Algorithmic Aspects of Cartogram Computation

Bettina Speckmann*

Abstract

In this note we describe some recent algorithms for the computation of rectangular and rectilinear cartograms.

1 Introduction

Cartograms. Cartograms are a useful and intuitive tool to visualize statistical data about a set of regions like countries, states or counties. The size of a region in a cartogram corresponds to a particular geographic variable. The most common variable is population: in a population cartogram, the sizes (measured in area) of the regions are proportional to their population. In a cartogram the sizes of the regions are not the true sizes and hence the regions generally cannot keep both their shape and their adjacencies. A good cartogram, however, preserves the recognizability in some way.

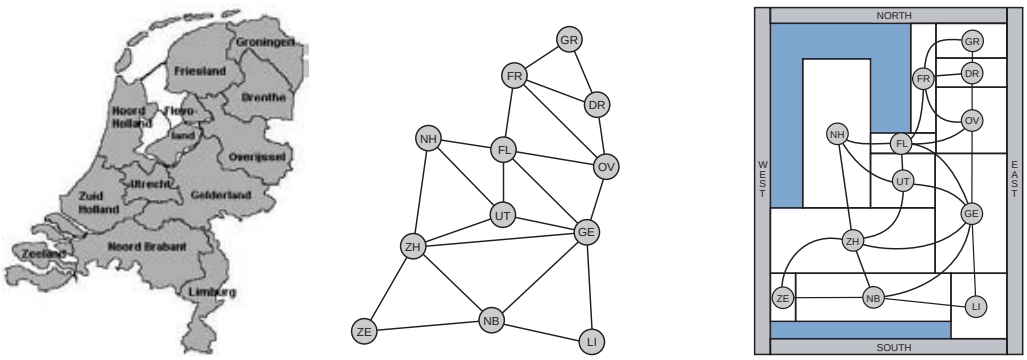

Figure 1: The provinces of the Netherlands, their adjacency graph, a population cartogram—here additional "sea rectangles" were added to preserve the outer shape.

Globally speaking, there are four types of cartogram. The standard type—also referred to as contiguous area cartogram—has deformed regions so that the

*Department of Mathematics and Computer Science, TU Eindhoven.
Email: speckman@win.tue.nl

Bulletin of the EATCS no 92, pp. 33–43, June 2007

desired sizes can be obtained and the adjacencies kept. Algorithms for such cartograms were given, among others, by Tobler [24], Dougenik et al. [10], Kocmoud and House [17], Edelsbrunner and Waupotitsch [11], Keim et al. [16], and Gastner and Newman [13].

The second type of cartogram is the non-contiguous area cartogram [12, 21]. The regions have the true shape, but are scaled down and generally do not touch anymore. Sometimes the scaled-down regions are shown on top of the original regions for recognizability. A third type of cartogram is based on circles and was introduced by Dorling [9]. The fourth type of cartogram is the rectangular cartogram introduced by Raisz in 1934 [22]. Each region is represented by a single rectangle, which has the great advantage that the sizes (area) of the regions can be estimated much better than with the first two types. However, the rectangular shape is less recognizable and it imposes limitations on the possible layout. Hybrid cartograms of the first and fourth type, so called *rectilinear cartograms* exist as well. Here, regions are rectilinear polygons with a small number of vertices instead of rectangles (see, for example, Fig. 10). In this note we focus on algorithms for *rectangular* and *rectilinear cartograms*.

Quality criteria. Whether a cartogram is good is determined by several factors. Two important criteria are the correct adjacencies of the regions of the cartogram and the *cartographic error* [10]. The first criterion requires that the dual graph of the cartogram is the same as the dual graph of the original map. Here the *dual graph* of a map—also referred to as *adjacency graph*—is the graph that has one node per region and connects two regions if they are adjacent, where two regions are considered to be adjacent if they share a 1-dimensional part of their boundaries (see Figure 1). The second criterion, the cartographic error, is defined for each region as $|A_c - A_s| / A_s$, where A_c is the area of the region in the cartogram and A_s is the specified area of that region, given by the geographic variable to be shown. Other criteria include suitable relative positions of the regions and bounded aspect ratio in the case of rectangular cartograms.

