Time-Space Maps from Triangulations

Sandra Bies and Marc van Kreveld

Department of Information and Computing Sciences,
Utrecht University, The Netherlands

Abstract. Time-space maps show travel time as distances on a map. We discuss the case of time-space maps with a single center; here the travel times from a single source location to a number of destinations are shown by their distances. To accomplish this while maintaining recognizability, the input map must be deformed in a suitable manner. We present three different methods and analyze them experimentally.

1 Introduction

Thematic maps are ways to visualize geographic data in alternative ways. For example, metro maps show connectivity of metro lines while abstracting from geographic reality (correct location) [11], and cartograms may show countries by using size (area) to depict total population [15]. Cartograms come in different types: contiguous area [7,9], non-contiguous area [12], rectangular [10], rectilinear [4], circle [6], and linear cartograms [5]. Except for the linear cartogram, all types use the area to show a variable of interest (often population). Linear cartograms do not try to get areas of regions correct, but distances between locations. Distance could represent travel time or travel cost, or any other variable that typically increases with distance. All types of cartogram deform geographic space to achieve the visualization, but attempt to keep correct locations or relative locations in some way.

Time-space maps are linear cartograms where distances correspond to travel time [1,8,14]. They are also called distance-by-time cartograms. Two types of time-space maps exist: centered and non-centered. The centered version has one specific location (city) as the center, and the map shows travel times from this city to other cities by distances. Its visualization can be enhanced by circles depicting increments of e.g. 30 min. travel time, see Fig. 1. The non-centered version shows travel times between all pairs of cities by distances. The centered version can be error-free, but non-centered time-space maps can seldom be error-free: four points with six pairwise distances can typically not be embedded in the plane.

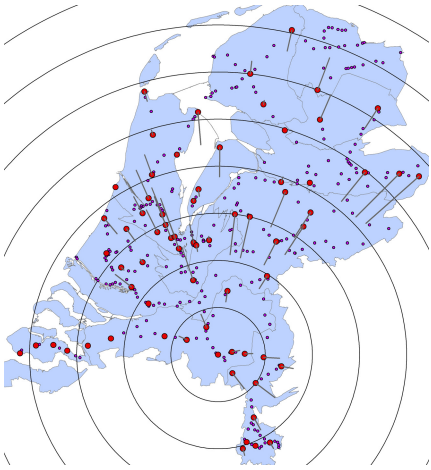


Fig. 1. Time-space map

W. Didimo and M. Patrignani (Eds.): GD 2012, LNCS 7704, pp. 511–516, 2013.

In this paper we focus on centered time-space maps. Let c be the central point and let S be a set of points whose travel time from c is given. Then the relative distances from all points in S to c are known on the time-space map. We can also maintain the direction from c to each point in S. This specifies the precise location of each point of S on the time-space map after choosing a global conversion that states what distance from c represents how much travel time. Fig. 1 has radial segments to show where cities originated in the input map.

In order to ensure that a deformation does not put cities outside the borders of a country, it is common to define a homeomorphism from the plane to the plane. This homeomorphism maps the points with known travel time to the correct location, and defines where all other points in the region are mapped. One way to define such homeomorphisms is by triangulations [13]. The vertices are the points of S, which move radially to the correct distance from c. The homeomorphism defines any point inside a triangle pqr to be mapped inside the deformed triangle using its barycentric coordinates. This defines a continuous mapping over the whole plane, but it may not be a homeomorphism: the deformation may collapse triangles, and some parts of the time-space map become multiply-covered. See Fig. 2 for an example.

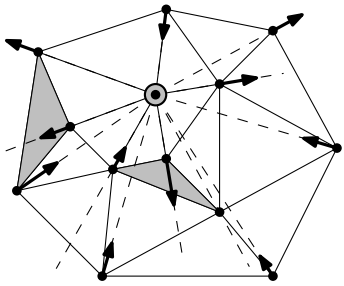


Fig. 2. Collapse of the grey triangles in the Delaunay triangulation when its vertices move

Our work is related to embedding graphs with specified edge lengths, but we are concerned with star graphs only, start out with an embedding, and need to extend the implied transformation to the whole plane.

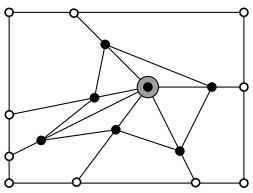
We present three triangulation-based methods to compute a homeomorphism that gives a correct time-space map in Section 2. Then we show the results of an implementation using data based on train travel time between stations in the Netherlands in Section 3. We also quantify the results by defining angle and distance deformation and applying it to the output of our implementations.

