On Rectangular Cartograms

Marc van Kreveld

Bettina Speckmann

institute of information and computing sciences, utrecht university

technical report UU-CS-2004-040

www.cs.uu.nl

On Rectangular Cartograms

Marc van Kreveld[1] Bettina Speckmann[2]

[1] Institute for Information and Computing Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
marc@cs.uu.nl
[2] Department of Mathematics and Computer Science, TU Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
speckman@win.tue.nl

Abstract

A rectangular cartogram is a type of map where every region is a rectangle. The size of the rectangles is chosen such that their areas represent a geographic variable (e.g., population). Good rectangular cartograms are hard to generate: The area specifications for each rectangle may make it impossible to realize correct adjacencies between the regions and so hamper the intuitive understanding of the map.

We present the first algorithms for rectangular cartogram construction. Our algorithms depend on a precise formalization of region adjacencies and build upon existing VLSI layout algorithms. Furthermore, we characterize a non-trivial class of rectangular subdivisions for which exact cartograms can be computed efficiently. An implementation of our algorithms and various tests show that in practice, visually pleasing rectangular cartograms with small cartographic error can be generated effectively.

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 fact, a cartogram is actually able to visualize more than one geographic variable at a time: it can depict a second geographic variable by shades of color [4, 14]. Cartograms are also called *value-by-area maps*.

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. Globally speaking, there are four types of cartogram. The standard type—also referred to as contiguous area cartogram—has deformed regions so that the desired sizes can be obtained and the adjacencies kept. Algorithms for such cartograms were given, among others, by Tobler [18], Dougenik et al. [6], Kocmoud and House [11], Edelsbrunner and Waupotitsch [7], Keim et al. [10]. The second type of cartogram is the non-contiguous area cartogram [8, 15]. The regions have the true shape, but are scaled down and generally do

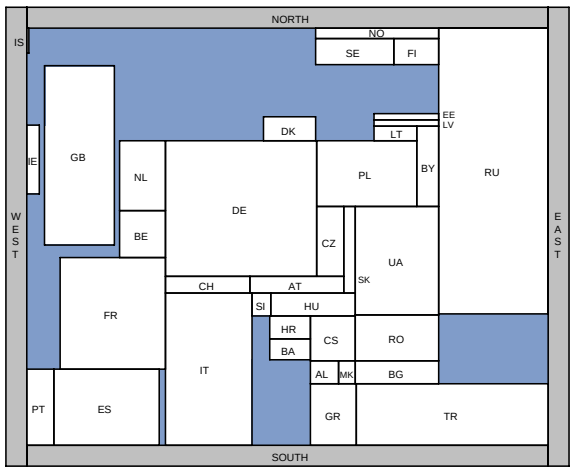

Figure 1: The population of Europe (country codes according to the ISO 3611 standard).

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 [5]. The fourth type of cartogram is the rectangular cartogram introduced by Raisz in 1934 [16]. 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 exist as well. There regions are rectilinear polygons with a small number of vertices instead of rectangles.

Quality criteria. Whether a rectangular cartogram is good is determined by several factors. One of these is the *cartographic error* [6, 7], 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 of that region, given by the geographic variable to be shown. The following list summarizes all quality criteria:

- Average and maximum cartographic error.
- Correct adjacencies of the rectangles (e.g., the rectangles for Germany and France should be adjacent and the rectangles for Germany and Spain should not be adjacent).
- Maximum aspect ratio.
- Suitable relative positions (e.g., the rectangle for The Netherlands should be West of the one for Germany).

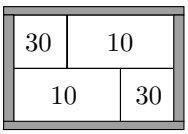

Figure 2: The values inside the rectangles indicate the preferred areas.

For a purely rectangular cartogram we cannot expect to simultaneously satisfy all criteria well. Figure 2 shows an example where correct adjacencies must be sacrificed in order to get a small cartographic error. In larger examples, bounding the aspect ratio of the rectangles also results in larger cartographic error.

Related work. Rectangular cartograms are closely related to *floor plans* for electronic chips and architectural designs. Floor planning aims to represent a planar graph by its *rectangular dual*, defined as follows. A *rectangular partition* of a rectangle R is a partition of R into a set $\mathcal{R}$ of non-overlapping rectangles such no four rectangles in $\mathcal{R}$ meet at the same point. A *rectangular dual* of a planar graph G is a rectangular partition $\mathcal{R}$, such that (i) there is a one-to-one correspondence between the rectangles in $\mathcal{R}$ and the nodes in G, and (ii) two rectangles in $\mathcal{R}$ share a common boundary if and only if the corresponding nodes in G are connected. A *triangle* is a cycle of G consisting of three arcs (a 3-cycle). A cycle C of G divides the plane into an interior and an exterior region. If C contains at least one vertex in its interior and in its exterior than C is called a *separating cycle*.

The following theorem was proven in [2, 12]:

Theorem 1 *A planar graph G has a rectangular dual R with four rectangles on the boundary of R if and only if*

1. *every interior face is a triangle and the exterior face is a quadrangle*
2. *G has no separating triangles.*

Planar graphs that do not have a rectangular dual can still be represented by using other shapes than rectangles. It was shown in [13] that L- and T-shapes in addition to rectangles are always sufficient to represent a planar graph. A rectangular dual is not necessarily unique.

