try to display time more clearly by distorting the base map. Two types of linear cartograms exist: centered and non-centered. The former has a "center" location and only distances to this location correspond to actual travel time. The latter type attempts to have all pairs of locations at travel-time-proportional distances, which

* K. Buchin, A. van Goethem, and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.207 (KB), no. 612.001.102 (AvG), and no. 639.023.208 (BS). M. Hoffmann is partially supported by the ESF EUROCORES programme EuroGIGA, CRP GraDR and SNF Project 20GG21-134306.

M. Duckham et al. (Eds.): GIScience 2014, LNCS 8728, pp. 18–33, 2014.

Fig. 1. Expected travel times in the Netherlands during morning rush hour. (a) Color coding. (b) Linear cartogram. (c) Linear cartogram with fixed vertex positions.

is generally not possible without error. By distorting the map, linear cartograms give a more intuitive sense as to what is nearby, in time, on the map. The distortion of the map, however, can make recognition and usage of the cartogram harder [6]. Items on the map might be far removed from their position on the base map, making it hard to find specific items (see Fig. 1 (b), which was computed with the method described in [7]).

We introduce an alternative model that is well suited to visualize travel times on road networks. Instead of distorting the base map, we keep the locations fixed and "distort" only the edges. We do so by using sinusoid curves (see Fig. 1 (c)). Our approach has the advantage that the base map remains undistorted, while length can still be used to quantify and visualize travel time. The resulting maps create a dramatic effect and can also be used in animations, where an increased travel time (delay) is shown by an increased amplitude or frequency. Of course the curves might intersect if not placed carefully. We study the algorithmic problem of generating crossing-free linear cartograms according to our new model.

Related Work. In addition to the results on linear cartograms [2–5], several other papers are also related. Weights of edges (but for very different applications) can also be visualized by the width [8]. When drawing planar graphs with fat edges, the occupation of space by these edges is the main concern. Drawing edges with curves (e.g., [9]) has received considerable attention. Lately, more specifically, the use of circular arcs for edges has received attention (e.g., [10]), in particular the creation of Lombardi drawings [11]. The use of regular sinusoid curves to indicate relative length was recently introduced by Nielsen *et al.* [12] for the visualization of connectivity graphs in genome sequencing. Lastly we note the topic of map labeling, where a suitable position of each label must be found among a set of candidate positions (see [13] for a survey). In particular the edge labeling version studied in [14, 15] is closely related (see Section 4).

Organization. Section 2 explains the model used to find suitable curves, and how to fit cubic Bézier splines. Section 3 discusses the optimal placement of curves, but also shows the restrictions of this approach. In Section 4 we prove that under a mild realistic input assumption, a non-overlapping choice of placement of the curves can be computed in $O(n \log n)$ time, if such a placement exists. We also study the problem of maximizing the centered curve positions under the condition that all curves can be

placed, which is NP-hard (Section 5). Section 6 discusses a heuristic approach to compute a solution maximizing the edges in centered position. In Section 7 we show how our techniques can be applied to create animations of time-dependent data such as traffic conditions. Finally, in Section 8 we discuss the advantages and disadvantages of the proposed method and look at possibilities for future work.

2 Preliminaries on Fitting Curves

Problem Setting. We assume a planar graph with a fixed embedding is given, along with the travel times for all edges. We compute a linear cartogram with the same topology and embedding, where all edges are drawn as sinusoid curves whose lengths are proportional to the specified travel times. For each edge e we define a region close to e and draw a curve with the specified length inside that region. To avoid intersections among the curves, we make sure that regions of different edges do not intersect. We consider three possible placements of the regions: above the edge, centered, or below the edge (see Fig. 2). When centered, the curve occupies a diamond-shaped region where the edge is a diagonal of the diamond. In the other two positions, the curve occupies a triangle-shaped region based on the edge. To minimize visual distortion we prefer to place curves in the centered, diamond-shaped position. Reducing the problem solely to the centered positions, however, is overly restrictive, as shown in Section 3.

We call the length of the diagonal of each region that is normal to its edge e the *width* w_e. The width is directly determined by the length of the edge, the associated travel time and the desired frequency. The two triangles and diamond of an edge induce four parts the region could occupy, called *zones* (see Fig. 2 (d)). The vertex of a triangle or diamond that is not part of the edge is called the *apex*. Any edge is associated with four apices, one of each triangle and two of its diamond.

Different Shapes. We represent the distorted edges with C^2-continuous cubic Bézier splines. This type of curve is already commonly present in many maps (e.g., [16]) and a continuous spline maintains continuity of edges. We fit the Bézier spline inside a zone representing the widened edge. This zone can be represented by various shapes. These shapes are not present in the final map and hardly influence the visual appearance (see Fig. 3). However, since space around vertices is limited, tapering shapes, such as triangles, are less likely to intersect at vertices and hence allow for a greater range of feasible solutions. Non-uniform shapes, such as a rectangle in the center of an edge, are also suitable.

