A Linear Programming Approach to Rectangular Cartograms

Bettina Speckmann[1], Marc van Kreveld[2], and Sander Florisson[3]

[1] Department of Mathematics and Computer Science, TU Eindhoven, speckman@win.tue.nl
[2] Department of Information and Computing Sciences, Utrecht University, marc@cs.uu.nl
[3] sanderq@xs4all.nl

Abstract. In [26], the first two authors of this paper presented the first algorithms to construct rectangular cartograms. The first step is to determine a representation of all regions by rectangles and the second—most important—step is to get the areas of all rectangles correct. This paper presents a new approach to the second step. It is based on alternatingly solving linear programs on the x-coordinates and the y-coordinates of the sides of the rectangles. Our algorithm gives cartograms with considerably lower error and better visual qualities than previous approaches. It also handles countries that cannot be present in any purely rectangular cartogram and it introduces a new way of controlling incorrect adjacencies of countries. Our implementation computes aesthetically pleasing rectangular and nearly rectangular cartograms, for instance depicting the 152 countries of the World that have population over one million.

Keywords: Rectangular cartogram, algorithm, linear programming.

1 Introduction

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. Cartograms are also called *value-by-area maps* [8, 20]. In a cartogram the sizes of the regions are not the true sizes and hence

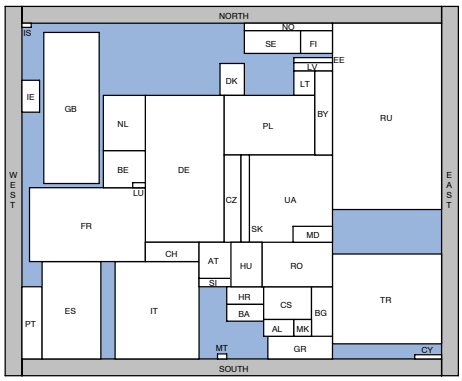

Fig. 1. The population of Europe (country codes according to the ISO 3611 standard).

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 [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 [23]. 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. The regions are rectilinear polygons with a small number of vertices instead of rectangles. Figure 1 shows a population cartogram where all but two countries are rectangular; these two countries are L-shaped.

Quality criteria. Whether a rectangular cartogram is good is determined by several factors. One of these is the *cartographic error* [10, 11], 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).

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.

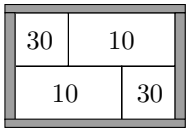

Fig. 2. The values inside the rectangles indicate the preferred areas.

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 that 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, then C is called a *separating cycle*. The following theorem was proven in [2, 18]:

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 [19] that L- and T-shapes in addition to rectangles are always sufficient to represent a planar graph. A rectangular dual is not necessarily unique. When converting a rectangular dual into a cartogram, the rectangles must obtain the specified areas while respecting all quality criteria.

The only algorithm for standard cartograms that can be adapted to handle rectangular cartograms is Tobler's pseudo-cartogram algorithm [24] combined with a rectangular dual algorithm. However, Tobler's method is known to produce a large cartographic error and is mostly used as a preprocessing step for cartogram construction [20]. Tobler states in a recent survey [25] that none of the existing cartogram algorithms are capable of generating rectangular cartograms. Shortly thereafter, the first two authors of this paper presented the first methods for the automated construction of rectangular cartograms [26]. One problem of those methods was the handling of sea regions, and an experimental study of possible solutions was performed in [27].

A few other papers exist that discuss constructions related to rectangular cartograms. Biedl and Genc [3] show NP-hardness of certain orthogonal layout problems with prescribed areas, while Heilmann et al. [14] and Rahman et al. [22] provide algorithms to construct rectangular or orthogonal layouts with prescribed areas. Recently, de Berg et al. [7] show how to construct orthogonal layouts with prescribed areas and correct adjacencies, while using at most a constant number of edges per region. Their result holds for any input graph.

Results. We present a new algorithm for the computation of rectangular cartograms which improves upon the previous ones in all aspects. In particular, our algorithm produces cartograms with a lower cartographic error, a smaller aspect ratio, and with better global shape. Several preprocessing steps are the same as described in [26], but the main computation of obtaining correct areas for all rectangles is significantly different. We also present new variations of the

method by allowing L-shaped regions in the cartogram. Furthermore, we show how to control the degree up to which incorrect adjacencies between countries are allowed. Our new method gives satisfactory rectangular cartograms of all countries of the World, whereas the previous method often does not.

