Efficient Cartogram Generation: A Comparison

Daniel A. Keim, Stephen C. North[*], Christian Panse[†], Jörn Schneidewind[‡]

Abstract

Cartograms are a well-known technique for showing geography-related statistical information, such as population demographics and epidemiological data. The basic idea is to distort a map by resizing its regions according to a statistical parameter, but in a way that keeps the map recognizable. In this paper, we deal with the problem of making continuous cartograms that strictly retain the topology of the input mesh. We compare two algorithms to solve the continuous cartogram problem. The first one uses an iterative relocation of the vertices based on scanlines. The second one is based on the Gridfit technique, which uses pixel-based distortion based on a quadtree-like data structure.

1 Introduction

Cartograms are a powerful way of visualizing geography-related information. A cartogram is a generalization of an ordinary map, which is distorted by resizing its regions according a geographically-related input parameter. Example applications include population demographics [7], election results [5], and epidemiology. Because cartograms are difficult to make by hand, the study of automated methods is of interest.Cartograms can also be seen as a general information visualization technique. They provide a mean for trading shape against area to improve a visualization by scaling polygonal elements according to an external parameter. In population cartograms, by allocating more space to densely populated areas, patterns that involve many people are highlighted, while those involving fewer people are emphasized less. Figure 1 shows a conventional map of the 2000 US presidential elections along with two population-based cartograms of the same information. The two cartograms are generated using the two methods which are compared in this paper. In the cartogram, the area of the states is scaled to their population, so it reveals the close result of a presidential election more effectively than the professionally designed map in figure 1(a). For a cartogram to be effective, a human being must be able to quickly understand the displayed data and relate it to the original map. Recognition depends on preserving basic properties, such as shape, orientation, and contiguity. This, however, is difficult to achieve and it has been shown that the cartogram problem is unsolvable in the general case [4]. Even when allowing errors in the shape and area representations, we are left with a difficult simultaneous optimization problem for which currently available algorithms are very time-consuming.

The Cartogram Problem The cartogram problem can be defined as a map deformation problem. The input is a planar polygon mesh (map) $\mathcal{P}$ and a set of values $\mathcal{X}$, one for each region. The goal is to deform the map into $\overline{\mathcal{P}}$ so that the area of each region matches the value assigned to it, doing this in such a way that the overall shape of the regions is preserved enough for them to remain recognizable. Intuitively, topology preservation means that the faces of the input mesh must stay the same, i.e. the cyclic order of adjacent edges in $\mathcal{P}$ must be the same as in $\overline{\mathcal{P}}$. This can be expressed formally by saying that the pseudo-duals[1] of the planar graphs represented by $\mathcal{P}$ and $\overline{\mathcal{P}}$ should be isomorphic. It has been shown that a simple variant of the cartogram problem, which even ignores issues of shape preservation, is NP-complete [8]. Since it may be impossible to simultaneously fulfill the area and shape constraints, the functions $f(\cdot\cdot), d_S(\cdot,\cdot)$ and $d_A(\cdot,\cdot)$ model the error of the output cartogram.

Previous Work There are several families of cartogram generators described in the literature. They range from simple non-continuous cartograms that merely scale and display disconnected polygons to sophisticated solutions that apply non-linear transformations or techniques from computational geometry to distort the map without breaking its topology[2]. None of the methods mentioned so far cap-

[*]AT&T Shannon Laboratory, Florham Park, NJ, USA – {keim, north}@research.att.com

[†]University of Constance, Germany – panse@informatik.uni-konstanz.de

[‡]University of Halle, Germany – schneide@informatik.uni-halle.de

[1]The *pseudo dual* of a planar graph is a graph that has one vertex for each face and an edge connecting two vertices if the corresponding faces are adjacent.

tures an explicit notion of shape-preservation. In contrast, the force-based approach in [5] by a non-linear optimization process.

Our Contribution In this paper we compare two new methods for generating cartograms. The first one is a recently-proposed scanline-based local repositioning of vertices [4]. The second one is a new approach based on the VisualPoints algorithm [3]. We compare both techniques and discuss their advantages and problems. Real application examples support the usefulness of the techniques.

2 Visualization Algorithm

2.1 The *CartoDraw* Solution

CartoDraw was recently proposed as a practical approach to cartogram generation [4]. In this section, we outline the main ideas behind it and introduce some useful variations.