Although every triangulated planar graph without separating triangles has a rectangular dual this does not imply that an error free cartogram for this graph exists. The area specification for

every rectangle, as well as other criteria for good cartograms, may make it impossible to realize. Only a careful trade-off between the various types of errors (for example incorrect areas or incorrect adjacencies) makes it possible to visualize certain data sets as rectangular cartograms.

The only algorithm for standard cartograms that can be adapted to handle rectangular cartograms is Tobler's pseudo-cartogram algorithm [18] combined with a rectangular dual algorithm. Tobler's pseudo-cartogram algorithm starts with a orthogonal grid superimposed on the map, after which the horizontal and vertical lines of the grid are moved to stretch and shrink the strips in between. Hence, if the input to Tobler's algorithm is a rectangular partition, then the output is also a rectangular partition. However, Tobler's method is known to produce a large cartographic error and is mostly used as a preprocessing step for cartogram construction [14]. Furthermore, Tobler himself in a recent survey [19] states that none of the existing cartogram algorithms are capable of generating rectangular cartograms.

Results. We present the first automated algorithms for the computation of rectangular cartograms. We formalize the region adjacencies based on their geographic location and use this information to construct a rectangular subdivision. The steps that lead us from the input data to an algorithmically processable rectangular subdivision are described in Section 2.

We describe three algorithms that compute a cartogram from a rectangular subdivision. The first is an easy and efficient heuristic which we evaluated experimentally. The visually pleasing results of our implementation (which compare favorably with existing hand-drawn or computer assisted cartograms) can be found in Section 6. Secondly, we show how to formulate the computation of a cartogram as a bilinear programming problem (see Section 5).

For our third algorithm we introduce an effective generalization of sliceable layouts, namely *L-shape destructible* layouts. We prove in Section 3 that the coordinates of an L-shape destructible layout are uniquely determined by the specified areas (if an exact cartogram exists at all for a specific set of area values). The proof immediately implies an efficient algorithm to compute an exact (up to an arbitrarily small error) cartogram for subdivisions that arise from actual maps.

2 Algorithmic Outline

Assume that we are given an administrative subdivision into a set of regions. The regions and adjacencies can be represented by nodes and arcs of a graph F, which is the face graph of the subdivision.

1. Preprocessing: To satisfy the conditions of Theorem 1 we have to ensure that the face graph F is triangulated and contains no internal nodes of degree less than four. F is in most cases already triangulated (except for its outer face). We triangulate any remaining non-triangular faces (for example the face formed by the nodes for Nevada, Utah, New Mexico, and Arizona). Our algorithms consider every possible triangulation of non-triangular faces, since each triangulation leads to a different rectangular dual.

It remains to preprocess internal nodes of degree less than four. There are several places on a country map of the world where a country does not have enough neighbors to allow rectangular faces only. For example, Luxembourg is adjacent only to Belgium, Germany, and France. Hence, in the adjacency graph, Belgium, Germany, and France form a separating triangle that separates Luxembourg from the other European countries. In this case, the most reasonable solution is to treat the union of Belgium and Luxembourg as a single region with summed area value and later decide which part of the lower right should be the rectangle for Luxembourg. Belgium will become L-shaped. Similar examples on the world map are Paraguay, Malawi, and Burundi.

Another problem are countries adjacent to only two other countries. These two adjacent countries must then have two adjacencies, one to either side of the country which they enclose. Examples are Mongolia, Andorra, Nepal, Moldova, and Bhutan. In these cases we also remove

the 2-adjacency country from the map and add its area to one of its neighbors. It will be inserted as a rectangle inside that neighbor after the rectangular cartogram algorithm has finished. The neighbor will become C-shaped. There is no guarantee that the insertion can preserve the correct adjacencies and areas.

Yet another problem are countries that are surrounded completely by one neighbor. Examples are Lesotho inside South Africa and the Bundesland Berlin inside Brandenburg. This case is easy to handle: we add the area of the country to its only neighbor, run the rectangular cartogram algorithm, and insert a rectangle of the correct size in the middle. The surrounding country will become O-shaped.

Finally, there are cases where regions are disconnected, like the Bundesland Bremen in Germany. We can choose one of its components to be its representative in the rectangular cartogram.

2. Directed edge labels: Any two nodes in the face graph have at least one direction of adjacency which follows naturally from their geographic location (for example, The Netherlands lies clearly West of Germany). While in theory there are four different directions of adjacency any two nodes can have, in practice only one or two directions are reasonable (for example, Germany can be considered to lie West or North of Austria). We employ a simple heuristic to extract the possible directions of adjacency from the administrative subdivision: we consider the line through the centers of mass of the two regions and let its orientation determine the directions.

Our algorithms go through all possible combinations of direction assignments and determine which one gives a correct or the best result. While in theory there can be an exponential number of options, in practice there is often only one natural choice for the direction of adjacency between two regions. We call a particular choice of adjacency directions a *directed edge labeling*.