Computing cartograms. From a graph-theoretic point of view constructing rectangular cartograms with correct adjacencies and zero cartographic error translates to the following problem. We are given a plane graph $\mathcal{G} = (V, E)$ (the dual graph of the original map) and a positive weight for each vertex (the required area of the region for that vertex). Then we want to construct a partition of a rectangle into rectangular regions whose dual graph is $\mathcal{G}$—such a partition is called a *rectangular dual* of $\mathcal{G}$—and where the area of each region is the weight of the corresponding vertex. As usual, we assume the input graph $\mathcal{G}$ is plane and triangulated, except possibly the outer face; this means that the original map did not

have four or more countries whose boundaries share a common point and that G does not have degree-2 nodes. Degree-2 nodes and four-country-points can easily be handled using suitable pre- and postprocessing steps, see [25] for details.

Unfortunately not every vertex-weighted plane triangulated graph admits a rectangular cartogram, even if we ignore the vertex weights and concentrate only on the correct adjacencies. The graph in Figure 2 (left), for instance, does not have a rectangular dual. The graph in the middle of Figure 2 does have a rectangular dual (Figure 2 (right)) but if, for example, the weight of vertex 1 and 3 is 10 and the weight of vertex 2 and 4 is 100, then no rectangular cartogram with correct adjacencies and zero cartographic error exists.

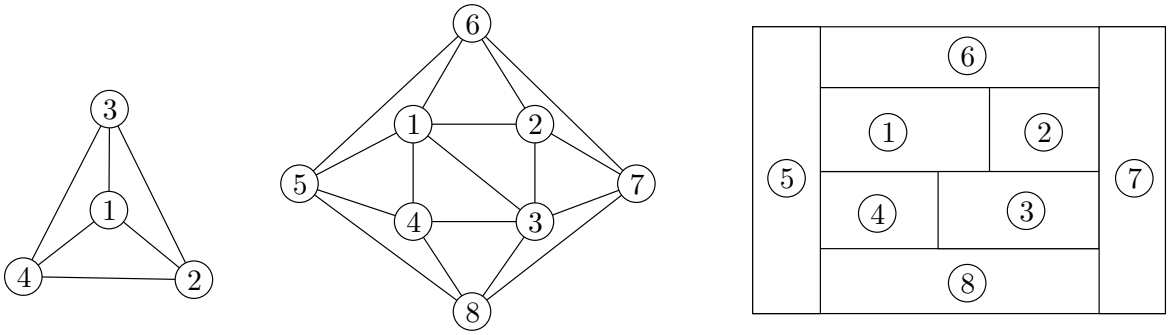

Figure 2: No rectangular dual (left); the graph in the middle does have a rectangular dual (right) but for certain weights no rectangular cartogram can be constructed.

There are several possibilities to address this problem. One is to relax the strict requirements on the adjacencies and areas. In Section 2 we discuss several algorithms that construct rectangular cartograms that in practice have only a small cartographic error and mild disturbances of the adjacencies. Heilmann et al. [14] gave an algorithm that always produces regions with the correct areas; unfortunately the adjacencies can be disturbed badly. The other extreme is to ignore the area constraints and focus only on getting the correct adjacencies—that is, to focus on rectangular duals rather than cartograms. This setting is relevant for computing floor plans in VLSI design. As mentioned above, ignoring the area constraints still does not guarantee that a solution exists. But, if the input graph is a triangulated plane graph with four vertices on the outer face and without separating triangles—a separating triangle is a 3-cycle with vertices both inside and outside the cycle—then a rectangular dual always exists [2, 18] and can be computed in linear time [15].

Another option is to use different shapes for the regions. Here we restrict our attention to rectilinear cartograms, which use rectilinear polygons as regions—see [8, 20] for some examples from the cartography community. If we now ignore the area requirement then things become much better: Any plane triangulated

graph admits a rectilinear dual. In fact, Liao et al. [19] showed that any plane triangulated graph admits a rectilinear dual with regions of small complexity, namely rectangles, L-shapes, and T-shapes. In Section 3 we discuss an algorithm that computes a rectilinear cartogram from such a rectilinear dual. There we show that any plane triangulated vertex-weighted graph admits a rectilinear cartogram, all of whose regions have constant complexity, and which has zero cartographic error and correct adjacencies.

