Mosaic drawings and cartograms

Eurographics Conference on Visualization (EuroVis) 2015
H. Carr, K.-L. Ma, and G. Santucci
(Guest Editors)

Volume 34 (2015), Number 3

Mosaic Drawings and Cartograms

R. G. Cano[1], K. Buchin[2], T. Castermans[2], A. Pieterse[2], W. Sonke[2], B. Speckmann[2]

[1] Institute of Computing, University of Campinas, Brazil, rgcano@ic.unicamp.br
[2] TU Eindhoven, The Netherlands. [k.a.buchin | b.speckmann]@tue.nl, [t.h.a.castermans | a.pieterse | w.m.sonke]@student.tue.nl.

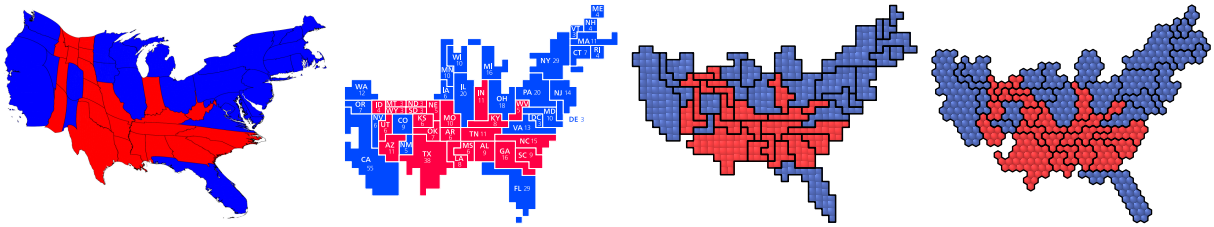

Figure 1: US election 2012, electorial college votes. Diffusion cartogram by Mark Newman (University of Michigan), square mosaic cartogram from Wikipedia, square and hexagonal mosaic cartograms computed by our algorithm.

Abstract

Cartograms visualize quantitative data about a set of regions such as countries or states. There are several different types of cartograms and – for some – algorithms to automatically construct them exist. We focus on mosaic cartograms: *cartograms that use multiples of simple tiles – usually squares or hexagons – to represent regions. Mosaic cartograms communicate well data that consist of, or can be cast into, small integer units (for example, electorial college votes). In addition, they allow users to accurately compare regions and can often maintain a (schematized) version of the input regions' shapes. We propose the first fully automated method to construct mosaic cartograms. To do so, we first introduce* mosaic drawings *of triangulated planar graphs. We then show how to modify mosaic drawings into mosaic cartograms with low cartographic error while maintaining correct adjacencies between regions. We validate our approach experimentally and compare to other cartogram methods.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Thematic maps are an effective way to visualize attributes or concepts which are associated with geographic locations. There is a variety of thematic maps that depict quantitative data about a set of regions such as countries or states: choropleth maps, proportional symbols maps, necklace maps, flow maps, and cartograms. Quantitative geo-referenced data is nowadays globally available, and hence automated methods to create thematic maps and other geo-visualizations for this type of data become increasingly more important.

In this paper we focus on a specific type of cartogram which uses multiples of simple tiles – usually squares or hexagons – to represent regions. In absence of a dedicated name in the literature (they are usually referred to simply as cartograms or – wrongly – as rectangular cartograms), we call such cartograms *mosaic cartograms*. Cartograms show a data value per input region by scaling each region such that its area is proportional to its data value. Mosaic cartograms show data in multiples of tiles, hence the input data must consist of, or be cast into, small integer units.

Mosaic cartograms using squares have been popularized by the New York Times, usually in the context of the US elections, but also to show changing demographics ("The Changing Face of American Catholics" [NYT]). Mosaic cartograms using hexagons are less frequent, good examples are

the illustrations in the "Indices of Deprivation 2010" published by the Leicestershire County Council [Lei] and the UK census maps by Thomas and Dorling [TD04].

Quality criteria. There are several quality criteria for cartograms. One of the most important ones is the *cartographic error* [DCN85, EW97], which 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 depending on the data value to be shown. In a mosaic cartogram a region is represented by an edge-connected set of tiles, which we call a *configuration*. Each configuration must be *simple*, that is, contain no holes. The *mosaic resolution* measures the average number of tiles used per region. We consider the following quality criteria in this paper:

- Average and maximum cartographic error
- Correct adjacencies of configurations: two configurations are adjacent if and only if the corresponding input regions are adjacent
- Shape of the regions
- Relative positions of regions
- Mosaic resolution

It is generally challenging to simultaneously satisfy all criteria well. We decided to enforce correct adjacencies in our algorithm, that is, we produce only mosaic cartograms which have exactly the same configuration adjacencies as the corresponding regions of the input map. There is a clear trade-off between mosaic resolution and recognizability of the map. A low mosaic resolution, that is a small to medium number of tiles per region, allows users to explicitly count tiles and compare regions. A high mosaic resolution makes it easier to preserve shapes and relative positions, and to achieve zero cartographic error. Fortunately, our method is often able to create cartograms with low, or even zero, cartographic error for relatively low mosaic resolutions. We have to note, though, that some rounding error is implicit in our method: whenever the input data does not consist of small integers, it first needs to be converted.

Most handmade mosaic cartograms neither preserve the adjacencies of all input regions nor their shapes. A common semi-automated approach is to take a map or a contiguous area cartogram and to overlay it with a suitable grid. This requires a rather high mosaic resolution, since otherwise rounding errors will easily destroy or create adjacencies. Furthermore, mosaic cartograms created in this way usually do not have zero cartographic error.

Results and organization. We propose the first method to create mosaic cartograms fully automatically. Our cartograms have correct adjacencies, can achieve low mosaic resolution, low or zero cartographic error, and preserve shapes and relative positions of regions rather well. To compute our cartograms we introduce *mosaic drawings* of triangulated planar graphs (Section 3). To compute mosaic cartograms, we start with a mosaic drawing of the (augmented) dual of the input map. We then use an iterative method to grow (or shrink) the configurations according to the input data. While doing so, we maintain the correct adjacencies at all times. The details of our algorithm can be found in Section 4. In Section 5 we validate our approach experimentally using various data sets and maps, also in comparison with the currently best methods for contiguous area and rectangular cartograms. We conclude with a discussion of open problems in Section 6.