Observation 2 *A face graph F 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.*
2. *when traversing the neighbors of any 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.*

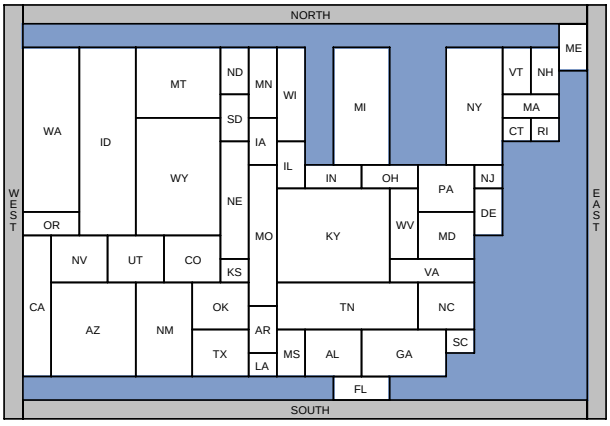

Figure 3: One of 4608 possible rectangular layouts of the US.

For example, a realizable directed edge labeling for the US cannot let Nevada have California to the West, Oregon and Idaho to the North, and Utah and Arizona to the East, because then Nevada would miss a South neighbor.

A realizable directed edge labeling constitutes a *regular edge labeling* for F as defined in [9] 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 regions* that help to preserve the original outline of the subdivision. Then we can employ the algorithm by He and Kant [9] to construct a *rectangular layout*, i.e., the unique

rectangular dual of a realizable directed edge labeling. The output of our implementation of the algorithm by He and Kant is shown in Figure 3.

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. If the answer is negative or if a solution cannot be found, then we still want to compute a cartogram that has a small cartographic error while maintaining reasonable aspect ratios and relative positions.

In this paper we introduce and discuss three methods for computing cartograms from rectangular layouts. We implemented (parts of) our algorithms and we present experimental results in Section 6. In the remainder of this section we give a short overview of our algorithms.

Segment moving heuristic. A simple but efficient heuristic: We iteratively move horizontal and vertical segments of the rectangular layout to reduce the maximum relative error. Additional details can be found in Sections 4 and 6.

Bilinear programming. The computation of a cartogram with the correct adjacencies and minimum relative error can be formulated as a *bilinear programming problem*, which unfortunately is nonconvex. Nevertheless, some non-linear programming methods may still be able to solve certain instances of the problem because the number of variables and constraints is only linear in the number of rectangles. Additional details can be found in Section 5.

L-sequence algorithm. For arbitrary layouts it is currently unknown if one can decide in polynomial time if an exact cartogram exists. But for certain types of rectangular layouts we can compute an exact or nearly exact cartogram. We first determine for a given rectangular layout a *maximal rectangle hierarchy*. The maximal rectangle hierarchy groups rectangles that together form a larger rectangle, as illustrated in Figure 4. It can be computed in linear time [17]. All groups in the hierarchy are independent and we will determine areas separately for each group.

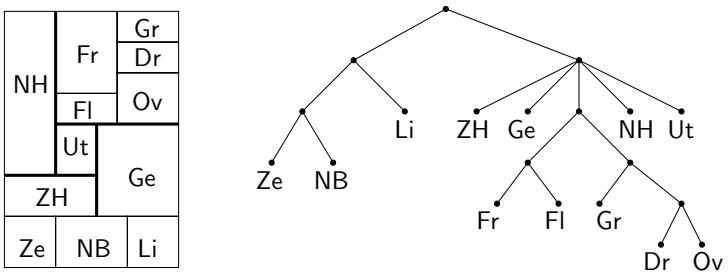

Figure 4: Rectangular layout of the 12 provinces of the Netherlands and a corresponding maximal rectangle hierarchy.

A node of degree 2 in the hierarchy corresponds to a sliceable group of rectangles. If the maximal rectangle hierarchy consists of slicing cuts only, then we can (in a top-down manner) compute the unique position of each slicing cut. Here we may introduce wrong adjacencies, but this will only happen if there is no correct rectangular cartogram.

Nodes of a degree higher than 2 (which is necessarily at least 5) require more complex cuts (see for example the four thick segments in Fig. 4). One of the main contributions of this paper is the characterization of a type of non-sliceable layout for which the coordinates are still uniquely determined by the specified areas. We describe an efficient algorithm to compute a cartogram for *L-shape destructible* layouts (see Section 3 for a precise definition). L-shape destructible layouts are a natural generalization of sliceable layouts with a clear practical value. For example, the rectangular layout of the 48 contiguous states of the US depicted in Figure 3 is not sliceable, but it is L-shape destructible.

Figure 5 shows the smallest non-sliceable layout, as well as the smallest non-L-shape destructible layout. In Section 3 we show that it is difficult to analytically solve the equations that arise from L-shape destructible layouts. But it is possible to produce a cartogram based on one guessed value (coordinate) and decide whether the initial choice was too large or too small in linear time.

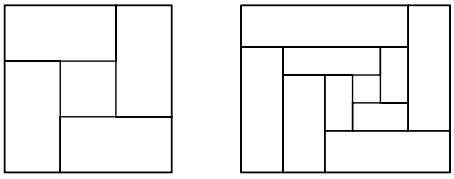

Figure 5: Smallest non-sliceable layout (left), smallest non-L-shape destructible layout (right).