2 Rectangular cartograms

In this section we sketch the algorithms presented in [23, 25]. They are iterative, semi-combinatorial, and the first algorithms for rectangular cartogram construction. Assume that we are given an administrative subdivision into a set of regions, that is, a map $\mathcal{M}$. As mentioned above the adjacencies of the regions can be represented by a graph $\mathcal{G}$, which is the dual graph of $\mathcal{M}$.

1. Preprocessing: The adjacency graph $\mathcal{G}$ is in most cases already triangulated (except for its outer face). In order to construct a rectangular dual of $\mathcal{G}$ we first have to process internal vertices of degree less than four and then triangulate any remaining non-triangular faces.

2. Directed edge labels: Any two nodes in the adjacency graph have at least one direction of adjacency which follows naturally from their geographic location. While in theory there are four different directions of adjacency any two nodes can have, in practice only one or two directions are applicable and, in fact, there is only a small number of adjacent regions where more than one direction is reasonable. The algorithms go through all possible combinations of direction assignments and determine which one gives a correct or the best result. We call a particular choice of adjacency directions a *directed edge labeling*.

Observation 1. *An adjacency graph $\mathcal{G}$ with a directed edge labeling can be represented by a rectangular dual if and only if*

1. *every internal region has at least one North, one South, one East, and one West neighbor, and*
2. *when traversing the neighbors of a node in clockwise order starting at the western most North neighbor we first encounter all North neighbors, then all East neighbors, then all South neighbors and finally all West neighbors.*

A realizable directed edge labeling constitutes a *regular edge labeling* for $\mathcal{G}$ as defined in [15] which immediately implies our observation.

3. Rectangular layout: To actually represent a face graph together with a realizable directed edge labeling as a rectangular dual we have to pay special attention to the nodes on the outer face since they may miss neighbors in up to three directions. To compensate for that we add four special regions NORTH, EAST, SOUTH, and WEST, as well as *sea rectangles* that help to preserve the original outline of the subdivision. Then we can employ the algorithm by He and Kant [15] to construct a *rectangular layout*, i.e., the unique rectangular dual of a realizable directed edge labeling.

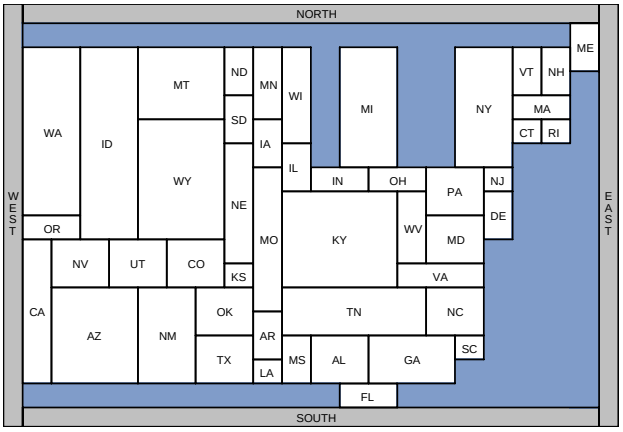

Figure 3: One of 4608 possible rectangular layouts of the US.

4. Area assignment: For a given set of area values and a given rectangular layout we would like to decide if an assignment of the area values to the regions is possible without destroying the correct adjacencies. Should the answer be negative or should the question be undecidable, then we still want to compute a cartogram that has a small cartographic error while maintaining reasonable aspect ratios and relative positions.

In [25] three algorithms are described that compute a cartogram from a rectangular layout. This work is extended in [23] where a fourth algorithm is introduced. The first algorithm presented in [25] is the simple segment moving heuristic, which loops over all maximal segments in the layout and moves each with a small step in the direction that decreases the maximum error of the adjacent regions. After a number of iterations, one can expect that all maximal segments have moved to a locally optimal position. However, there is no proof that the method reaches the global optimum or that it even converges. Secondly, if the rectangular layout is L-shape destructible (see [25]) then one can compute a zero-error cartogram if one exists. The third method is based on bilinear programming and can produce a cartogram with minimum maximal error, provided a good bilinear program solver is available. Unfortunately, bilinear programs are notoriously difficult and none of the available solvers is guaranteed to work [1].