2. Related work

Cartograms. The most common cartograms are *contiguous area cartograms*. Here the regions are deformed in such a way that adjacencies are kept. Various methods to construct cartograms of this type have been proposed [DCN85, EW97, GN04, KNP04, KH98, Tob86]. Contiguous area cartograms perform well if the data values are positively correlated to the land areas of the input regions, but producing good cartograms if this is not the case remains a challenge.

A cartogram should convey the values per region, so the user should easily be able to estimate region sizes. The size of regions in contiguous area cartograms is generally hard to judge. To remedy this situation several types of cartograms depict regions using simple geometric shapes. *Circular cartograms* [Dor96] represent each region by a circle which is scaled such that its area is proportional to the data value. Similarly, *rectangular cartograms* [Rai34] represent each region by a rectangle. *Demer cartograms* use squares instead of circles. The shapes (circles, squares, or rectangles) are placed without overlap such that two shapes touch if the corresponding regions are adjacent. Relative positions between regions are preserved as well as possible. Circular and Demer cartograms remain a challenge to produce: the circles and squares are often placed via iterative methods and might be moved far from their original position and in arbitrary directions, making it difficult to find the circle (or square) of a particular region. Algorithms to construct rectangular cartograms [KS07, SvKF06, BSV11, BSV12] use graph-theoretic approaches to preserve adjacencies and relative positions. However, the rectangular shape imposes limitations on the possible layout. Therefore, other related approaches which loosen the adjacency requirements but keep some spatial proximity of the rectangles are *rectangular map approximations* [HKPS04] and *spatial treemaps* [WD08].

The rectangular shape is not very recognizable and furthermore, rectangular cartograms exist only for maps with a dual graph that does not contain separating 3-cycles (no countries with only three neighbors). Mumford et al. [dBMS09] hence initiated the study of rectilinear cartograms where each region is represented by a rectilinear polygon. They showed that for any triangulated planar graph and any set of positive input weights, a rectilinear cartogram of bounded complexity, with correct adjacencies and with zero cartographic error, exists. Alam et al. [ABF*13] proved

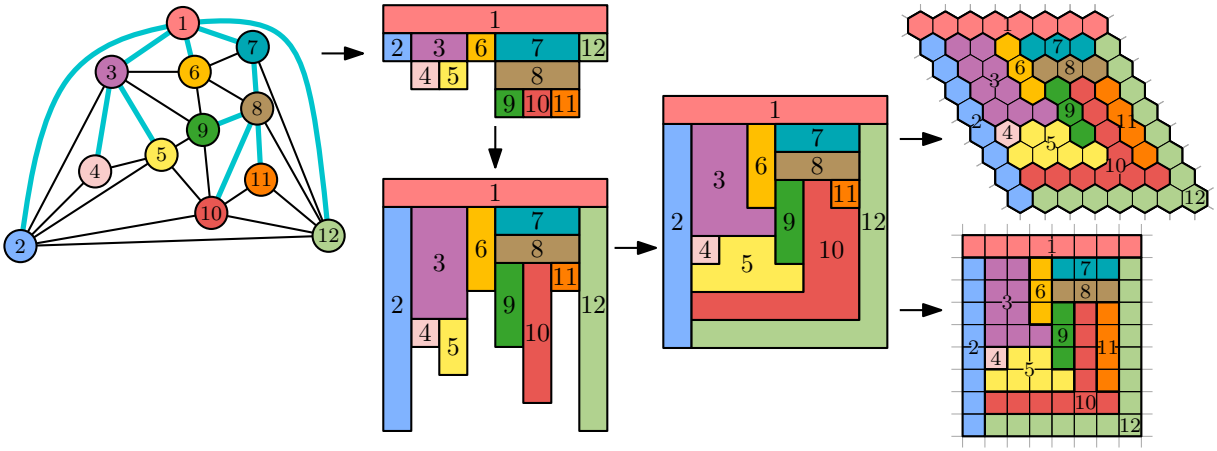

Figure 2: Constructing simple mosaic drawings for square and hexagonal grids via an orderly spanning tree (adapted from [CLL05]).

that complexity 8 (each region has at most 8 vertices) is sufficient. This bound is tight, since complexity 8 is necessary to represent every triangulated planar graph. Mumford et al. [dBMS10] also developed a practical method for rectilinear cartograms, the resulting maps use mostly rectangles and require only few shapes of higher complexity.

Hexagon mosaic maps, closely related in name to mosaic cartograms, are in fact a type of thematic map which uses (parts of) hexagonal tilings as an abstract thematic overlay to show broad patterns [COW92].

3. Mosaic Drawings

We define mosaic drawings for *plane triangulated graphs*, that is, planar graphs with a given embedding where every interior face is a triangle. Mosaic drawings are drawn on a *tiling* of the plane. Of particular interest are the uniform, and especially the regular tilings, although other types of tilings might also result in intriguing drawings. There are three types of regular tilings: the triangular, the square, and the hexagonal tiling. The triangular tiling uses two different rotations of the basic triangular shape and hence is visually a little more complex than the square or the hexagonal tiling.

We call a set of edge-connected tiles of a tiling $\mathcal{T}$ a *configuration*. We say that a configuration C is *simple* if the tiles in C are simply connected (C has no holes). Two configurations C_1 and C_2 are adjacent if and only if there is at least one tile $t_1 \in C_1$ and at least one tile $t_2 \in C_2$ such that t_1 and t_2 are edge-connected. A *mosaic drawing* $\mathcal{D}_\mathcal{T}(G)$ of plane triangulated graph $G = (V, E)$ on $\mathcal{T}$ represents every vertex $v \in V$ by a simple configuration $C(v)$ of edge-adjacent tiles from $\mathcal{T}$ in such a way that two vertices v and u of G are connected by an edge $e = (v, u)$ if and only if the configurations $C(v)$ and $C(u)$ are adjacent. We say that a mosaic drawing is *simple* if the union of all configurations is simply connected, that is, the drawing has no holes (see Fig. 3). Mosaic drawings can be seen as a type of *contact representation*, since they do not draw edges explicitly, but imply them by the contact of the configurations. There is an extensive body of work on contact representations in the Graph Drawing community, surveying it here goes beyond the scope of this paper.

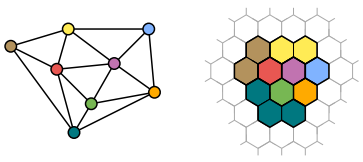

Figure 3: A simple mosaic drawing on a hexagonal tiling.