3 L-shape destructible layouts

In this section we show that for certain non-sliceable rectangular layouts we can determine an exact or nearly exact rectangular cartogram efficiently (if one exists). Recall that a rectangular layout is a partition of a rectangle R into a set $\mathcal{R}$ of non-overlapping rectangles. We call a rectangular layout $\mathcal{R}$ *irreducible* if no proper subset of $\mathcal{R}$ (of size > 1) forms a rectangle. Furthermore, we call a rectilinear simple polygon with at most 6 vertices *L-shaped*. We say that an L-shaped polygon is *rooted* at a vertex p if one of its convex vertices is p.

Definition 1 (L-shape destructible) *An irreducible rectangular layout $\mathcal{R}$ of a rectangle R is* L-shape destructible *if there is a sequence starting at a corner s of R in which the rectangles of $\mathcal{R}$ can be removed from R such that the remainder forms an L-shaped polygon rooted at the corner t of R opposite to s after each removal.*

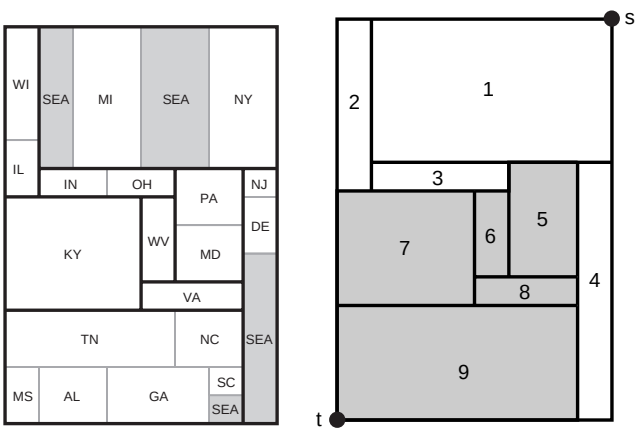

Figure 6: An L-shape destructible layout of the Eastern US. The shaded area shows the L-shaped polygon after the removal of rectangles $1-4$.

Note that the removal sequence is not necessarily unique. However, an incremental algorithm cannot make a bad decision, so the sequence can easily be computed in linear time. See Figure 6 for an example of a removal sequence for an irreducible layout obtained from the maximal rectangle hierarchy of the rectangular layout of the Eastern United States depicted in Figure 3.

The easiest type of non-sliceable but L-shape destructible layouts are *windmill* layouts (see Fig. 7). For a given set of area values $\{A_1, \ldots, A_5\}$ we can compute the (unique) realization analytically because this only requires finding roots of a polynomial of degree 2. We can scale the axes of any instance such that the outer rectangle has coordinates $(0,0)$, $(0,1)$, $(A,0)$, and $(A,1)$, where $A = A_1 + A_2 + A_3 + A_4 + A_5$. Then we get the five equations

Solving these equations with respect to, say, y_1, we obtain a quadratic equation in y_1. A windmill layout always has a realization as a cartogram, independent of the area values, which follows from the proof of Theorem 3.

Already for only slightly more complex types, for example Figure 8, we cannot determine the coordinates exactly anymore, because this requires finding the roots of a polynomial of degree 6. Although the degree of the polynomials suggests that there may be more than one solution, we show that every L-shape destructible layout has at most

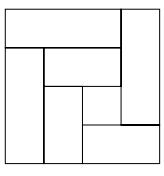

Figure 8: An L-shape destructible layout.

$$
\begin{aligned}
x_1 \cdot y_1 &= A_1 \\
x_2 \cdot (1 - y_1) &= A_2 \\
(x_2 - x_1) \cdot (y_1 - y_2) &= A_3 \\
(x_3 - x_1) \cdot y_2 &= A_4 \\
(x_3 - x_2) \cdot (1 - y_2) &= A_5
\end{aligned}
$$

where also $x_3 = A$.

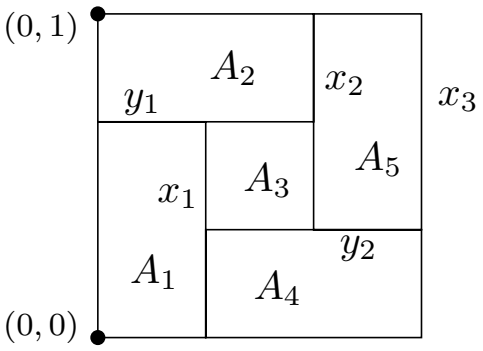

Figure 7: A windmill layout.

one unique realization as a cartogram for a given set of area values. Furthermore we can compute the coordinates of this realization with an easy, iterative method.

Theorem 3 *An L-shape destructible layout with a given set of area values has exactly one or no realization as a cartogram.*

Proof. Without loss of generality we assume that the L-shape destruction sequence begins with a rectangle R_1 that contains the upper left corner of R. We denote the height of R_1 with y_1 (see Fig. 9) and the L-shaped polygon that remains after the removal of R_1 by L_1.

If we consider the edge lengths of L_1 to be a function of y_1 then we observe that these functions are either monotonically increasing in y_1 (denoted by a plus sign near the edge) or monotonically decreasing in y_1 (denoted by a minus sign near the edge). This is true because the area of R_1 is specified. (Note here that the length of the right and bottom edges in Fig. 9 is actually fixed. We simply declare them to be monotonically increasing and decreasing respectively.) In fact, we will prove a much stronger statement: The lengths of the edges of the L-shaped polygons remaining after each removal of a rectangle in an L-shape reduction sequence are monotone functions in y_1.