We give the outline of the algorithm from [26] in Section 2, and present the new, linear programming based algorithm—including variations—extensively in Section 3. We implemented the linear programming approach, evaluated its output for different settings, and compared it with the approach from [26], which is reported in Section 4.

2 Algorithmic outline

Assume that we are given an administrative subdivision into a set of regions, like the countries of the World or the States of the USA. 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 and for inner seas that are adjacent to several regions. We triangulate any remaining non-triangular faces, like the face formed by the nodes for Nevada, Utah, New Mexico, and Arizona, and the four-country point in Africa.

Regions with only three neighbors and no adjacency to the seas, like Luxembourg and Burundi, cannot be represented in a rectangular cartogram where all adjacencies are correct. The same is true for regions with only two neighbors, like Mongolia. We discuss these issues further in Section 3.

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 that 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. If two directions are reasonable, we have a so-called *layout option*. While in theory many pairs of adjacent regions can have a layout option, in practice there is often only one natural choice for the direction of adjacency between two regions.

Our algorithm will go through all possible combinations of layout options and determines which one gives a correct or the best result. If the input subdivision has m layout options, we will test 2^m different combinations of adjacency directions. A particular choice of adjacency directions gives rise to a *directed edge labeling*.

Observation 1. *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.*

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 [15] which immediately implies our observation.

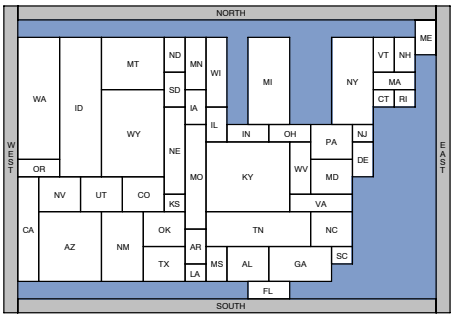

Fig. 3. One of 4608 possible rectangular layouts of the US.

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 this we add four special nodes NORTH, EAST, SOUTH, and WEST. Furthermore, we add additional nodes (*sea nodes*) to F that represent bodies of water and so help to preserve the original outline. For example, our face graph of the World contains one node each for the Black and the Caspian Sea. Larger bodies of water (like the oceans) and bodies of water with a more complex shape (like the Mediterranean Sea) are represented by many sea nodes.

Now we can employ the algorithm by Kant and He [15] to construct a *rectangular layout*, i.e., the unique rectangular dual of a realizable directed edge labeling. Every sea node is represented by a *sea rectangle*. The land and the sea rectangles together form a partitioning of the whole map (a rectangle) into smaller rectangles. The output of our implementation of the algorithm by Kant and He is shown in Figure 3.

4. Area assignment: For a given set of area values and a given rectangular layout we would like to compute a rectangular cartogram that has a small cartographic error, while maintaining reasonable aspect ratios and relative positions. In this step the main information to be conveyed by the cartogram—the rectangle sizes—is determined.

In a previous paper [26] we introduced three methods for computing cartograms from rectangular layouts. One is the simple segment moving heuristic, which incrementally moves segments by fixed steps if this reduces the error. This method was implemented. Secondly, if the rectangular layout is

L-shape destructible (see [26]) we 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 this paper we formulate area assignment as a linear program which takes only the vertical or only the horizontal segments into account. We then alternatingly solve a linear program for the vertical and the horizontal segments. We implemented this approach—using the well-known CPLEX program [6]—and it clearly improves upon the segment moving heuristic. The next section shows how to formulate area assignment as a linear program if either the horizontal or the vertical segments are considered to be at fixed positions.

3 Linear programming

Assume that we are given a rectangular layout $\mathcal{L}$. $\mathcal{L}$ is uniquely determined by the x-coordinates of its maximal vertical segments and the y-coordinates of its maximal horizontal segments. (See for example Fig. 3: the horizontal segment which has Illinois, Indiana, and Ohio above and Kentucky and Wyoming below is a maximal horizontal segment.) The area of a rectangle R of $\mathcal{L}$ is determined by the coordinates of exactly four maximal segments: the y-coordinates of the two horizontal segments bounding it from above and below and the x-coordinates of the two vertical segments bounding it from the left and the right. Similarly, the aspect ratio of a R is also determined by these four segments. Minimizing area error, bounding the aspect ratio, and preserving adjacencies of rectangles can all be formulated using bilinear constraints in the x- and y-coordinates of the maximal segments. These constraints are linear in x if the y's are treated as constants and vice versa.