Square and hexagonal grids. To show that any plane triangulation has a simple mosaic drawing on a square and hexagonal tiling we use a method which is based on *orderly spanning trees* [CLL05] (see Fig. 2). Orderly spanning trees are spanning trees with certain desirable order relations between the nodes. Chiang et al [CLL05] show how to compute a orderly spanning tree ST for a planar triangulation G. They then construct a (vertical) visibility drawing for ST, which is stretched into a 2-visibility drawing of G. Finally they grow horizontal branches to fill up gaps. The result is a square mosaic drawing of G. Since at most three regions meet at each intersection, we can directly shear the same drawing onto a hexagonal grid and so obtain a hexagonal mosaic drawing with the same complexity.

4. Computing Mosaic Cartograms

Our input is a map M represented as a planar subdivision and a graph G_M which is the (weak) dual of M. We assume that

G_M is a planar triangulation, or otherwise simply triangulate G_M. Each vertex v of G_M has a positive weight $w(v)$ which corresponds to the data value that is to be represented in the cartogram. The face of M associated with v is denoted by $F(v)$. A mosaic cartogram for G_M is a simple mosaic drawing of G_M where each configuration $C(v)$ consists of exactly $w(v)$ tiles for each vertex $v \in V$.

The starting point for our approach is a simple mosaic drawing $\mathcal{D}_\mathcal{T}(G_M)$ for a given tiling $\mathcal{T}$. The steps that follow transform $\mathcal{D}_\mathcal{T}(G_M)$ in such a way that the final result is still a simple mosaic drawing. In particular, every operation we execute guarantees that all adjacencies are maintained and all configurations remain connected (although they might sometimes contain holes at an intermediate step).

The transformation process has two main steps. First, given the initial mosaic drawing $\mathcal{D}_\mathcal{T}(G_M)$, we use an iterative method to grow (or shrink) the configurations in $\mathcal{D}_\mathcal{T}(G_M)$ according to the input data. While doing so, we maintain the correct adjacencies at all times. We use so-called *guiding shapes* – configurations which represent the desired final state of each configuration $C(v)$ – to slowly nudge each configuration towards the correct number of tiles and the correct shape. Second, we finalize the cartogram by correcting the size of each configuration while trying to preserve the previously computed shapes as much as possible.

begin `MosaicCartograms`
  Compute mosaic drawing $\mathcal{D}_\mathcal{T}(G_M)$ (Sect. 4.1)
  Move and reshape configurations (Sect. 4.2)
  Correct sizes of configurations (Sect. 4.3)

4.1. Compute mosaic drawing $\mathcal{D}_\mathcal{T}(G_M)$

We compute a simple mosaic drawing $\mathcal{D}_\mathcal{T}(G_M)$ of G_M using the drawing algorithm based orderly spanning trees [CLL05] (see Fig. 2). Unfortunately, the orderly spanning trees created by Chiang et al. [CLL05] have a tendency to "curl inwards". The resulting mosaic drawings often do not retain any of the relative positions of the input nodes and are hence a rather poor starting point for mosaic cartograms (see Fig. 4 left). However, the spanning trees induced by a Schnyder labeling are also orderly spanning trees [MAN05] and lead to mosaic drawings which do capture a significant part of the relative positions of the input (see Fig. 4 right top).

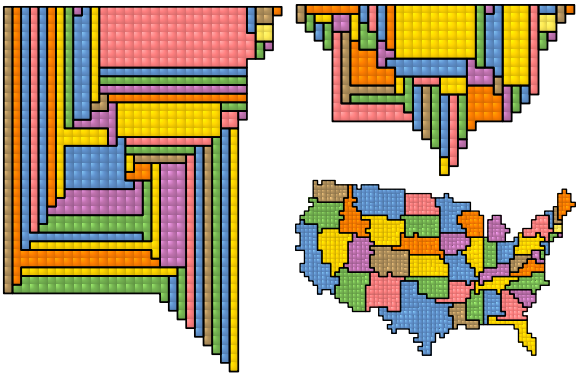

Figure 4: US: mosaic drawing based on Chiang et al. [CLL05] (left) and on Schnyder labeling (top right), area cartogram with 1 square $\approx$ 5000 km^2.

4.2. Move and reshape configurations

Guiding shapes. Given a vertex v of G_M, a guiding shape $S(v)$ is a configuration that represents the desired final state of $C(v)$ (see Fig. 5). $S(v)$ has the exact number of tiles that must be in $C(v)$ and its outline resembles as much as possible the outline of $F(v)$. We compute guiding shapes as follows.

Let $a(\mathcal{T})$ be the area of each tile of $\mathcal{T}$ and let $n_S(v)$ be the desired number of tiles for $S(v)$. We compute a guiding shape for v by first scaling $F(v)$ so that its area becomes $a(\mathcal{T}) * n_S(v)$, which is the target area for $S(v)$. We denote the scaled shape by $F'(v)$. We overlay $F'(v)$ with the tiling and assign to $S(v)$ a set of $n_S(v)$ connected tiles which have the largest intersections with $F'(v)$. We observe that the exact placement of $F'(v)$ has an influence on the result, especially for low-resolution cartograms. Therefore, we repeat this procedure for several slightly translated copies of $F'(v)$ to obtain different candidates for $S(v)$. We then choose the one with minimum symmetric difference with respect to $F'(v)$.

Iteratively moving and reshaping. This iterative procedure gradually transforms the mosaic drawing seeking to minimize the symmetric difference between each configuration and its associated guiding shape.

begin `MoveAndReshape` $(\mathcal{D}_\mathcal{T}(G_M))$
  while *termination condition not reached* **do**
    Transform configurations of $\mathcal{D}_\mathcal{T}(G_M)$
    Move guiding shapes

Figure 6 illustrates the execution of the move and reshape algorithm. Each guiding shape $S(v)$ is placed on the grid in such a way that it contains at least one tile from configuration $C(v)$. The transformation step tries to assign the tiles

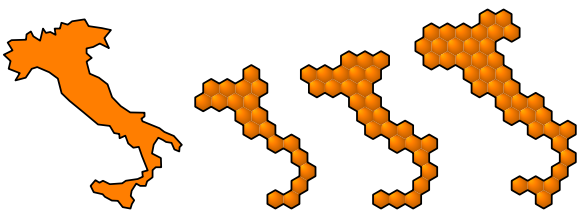

Figure 5: Guiding shapes for Italy with 20, 28 and 40 tiles.