The *CartoDraw* Algorithm The basic idea of *CartoDraw* is to incrementally reposition the vertices of the map's polygons by means of scanlines. Local changes are applied if they reduce total area error without introducing an excessive shape error. The main loop iterates over a set of scanlines. For each scanline, it computes a candidate transformation of the polygons, and checks it for topology and shape preservation. If the candidate transformation passes the tests, it is made persistent, otherwise it is discarded. The order of scanline processing depends on their potential for reducing area error. The algorithm iterates over the scanlines until the area error improvement over all scanlines falls below a threshold.

Scanline-based Local Repositioning The input scanlines are arbitrary lines and may be computed automatically or entered interactively. The idea is to use line segments (called *cutting lines*) perpendicular to scanlines at regular intervals. Consider the two edges on the boundary of the polygon intersected by a cutting line on either side of the scanline. These edges divide the polygon boundary into two connected chains. Now, if the area constraints require that the polygon expanded, the algorithm applies a translation *parallel* to the scanline to each vertex on the two connected pieces of the boundary (in opposite directions) to *stretch* the polygon at that point. Similarly, if a contraction is called for, the direction of translation is reversed.

2.2 The *VisualPoints* Solution

The *VisualPoints* system [3] was developed to address the problem of overplotting spatially referenced data. It works by moving points that would be drawn on already-occupied pixels to nearby unoccupied pixels instead of overplotting. *VisualPoints* assumes a hierarchical partitioning of the data space to support an efficient repositioning of the data points while preserving their distances and positions. In the following, we show how a similar idea can be applied to efficient cartogram generation. The basic idea is to insert a number of points proportional to its target area inside each polygon. The points are inserted into a hierarchical data structure, and the distortion implied by the data structure is then applied to reposition the vertices of the map.

The *VisualPoints* Algorithm In each step of the *VisualPoints* construction, the data set is recursively partitioned into four subsets containing the data points in four equally-sized subregions. Since the data points may not fit into the four equally size subregions, we have to determine new extents of the four subregions (without changing the four subsets of data points) such that the data points in each subset can be visualized in their corresponding subregion. For an efficient implementation, a quadtree-like data structure manages the required information and supports the recursive partitioning process. The partitioning process works as follows. Starting with the root of the quadtree, in each step the data space is partitioned into four subregions. The partitioning is made such that the area occupied by each of the subregions (in pixels) is larger than the number of pixels belonging to the corresponding subregion.

Generating Cartograms with *VisualPoints* After the quadtree is constructed, it is applied to distort the vertices of the polygon mesh. Each vertex is repositioned separately: First the cell of the quadtree containing the vertex is determined. Then the new position of the vertex is calculated by scaling the cells of the quadtree on each level according to the desired size of the cells (corresponding to the number of pixels). By repositioning each vertex, we iteratively construct the distorted polygon mesh.

3 Comparison and Evaluation

The *CartoDraw* algorithm described in section 2.1 was implemented in C++ using the LEDA library [6] and the *VPCarto* algorithm described in section 2.2 was implemented in Java. The tests reported in this section were performed on a 1 GHz Pentium computer with 512 Mbytes of main memory. In the following, we report and discuss the results, and compare the efficiency and effectiveness of both approaches.

Efficiency and Effectiveness Figure 2 shows the measured efficiency and effectiveness results. The total run

time varies between 3 seconds for the new *VisualPoints* approach, 25 seconds for the automatic scanline approach, and 16 hours for the non-linear optimization approach by Kocmoud and House [5][2]. Note that the scale on the y-axis of figure 2 is logarithmic. The *VPCarto* approach is more than four orders of magnitude faster than the Kocmoud and House approach, about two orders of magnitude faster than the interactive scanlines, and about one order of magnitude faster than the automatic scanlines. Since the *VPCarto* algorithm has no explicit notion of shape, the shape quality of the results is not as good as the results of *CartoDraw*. Figure 2(b) shows the shape versus area error results of call-volume cartograms of *VPCarto* versus interactive scanlines for the four call-volume cartograms shown in figure 3.[3] The results clearly indicate that the shape error of *CartoDraw* (interactive scanlines) is always better than that of *VPCarto*, but its area error is slightly worse. Since the total shape error is basically an average over the state-wise area error, in figure 2(c) we show the shape error state by state, sorted according to the shape error. Figure 2(c) reveals that the *CartoDraw* algorithm consistently provides a lower shape error than the *VPCarto* algorithm.