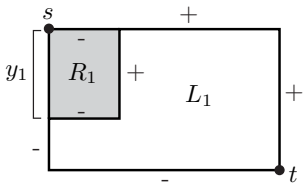

Figure 9: Setting the height of the first rectangle in an L-shape destruction sequence.

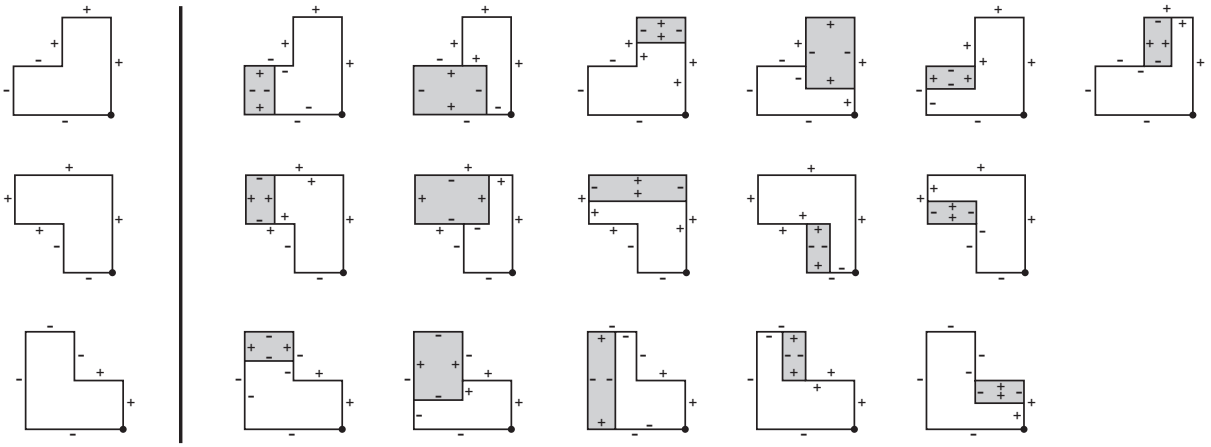

Figure 10: Removing a rectangle from a rooted L-shaped polygon. The plus and minus signs next to an edge indicate that its length is a monotonically increasing or monotonically decreasing function of y_1.

Clearly this statement is true after the removal of R_1. L_1 is of the L-shape type depicted in the upper left corner of Figure 10. We will now argue by means of a complete case analysis that during a destruction sequence only L-shaped polygons of the three types depicted in the left column of Figure 10 can ever occur. Note that the type of an L-shaped polygon also describes which of its edges are monotonically increasing or decreasing in y_1.

For each type of L-shaped rectangle there is only a constant number of possible successors in the destruction sequence, all of which are depicted in Figure 10. We invite the reader to carefully study this figure to convince him or herself that it indeed covers all possibilities. Note that we never need to know exactly which functions describe the edge lengths of the L-shaped polygons—it is sufficient to know that they are always monotone in y_1.

During the destruction sequence it is possible that we realize that our initial choice of y_1 was incorrect. Every rectangle R_i should be placed according to exactly one of the layouts depicted in Figure 10. If the area of R_i is too large or too small for it to be placed in the proper layout then we can infer from the signs of the edges whether y_1 needs to be increased or decreased to achieve correct adjacencies. For example, assume that a rectangle R_i should be placed according to the layout of the second case of the top row of Figure 10 but its area is too small and it would in fact be placed according to the layout of the first case of of the top row. Then the signs of the edges show that y_1 must be increased to obtain correct adjacencies. Should no further increase of y_1 be possible, then we know that the correct adjacencies can not be realized for this particular layout and area values. All other cases can be treated in a similar manner.

As soon as the L-shaped polygon consists of only two rectangles we can make a decision concerning our initial choice of y_1. The edge lengths of one of the two rectangles are either both monotonically increasing or monotonically decreasing. A comparison of its current area with its prescribed area tells us if we have to increment or decrement y_1.

A binary search on y_1 either leads to a unique solution or we recognize that the rectangular layout can not be realized as a cartogram with correct areas. □

4 Segment moving heuristic

A very simple but effective heuristic to obtain a rectangular cartogram from a rectangular layout is the following. Consider the maximal vertical segments and maximal horizontal segments in the layout, for example the vertical segment in Figure 3 that has Kentucky (KY) to its left and West Virginia (WV) and Virginia (VA) to its right. This segment can be moved a little to the left, making Kentucky smaller and the Virginias larger, or it can be moved to the right with the opposite effect.

The segment moving heuristic loops over all maximal segments 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, we have no proof that the method reaches the global optimum or that it even converges.

The segment moving heuristic has some important advantages: (*i*) it can be used for any rectangular layout, (*ii*) one iterative step for all maximal segments takes $O(n)$ time, (*iii*) no area needs to be specified for sea rectangles, (*iv*) a bound on the aspect ratio can be specified, and (*v*) adjacencies between the rectangles can be preserved, but need not be. Not preserving adjacencies can help to reduce cartographic error.