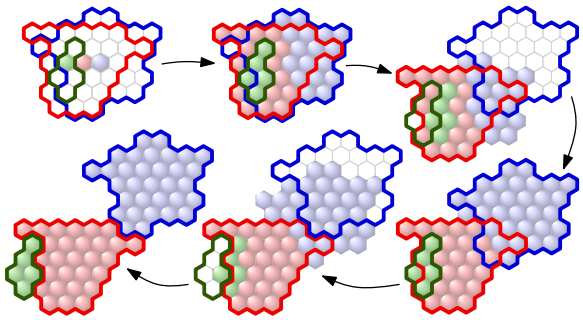

Figure 6: The move and reshape algorithm executed on an instance containing Portugal, Spain and France.

contained in each each $S(v)$ to $C(v)$. Although configurations can never overlap, guiding shapes might. Thus, different configurations often compete for the same tiles (see steps 2 and 4 in Fig. 6). After assigning tiles to each configuration $C(v)$ we move the guiding shapes (steps 3 and 5 in Fig. 6) to minimize overlap.

Reshaping. We reshape configurations with two basic operations. Given a tile $t \in \mathcal{T}$ and a vertex v of G_M, `set (t, v)` assigns t to $C(v)$ (and, consequently, removes t from any other configuration) and `unset (t)` removes t from $\mathcal{D}_{\mathcal{T}}(G_M)$. An operation is *valid* if after its execution all adjacencies remain correct and all configurations remain connected (holes are permitted in this step to give more freedom to the algorithm). Let $\Delta(v)$ be the symmetric difference between $C(v)$ and $S(v)$, and let $\Delta(G_M) = \sum_{v \in G_M} \Delta(v)$. We execute all valid operations that cause $\Delta(G_M)$ to decrease.

Given two vertices u and v of G_M and a tile $t \in C(u)$, the value of $\Delta(G_M)$ remains constant after executing `set (t, v)` if $t \in S(u) \cap S(v)$. To decide whether or not to execute the operation, we consider the symmetric difference normalized by the desired final size of the configurations $\Delta_{\text{norm}}(v) = \Delta(v)/n_S(v)$. We want to reduce the largest normalized distance. Thus, we execute the operation only if it decreases the value of $\max(\Delta_{\text{norm}}(u), \Delta_{\text{norm}}(v))$.

Moving. Once no more reshaping is possible, we move the guiding shapes with a force-directed algorithm. Each guiding shape $S(v)$ is represented by a point $p(v)$ which is initially placed on the barycenter of $S(v)$. We define a time step t_{step} and iteratively compute the force $\vec{f}(v)$ acting on $p(v)$ during each time interval. In each iteration, we update the position of $p(v)$ and every time the tile that contains $p(v)$ changes, we move the guiding shape $S(v)$ accordingly (see FORCEDIRECTEDALGORITHM).

Each point $p(v)$ is affected by attraction and repulsion forces. Two points $p(u)$ and $p(v)$ attract each other if and only if u and v are adjacent in G_M. We define distance$(S(u), S(v))$ as the length of the shortest path in $\mathcal{T}$

begin `ForceDirectedAlgorithm`
  while *not all forces are null and*
      no guiding shapes have been moved **do**
    foreach $v \in G_M$ **do**
      Compute $\vec{f}(v)$
      Let $t \in \mathcal{T}$ be the tile that contains $p(v)$
      $p(v) = p(v) + t_{\text{step}} \cdot \vec{f}(v)$
      Let $t' \in \mathcal{T}$ be the new tile that contains $p(v)$
      if $t \neq t'$ **then**
        Let m be a translation that takes t to t'
        Apply m to all tiles in $S(v)$

that connects any tile in $S(u)$ to any tile in $S(v)$ if $S(u) \cap S(v) = \emptyset$, or 1 otherwise. The direction of the force follows the line that contains the barycenters of $S(u)$ and $S(v)$. Its intensity is given by $100 \cdot \text{distance}(S(u), S(v))$.

A pair of points $p(u)$ and $p(v)$ is affected by repulsion forces whenever there is an overlap between $S(u)$ and $S(v)$. Reversing the construction of the guiding shapes, each tile $t \in S(v)$ can be mapped to a point $q_v(t)$ in the input map M (see Fig. 7). A tile t which is contained in $S(u) \cap S(v)$ is mapped to two different points $q_u(t)$ and $q_v(t)$. The relative position of $q_u(t)$ and $q_v(t)$ determines the direction of the repulsion force: $\sum_{t \in S(u) \cap S(v)} \frac{q_u(t) - q_v(t)}{||q_u(t) - q_v(t)||}$. The intensity of the repulsion force for $p(v)$ depends on the number of overlapping tiles in $S(u)$ and $S(v)$, which we denote by $\text{card}(S(u) \cap S(v))$. Let $k_{uv} = \text{card}(S(u) \cap S(v))/n_S(v)$. The intensity of the repulsion force that acts on $p(v)$ is given by $500 \cdot (1 + k_{uv})$ if u and v are adjacent in G_M, and $200 \cdot (1 + k_{uv})$ otherwise.

To avoid guiding shapes moving too far from their associated configurations, we stop the moving procedure as soon as the first guiding shapes move. Furthermore, to avoid convergence towards a bad position, we randomly change the position of every guiding shape $S(v)$ for which $\Delta_{\text{norm}}(v) \geq 1$ for more than 50 time intervals.

Termination. MOVEANDRESHAPE stops if the system converges to a rest point in which all forces are null. Since we

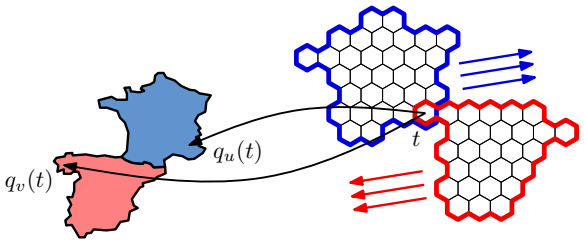

Figure 7: Tile t is mapped to two points in the input map M and the corresponding direction of the repulsion forces.

cannot guarantee convergence, we also set a maximum number of iterations after which the procedure is interrupted.

If the target mosaic resolution is high we run MOVEANDRESHAPE several times with increasing mosaic resolution for the guiding shapes, using the result of the previous run as input for the next. We initially use guiding shapes with an average of 20 tiles per configuration and increase sizes by a factor of $\sqrt{2}$ until the target resolution is reached.