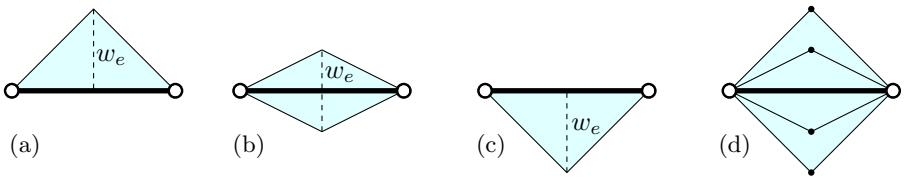


Fig. 2. (a)–(c) The regions with width w_e for an edge e. (d) The zones and apices of e.

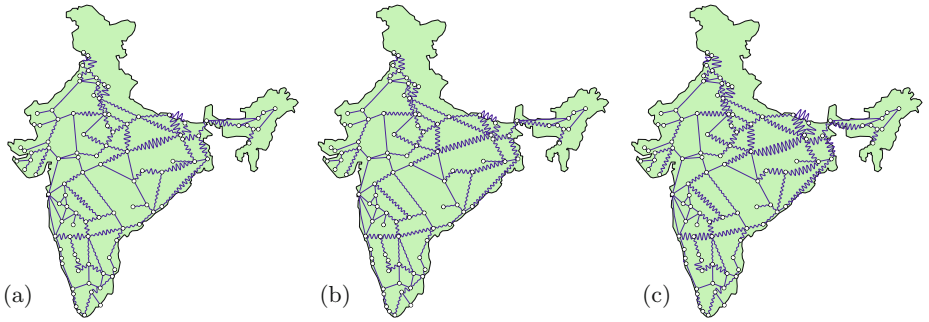


Fig. 3. Different edge shapes for the Indian railroad network. Frequencies are exaggerated for display purposes. (a) Triangles. (b) Rectangles. (c) Ellipsoids.

We study the algorithmic aspects of fitting curves using triangular and diamond-shaped areas. Both the algorithm of Section 4 and the NP-hardness proof of Section 5 also hold for uniform rectangular shapes. Our algorithm does not work for ellipses, however, in this case there is a trivial $O(n^2)$ algorithm.

Relating Width to Curve Length in the Triangle Model. One can fit a C^2-continuous sinusoid cubic Bézier spline such that the length of the fitted curve is (nearly) linearly proportional to the width of the edge. The sinusoid spline starts and ends at the endpoints of the edge, so the number of oscillations is a multiple of one half. We let the number of oscillations used depend on the edge length and the specified travel time and we either keep the length of all oscillations (the frequency) or the edge width equal across the network.

Each oscillation is represented by two cubic Bézier curves. The control points of the Bézier curves are evenly distributed along the outer edge of the zone (Fig. 4 (a)) to ensure that the resulting spline uses the full available area. As the control points are equally spaced and the tangents of two consecutive curves are aligned, the connection between consecutive Bézier curves is C^2-continuous. A degenerate case occurs when two curves c_1 and c_2 connect at the apex p (Fig. 4 (b)). To keep the connection C^2-continuous, we select as the connection point the center of the third control point of c_1 and the second of c_2 (Fig. 4 (c)).

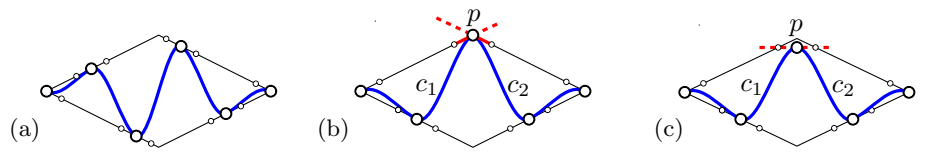
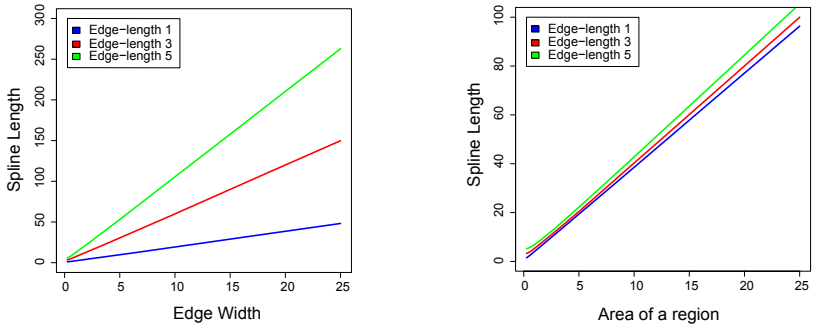


Fig. 4. (a) A sinusoid curve that is fitted to a diamond-shaped area. (b) The spline is not C^2-continuous as the tangents of curve c_1 and c_2 do not line up. (c) The continuity is restored when point p is moved down.



(a) (b)

Fig. 5. (a) Relation between the width of an edge and the length of the fitted spline for different edge lengths. (b) Relation with the size of the area that is fitted.

The exact length of a cubic Bézier curve cannot be computed by a closed formula, but it can be ϵ-approximated by regularly sampling the curve. In Fig. 5 (a) and 5 (b) the relation between the width, respectively area, of a zone and the length of the fitted Bézier spline is plotted for different edge lengths. We note that the relationship is nearly linear.