To preserve adjacencies, we have to make sure that two vertical segments incident to the same horizontal segment do not swap in x-order. For example, the vertical segment between Kentucky and the Virginias (Fig. 3) should not go further left than the segment bounding Kentucky to its left, but also no further than the segment between Indiana and Ohio. Otherwise, Indiana and West Virginia become adjacent. In this case we still have a rectangular partition, but it yields a rectangular cartogram with false adjacencies. We examine the cartograms produced by the

segment moving heuristic with correct and false adjacencies in Section 6.

5 Bilinear programming

Once a rectangular layout is fixed, the rectangular cartogram problem can be formulated as an optimization problem. The variables are the x-coordinates of the vertical segments and the y-coordinates of the horizontal segments. For proper rectangular layouts (no four-rectangle junctions, inside an outer rectangle) with n rectangles, there are $n-1$ variables. The area of each rectangle is determined by four of the variables.

We can formulate the minimum error cartogram problem as a *bilinear program*: the rectangle constraints for a rectangle R are:

$$(x_j - x_i) \cdot (y_l - y_k) \geq (1 - \epsilon) \cdot A_R \,,$$
$$(x_j - x_i) \cdot (y_l - y_k) \leq (1 + \epsilon) \cdot A_R$$

where x_i and x_j are the x-coordinates of the left and right segment, y_k and y_l are the y-coordinates of the bottom and top segment, A_R is the specified area of R, and ϵ is the cartographic error for R. An additional $O(n)$ linear constraints are needed to preserve the correct adjacencies of the rectangles (e.g., along sliceable cuts). Also, bounded aspect ratio (height-width) of all rectangles can be guaranteed with linear constraints. The objective function is to minimize ϵ. The formulation is a quadratic programming problem with quadratic constraints that include non-convex ones, and a linear optimization function. Since there are no x_i^2 or y_j^2 terms, only $x_i y_j$ terms, it is a bilinear programming problem. Several approaches exist that can handle such problems, but they do not have any guarantees [1].

6 Implementation and experiments

We have implemented the segment moving heuristic and tested it on several data sets. The main objective was to discover whether rectangular cartograms with reasonably small cartographic error exist, given that they are rather restrictive in the possibilities to represent all rectangle areas correctly. Obviously, we can only answer this question if the segment moving heuristic actually finds a good cartogram if it exist. Secondary objectives of the experiments are to determine to what extent the cartographic error depends on maximum aspect ratio and correct or false adjacencies. We were also interested in the dependency of the error on the percentage of area used by the sea.

Our layout data sets consist of the 36 countries of Europe, and the 48 contiguous states of the USA. For Europe, we joined Belgium and Luxembourg, and Ukraine and Moldova, because rectangular duals do not exist if Luxembourg or Moldova are included as a separate country. Europe has 16 sea rectangles and the US data set has 9. For Europe we allowed 10 pairs of adjacent countries to be in different relative position, leading to 1024 possible layouts. Of these, 768 correspond to a realizable directed edge labeling. For the USA we have 13 pairs, 8192 possible layouts, and 4608 of these are realizable. In the experiments, all 768 or 4608 layouts are considered and the one giving the lowest average error is chosen as the cartogram.

As numeric data we considered for Europe the *population* and the *electricity production*, taken from [3]. For the USA we considered *population*, *native population*, number of *farms*, and total length of *highways*. The data is provided by the US census bureau in the *Statistical Abstract of the United States*.[1]

Preliminary tests on all data sets showed that the false adjacency option always gives considerably lower error than correct adjacencies. The false adjacency option always allowed cartograms

[1] `http://www.census.gov/statab/www/`

Data set	Sea	Aspect ratio	Ave. error	Max. error
Eu elec.	20%	8	0.071	0.280
Eu elec.	20%	9	0.070	0.183
Eu elec.	20%	10	0.067	0.179
Eu elec.	20%	11	0.065	0.155
Eu elec.	20%	12	0.054	0.137
Eu elec.	10%	10	0.098	0.320
Eu elec.	15%	10	0.076	0.245
Eu elec.	20%	10	0.067	0.179
Eu elec.	25%	10	0.049	0.126

Table 1: Errors for different aspect ratios and sea percentages (correct adjacencies).

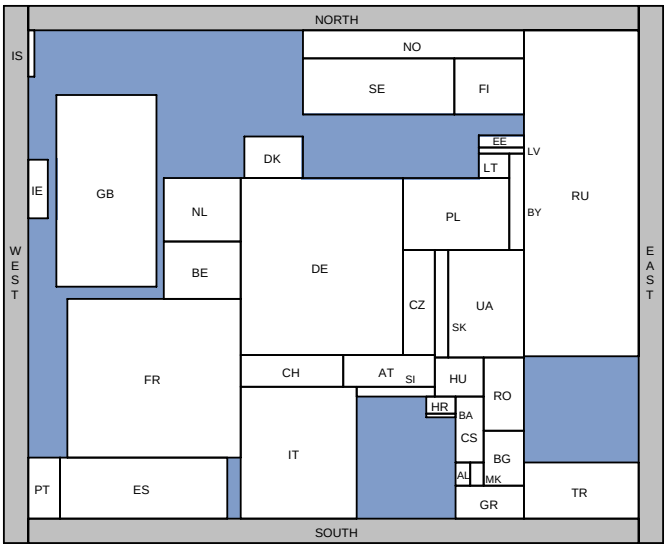

Figure 11: A cartogram depicting the electricity production of Europe.

with average error of only a few percent. A small part of the errors is due to the discrete steps taken when moving the segments. Since cartograms are interpreted visually and show a global picture, errors of a few percent on the average are acceptable. Errors of a few percent are also present in standard, computer-generated contiguous cartograms [6, 7, 10, 11]. We note that most hand-made rectangular cartograms also have false adjacencies and that aspect ratios of more than 20 can be observed.