4.3. Correct sizes of configurations

The previous step creates a mosaic drawing $\mathcal{D}_{\mathcal{T}}(G_M)$ with correct adjacencies and configurations $C(v)$ which are as good a match as possible with their guiding shapes $S(v)$. We now show how to correct the size and shape of each configuration to obtain a simple mosaic cartogram with low or zero cartographic error. We say that a tile $t \in \mathcal{T}$ is a *sea tile* if it is not part of $C(v)$ in $\mathcal{D}_{\mathcal{T}}(G_M)$ for any $v \in G_M$. All the other tiles are *land tiles*. We model the correction of configuration sizes as a Minimum Cost Flow Problem (MCFP) which pushes excess land tiles out towards the sea and pulls in extra sea tiles to grow configurations in need.

Let $D = (N,A)$ be a directed graph in which each node $i \in N$ has capacity u_i (the maximum amount of flow allowed through i) and supplies b_i units of flow to the network (if b_i is negative, it actually demands flow from the network). Each arc $(i,j) \in A$ has capacity u_{ij} (the maximum amount of flow allowed through (i,j)) and cost c_{ij} (the cost per unit of flow on (i,j)). A land tile $t \in C(v)$ is a *boundary land tile* if it has at least one neighbor $t' \notin C(v)$. Similarly, a sea tile is a *boundary sea tile* if it has at least one neighbor which is a land tile. All other tiles are called *inner tiles*. The set of boundary tiles of a configuration $C(v)$ is denoted by $B(v)$. The set of boundary sea tiles is denoted by B_{sea}.

Each tile $t \in \mathcal{T}$ is represented by a node n_t in D. Nodes that represent adjacent tiles are connected with incoming and outgoing arcs (see Fig. 8). Let $s \in C(u)$ and $t \in C(v)$ be two adjacent land tiles. A unit of flow from n_s to n_t represents the operation of adding tile s to $C(v)$ and removing it from $C(u)$. This is equivalent to performing the operation `set` (s, v) (see Sect. 4.2). If s is a land tile and t is a sea tile, adding s to the sea is the same as executing the operation `unset` (s).

If s and t belong to the same configuration, any flow between the corresponding vertices is irrelevant. That is, we do not need to represent inner tiles in D. Moreover, since flow on an arc is analogous to the operations `set` and `unset`, we add an arc to D only if the corresponding operation is valid. Finally, a tile t can be assigned only to one unique configuration. Thus, we allow at most one unit of flow through node n_t and any incident arcs also have a capacity of one.

We also add one supply node n_v per configuration $C(v)$. This supply node encodes the amount of tiles to be added to or removed from $C(v)$. Because of the capacity restrictions we impose on boundary nodes, it is possible that no feasible flow exists. Hence we add a "feasibility node" with infinite capacity and arcs with infinite cost to and from every other node. We can now describe all nodes and arcs of D:

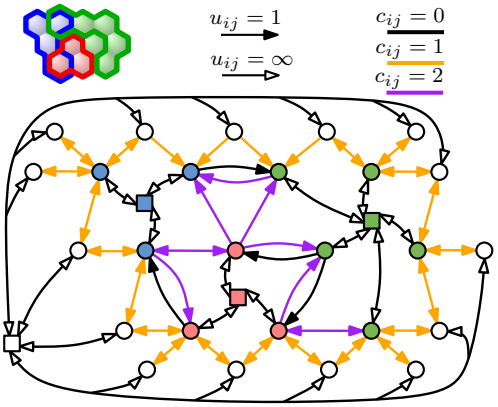

Figure 8: A mosaic drawing, guiding shapes as overlay, and the associated MCFP instance. The feasibility node is omitted for simplicity. Boundary nodes are represented by circles and supply nodes by squares. Sea nodes are white. The supplies are $b_{\text{red}} = b_{\text{green}} = 0$, $b_{\text{blue}} = -1$ and $b_{n_{\text{sea}}} = 1$.

- Boundary nodes: a node n_t for each boundary tile t with capacity $u_{n_t} = 1$ and supply $b_{n_t} = 0$;
- Supply nodes: a node n_v for each vertex v of G_M and a node n_{sea} for the sea. All supply nodes have infinite capacity. Let $n_C(v)$ be the number of tiles in $C(v)$ and $n_S(v)$ be the number of tiles in $S(v)$. Given a vertex v of G_M, the supply of n_v is $b_{n_v} = n_C(v) - n_S(v)$. The supply of n_{sea} is $b_{n_{\text{sea}}} = \sum_{v \in G_M} n_S(v) - n_C(v)$.
- Feasibility node: one node n_{feas} with infinite capacity and zero supply to guarantee that the instance will always have a feasible flow.

- Given two boundary nodes n_s and n_t, we add arcs (n_s, n_t) and (n_t, n_s) if and only if s and t are not in the same configuration and if the corresponding `set`/`unset` operations are valid. Both arcs have capacity of one (costs are discussed below).
- Given a supply node n_v and a boundary node n_t, we add arcs (n_v, n_t) and (n_t, n_v) if and only if $t \in C(v)$. Both arcs have infinite capacity and zero cost.
- Given any node n of D, we add arcs (n, n_{feas}) and (n_{feas}, n), both with infinite capacity and infinite cost.

Since all arcs incident to n_{feas} have infinite cost, they will be used only as a last resort. But the capacity restrictions we impose on boundary nodes often force the use of some of these arcs. After solving the flow problem we execute the operations according to the flow on the arcs between boundary nodes. If the cartographic error is greater than zero, we rebuild the model with the modified drawing and repeat until

either we obtain a cartogram with zero cartographic error, or the flow model cannot make any more changes.

We want to prevent the model from deforming the shapes of the configurations. If we need to choose a tile t to be added to some configuration $C(v)$, we would like to choose it such that $t \in S(v)$ (or, if not possible, not too far from $S(v)$). Similarly, if we need to remove a tile t from $C(v)$, we would like to choose one that is not contained in $S(v)$ (or, if not possible, not too deep inside $S(v)$). Hence we assign costs to the arcs between boundary nodes to encourage this behavior. Specifically, for a tile $t \notin S(v)$ and a guiding shape $S(v)$, we define distance$(t, S(v))$ as the length of a shortest path in $\mathcal{T}$ from t to any tile of $S(v)$. For $t \in S(v)$, we define depth$(t, S(v))$ as the length of a shortest path in $\mathcal{T}$ from t to any tile $t' \notin S(v)$. Let $s \in C(u)$ and $t \in C(v)$ be two adjacent boundary tiles. The cost of the arc (n_s, n_t) is initially zero. If $s \in S(u)$, we add depth$(s, S(u))$ to the cost, otherwise we subtract distance$(s, S(u))$. If $t \in S(v)$, we subtract depth$(t, S(v))$ from the cost, otherwise we add distance$(t, S(v))$.