In [23] area assignment is formulated as a linear program which takes only the vertical or only the horizontal segments into account. The algorithm then alternatingly solves a linear program for the vertical and the horizontal segments. The segment moving heuristic and the linear programming based approach were implemented—the latter using the well-known CPLEX program [3]. Both meth-

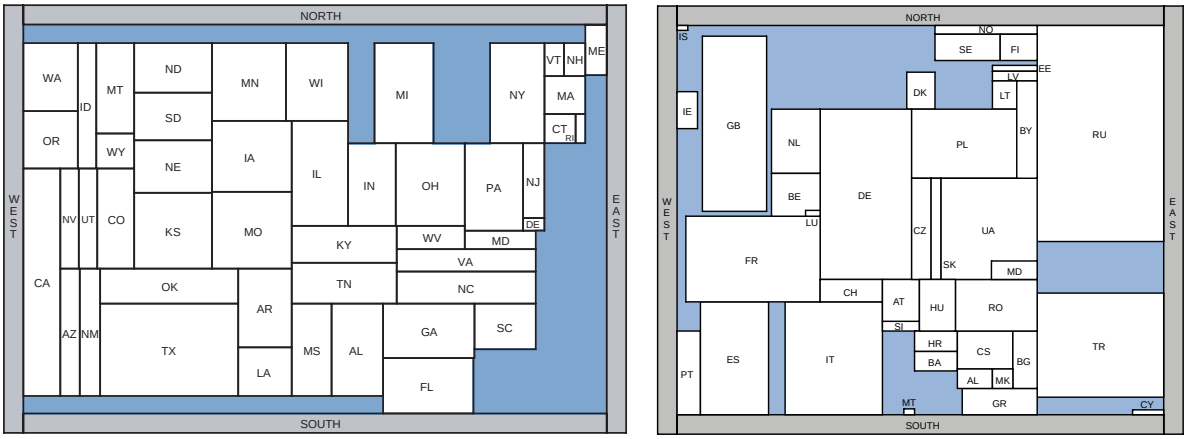

Figure 4: Highway kilometers of the US; the population of Europe (country codes according to the ISO 3166 standard).

ods can relax the adjacency constraints to compute cartograms with lower cartographic error at the cost of mild disturbances in the adjacencies.

Although the segment moving heuristic often gives aesthetically pleasing cartograms with small error (see Fig. 4 (left)), the linear programming based approach in general produces cartograms with a lower cartographic error, a smaller aspect ratio, and a better global shape. Furthermore, it is also able to handle L-shaped regions in a cartogram (see Fig. 4 (right)) and to compute satisfactory cartograms of all countries of the World (see Fig. 5).

3 Rectilinear cartograms

Rectilinear cartograms are a generalization of rectangular cartograms where regions can be rectangles, or L-shapes, or any other type of rectilinear polygon. Recall that even if a plane triangulated graph $\mathcal{G}$ has a rectangular dual then this does not imply that an error free cartogram for $\mathcal{G}$ exists. However, if we allow the regions to be rectilinear polygons then a cartogram with zero cartographic error and correct adjacencies exists for any plane triangulated graph $\mathcal{G}$ and any assignment of areas to regions.

More specifically, let $\mathcal{G} = (V, E)$ be a plane triangulated graph—the adjacency graph of the input map—where each vertex is assigned a positive weight—the preferred area of the region that corresponds to this vertex. A rectilinear dual of $\mathcal{G}$ is a partition of a rectangle into $|V|$ simple rectilinear regions, one for each vertex, such that two regions are adjacent if and only if the corresponding vertices are connected by an edge in E. A rectilinear cartogram is a rectilinear dual where the area of each region is equal to the weight of the corresponding vertex. The following theorem is proven in [5, 7].