Application Examples We applied both algorithms for continuously monitoring telephone call volume data. Figure 3 shows the results of the telephone call volume (normalized by population) at four different times during one day. The color is redundantly mapped to the normalized call volume with brighter colors corresponding to smaller call volumes. The resulting visualizations clearly reflect the different time zones of the US, and show interesting patterns of phone usage as it proceeds during the day. For example, we see the western part of the country shrink in size in the early part of the day (6 am EST) and slowly increase in size as the day goes on, reflecting increasing traffic originated in that part of the country. It is interesting that the call volume is especially high in the morning and in the evening (see figure 3(a–e) and (d–h): 6 am on the east coast and 0 am on the west coast), while it is slightly lower during the day. Again, the *VPCarto* algorithm has a slightly lower area error, while the *CartoDraw* algorithm (interactive scanlines) provides a better shape preservation. The evaluation and comparison shows that both approaches have their advantages and disadvantages: While the *CartoDraw* algorithm provides significantly better shape preservation, it also needs elaborate geometric algorithms and therefore requires a significantly higher run time. In contrast, the *VPCarto* algorithm runs in interactive time and does not depend on the number of polygons involved.

[2]Our experiements were performed on a 1 GHz Pentium computer. For the comparison, the performance results were linearly scaled to 120 MHz to make them comparable to the results of Kocmoud and House.

[3]The four points are connected by a spline to make clear which points belong to which approach. The spline has no specific meaning.

4 Conclusions and Future Work

In this study we analyzed and discussed the problem of efficient cartogram drawing, and proposed a new cartogram drawing algorithm which is based on the *VisualPoints* algorithm. The new algorithm provides better area error results and a significantly better time performance but is slightly worse with respect to the shape error. The experiments show that both algorithms offer good results for a variety of applications, and the speed of our new algorithm even allows an interactive display of call volume in telecommunication applications.

Although the proposed algorithm is a significant step toward fast, reliable, and effective cartogram generation, there are a number of promising directions for further research. An important issue is that shape preservation in the *VPCarto* approach needs improvement. One possibility would be to combine both approaches and use a cartogram generated by automatic scanlines and further refine it using the *VPCarto* algorithm. An important related question is how to automatically determine scanlines. If we could automatically determine useful scanlines, we would probably be able to achieve speed and area error similar to the *VPCarto* approach with superior shape preservation.

References

[1] H. Central. *www.multied.com/elections*. Mar. 2002.

[2] H. Edelsbrunner and R. Waupotitsch. A combinatorial approach to cartograms. *Computational Geometry*, pages 343–360, 1997.

[3] D. A. Keim and A. Herrmann. The gridfit algorithm: An efficient and effective approach to visualizing large amounts of spatial data. *IEEE Visualization, Research Triangle Park, NC*, pages 181–188, 1998.

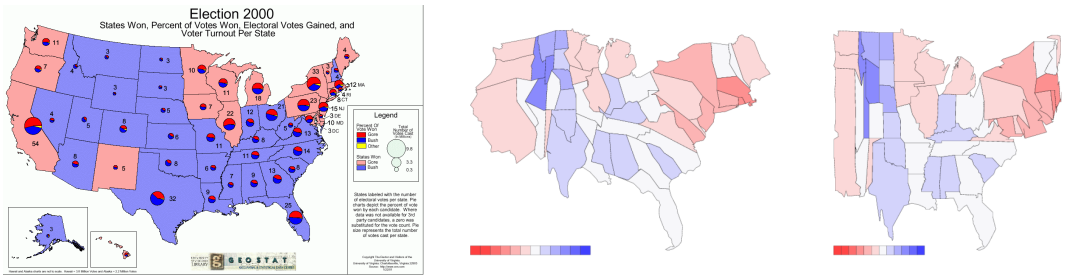
[4] D. A. Keim, S. C. North, and C. Panse. Cartodraw: A fast algorithm for generating contiguous cartograms. Information Visualization Reseach Group, AT&T Laboratories, Florham Park, 2002.

[5] C. J. Kocmoud and D. H. House. Continuous cartogram construction. *Proceedings IEEE Visualization*, pages 197–204, 1998.

[6] K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1st edition, 1999. `http://www.mpi-sb.mpg.de/~mehlhorn/LEDAbook.html`.

[7] W. Tobler. Cartograms and cartosplines. *Proceedings of the 1976 Workshop on Automated Cartography and Epidemiology*, pages 53–58, 1976.