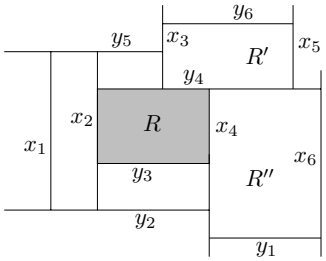

Fig. 4. Linear constraints for rectangle R.

We create constraints for every rectangle R of a given rectangular layout $\mathcal{L}$. The specified area of the region represented by R is denoted by $A_s(R)$. Please refer to Figure 4 for the notation used in the following paragraphs. Note that the horizontal segment with y-coordinate y_4 is simultaneously the top of R and R'', and the bottom of R' and two other rectangles. Hence the constraints associated with several different rectangles will impose restrictions on the value of y_4.

There are four different types of constraints associated with every rectangle R: *error constraints*, *aspect ratio constraints*, *planarity preserving constraints*, and *adjacency preserving constraints*. We explain these constraints in detail below, including the modifications we make for sea rectangles.

Area constraints. The area of R in the layout is $(x_4 - x_2) \cdot (y_4 - y_3)$. This implies that the error of R is

$$\frac{|(x_4 - x_2) \cdot (y_4 - y_3) - A_s(R)|}{A_s(R)} .$$

We introduce a variable $\mathrm{ERR}(R)$ that bounds the error from above, that is

$$\mathrm{ERR}(R) \geq \frac{|(x_4 - x_2) \cdot (y_4 - y_3) - A_s(R)|}{A_s(R)} .$$

Absolute values cannot be used in linear programs, so we replace this constraint by two others:

$$(x_4 - x_2) \cdot (y_4 - y_3) \geq (1 - \mathrm{ERR}(R)) \cdot A_s(R) ,$$
$$(x_4 - x_2) \cdot (y_4 - y_3) \leq (1 + \mathrm{ERR}(R)) \cdot A_s(R) .$$

Note that these constraints are indeed linear if either the x's or the y's are treated as constants.

Aspect ratio constraints. The aspect ratio of R is the maximum of its *height:width* ratio and its *width:height* ratio, so it is always at least 1. Given a maximum aspect ratio D for all rectangles, we obtain the following two constraints for R:

$$(x_4 - x_2) \leq (y_4 - y_3) \cdot D ,$$
$$(y_4 - y_3) \leq (x_4 - x_2) \cdot D .$$

Planarity constraints. These constraints preserve the planarity of the layout. For example, no solution with $x_2 > x_4$ is acceptable, because R would be "inverted", and the layout would no longer be planar. Hence, for every rectangle we require its left segment to be left of its right segment, and its bottom segment to be below its upper segment:

$$x_2 < x_4 \quad \text{and} \quad y_3 < y_4 .$$

Only one of these constraints is in effect in any linear program, because only the x's or only the y's are the variables.

Adjacency constraints. To make sure that rectangles R and R' remain adjacent when vertical segments (x-coordinates) are changed, we require that $x_3 < x_4$. Note that violating this constraint is not disastrous for the final cartogram, because planarity is not affected. Only the adjacency of rectangles is influenced.

Sea rectangles. There are no error constraints or aspect ratio constraints associated with sea rectangles. However, planarity and adjacency constraints must be defined. Adjacency constraints can become less strict. If in Figure 4, both R and R'' are sea rectangles, then the constraint $x_3 < x_4$ can be dropped: the segment that separates R and R'' is not shown in the final cartogram anyway. Finally, we add two new constraints for every sea rectangle R to

make sure that it does not get zero height or width (otherwise a sea may become invisible and opposite regions would visually be adjacent):

$$x_2 < x_4 - d \quad \text{and} \quad y_3 < y_4 - d\,.$$

These constraints guarantee that any sea rectangle has at least some width and height d, where d is some well-chosen constant.