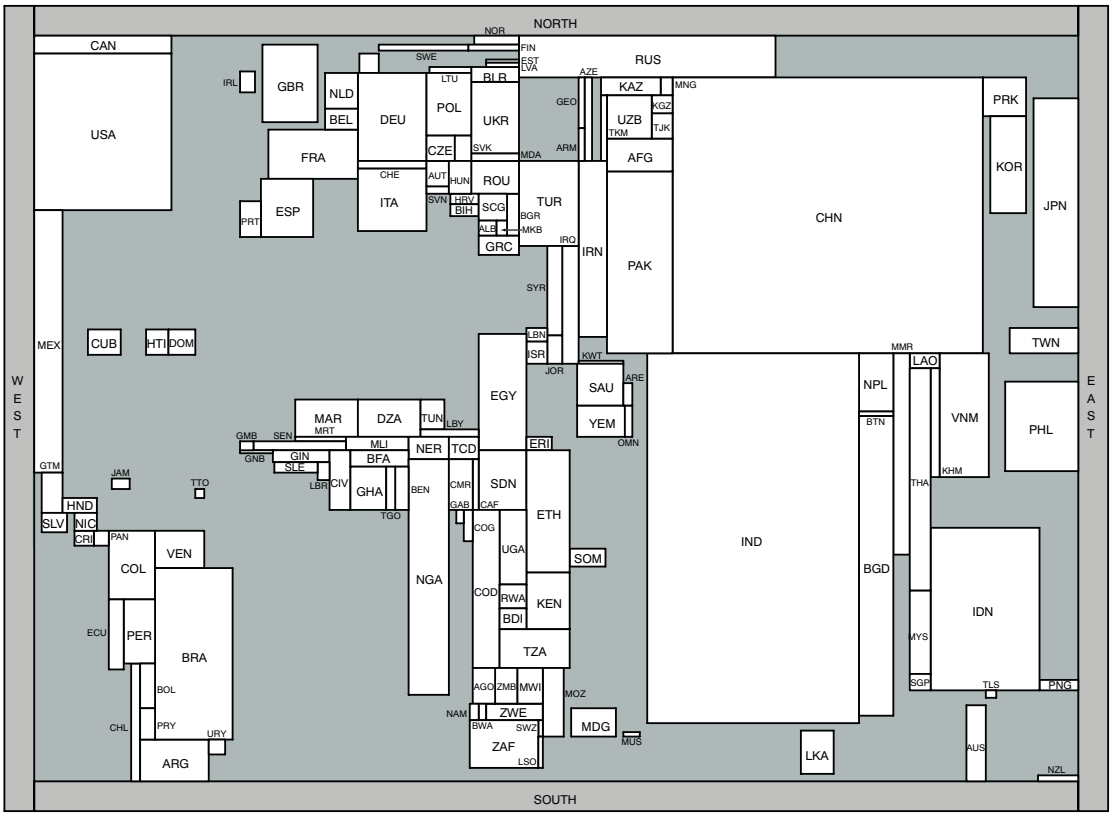

Figure 5: A population cartogram of the World .

Theorem 1. *Every vertex-weighted plane triangulated graph $\mathcal{G}$ admits a rectilinear cartogram of constant complexity, that is, a cartogram where the number of vertices of each region is constant.*

The proof of this theorem is constructive. The underlying algorithm has been implemented and evaluated experimentally—the results can be found in [6]. We briefly sketch the algorithm in the following paragraphs. It produces regions of very small complexity for real world data sets—in fact, most regions are rectangles (see Fig. 10). Assume again that we are given a subdivision, that is, a map $\mathcal{M}$, and its dual graph $\mathcal{G}$.

1. Preprocessing: Just as in the previous section we preprocess $\mathcal{G}$ to ensure that it has a rectangular dual. That is, we triangulate faces that are not triangulated yet, we add additional NORTH, EAST, SOUTH, and WEST vertices to form the boundary of the graph, and we split vertices to remove separating triangles. This can be done in such a way, that the final cartogram is guaranteed to have regions of bounded complexity.

2. Constructing a rectangular layout: As before we use the algorithm by Kant and He [15] to construct a rectangular layout $\mathcal{M}_1$ for $\mathcal{G}$. In fact, we again try

all reasonable rectangular layouts, that is, those layouts that correspond to the geographic situation as much as possible.