Table 1 shows errors for various settings for the electricity production data set. The rectangular layout chosen for the table is the one with lowest average error. The corresponding maximum error is shown only for completeness. In the table we observe that the error goes down with a larger allowed aspect ratio, as expected. For Europe and population (not shown in the table), errors below 0.1 on the average with correct adjacencies were only obtained for aspect ratios greater than 15. The table also shows that a larger sea percentage brings the error down. This is as expected because sea rectangles can grow or shrink to reduce the error of adjacent countries, while a sea rectangle cannot have an error in its area. So, more sea means more freedom to reduce errors. However, sea rectangles should not become so small that they visually (nearly) disappear.

Table 2 shows errors for various settings for two US data sets. Again, we choose the rectangular layout giving the lowest average error. In the US highway data set, aspect ratios above 8 do not seem to decrease the error below a certain value. In the US population data set, correct adjacencies give a larger error than is acceptable. Even an aspect ratio of 40 gave an average

Data set	Adjacency	Aspect ratio	Ave. error	Max. error
US population	false	8	0.104	0.278
US population	false	9	0.085	0.193
US population	false	10	0.052	0.295
US population	false	11	0.030	0.091
US population	false	12	0.022	0.056
US population	correct	12	0.327	0.618
US population	correct	13	0.319	0.608
US population	correct	14	0.317	0.612
US population	correct	15	0.314	0.569
US population	correct	16	0.308	0.612
US highway	correct	6	0.073	0.188
US highway	correct	7	0.059	0.111
US highway	correct	8	0.058	0.101
US highway	correct	9	0.058	0.101
US highway	correct	10	0.058	0.101

Table 2: Errors for different aspect ratios, and correct or false adjacencies. Sea 20%.

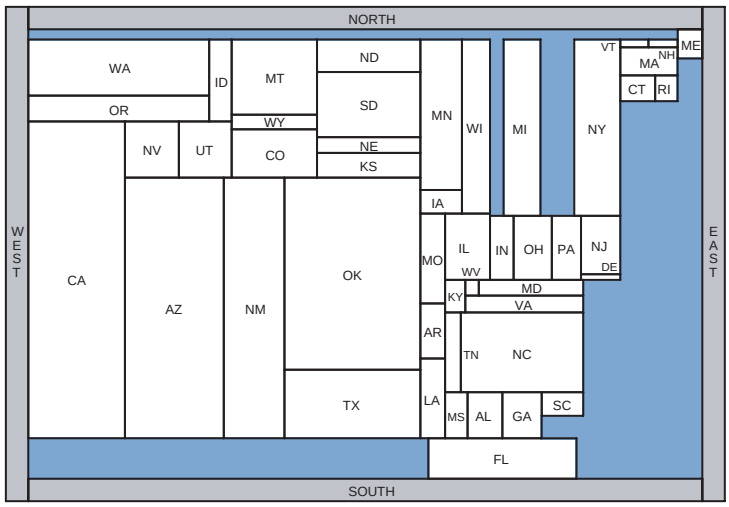

Figure 12: A cartogram depicting the native population of the United States.

error of over 0.3. We ran the same tests for the native population data and again observed that the error decreases with larger aspect ratio. An aspect ratio of 7 combined with false adjacency gives a cartogram with average error below 0.04 (see Fig. 12). Only the highways data allowed correct adjacencies, small aspect ratio, and small error simultaneously.

Figures 12, 13, 14, and 15 show rectangular cartograms of the US. Three of them have false adjacencies, but we can observe that adjacencies are only slightly disturbed in all cases (which is the same as for hand-made rectangular cartograms). The data sets allowed an aspect ratio of 10 or lower to yield an average error between 0.03 and 0.06, except for the farms data. Here an aspect ratio of 20 gives an average error of just below 0.1. Figures 1 and 11 show rectangular cartograms for Europe. The former has false adjacencies and aspect ratio bounded by 12, the latter has correct adjacencies and aspect ratio bounded by 8. The average error is roughly 0.06 in both cartograms.

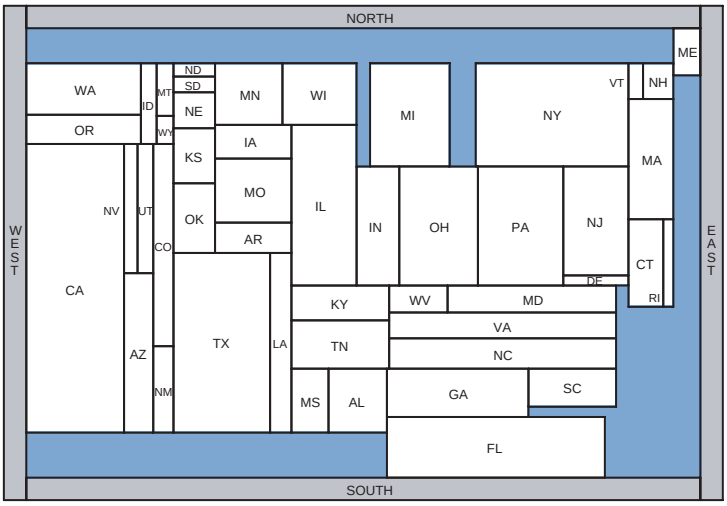

Figure 13: The population of the US.