To generate a cartogram that has minimum error, we must minimize $\mathrm{ERR}(R)$ for all rectangles R. The objective function of the linear program therefore is

$$\text{minimize} \sum_{\text{rectangles } R} \mathrm{ERR}(R)\,.$$

This is a linear objective function, and hence we have derived a linear program for minimizing the total or, equivalently, the average error of the rectangles.

Our whole algorithm is as follows. Starting with the rectangular layout produced by the algorithm of Kant and He [15], generate all constraints for the vertical segments and solve a linear program where all x's are variables. Then fix the new x-coordinates and generate all constraints for the horizontal segments and solve a linear program where all y's are variables. Repeat this for a fixed number of iterations, or until no more reduction of error occurs.

Nearly rectangular cartograms. In Section 2 we mentioned the issue of regions with three or fewer neighbors, and the fact that they cannot be represented in a rectangular cartogram with correct adjacencies. One option is to use more general shapes than rectangles, like L-shaped and C-shaped regions. For example, we can place Luxembourg in the lower right corner of Belgium, which then becomes L-shaped. It is straightforward to set up error, aspect ratio, planarity, and adjacency constraints for Belgium and Luxembourg in this case. Hence, we can still use the linear programming approach. Similar examples on the world map are Paraguay, Malawi, and Burundi.

There are also countries that are adjacent to only two other countries, like Nepal and Moldova. Incorporating these can be done by using C-shaped regions. A country like Lesotho inside South Africa can be incorporated by making South Africa O-shaped.

In cases where L-shaped, C-shaped and O-shaped regions appear instead of rectangles, we must adapt the constraints and optimization function for the linear program. For example, in Figure 5, the area of an L-shaped region R is given by $(x_2-x_1)\cdot(y_5-y_1)+(x_4-x_2)\cdot(y_5-y_2)$, and this can be converted into a bilinear constraint including an error term $\mathrm{ERR}(R)$ as before. The changes required are straightforward and we omit them from this paper.

The other option is to maintain a purely rectangular layout and change the correct adjacencies locally. For example, Luxembourg could be made adjacent to the North Sea between Belgium and France (which would no longer be adjacent), or Luxembourg could be made adjacent to the Netherlands, in which case Belgium and Germany are no longer adjacent. If the initial layout

is adapted in this manner, a purely rectangular layout can be made. Regions with only two adjacent regions like Mongolia can be handled in a similar way. An advantage of using only rectangles over using L-shaped, C-shaped, and O-shaped regions besides rectangles is that areas can be estimated better, which important for cartograms.

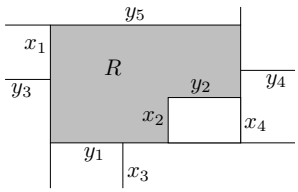

Fig. 5. Linear constraints for L-shaped region R.

Limiting incorrect adjacencies.
The example in Figure 2 shows that requiring correct adjacencies can make rectangular cartograms with low error impossible. Therefore, it is often necessary to sacrifice correct adjacencies. As noted before, not including the adjacency constraints in the linear program leads to rectangular cartograms that are still planar, rectangular layouts, but where two countries may no longer be adjacent, or are adjacent although they are not in reality. Consider the cartogram of Figure 1. Without adjacency constraints and a theme where Norway and Turkey have very high values, it can happen that these two countries become adjacent. Similarly, in Figure 3, North Dakota (ND) and Louisiana (LA) can potentially become adjacent.

So clearly, if we omit all adjacency constraints, the relative positions of regions can be disturbed too much. Therefore, we introduce *levels* of adjacency error and the corresponding linear constraints. For consistency, we call the correct adjacency constraints *level*-0 *adjacency constraints*, like $y_1 < y_2$ in Figure 4. A *level*-1 *adjacency constraint* allows each vertical or horizontal segment to pass at most 1 other vertical or horizontal segment, but no more. An example is $y_1 < y_3$ in Figure 4. There are no other level-1 adjacency constraints in the figure, and also no constraints of even higher level. Figure 6 shows the situation for a level-k adjacency constraint. If k is set to a high value, corresponding to the situation without any adjacency constraints, we speak of *false adjacencies*.

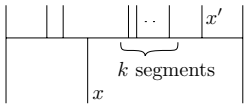

Fig. 6. Illustration of the level-k adjacency constraint $x < x'$.