3. Constructing a BSP: Our algorithm works for so-called *sliceable* layouts, that is, layouts which can be obtained by recursively slicing a rectangle by horizontal and vertical lines. To turn the layout $\mathcal{M}_1$ that results from Step 2 into a sliceable layout we compute a rectilinear binary space partition (BSP) for $\mathcal{M}_1$. We would like the BSP to avoid cutting regions as much as possible, hence we use the BSP-construction algorithm by d'Amore and Franciosa [4], which guarantees that each rectangle in $\mathcal{M}_1$ is cut into at most four pieces. We use several heuristics that subdivide the weight in a suitable manner for each region that is cut by the BSP. This step results in a sliceable layout $\mathcal{M}_2$ (see Fig. 6, for simplicity we depict a graph $\mathcal{G}$ that has a sliceable layout).

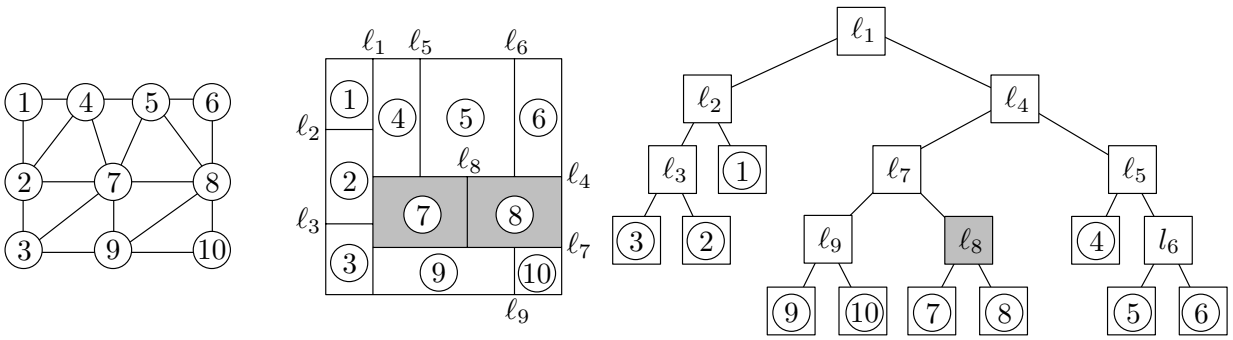

Figure 6: A graph $\mathcal{G}$, the layout $\mathcal{M}_2$, and the BSP tree $\mathcal{T}$.

4. Getting the areas right: The input to Step 4 is a BSP tree $\mathcal{T}$ for the rectangular layout $\mathcal{M}_2$ (see Fig. 6). Recall that each internal node v in a BSP tree stores a splitting line; we denote this line by $\ell(v)$. Also recall that each leaf in the BSP corresponds to a cell in the BSP subdivision (which in our case is contained in one of the regions of $\mathcal{M}_2$). For an internal node v we define $R(v)$ to be the union of all the cells corresponding to leaves in the subtree of v. This implies that $R(\text{root}(\mathcal{T}))$ is simply the whole map area, and that the splitting line $\ell(v)$ splits $R(v)$ into $R(\text{leftchild}(v))$ and $R(\text{rightchild}(v))$.

We traverse $\mathcal{T}$ top down. At each node v the splitting line $\ell(v)$ is repositioned while maintaining the following invariant: when a node v is handled, the current area of $R(v)$ is equal to the sum of the weights of the required areas of the leaf cells in the subtree $\mathcal{T}(v)$ rooted at v. We start at the root and scale $\mathcal{M}_2$ for R to have the required area. Then for each internal node v we position $\ell(v)$ such that $R_{\text{rightchild}(v)}$

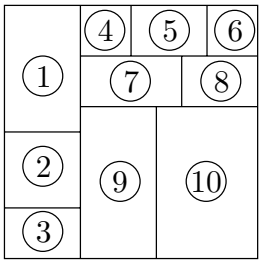

Figure 7: Layout $\mathcal{M}_3$.

and $R_{\text{leftchild}(v)}$ have correct areas. When we arrive at the leaves the corresponding cells have correct areas. See, for example, Fig. 7 that shows the layout $\mathcal{M}_3$ for the example in Fig. 6 and the weights $[w(1), \ldots, w(10)] = [0.15, 0.09, 0.06, 0.04, 0.06, 0.04, 0.08, 0.06, 0.18, 0.24]$.