3 Optimal Assignment of Centered Curves

To minimize the distortion introduced by lengthening edges, curves should preferably be centered on the edge they represent. Here we explore the effect of placing all curves in their centered region. We assume that edge length and travel time (the desired edge length) are given as input variables. This leaves two variables that can be used to fit curves of the correct length: edge width (amplitude) and curve frequency.

To minimize the number of visual variables in the map, we fix either the frequency or the edge width in the network. This directly determines a function relating the required curve length to the opposing variable. When fitting curves, there should be no overlap between different curves, and thus regions.

By fixing a uniform edge width across the network, the length of a curve is directly related to the number of oscillations on the edge. Using basic geometry we can compute the maximum edge width that still allows a non-overlapping solution. A solution with a maximum width has a minimum frequency, which may be preferable to distinguish oscillations (see Fig. 6 (a)). Any solution with a smaller width can be obtained through scaling.

Instead of fixing the edge width, we can also fix the curve frequency for all edges. This directly implies an edge width for all edges, though a too low frequency may cause overlap between regions. To prevent overlap, we compute the minimum feasible uniform frequency (see Fig. 6 (c)). As frequency maintains the relative edge widths, we can apply similar geometry as before.

Fig. 6. Expected travel times on the main highway network in the USA. The dense area in the East of the USA overly restricts the results in the rest of the map. Placement regions are indicated in the insets. Bottleneck regions are indicated in red. (a) Using equal edge widths. (b) Using equal frequency.

Critical areas with a high edge density are often restricted to only a small section of the map. The high edge density in combination with a high likelihood of delays, however, can cause the minimum frequency for the entire map to be highly constrained. We can add flexibility to the solution by also using the two outer triangular regions to place the curve, instead of only the centered diamond. This allows solutions with a lower minimum frequency, but changes the problem into an assignment problem which we discuss in the next section.

4 Efficiently Computing Placements of Curves

For each edge we have three candidate regions, a diamond and two triangles, and we must choose one per edge so that the choices do not intersect. Essentially the same problem was studied by Poon *et al.* [14] for rectilinear map labeling. They showed that the problem of selecting non-overlapping regions can be transformed to a 2-satisfiability (2-SAT) instance. In a 2-SAT instance a Boolean formula is given that contains a conjunction of

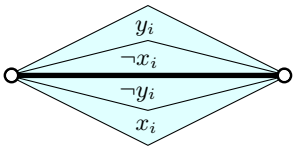


Fig. 7. Assignment of variables to the zones of an edge

disjunctions, where each disjunction consists of two Boolean variables or their negation. An example of such a formula would be $(\neg a \vee b) \wedge (\neg c \vee \neg b) \wedge (a \vee \neg c)$. Each disjunction consists of at most two variables or their negation, but variables can be present multiple times in the formula.

For each edge e_i we use two Boolean variables x_i and y_i, where $x_i = \text{TRUE}$ if we use the one triangle and $y_i = \text{TRUE}$ if we use the other triangle. Hence, $x_i = y_i = \text{FALSE}$ means we use the diamond (see Fig. 7). To enforce the use of one of the three options, we add $(\neg x_i \vee \neg y_i)$. Moreover, to ensure that we never choose intersecting triangles or diamonds of two different edges, we make corresponding 2-SAT clauses representing this requirement. The conjunction of all 2-SAT clauses gives a 2-SAT formula that is satisfiable if and only if there is a choice of regions, one per edge.

Satisfiability of 2-SAT formulas can be tested in time linear in their length [17]. The values of the Boolean variables show which placement to take for each edge. There can be quadratically many pairs of intersecting regions, so in the worst case the formula has quadratic length. This leads to a straightforward quadratic time solution using standard techniques. An improvement to $O(n^{4/3} \operatorname{polylog} n)$ time is possible using advanced and rather impractical techniques [15] (see also [10]).

4.1 Reduced Time Algorithm

We show that $O(n \log n)$ time can be achieved under a very mild realistic input assumption: for each edge, the apices of its triangles and diamond have angles at least β, for some constant $\beta > 0$. That is, regions may not be arbitrarily wide compared to their edge length. The improvement is based on computing an equivalent 2-SAT formula that has only linear length and that can be constructed in $O(n \log n)$ time. More precisely, the 2-SAT formula has $O(n/\beta^2)$ clauses and is constructed in $O((n \log n)/\beta)$ time. An overview is given in Algorithm 1.

Algorithm 1. `Compute area arrangement`(S, k)

1: Create $2\pi/\beta$ trapezoidal decompositions based on the edges.
2: Mark all zones that are intersected by an edge.
3: Create the arrangement of the valid zones.
4: Detect all overlaps between zones.
5: Generate the corresponding 2-SAT formula.

We first observe that if any edge e intersects any zone of a different edge e', then that zone cannot be used. Hence one or both Boolean variables of e' must be set a certain way to make the formula satisfiable. After finding and removing these zones, we build the arrangement of the remaining zones and show that it has complexity $O(n/\beta^2)$, implying that there will be at most that many clauses in the Boolean formula.