4 Implementation and test results

The linear programming approach to construct rectangular cartograms has been implemented and tested. The main objective was to test if it produces cartograms that are both visually pleasing and have low error. Also, we were interested in the difference between purely rectangular cartograms and cartograms with L-shaped regions, and we wanted to know how the linear programming approach compares with the segment moving heuristic. Finally, we wanted to investigate if level-1 adjacency constraints provide a good compromise between low error and only mildly incorrect adjacencies.

We used three different subdivisions: the contiguous states of the USA, the countries of Europe with population over 100,000, and the countries of the World with population over 1,000,000. Themes that were tested are population, area, gross domestic product, and total highway length (data was taken from the CIA World Factbook 2005 [5]). The USA has 13 layout options, Europe has 10, and the World 22. For Europe and the World, we considered both purely rectangular layouts and layouts with L-shaped countries.

The rectangular cartograms in the figures have regions that are colored based on their error[1]. Shades of red show that a region is too small and shades of blue show that a region is too large. If the error is below 0.05, the region is white. The lightest shades of red and blue are used for errors between 0.05 and 0.1, and darker shades are used for errors between 0.1 and 0.2, between 0.2 and 0.3, and above 0.3.

During implementation, we made two adaptations to the method as described so far. Test runs of the linear programming approach often gave output where one or two countries had a high error, while all others had no or negligible error. The reason is that the optimization function minimizes the sum of errors, which means that it is equally good to have one country with error 0.4 and another country with error 0.0, as to have those two countries with error 0.2 each. However, we clearly prefer to have two countries with error 0.2, and, more generally, to have low maximum error. This is easily done by minimizing the sum of *squared* errors instead of the sum of errors. CPLEX [6] allows for minimizing the sum of squared errors, even though technically speaking, we no longer have a linear program. All results given in this section have been computed by minimizing the sum of squared errors.

The other adaptation is an efficiency issue. Recall that our algorithm allows for layout options, that is, a region may for instance be either North or East of an adjacent region in a rectangular layout. For the USA subdivision we have 13 of these layout options, implying that we try to generate $2^{13} = 8,192$ different rectangular layouts on which the algorithm is run. Not all rectangular layouts are realizable, so fewer layouts are actually generated. For the World data set, there are 22 layout options, potentially giving over four million layouts. This would make the cartogram computation prohibitively time-consuming.

To deal with this problem we partition the layout options into groups. The idea is that layout options in Europe do not have to be tested in combination with layout options in South America, for instance. So the best layout option for Europe can be determined independently from the best layout option for Europe. We partitioned the 22 layout options into 7 groups of sizes up to 4, which leads to testing only 74 rectangular layouts. This adaptation may result in slightly larger errors in the cartogram, but the time savings are drastic.

Figure 7 shows two cartograms of the World with the theme population. Only countries with population over one million are included. The first map

[1] The electronic version of this paper contains color figures and can be found at `http://www.win.tue.nl/~speckman/pub.html`.

Fig. 7. A purely rectangular cartogram of the World with the theme population, and a corresponding cartogram that includes L-shaped regions.

is based on a purely rectangular layout whereas the second uses L-shaped regions for Moldova, Mongolia, Gambia, Lesotho, Swaziland, Malawi, and Burundi. Both cartograms were made with the linear programming approach. The maximum aspect ratio is 12, the adjacency constraints are of level 1, the overall sea area is 40% of the map, and 50 iterations (linear programs on x and y per layout) were used. The purely rectangular cartogram has slightly higher errors (average 0.003 and maximum 0.099 in China) than the one with L-shaped regions (average 0.002 and maximum 0.063 in China). We observed the same small differences on tests with other data (gross domestic product, total highway length) and other maximum aspect ratios (8, 16, and 20). Usually the cartograms with L-shaped regions have slightly lower errors.

We used basically the same settings to produce a purely rectangular cartogram with the segment moving heuristic. However, false adjacencies are allowed, and 400 iterations were used. The average error is 0.103 and the maximum is 0.534, which is significantly worse than what we obtained with the linear programming approach, despite the fact that adjacencies may be disturbed more. The corresponding cartogram (not shown) is also visually much worse than the cartogram produced by linear programming.