2 Three Methods for Computing Time-Space Maps

While a triangulation of the points of S can define a mapping for all points inside the convex hull of S, we typically want to deform the borders of a region as well. When coastal cities are well reachable from c, the coastline near these cities should also deform towards c. This means that we need to extrapolate the deformation function. This is done by a bounding box B that is sufficiently much larger than the convex hull of $S \cup \{c\}$. The bounding box has extra vertices that are used in the triangulation as well, but their deformation is zero (they don't move). This will mean that the deformation dies out gradually, close to the bounding box. All three methods described in this section use bounding boxes.

Radial Triangulation. We have seen that the Delaunay triangulation can lead to overlap and hence may not give a homeomorphism. But we can triangulate S, c and the

bounding box vertices such that any travel time specification gives a homeomorphism. The idea is to triangulate "as radial as possible".



Assume no two points of S are at the same angle from c. Add edges between the points of S in angular order around c, forming a polygon. Add an edge $\overline{cp}$ for any $p \in S$. Extend this edge outward to the bounding box. Add a new vertex p' on B and an extra edge $\overline{pp'}$. Then triangulate the remaining non-triangular faces.

It is easy to see that any travel time specification of the points of S gives homeomorphism (assuming the global scaling factor is chosen such that all points of S remain inside B); in this sense the triangulation is universal. It is also easy to see that the efficiency of computing the deformation is $O(n \log n)$ time, if S has n points. To compute a time-space map, we need to deform all map features (borders, rivers, lakes, etc.). Assuming we transform the m vertices defining the map features, we can compute the time-space map in $O((n + m) \log n)$ time using planar point location. So the method is correct and efficient, but later we will see that the poor shapes of the triangles lead to poor time-space maps.

Quasi-Delaunay Triangulation. Since the radial triangulation has poorly-shaped triangles, but the Delaunay triangulation may not give a homeomorphism, we can try to find a triangulation that is "as much Delaunay as possible" while ensuring a homeomorphism for the given travel time data. If we denote by $S(0)$ the points of S in the input and by $S(1)$ the points of S at their output location, we can imagine the process of moving the points of $S(0)$ to $S(1)$. Let $S(\frac{1}{2})$ denote the locations half-way: we will attempt to make $S(\frac{1}{2})$ as Delaunay as possible, but its structure should be valid for $S(z)$ for all $0 \leq z \leq 1$. The concept is akin to *joint* or *compatible* triangulations (see e.g. [3,13]):

We begin with the radial triangulation of $S(\frac{1}{2})$. Then we incrementally flip edges if two conditions hold: (i) the flip gives two triangles that do not collapse when transforming $S(0)$ into $S(1)$; (ii) the flip is a standard Lawson flip in $S(\frac{1}{2})$. To determine whether a collapse takes place, we parametrize the positions of the four points involved with a parameter z that increases from 0 to 1. The expression for a collapse (collinearity) is quadratic in z; if it has solutions in $[0, 1]$, then the flip will not be done.

The order in which flips take place is important. Suppose that $\overline{pq}$ is an edge that we may flip to $\overline{rs}$. If s lies inside the circumcircle of pqr, then it is a Lawson flip, and the closer s lies to the center of the circumcircle of pqr, the more eager we are to flip $\overline{pq}$. The flip priority is based on this distance. We continue flipping in decreasing order of priority until no more flips satisfy (i) and (ii).

The algorithm to construct the deformation takes $O(n^2 \log n)$ time because there can be at most quadratically many flips until a Delaunay triangulation is obtained. The logarithmic factor comes from heap operations based on the priority. The total time-space map construction is $O((n^2 + m) \log n)$.

Dynamic Delaunay Triangulation. It is not necessary to use one single triangulation to define the homeomorphism. We present a method that maintains the Delaunay triangulation for moving points, starting with $S(0)$ and ending with $S(1)$. Each point moves with constant speed on its ray. When the movement causes an empty-circumcircle test to

be violated, we flip and continue with the new Delaunay triangulation that is valid until the next empty-circumcircle violation. It is known that the process causes $O(n^3 \cdot 2^{\alpha(n)})$ flips [2],[1] although a tight bound for the maximum number of flips for moving points is not known.[2]

To transform a map feature vertex through the sequence of triangulations that make the homeomorphism, we simply trace the vertex using its barycentric coordinates. When a flip puts it in a new triangle, we compute its barycentric coordinates with respect to the new triangle and trace it further.