To determine all zones that intersect some edge, we use a number of fixed-direction ray shooting data structures in the set of edges (edges are disjoint since our input is planar). A fixed-direction ray shooting structure is simply a trapezoidal decomposition preprocessed for planar point location; it can be built in $O(n \log n)$ time and supports queries in $O(\log n)$ time. We choose a set D of $\delta = O(1/\beta)$ equal-spaced directions, ensuring that every apex of a triangle or diamond has a direction in D that points to its inside. For each direction in D we build a ray-shooting structure for that direction.

We perform δ ray-shooting queries from each endpoint p of each edge. Whenever the ray hits some edge e, we test if any zone of e contains p, and if so, we remove that zone. Then we perform another $4n$ ray-shooting queries, one for each apex of each edge. The direction of a ray r is toward the defining edge e. If r does not hit e as the first edge, then we can remove that zone of e. Together all ray-shooting queries identify all zones that cannot be used in a solution. This step takes $O((n \log n)/\beta)$ time in total.

We build the arrangement of all remaining zones to find intersecting pairs efficiently [18]. Vertices in the arrangement are either from the remaining zones or intersection points of remaining zones of different edges. The latter give rise to a clause to be included in the 2-SAT formula. We can show with a packing argument that the arrangement of the remaining zones has complexity $O(n/\beta^2)$ (see Section 4.2). Hence, we add at most a linear number of clauses to the 2-SAT formula.

Theorem 1. *Let E be a set of n disjoint edges in the plane, each with an isosceles triangle to one side, an isosceles triangle to the other side, and a diamond with the edge as its diagonal. If the apices of the triangles and diamond have an angle at least $\beta > 0$, then we can find in $O((n \log n)/\beta + n/\beta^2)$ time a choice of a triangle or diamond for each edge in E such that choices of different edges do not intersect.*

4.2 Packing Argument

We assume that all edges have at least an angle β at their apex and that β integrally divides 2π. If this is not the case we can set β to be the largest value smaller than β that does. We show that this requirement restricts the total number of overlaps possible between remaining zones. Recall that no remaining zone intersects an edge.

Lemma 1. *Let set S be a set of isosceles triangles intersecting an isosceles triangle T, where each element $s \in S$ has the same length sides as T. The total area that can be covered by S is bounded by $O(\|e\|^2)$, where $\|e\|$ is the length of a side of T.*

Proof. As all triangles considered are isosceles triangles, we know that if the sides have length $\|e\|$, the length of the base is bounded by $2 * \|e\|$. All points that can be covered by the intersecting triangles of T lie within a rectangle with sides proportional in e (see Fig. 8). Hence, the maximum area covered by intersecting isosceles triangles is bounded by $O(\|e\|^2)$. □

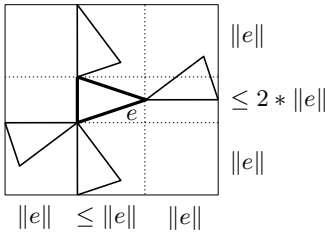


Fig. 8. The maximum area that can be covered is bounded by $O(\|e\|^2)$

Lemma 2. *The number of remaining, larger isosceles triangles intersecting an isosceles triangle T with sides of length e is bounded by $O(1/\beta^2)$.*

Proof. Define for each intersecting triangle an angle γ corresponding to the counterclockwise angle between its edge when encountered clockwise and the unit vector $(1, 0)$. We partition all intersecting triangles by the angle γ into of a set of intervals $[\alpha, \alpha + \beta]$, where $\alpha \in \{i * \beta : i \text{ is integral and } 0 \leq i \leq 2\pi/\beta - 1\}$. We bound the number of triangles within one interval by $O(1/\beta)$.

Within an interval, triangles cannot intersect. All triangles intersecting an edge have been filtered out in the first step and their relative direction lies within an interval of width β. All intersecting triangles are at least as large as T. We only look at the top part of the intersecting triangles up to a length e along the sides. The total area covered by all triangles within an interval $[\alpha, \alpha + \beta]$ is at most as large as the area covered by all intersecting triangles. By Lemma 1 this area is bounded by $O(e^2)$. As no triangles overlap and each triangle has a size of at least $\theta(e^2 * \beta)$, there can be at most $O(1/\beta)$ triangles in a partition.

As each partition has $O(1/\beta)$ intersecting triangles and there are $O(1/\beta)$ partitions, the number of triangles intersecting T is bounded by $O(1/\beta^2)$. □

Lemma 3. *When all areas are valid and have at least an angle β at their apex, the number of overlapping pairs of areas is at most $O(n/\beta^2)$.*

Proof. For each pair of overlapping outer areas, we count the overlap towards the smaller triangle. By Lemma 2 it follows that the number of outer areas pairwise overlapping is bounded by $O(1/\beta^2)$ for each outer area. Thus, the total number of outer areas pairwise overlapping is at most $O(n/\beta^2)$. As the number of overlaps for the outer areas is bounded by $O(n/\beta^2)$, the same must hold for intersections with the inner area. The total number of areas overlapping is bounded by $O(n/\beta^2)$. □

5 NP-Hardness

The algorithm in Section 4 computes a valid choice of regions if one exists. It does not, however, have any preference for what region is selected on an edge. Ideally we would want to maximize the number of regions placed in a centered position, the diamonds. Unfortunately, maximizing this number is NP-hard. We first prove that the problem is NP-hard even if all edges have a uniform length. Our construction, however, requires that some edges have a width to length ratio that is strictly larger than 1. We then prove that the problem is NP-hard even if all edges have an arbitrarily small width to length ratio. This, however, requires edges to have non-uniform lengths. Both versions of the problem are reduced from planar maximum 2-SAT [19]. We describe the *gadgets* that we use to encode variables, clauses, and wires between variables and clauses.