5. Experimental Evaluation

In this section we discuss the results of our algorithm, also in comparison with other types of cartograms. We created a proof-of-concept implementation in Java. Computing the small cartograms takes less than a minute, the larger, high resolution ones, take 10-15 min.

The mosaic resolution has a clear influence on the look & feel of the cartograms. In Fig. 9 we show three hexagonal mosaic cartogram of the population of continental Europe (including the European parts of Russia). The leftmost cartogram has the lowest possible resolution to include all countries. It is rather schematic and the shapes of small countries are not recognizable. As the resolution increases, all shapes improve. However, individual hexagons are not identifiable (let alone countable) in the highest resolution cartogram. The middle realizes a reasonable trade-off between shape and resolution.In the limit – if the mosaic resolution tends to infinity – mosaic cartograms essentially turn into contiguous area cartograms.

Our algorithm often produces mosaic cartograms with zero cartographic error even for low mosaic resolutions. Of course it is interesting to see how low we can go. In Fig. 10 we show three hexagonal mosaic cartograms of the area of the contiguous Unites States. The top figure is the smallest resolution we can use without losing the smallest state. Since the East Coast contains many of the smallest states, we remove standard federal regions $I - -IV$. Now Indiana is the smallest state. If we use 1 hexagon for Indiana, then our algorithm is not able to solve without error. In particular, in Fig. 10 bottom right all adjacencies are still correct, but 4 hexagons are allocated incorrectly. Both Oregon and Montana would like an extra hexagon, and Oklahoma and Missouri would like one hexagon less. If we give 2 hexagons to Indiana, then our algorithm finds a solution without error.

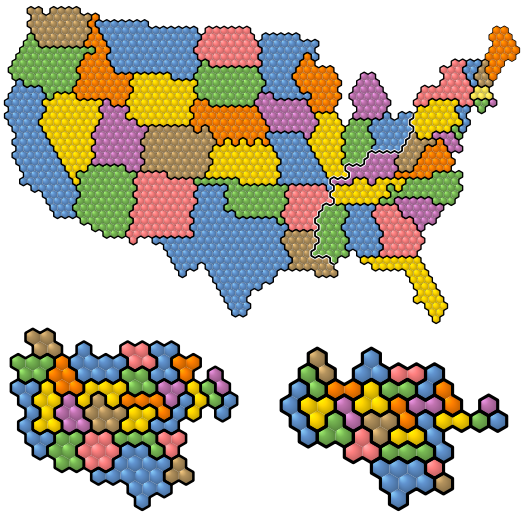

Figure 10: US area. All contiguous states, 1 hexagon $\approx$ Rhode Island $\approx$ 4001 km^2, 1928 hexagons (top), all contiguous states with standard federal regions I–IV removed, 2 resp. 1 hexagons $\approx$ Indiana $\approx$ 94326 km^2, 130 and 64 hexagons (bottom). Cut line indicated in top figure.

Table 1 lists the results of a quantitative comparison of our mosaic cartograms with the currently best methods for con-

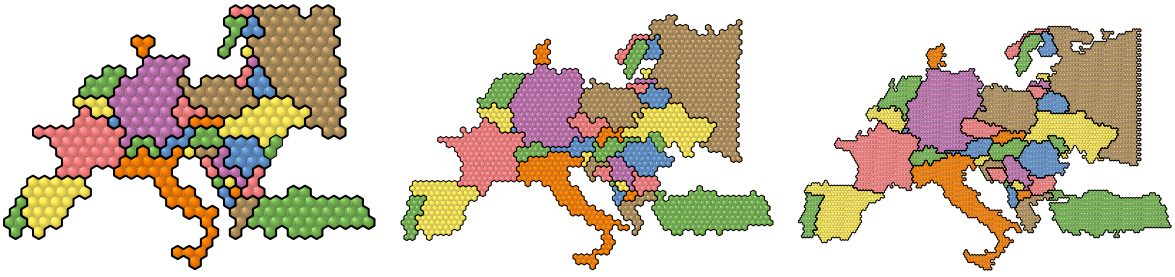

Figure 9: Continental Europe population in 2010: 1 hexagon per 2 million, 377 hexagons (left), 1 hexagon per 500.000, 1529 hexagons (middle), 1 hexagon per 125.000, 5971 hexagons (right).

data set	avg. cartographic error				max. cartographic error				avg. symm. diff.		
	RC	DF	MH	MO	RC	DF	MH	MO	DF	MH	MO
US Census data (2010)											
resident population 2000	0.05	0.13	0.01 (0.001)	0.02 (0.003)	0.34	8.64	0.50 (1)	0.50 (1)	0.55	0.28	0.26
RP $<$ 5 years (%)	0.04	0.07	0.00 (0.000)	0.00 (0.000)	0.16	7.41	0.00 (0)	0.00 (0)	0.40	0.19	0.20
RP total females (%)	0.03	0.07	0.00 (0.000)	0.00 (0.000)	0.16	7.77	0.00 (0)	0.00 (0)	0.41	0.20	0.20
businesses without empl.	0.00	0.13	0.01 (0.001)	0.02 (0.004)	0.03	0.89	0.50 (1)	0.50 (1)	0.56	0.33	0.34
population (per m^2)	0.08	0.23	0.06 (0.004)	0.10 (0.008)	0.65	15.07	1.00 (1)	1.00 (1)	0.81	0.44	0.46
World Factbook data (Europe, 2013)											
GDP real growth rate	0.11	0.42	0.04 (0.040)	0.13 (0.144)	0.60	3.55	0.88 (21)	1.00 (17)	0.93	0.61	0.50
electricity – production	0.00	0.17	0.07 (0.004)	0.24 (0.013)	0.03	1.51	1.00 (1)	4.00 (4)	0.62	0.25	0.51
exports	0.01	0.36	0.15 (0.008)	0.31 (0.018)	0.03	5.99	2.00 (2)	4.00 (4)	0.97	0.41	0.58
population	0.00	0.18	0.02 (0.002)	0.00 (0.000)	0.01	2.90	1.00 (2)	0.00 (0)	0.59	0.19	0.26
current account balance	0.05	0.52	0.17 (0.013)	0.33 (0.042)	0.49	9.35	3.00 (6)	4.00 (15)	1.08	0.45	0.57

Table 1: Average cartographic error, maximum cartographic error, average symmetric difference and mosaic resolution for rectangular cartograms (RC), diffusion cartograms (DF) and mosaic cartograms on the hexagonal grid (MH) and the orthogonal grid (MO). The values for rectangular cartograms are taken from Buchin et al. [BSV12]; the average of 20 runs is used. Mosaic cartograms were constructed with an average number of 20 tiles per region. Values for symmetric differences are normalized by the size of each region.