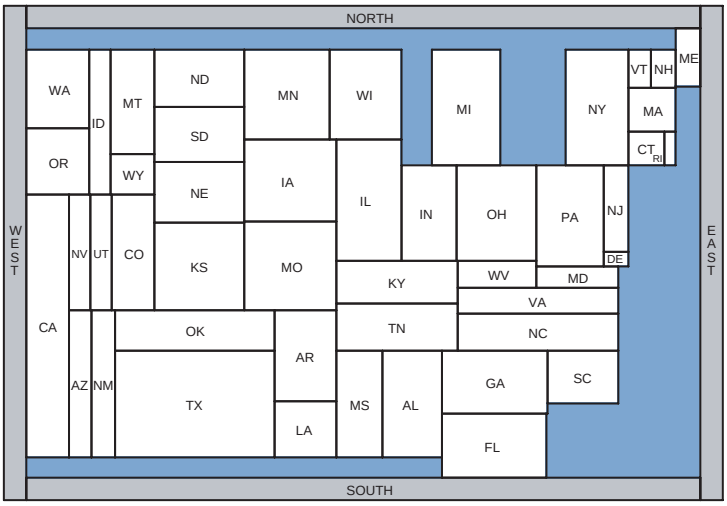

Figure 14: The highway kilometers of the US.

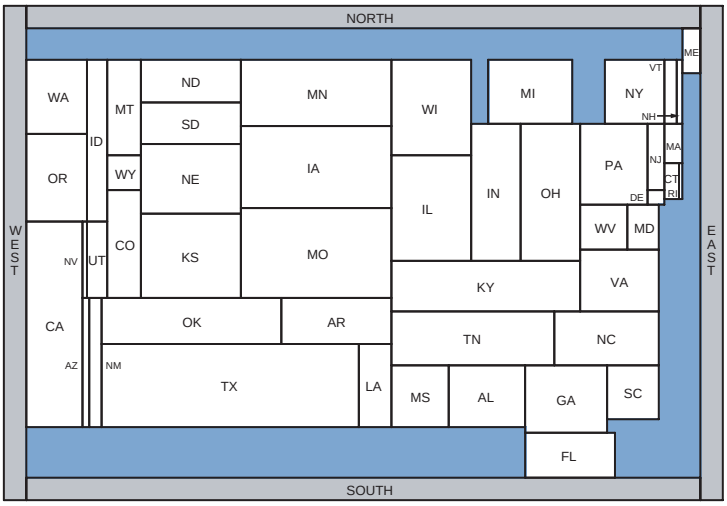

Figure 15: The farms of the US.

7 Conclusion

In this paper we presented the first algorithms to compute rectangular cartograms. We showed how to formalize region adjacencies in order to generate algorithmically processable layouts that represent the positions of the geographic regions well. Then we gave three different methods to generate correct areas for all rectangles. The first one is a segment moving heuristic and the second one is based on bilinear programming. The third method uses an incremental placement order and applies to so called L-shape destructible layouts which generalize sliceable layouts. For these layouts, we showed how to efficiently generate exact or nearly exact cartograms. For all three methods, bounds on the aspect ratio of the rectangles can be respected as well.

We experimentally studied the quality of our segment moving heuristic and showed that it is very effective in producing aesthetic rectangular cartograms with only small cartographic error. Our tests show the dependency of the error on the aspect ratio, correct adjacencies, and sea percentage. The quality of the cartograms generated is comparable to hand-made rectangular cartograms.

Since this is the first paper that discusses algorithms for rectangular cartograms, several extensions are possible and various open problems remain. We showed that L-shape destructible layouts either have no solution or a unique solution. An interesting open question is whether the same holds for non-sliceable, not L-shape destructible layouts. Another open problem is whether rectangular cartogram construction (correct or minimum error) can be done in polynomial time, even if the rectangular layout is given. Finally, since not all face graphs have a rectangular dual, it is also interesting to study cartograms that allow L- and T-shaped regions.

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] *De Grote Bosatlas*. Wolters-Noordhoff, Groningen, 52nd edition, 2001.

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

[5] 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.

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

[7] H. Edelsbrunner and E. Waupotitsch. A combinatorial approach to cartograms. *Comput. Geom. Theory Appl.*, 7:343–360, 1997.

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

[9] 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.

[10] 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.

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

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

[13] C.-C. Liao, H.-I. Lu, and H.-C. Yen. Floor-planning using orderly spanning trees. *J. Algorithms*, 48:441–451, 2003.

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

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

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

[17] S. Sur-Kolay and B. Bhattacharya. The cycle structure of channel graphs in nonslicable floorplans and a unified algorithm for feasible routing order. In *Proc. IEEE International Conference on Computer Design*, pages 524–527, 1991.

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

[19] W. Tobler. Thirty-five years of computer cartograms. *Annals of the Assoc. American Cartographers*, 94(1):58–71, 2004.