Bounded Length to Width Ratio. Fig. 9 illustrates the construction of the first version of the problem. The *variable gadget* consists of four edges whose diamonds intersect in a cycle (see Fig. 9 (a)). A variable gadget has two valid configurations, neither of which uses diamonds (see Fig. 9 (c)). These configurations encode the TRUE and FALSE states of the variable. The number of edges in a variable gadget can be increased to allow more wires to connect.

The *clause gadget* consists of two parallel edges with overlapping diamonds (see Fig. 9 (b)). The two incoming wires represent the two literals used in the clause. If a literal in a clause is FALSE (resp. TRUE), we ensure that the last edge in the wire gadget must choose (resp. need not choose) a triangle region intersecting the clause gadget.

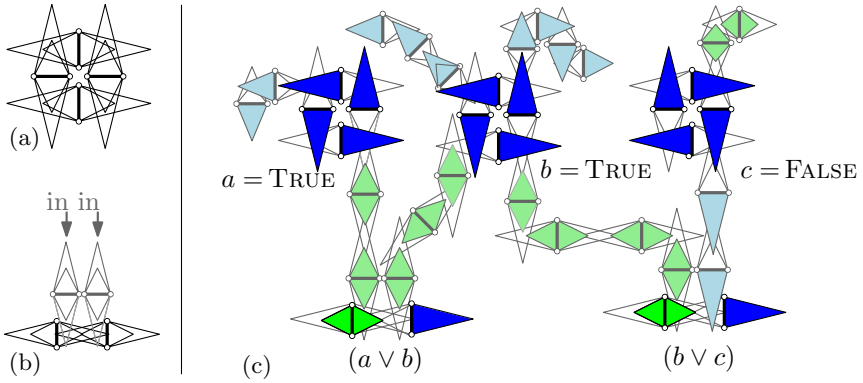


Fig. 9. Gadgets for variables and clauses, and solution for the formula $(a \vee b) \wedge (b \vee c)$

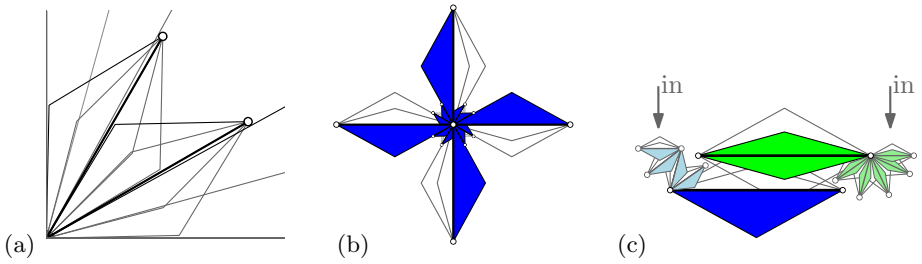


Fig. 10. Gadgets for second version: (a) zoomed in on part of the variable, (b) variable, (c) clause

Consequently, the diamond of the corresponding edge in the clause gadget cannot (resp. can) be selected (see Fig. 9 (c)). If both literals in a clause are TRUE, still only one of the diamonds can be selected as they overlap. Hence, if a clause is satisfied, the clause gadget can select one diamond; otherwise none.

The *wire gadget* consists of a sequence of edges with overlapping regions. Each diamond of an edge intersects the triangles of the previous and next edge. A wire gadget has two valid states: either all diamonds are selected or all triangles pointing from the variable to the clause. We connect each wire to a clause-variable pair such that the center regions of the wire can only be selected if the corresponding literal is TRUE. For each wire gadget that we use, we introduce a *counter-wire gadget*. It has the same length as the wire gadget and solely connects to the opposite assignment of the variable. Therefore, the summed number of diamonds selected in a wire gadget and its counter-wire gadget does not depend on the truth assignment.

Hence, maximizing the number of diamonds in the complete construction (see Fig. 9 (c)) corresponds to maximizing the number of satisfied clauses.

Arbitrary Length to Width Ratio. Now, edges may have an arbitrary length to width ratio, but also different lengths. Our NP-hardness construction is equal to the first version but uses slightly different gadgets.

The *variable gadget* is shown in Fig. 10 (b). Edges are placed around a common vertex, such that consecutive diamonds intersect, while the triangles do not cross the neighboring edges (see Fig. 10 (a)). This is always possible as the angle γ of a triangle is smaller than the sum of the angles of the two involved halves of the neighboring diamonds (2α) (see Fig. 11). By interspersing long edges with short edges in the gadget, we leave space for connecting wires.

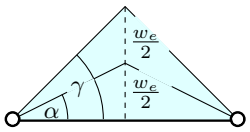


Fig. 11. $2\alpha < \gamma$

The clause gadget is similar to the clause gadget in the first reduction, but rotated by 90 degrees and slightly shifted in opposite directions (see Fig. 10 (c)). As the two edges in the clause gadget are shifted, we can always connect the wires to the clause.