We also compared the segment moving heuristic and linear programming approach on the USA data set with the theme population, and on the Europe data set with the theme highway length, see Table 1. In all cases the maximum aspect ratio was set to 12 and the total sea area to 20%. We observe that the linear programming approach always gives a lower average error. The maximum occurring error can be better for either method. We also observe that for the linear programming approach, inclusion of level-1 adjacency constraints gives a much lower error than correct adjacencies, albeit not as good as false adjacencies. Corresponding cartograms for four cases are given in Figure 8.

We present one more rectangular cartogram related to the FIFA World Cup in 2006. Figure 9 shows all countries or regions with population over one million who participated in the qualification rounds. The qualified 32 countries are shown by their FIFA country code and flag with a fixed size, whereas all other countries are shown in grey by their actual area. For qualified

Method	Data	Adjacencies	Av. error	Max. error
LP	US pop.	correct	0.086	0.873
LP	US pop.	level 1	0.032	0.189
LP	US pop.	false	0.013	0.107
Segment moving	US pop.	correct	0.204	0.443
Segment moving	US pop.	false	0.018	0.058
LP	EU highway	correct	0.022	0.166
LP	EU highway	level 1	0.000	0.001
LP	EU highway	false	0.000	0.000
Segment moving	EU highway	correct	0.099	0.375
Segment moving	EU highway	false	0.052	0.151

Table 1. Linear programming versus segment moving.

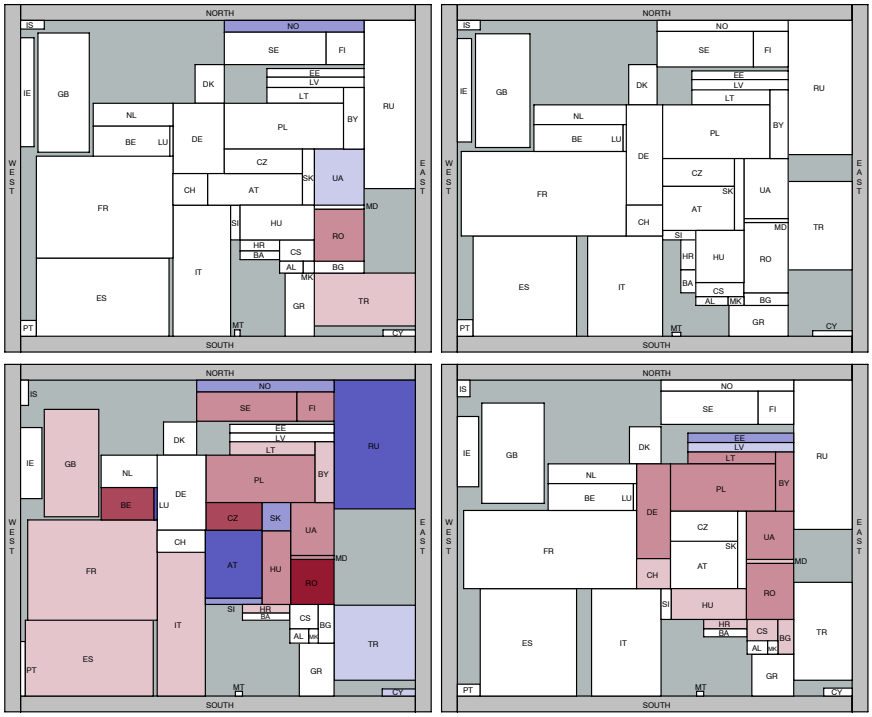

Fig. 8. Purely rectangular cartograms with correct adjacencies (left) and false adjacencies (right) produced by linear programming (top) and segment moving (bottom).

countries we added minimum width and height constraints to make sure that the flag and name fit inside the rectangle. During the World Cup we will grow countries that do well to show their increased chance of winning the cup.

5 Conclusions

We presented a new solution for the automated construction of rectangular cartograms. It is based on iterative linear programming, and produces better results than the previous, segment moving method. We also presented and implemented two extensions. The first allows L-shaped countries in the cartogram, and the second limits the adjacency errors of rectangles. The new algorithm makes it possible to generate rectangular cartograms of the World that both have low error and are aesthetically pleasing.