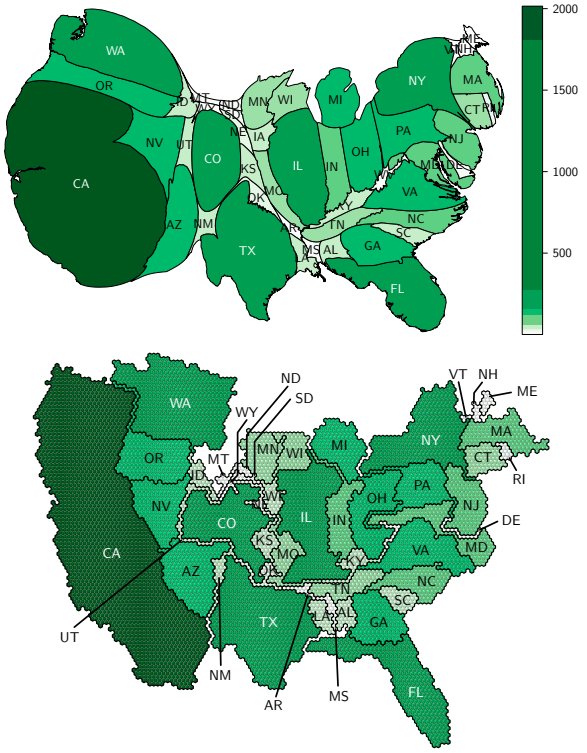

Figure 11: US Starbucks locations in 2009. Diffusion cartogram using the method from [GN04], hexagonal mosaic map with 1 hexagon per location (bottom).

tiguous area [GN04] and rectangular cartograms [BSV12]. The data sets are a subset of those used in [BSV12] ([CIA] and [USA]) and showcase a variety of simple and hard instances. To be consistent with the rectangular cartogram method we represent each region by on contiguous shape, even if it is in fact the union of several disjoint shapes (Michigan or Greece). All three methods keep correct adjacencies. For contiguous area cartograms we measure the quality of shape by computing the average symmetric difference of the (scaled) original regions and the corresponding regions in the cartogram. For mosaic cartograms we measure the average symmetric difference between configurations and their guiding shapes.

Contiguous area cartograms computed with the diffusion method can have very high cartographic error in certain regions and also show a high symmetric difference. The mosaic cartograms were computed with a low mosaic resolution. Hence small regions can create a large cartographic error, for example, by using two instead of one hexagon. Hence we included in brackets the maximum number of "misplaced" hexagons for the worst region (under the maximum), and the total number of "misplaced" hexagons divided by the total number of hexagons (under the averages)

We excluded rectilinear cartograms from the comparison in Table 1. Rectilinear cartogram always have correct adjacencies and zero cartographic error. Furthermore, it is unclear how to measure the quality of the region shapes, given that the rectilinear cartograms use mostly rectangles, with some thin connectors to guarantee correct adjacencies. In Fig. 12 we show the same data sets in all four methods to allow for visual comparison.

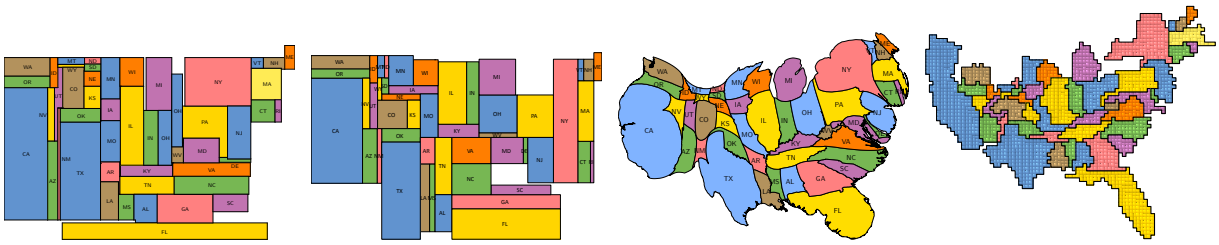

Figure 12: Businesses without paid employees in the US: rectilinear [dBMS10], rectangular [BSV12], contiguous area [GN04], and mosaic cartogram. The mosaic cartogram uses 1 square per 10.000 businesses. See Table 1 for statistics.

Fig. 11 visually compares contiguous area and mosaic cartograms on a challenging data set, namely the number of Starbucks locations in the US in 2009. The contiguous area cartogram exhibits the characteristic bulby and spiky artifacts of the diffusion method, whereas the mosaic cartogram has many well-shaped regions. However, to preserve correct adjacencies, some regions in the mosaic cartogram have to grow tentacles (Oregon), while others are separated by thin channels (Texas and Arizona). Some of the smaller regions also essentially turn into thin, hardly recognizable "snakes". The diffusion cartogram has average cartographic error 0.17 (with the incredibly high maximum of 2.18 in Vermont) and average symmetric difference 72.93 % (with maximum 193.38 %). The mosaic cartogram uses 8080 tiles and has zero cartographic error, with an average symmetric difference of 47 % (with maximum 145 %).

6. Discussion and Future Work

In many cases our method performs well: the cartographic error in many cases is zero or very low and the shapes of regions are close to their original shapes. However, in cases in which shapes need to drastically change to achieve low cartographic error, the method still shows some undesirable behavior. Firstly, in some cases the guiding shapes seem too rigid. For instance in cartograms showing the states of the US the costal states typically have to be enlarged relative to the other states. Our current guiding shapes aim to achieve this by moving these states but keeping their shape fixed. It would, however, seem the better option to allow some non-uniform scaling in these cases. For instance, in Figures 11 and 12 'fattening' California would provide the necessary space to place Oregon and Washington north of California. Secondly, the last step of the algorithm is intended make minor corrections to the sizes of configurations, and can result in long, thin extensions of regions if larger corrections are necessary. This problem in particular occurs if the mosaic resolution is high, and therefore many small mosaics need to be transferred in this step. This can be seen in Figure 11 where several of the smaller regions have long thin extensions to guarantee correct adjacencies. While we hope that this could be improved by less rigid guiding shapes, this seems a rather challenging problem. Note that the diffusion method shows the same issue: For instance, New Mexico (NM) connects to Utah (UT) and Oklahoma (OK) via narrow (essentially, zero-width) corridors. In a mosaic cartogram, however, any such corridor will have at least the width of the tiles, which gives more visual emphasis to this issue.