Theorem 2. *Let E be a set of n disjoint edges in the plane, each with an isosceles triangle to one side, an isosceles triangle to the other side, and a diamond with the edge as its diagonal. The problem of choosing a triangle or diamond for each edge in E such that choices of different edges do not intersect and maximizing the number of diamonds chosen is NP-hard.*

6 Heuristics

Exactly computing the optimal setting that maximizes the number of regions in center position is not feasible in polynomial time. Heuristics try to optimize the number of centered edges, but give no optimality guarantee. Thus, they are able to reach polynomial running-times. In this section we present several heuristics that aim to maximize the number of centered edges.

First, note that if an edge intersects a zone of a different edge, this zone can never be part of a solution. We remove these zones by setting the corresponding variables, thus reducing the search space. Edges where the center zone does not intersect any other zone, can safely be selected and are also set. The same holds for clusters of edges that have non-overlapping center regions and only overlap the rest of the instance with their off-center regions.

Second, note that it is never advantageous to set an edge to an off-center position. Hence, we check only the effects of setting an edge to the center position. This may, however, force other edges to be set off-center. We test three heuristics that attempt to minimize the constrains placed on the solution space:

H1 Select the edge that invalidates the fewest regions among other edges.
H2 Select the edge that invalidates the fewest possible center regions.
H3 Select the edge that has the lowest ratio between option 2 and 1.

As selecting an area may have consequences that reach far across the network, we cannot test the result of a selection locally. Instead we use the 2-SAT representation of the input. The variables that correspond to the selected area are set and this information is propagated across the 2-SAT formula. Subsequently simplifying the 2-SAT formula prevents areas from being selected that would lead to invalid results. Using this approach we can guarantee that we always obtain a valid solution for the problem,

Table 1. Number of edges in center position for the different heuristics

Scenario (*number of edges*)	Optimal	H1	H2	H3
Los Angeles Highway Network (*133*)	128	128	128	128
Netherlands Highway Network (*118*)	105	105	105	105
India Railroad Network (*148*)	143	143	143	143
USA Main Highways (*136*)	118	118	118	118
Percentage of cases solved optimal (50 cases) - average error				
Random square 11 (*120*)	100% - 0	66% - 1.2	80% - 1.2	88% - 1.0
Random square 13 (*168*)	100% - 0	70% - 1.0	76% - 1.0	86% - 1.0
Random square 15 (*224*)	100% - 0	77% - 1.2	86% - 1.0	93% - 1.3

if one exists. Given the realistic input assumptions discussed in Section 4.1, testing the effect of setting an edge can be done in $O(n^2)$ time, where n is the number of edges. Hence, we obtain an algorithm for all three heuristics that runs in $O(n^3)$ time. As the expected input scenarios are relatively small, this running time is reasonably fast. By comparison, all heuristics solved the test-scenarios within a few seconds, the brute-force approach, however required up to half a day to optimally solve a scenario.

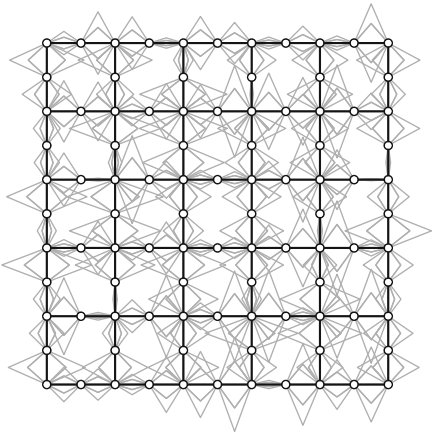


Fig. 12. Random grid-based scenario of size 11 by 11

As an informal use case test, we tested our heuristics on three scenarios based on real-life data. These scenarios are relatively easy to solve due to the inherent uneven distribution of dense areas. In Los Angeles, for example, nearly 90% of the center regions do not intersect any other edge. To test the behavior of the heuristic in more complex situations we also compute solutions for several more complex, randomly-generated scenarios. Each random scenario consists of a grid of vertices of size m by m, where all odd columns and rows are connected by edges (see Fig. 12). Edges are attributed a random width in the range $(0, 10)$ and then scaled to the largest size that allows a solution. For comparisons, we compute the optimum solution using a brute-force approach. We create 50 random scenarios of each type and compute the percentage of scenarios solved to optimality by the heuristics, as well as the average error when optimality is not reached, see Table 1.

For the real-life scenarios all heuristics are able to solve the problem to optimality, since the uneven distribution of density makes them considerably more tractable. Yet even for the more complex, artificial scenarios we manage high accuracy, which is probably caused by the geometry of the problem. The center position of an edge is likely to least constrain the rest of the problem and complex constructions that favor other allocations are less likely to occur in practice.