Computing the deformation takes $O(n^3 \cdot 2^{\alpha(n)} \log n)$ time, and tracing the m vertices of map features through the deformation takes $O(n^3 \cdot 2^{\alpha(n)} m)$ time.

3 Experimental Results

We implemented the three methods to determine how the results compare visually. We also defined two deformation measures to quantify which of the methods score better. Thirdly, the implementation gives an indication of how many flips are done in the dynamic Delaunay method. An input map and the time-space maps produced by the three methods is shown in Fig. 3.

Data and Output Maps. Our data set consists of the country border of the Netherlands from the Dutch cadastre, the Dutch train schedule, 68 InterCity stations of the Dutch Railways (NS), and 204 regular stations. We took twelve InterCity stations spread over the country as central location, and computed the time-space map for the InterCity stations only as destinations, or for all 271 other stations. This resulted in 72 time-space maps. Visually, it appears that the radial method has many artifacts, the quasi-Delaunay method has fewer artifacts, and the dynamic Delaunay method has some artifacts, but these can always be explained from the input data. The dynamic Delaunay method always gives the visually best maps. The maps appear good even for the situation with all 272 train stations, a larger and more difficult scenario than reported before in the literature [1,8,14].

Deformation Measures. The distance deformation (DD) captures how long a straight line segment on the input map becomes in the output map (where it will not be straight anymore). We sampled 500 pairs of points, each giving a line segment with a certain length on the input map. The line segment becomes a polygonal line on the time-space map, which also has a length. The ratio of the latter length and the former length is the distance deformation, and we compute the average of the ratios. The input line segment lengths are in five classes: 0–10 km, ..., 40–50 km, and there are 100 pairs in each class.

The angle deformation (AD) definition is based on a triple of randomly chosen points pqr on the input map, where the angle $\measuredangle pqr = 60°$. We transform pqr and measure the same angle on the time-space map. The absolute difference in angle is the angle distortion. We only take triples of points that are relatively close together (at most 10 km) to respect the fact that angle deformation should be a local measure. We average over 500 triples.

[1] $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function.
[2] The lower bound is $\Omega(n^2)$, which also holds for radially moving points.