[8] S. Venkatasubramanian. Computing integer cartograms is NP-complete. *Technical Report, ATT Research Labs*, 2002.

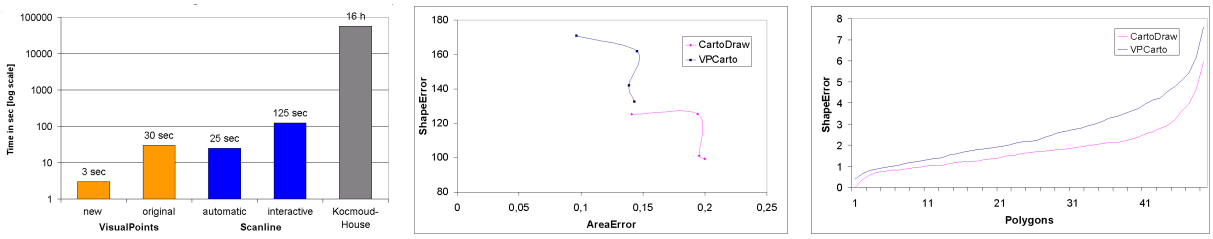


(a) Traditional Map Design [1]

(b) Population Cartogram *(CartoDraw)*

(c) Population Cartogram *(VisualPoints)*

Figure 1. Results of the 2000 US Presidential Elections

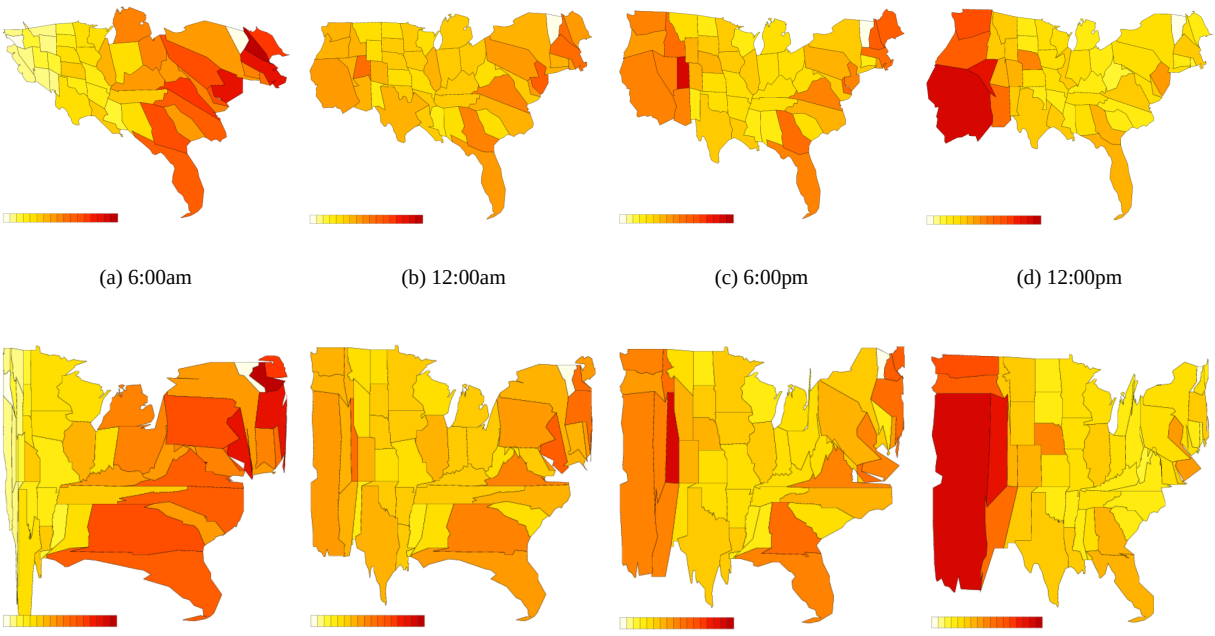


(a) Run Time Comparison

(b) Shape Error Versus Area Error

(c) Sorted Shape

Figure 2. Efficiency and Effectiveness Results



(a) 6:00am

(b) 12:00am

(c) 6:00pm

(d) 12:00pm

(e) 6:00am

(f) 12:00am

(g) 6:00pm

(h) 12:00pm

Figure 3. Long Distance Call Volume Data – computed with *CartoDraw* (a–d) and *VisualPoints* (e–h)