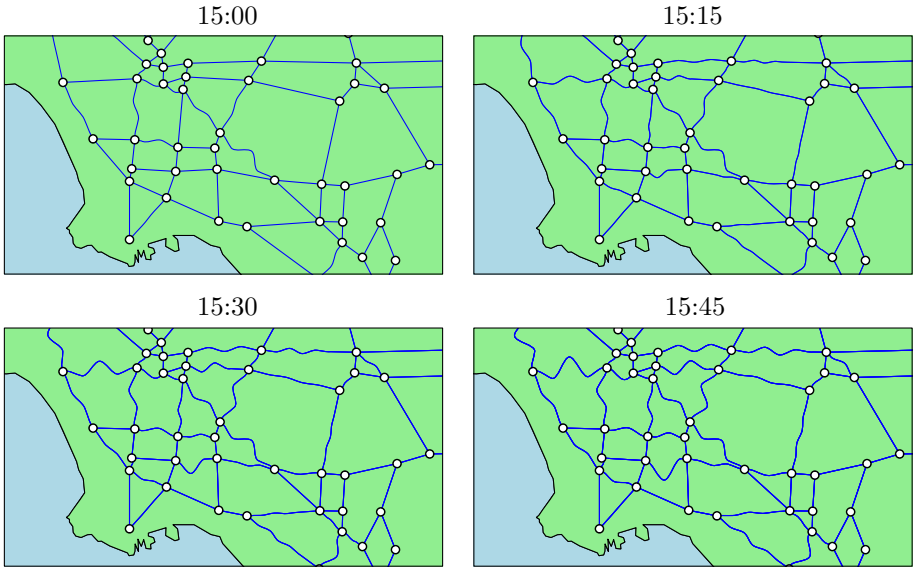


Fig. 13. By interpolating the control points of the curve we can generate fluent transitions in both frequency and amplitude. A fraction of the LA dataset is shown.

7 Animating Time-Dependent Data

When the distorted edge length to be displayed changes over time, for example due to traffic conditions, an animation of these changes may give more intuitive information on the underlying processes. Our linear cartograms are very suitable for animations. Here we describe how to use the heuristics from the previous section to create animated data. Further results can be found online[1].

Once more, we first look at the fixed width version and then investigate the fixed frequency version. For animated data, however, the fixed frequency approach appears to be less suitable. There is an increased computational requirement to maintain legal solutions and if the algorithm is run online, we can not guarantee a single consistent frequency across the network.

Fixed Width. Given an input graph we can compute either the maximum edge-width that allows a feasible solution or the maximum edge-width that allows all edges to be centered (see Section 3). As this width is independent of the delay along the edges, it can be computed beforehand and is maintained throughout the animation. All future steps solely alter the frequency along edges. We apply a fluent transition between different states by linearly interpolating the control points of the curves. Additional oscillations are introduced by adding degenerate curves, having all control points at either endpoint of the edge, before interpolating (see Fig. 13).

[1] `http://www.win.tue.nl/~agoethem/linear_cartograms/`, May, 2014

A disadvantage of fixed edge-widths is that the frequency on some edges may become too high. In extreme cases, increasing the frequency can cause problems with visualization.

Fixed Frequency. An alternative to fixed edge-width is using a fixed frequency. We assume, for now, that in each situation a valid configuration exists. Over time the data, and thus the width of edges, will change. If the changes are significant this may cause the current configuration to become invalid. While we can compute a valid configuration, a large change in configuration will create overly complex animations. The number of changes in the selected zones should be minimized. To minimize zone changes we, once more, make use of the heuristics described in Section 6. Instead of the center position, the number of edges that maintain their "current" position is maximized.

If the width of the edges is significantly increased, no valid configuration may exist anymore. To obtain a feasible solution the curve frequency must be increased. Increasing the global frequency, however, creates a large overall change. This risks a visual disconnection between two consecutive states and causes undue attention to be drawn to areas without any significant change. Instead, we increase the frequency of only the invalidating edges. To maintain visual balance, all invalidating edges are set at the same frequency, which is the minimal frequency that generates a feasible solution. As a consequence, at any moment during the animation we display at most two discrete frequencies. A different approach would be to compute the minimum frequency that satisfies all steps beforehand. However, overly increasing the frequency to suit the most restricting time step may reduce visual clarity in less constricted time steps and prevents the algorithm from being run online.

8 Discussion

We introduced a new type of linear cartogram where edges are drawn as sinusoid curves such that their length corresponds to travel time. We assumed that the regions occupied by these curves are triangular or diamond-shaped, and that the curves are C^2-continuous cubic Bézier splines. However, our approach applies to other visual styles as well (e.g., a piecewise linear curve zigzagging in the middle of an edge). The extension to using more than three regions per edge is straightforward, similar to [14]. Interestingly, our improvement under the realistic input assumption can also be used to speed up the line labeling problem in [15].

The optimization problem of maximizing the number of edges that use the centered region is NP-hard. Hence we introduced several heuristics and showed that they work efficiently in practice. Finally, we discussed how to create animations based on time-dependent data. The results give a clear and concise representation of the data without compromising the integrity of the map.