Overall our method relies on a good result from the iterative moving and reshaping. Since for some data sets using congruent copies of the regions seems too restricted, a crucial question is which deformations are most suitable to add the necessary flexibility, while still maintaining the shape well. Ultimately, the readability of all cartogram methods and the influence of the shape of the regions should be evaluated in an extensive user study.

An obvious direction for future work are other types of tilings, most notably the triangular tiling. Our method in principle extends to any uniform tiling, however, one first has to prove similar results for the existence of mosaic drawings on these tilings. It would also be interesting to develop a method suitable for morphs between different data sets. We cannot simply use our current method with one cartogram as the input for the other, since our guiding shapes move rather abruptly, tearing holes into the intermittent drawings. This behavior is necessary to avoid local minima, but is completely unsuitable for smooth morphs.

Acknowledgements. R. Cano is supported by FAPESP under projects 2012/00673-2 and 2013/23571-3. K. Buchin and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.106 and 639.023.208, respectively.

References

[ABF*13] ALAM M. J., BIEDL T. C., FELSNER S., KAUFMANN M., KOBOUROV S. G., UECKERDT T.: Computing cartograms with optimal complexity. *Discrete & Computational Geometry 50*, 3 (2013), 784–810. 2

[BSV11] BUCHIN K., SPECKMANN B., VERDONSCHOT S.: Optimizing regular edge labelings. In *Proc. 18th International Symposium on Graph Drawing (GD 10)* (2011), LNCS 6502, pp. 117–128. 2

[BSV12] BUCHIN K., SPECKMANN B., VERDONSCHOT S.: Evolution strategies for optimizing rectangular cartograms. In *Proc. 6th International Conference on Geographic Information Science (GIScience)* (2012), vol. 7478 of *Lecture Notes in Computer Science*, Springer, pp. 29–42. 2, 8, 9

[CIA] CIA WORLD FACTBOOK:. accessed 2015/03/06. URL: http://web.archive.org/web/20120313123513/https://www.cia.gov/library/publications/the-world-factbook/. 8

[CLL05] CHIANG Y.-T., LIN C.-C., LU H.-I.: Orderly spanning trees with applications. *SIAM Journal on Computing 34*, 4 (2005), 924Ű–945. 3, 4

[COW92] CARR D. B., OLSEN A. R., WHITE D.: Hexagon mosaic maps for display of univariate and bivariate geographical data. *Cartography and Geographic Information Systems 19*, 4 (1992), 228–236. 3

[dBMS09] DE BERG M., MUMFORD E., SPECKMANN B.: On rectilinear duals for vertex-weighted plane graphs. *Discrete Mathematics 309*, 7 (2009), 1794–1812. 2

[dBMS10] DE BERG M., MUMFORD E., SPECKMANN B.: Optimal BSPs and rectilinear cartograms. *International Journal of Computational Geometry and Applications 20*, 2 (2010), 203–222. 3, 9

[DCN85] DOUGENIK J. A., CHRISMAN N. R., NIEMEYER D. R.: An algorithm to construct continous area cartograms. *Professional Geographer 3* (1985), 75–81. 2

[Dor96] DORLING D.: *Area Cartograms: their Use and Creation*, vol. 59 of *Concepts and Techniques in Modern Geography*. University of East Anglia, Environmental Publications, Norwich, 1996. 2

[EW97] EDELSBRUNNER H., WAUPOTITSCH E.: A combinatorial approach to cartograms. *Computational Geometry: Theory and Applications 7* (1997), 343–360. 2

[GN04] GASTNER M., NEWMAN M.: Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences of the United States of America (PNAS) 101*, 20 (2004), 7499–7504. 2, 8, 9

[HKPS04] HEILMANN R., KEIM D. A., PANSE C., SIPS M.: Recmap: Rectangular map approximations. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS)* (2004), pp. 33–40. 2

[KH98] KOCMOUD C., HOUSE D.: A constraint-based approach to constructing continuous cartograms. In *Proceedings International Symposium on Spatial Data Handling (SDH)* (1998), pp. 236–246. 2

[KNP04] KEIM D., NORTH S., PANSE C.: Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE Transactions on Visualization and Computer Graphics 10* (2004), 95–110. 2

[KS07] KREVELD M. V., SPECKMANN B.: On rectangular cartograms. *Computational Geometry: Theory and Applications 37*, 3 (2007), 175–187. 2

[Lei] URL: http://www.lsr-online.org/uploads/id2010---headline-results-080411.pdf. 2

[MAN05] MIURA K., AZUMA M., NISHIZEKI T.: Canonical decomposition, realizer, Schnyder labeling and orderly spanning trees of plane graphs. *International Journal of Foundations of Computer Science 16*, 1 (2005), 117–141. 4

[NYT] URL: http://www.nytimes.com/interactive/2008/04/13/us/20080412_CATHOLIC_GRAPHIC.html. 1

[Rai34] RAISZ E.: The rectangular statistical cartogram. *Geographical Review 24* (1934), 292–296. 2

[SvKF06] SPECKMANN B., VAN KREVELD M., FLORISSON S.: A linear programming approach to rectangular cartograms. In *Proceedings 12th International Symposium on Spatial Data Handling (SDH)* (2006), pp. 527–546. 2

[TD04] THOMAS B., DORLING D.: Advances in the human cartography of the uk. *The Cartographic Journal 41*, 2 (2004), 109–115. 2

[Tob86] TOBLER W.: Pseudo-cartograms. *The American Cartographer 13* (1986), 43–50. 2

[USA] USA CENSUS BUREAU:. accessed 2015/03/06. URL: http://web.archive.org/web/20111020043143/http://quickfacts.census.gov/qfd/download/DataSet.txt. 8

[WD08] WOOD J., DYKES J.: Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics 14* (2008), 1348–1355. 2