The repositioning of the splitting lines may destroy some of the adjacencies. This will be remedied in the later steps.

5. Tailing: Repositioning the splitting lines during the previous step may have caused some of the adjacencies to be broken. We fix the adjacencies using so-called *tails*. These are very thin rectangles that are added to a region to connect it to some other region. The region then becomes an L-shape or something more complicated if a region gets several tails (see Fig. 8 which shows the result of Step 5 on our running example).

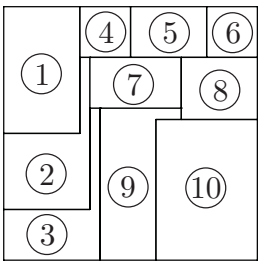

Figure 8: Layout $\mathcal{M}_4$.

Every tail introduces area errors. To prove that we can fix these errors in the following step without destroying any adjacencies we assume that all tails are extremely thin. However, when computing actual rectilinear cartograms we use several heuristics to widen the tails as far as possible while still producing zero error cartograms.

6. Correcting the areas again: The errors in the areas are corrected by moving the splitting lines —which have by now become polylines—of the BSP, similar to Step 4. This time, however, the errors to be repaired are so small that one can prove that the splitting lines have to be moved only so little that no adjacencies are destroyed. Fig. 9 shows the final rectilinear cartogram for our example. All figures were computed by the implemented algorithm and hence use heuristics to widen tails while keeping adjacencies and guaranteeing zero error.

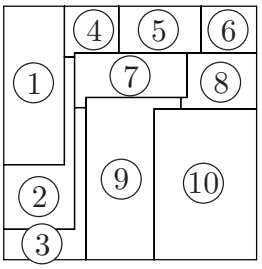

Figure 9: Layout $\mathcal{M}_5$.

4 Conclusions

The algorithms discussed in this note are the first algorithms for the computation of rectangular and rectilinear cartograms. They do produce aesthetically pleasing cartograms of high quality. However, some interesting open question related to rectangular cartogram computation still remain. For example, can one determine

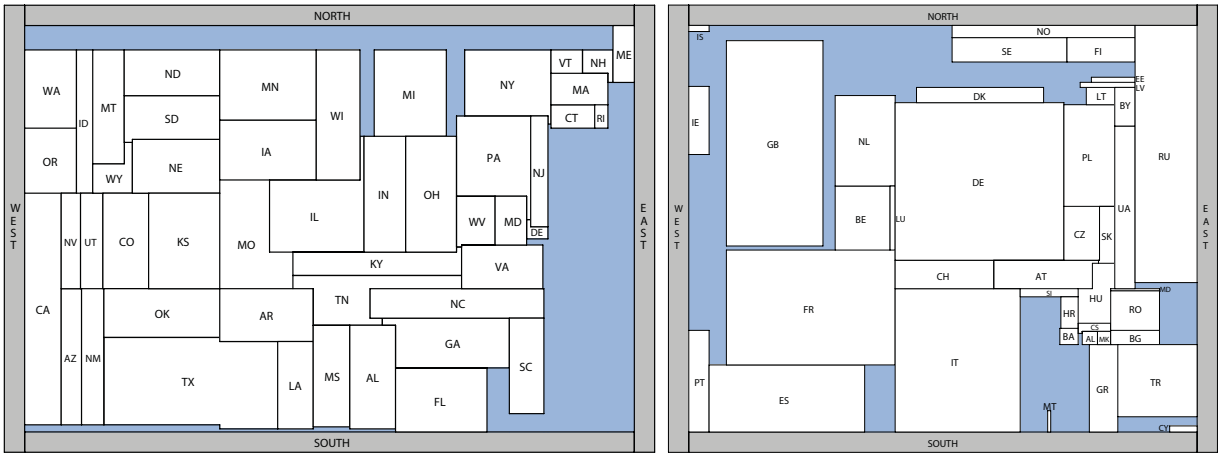

Figure 10: Highway kilometers of the US; gross domestic product (GDP) of Europe.

in polynomial time if there is a rectangular cartogram with zero cartographic error and correct adjacencies for a given plane triangulated graph with arbitrary weights? Secondly, the algorithms work particularly well for graphs that have a sliceable rectangular dual. Is there a complete characterization of this class of graphs?