The benefit of our method over traditional edge coloring lies in the comparison of different paths through the network. As length is additional, in contrast to color, it is simpler to compare the required time investment of different paths. This observation, however, hinges on two key aspects. Firstly, the perceived distance between two points in a network should equal the sum of the edge lengths. Research into the distance-similarity metaphor [20] appears to indicate that in a network indeed edge length is

used as a measure of distance. Secondly, users should be able to accurately estimate the length of a regular sinusoid curve. When the frequency is fixed across the network, the edge *area*, instead of edge width, is related to the length of an edge. It is unclear if users intuitively will compare the size of the described areas and, therefore, are able to accurately compare the length of different edges. For a fixed edge width the comparison appears to be more intuitive, inherently turning into a symbolization for density on the edges.

While a formal user evaluation was not the main goal of this paper, for future work we recommend further exploration of the effects of this new method. To validate the applicability the perceived length of both types of regular sinusoid curves should be investigated. We note that our method is not restricted to using the exact length. Overemphasizing or underemphasizing edge length to compensate for the perceived length can easily be integrated in the method. Furthermore, a more complete and systematically selected dataset should be evaluated to explore the possible interplay effects with different scenarios.

From an algorithmic perspective for future work it would be interesting to see if a polynomial-time approximation algorithm would be possible maximizing the number of edges in center position. The problem, however, appears to be quite hard as choices can have consequences that reach far across the network. In contrast to many map-labeling problems, we require that all edges select an option, causing choices to propagate along the network. Another direction for future work is to determine if there are restrictions under which we can solve the problem optimally. All real-life scenarios investigated were solved near optimal by the heuristics introduced. If we could show that (most) real-life scenarios adhere to stricter input restrictions, the problem might not be NP-hard under those restrictions.

Acknowledgments. The authors would like to thank all reviewers for their extensive and insightful feedback which helped to improve the paper.

References

1. Robinson, A., Morrison, J., Muehrcke, P., Kimerling, J., Guptill, S.: Elements of cartography. John Wiley & Sons (1995)
2. Bies, S., van Kreveld, M.: Time-space maps from triangulations. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 511–516. Springer, Heidelberg (2013)
3. Cabello, S., Demaine, E., Rote, G.: Planar embeddings of graphs with specified edge lengths. Journal of Graph Algorithms and Applications 11(1), 259–276 (2007)
4. Kaiser, C., Walsh, F., Farmer, C., Pozdnoukhov, A.: User-centric time-distance representation of road networks. In: Fabrikant, S.I., Reichenbacher, T., van Kreveld, M., Schlieder, C. (eds.) GIScience 2010. LNCS, vol. 6292, pp. 85–99. Springer, Heidelberg (2010)
5. Shimizu, E., Inoue, R.: A new algorithm for distance cartogram construction. International Journal of Geographical Information Science 23(11), 1453–1470 (2009)
6. Langlois, P., Denain, J.C.: Cartographie en anamorphose. Cybergeo: European Journal of Geography (1996)
7. Bouts, Q., Dwyer, T., Dykes, J., Speckmann, B., Riche, N., Carpendale, S., Goodwin, S., Liebman, A.: Visual encoding of dissimilarity data via topology preserving map deformation (in preparation, 2014)

8. Barequet, G., Goodrich, M., Riley, C.: Drawing planar graphs with large vertices and thick edges. Journal of Graph Algorithms and Applications 8(1), 3–20 (2004)
9. Goodrich, M., Wagner, C.: A framework for drawing planar graphs with curves and polylines. Journal of Algorithms 37(2), 399–421 (2000)
10. Efrat, A., Erten, C., Kobourov, S.: Fixed-location circular-arc drawing of planar graphs. Journal of Graph Algorithms and Applications 11(1), 145–164 (2007)
11. Duncan, C., Eppstein, D., Goodrich, M., Kobourov, S., Nöllenburg, M.: Lombardi drawings of graphs. Journal of Graph Algorithms and Applications 16(1), 37–83 (2012)
12. Nielsen, C., Jackman, S., Birol, I., Jones, S.: Abyss-explorer: visualizing genome sequence assemblies. IEEE Transactions on Visualization and Computer Graphics 15(6), 881–888 (2009)
13. Wolff, A., Strijk, T.: The map labeling bibliography (2009), `http://liinwww.ira.uka.de/bibliography/Theory/map.labeling.html`
14. Poon, C.K., Zhu, B., Chin, F.: A polynomial time solution for labeling a rectilinear map. Information Processing Letters 65(4), 201–207 (1998)
15. Strijk, T., van Kreveld, M.: Labeling a rectilinear map more efficiently. Information Processing Letters 69(1), 25–30 (1999)
16. Saux, E., Daniel, M.: Data reduction of polygonal curves using B-splines. Computer-Aided Design 31(8), 507–515 (1999)
17. Aspvall, B., Plass, M., Tarjan, R.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Information Processing Letters 8(3), 121–123 (1979)
18. Halperin, D.: Arrangements. In: Handbook of Discrete and Computational Geometry. Chapman & Hall/CRC (2004)
19. Guibas, L., Hershberger, J., Mitchell, J., Snoeyink, J.: Approximating polygons and subdivisions with minimum-link paths. International Journal of Computational Geometry & Applications 3(4), 383–415 (1993)
20. Fabrikant, S., Montello, D., Ruocco, M., Middleton, R.: The distance–similarity metaphor in network-display spatializations. Cartography and Geographic Information Science 31(4), 237–252 (2004)