The difference in error and visual quality of purely rectangular cartograms, and cartograms with L-shaped regions is not significant, and which type is preferred is a matter of taste. Concerning adjacency constraints, it appears that level-1 provides a significant improvement in error over correct adjacencies, but a higher level does not seem worthwhile and may give cartograms of

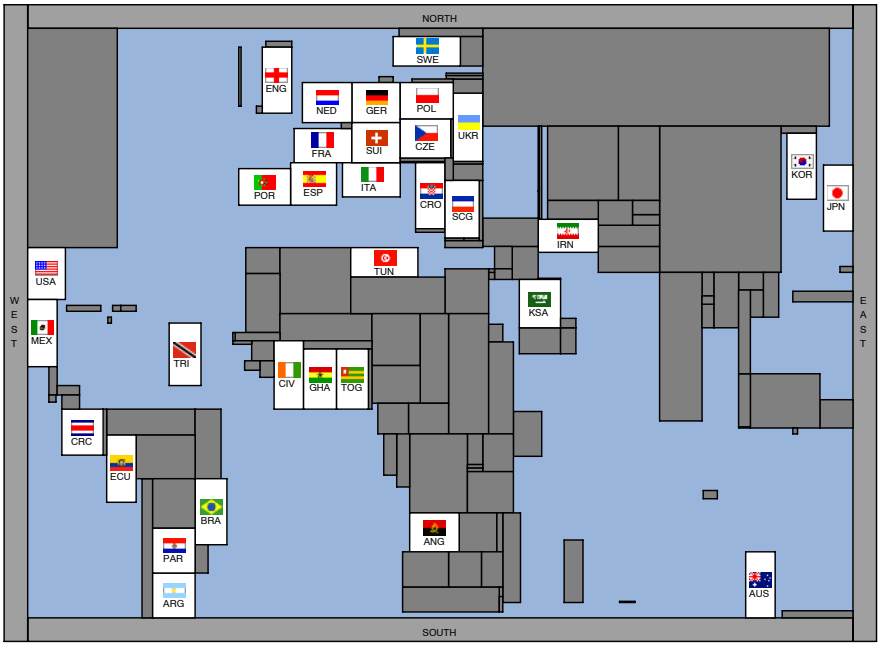

Fig. 9. FIFA World Cup 2006 qualified countries.

lower visual quality. We note that the rectangular cartograms of the Bosatlas [4] have incorrect adjacencies of various types, and our level-1 constraints give at least as good results in terms of relative positions.

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. T. Biedl and B. Genc. Complexity of orthogonal and rectangular cartograms. In *Proc. 17th Canad. Conf. on Computational Geometry*, pages 117–120, 2005.
4. *De Grote Bosatlas*. Wolters-Noordhoff, Groningen, 52nd edition, 2001.
5. CIA World Factbook, 2005. `http://www.cia.gov/cia/publications/factbook`.
6. CPLEX: High-performance software for mathematical programming and optimization. `http://www.ilog.com/products/cplex/`.
7. M. de Berg, E. Mumford, and B. Speckmann. On rectilinear duals for vertex-weighted plane graphs. In *Proc. 13th Int. Sympos. on Graph Drawing*, number 3843 in LNCS, pages 61–72, 2005.
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. A. Dougenik, N. R. Chrisman, and D. R. Niemeyer. An algorithm to construct continous area cartograms. *Professional Geographer*, 37:75–81, 1985.
11. H. Edelsbrunner and E. Waupotitsch. A combinatorial approach to cartograms. *Comput. Geom. Theory Appl.*, 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 Vis.*, 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. Symp. 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. Floor-planning using orderly spanning trees. *J. Algorithms*, 48:441–451, 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. M. Rahman, K. Miura, and T. Nishizeki. Octagonal drawings of plane graphs with prescribed face areas. In *Proc. 30th Graph-Theoretic Comcepts in Computer Science*, number 3353 in LNCS, pages 320–331, 2004.
23. E. Raisz. The rectangular statistical cartogram. *Geogr. Review*, 24:292–296, 1934.
24. W. Tobler. Pseudo-cartograms. *The American Cartographer*, 13:43–50, 1986.
25. W. Tobler. Thirty-five years of computer cartograms. *Annals of the Assoc. American Cartographers*, 94(1):58–71, 2004.
26. M. van Kreveld and B. Speckmann. On rectangular cartograms. In *Proc. 12th Europ. Sympos. Algorithms*, number 3221 in LNCS, pages 724–735, 2004.
27. M. van Kreveld and B. Speckmann. Rectangular cartogram computation with sea regions. In *Proc. 22st Int. Cartographic Conference*, 2005.