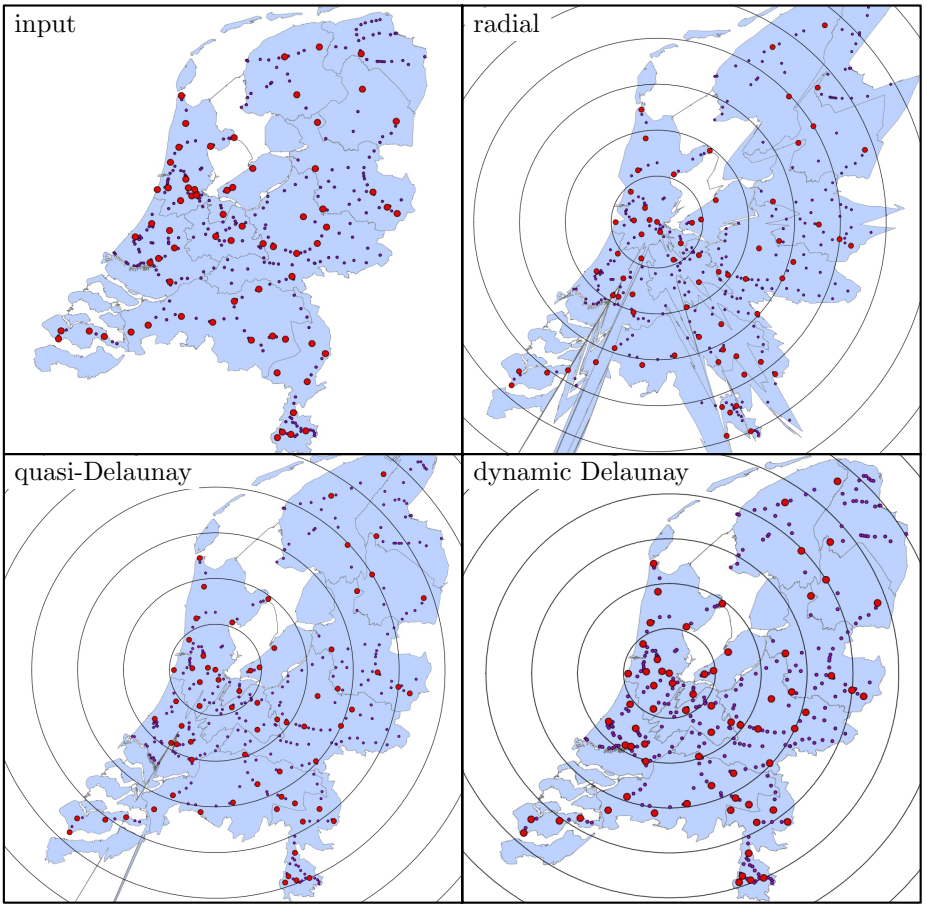


Fig. 3. Input map and output time-space maps produced by the three different methods. Amsterdam is the center, distances to other InterCity stations are used, and the circles correspond to 30 min., 60 min., 90 min., etc., travel time.

Table 1. Average deformations for the three methods (standard deviation in brackets). The number of flips shows either the number of flips to get from radial to quasi-Delaunay, or the flips to maintain the Delaunay triangulation during movement.

	Avg. DD in km	Avg. AD in deg.	no. of flips
InterCity, radial	58.8 (24.6)	37.0 (4.45)	N/A
InterCity, quasi-D	27.3 (24.4)	26.3 (5.02)	468 (56)
InterCity, dyn-D	7.0 (2.88)	19.6 (4.16)	130 (56)
All stations, radial	217.1 (66.4)	45.6 (3.06)	N/A
All stations, quasi-D	74.4 (49.1)	36.5 (2.16)	1572 (154)
All stations, dyn-D	10.2 (2.5)	25.3 (2.74)	921 (200)

Table 1 shows clearly that the dynamic Delaunay method produces the best results. For example, line segment of average length 25 km on the InterCity stations only data set becomes on the average about 7 km longer on the time space-map, whereas this is 27.3 km for Quasi-Delaunay and 58.8 km for radial triangulations. The average angle deformation is also smaller for the dynamic Delaunay method. These quantitative results support the visual findings that time-space maps produced by the dynamic Delaunay method are best.

References

1. Ahmed, N., Miller, H.: Time-space transformations of geographic space for exploring, analyzing and visualizing transportation systems. J. of Transport Geography 15, 2–17 (2007)
2. Albers, G., Guibas, L.J., Mitchell, J.S.B., Roos, T.: Voronoi diagrams of moving points. Int. J. Comput. Geometry Appl. 8(3), 365–380 (1998)
3. Aronov, B., Seidel, R., Souvaine, D.L.: On compatible triangulations of simple polygons. Comput. Geom. 3, 27–35 (1993)
4. de Berg, M., Mumford, E., Speckmann, B.: Optimal BSPs and rectilinear cartograms. Int. J. Comput. Geometry Appl. 20(2), 203–222 (2010)
5. Cabello, S., Demaine, E.D., Rote, G.: Planar embeddings of graphs with specified edge lengths. J. Graph Algorithms Appl. 11(1), 259–276 (2007)
6. Dorling, D.: Area Cartograms: their Use and Creation. Concepts and Techniques in Modern Geography, vol. 59. Environmental Publications, Norwich (1996)
7. Dougenik, J., Chrisman, N., Niemeyer, D.: An algorithm to construct continuous area cartograms. Prof. Geographer 37, 75–81 (1985)
8. Kaiser, C., Walsh, F., Farmer, C.J.Q., 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)
9. Keim, D., North, S., Panse, C.: CartoDraw: A fast algorithm for generating contiguous cartograms. IEEE Trans. Visu. and Comp. Graphics 10, 95–110 (2004)
10. van Kreveld, M., Speckmann, B.: On rectangular cartograms. Comput. Geom. 37(3), 175–187 (2007)
11. Nöllenburg, M., Wolff, A.: A Mixed-Integer Program for Drawing High-Quality Metro Maps. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 321–333. Springer, Heidelberg (2006)
12. Olson, J.: Noncontiguous area cartograms. Prof. Geographer 28, 371–380 (1976)
13. Saalfeld, A.: Joint triangulations and triangulation maps. In: Proc. 3rd ACM Symposium on Computational Geometry, pp. 195–204 (1987)
14. Shimizu, E., Inoue, R.: A new algorithm for distance cartogram construction. International Journal of Geographical Information Science 23(11), 1453–1470 (2009)
15. Tobler, W.: Thirty-five years of computer cartograms. Annals of the Assoc. American Cartographers 94(1), 58–71 (2004)