References

[1] M. Bazaraa, H. Sherali, and C. Shetty. *Nonlinear Programming - Theory and Algorithms*. John Wiley & Sons, Hoboken, NJ, 2nd edition, 1993.

[2] J. Bhasker and S. Sahni. A linear algorithm to check for the existence of a rectangular dual of a planar triangulated graph. *Networks*, 7:307–317, 1987.

[3] CPLEX: High-performance software for mathematical programming and optimization. `http://www.ilog.com/products/cplex/`.

[4] F. d'Amore and P. G. Franciosa. On the optimal binary plane partition for sets of isothetic rectangles. *Information Processing Letters*, 44(5):255–259, 1992.

[5] M. de Berg, E. Mumford, and B. Speckmann. On rectilinear duals for vertex-weighted plane graphs. In *Proc. 13th International Symposium on Graph Drawing*, number 3843 in LNCS, pages 61–72, 2005.

[6] M. de Berg, E. Mumford, and B. Speckmann. Optimal bsps and rectilinear cartograms. In *Proc. 14th International Symposium on Advances in Geographic Information Systems*, pages 19–26, 2006.

[7] M. de Berg, E. Mumford, and B. Speckmann. On rectilinear duals for vertex-weighted plane graphs. *Discrete Mathematics*, 2007 (to appear).

[8] B. Dent. *Cartography - thematic map design*. McGraw-Hill, 5th edition, 1999.

[9] D. Dorling. *Area Cartograms: their Use and Creation*. Number 59 in Concepts and Techniques in Modern Geography. University of East Anglia, Environmental Publications, Norwich, 1996.

[10] J. Dougenik, N. Chrisman, and D. Niemeyer. An algorithm to construct continuous area cartograms. *Professional Geographer*, 37:75–81, 1985.

[11] H. Edelsbrunner and E. Waupotitsch. A combinatorial approach to cartograms. *Computational Geometry Theory and Applications*, 7:343–360, 1997.

[12] S. Fabrikant. Cartographic variations on the presidential election 2000 theme, 2000. `http://www.geog.ucsb.edu/~sara/html/mapping/election/map.html`.

[13] M. Gastner and M. Newman. Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 101(20):7499–7504, 2004.

[14] R. Heilmann, D. Keim, C. Panse, and M. Sips. Recmap: Rectangular map approximations. In *Proc. IEEE Symp. on Information Visualization*, pages 33–40, 2004.

[15] G. Kant and X. He. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theor. Comp. Sci.*, 172:175–193, 1997.

[16] D. Keim, S. North, and C. Panse. Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE Trans. Visu. and Comp. Graphics*, 10:95–110, 2004.

[17] C. Kocmoud and D. House. A constraint-based approach to constructing continuous cartograms. In *Proc. International Symposium on Spatial Data Handling*, pages 236–246, 1998.

[18] K. Koźmiński and E. Kinnen. Rectangular dual of planar graphs. *Networks*, 5:145–157, 1985.

[19] C.-C. Liao, H.-I. Lu, and H.-C. Yen. Compact floor-planning using orderly spanning trees. *J. Algorithms*, 48(2):441–451, 2003.

[20] NCGIA / USGS. Cartogram Central, 2002. `http://www.ncgia.ucsb.edu/projects/Cartogram_Central/index.html`.

[21] J. Olson. Noncontiguous area cartograms. *Prof. Geographer*, 28:371–380, 1976.

[22] E. Raisz. The rectangular statistical cartogram. *Geogr. Review*, 24:292–296, 1934.

[23] B. Speckmann, M. van Kreveld, and S. Florisson. A linear programming approach to rectangular cartograms. In *Proc. International Symposium on Spatial Data Handling*, pages 529–546, 2006.

[24] W. Tobler. Pseudo-cartograms. *The American Cartographer*, 13:43–50, 1986.

[25] M. van Kreveld and B. Speckmann. On rectangular cartograms. *Computational Geometry: Theory and Applications*, 37(3):175–187, 2007.