First publ. in: IEEE computer graphics and applications 25 (2005), 3, pp. 60-68

# Medial-Axis-Based Cartograms

**Daniel A. Keim and Christian Panse**
*University of Constance, Germany*

**Stephen C. North**
*AT&T Shannon Laboratory*

**A method for generating cartograms that combines iterative relocation of a map's vertices with medial-axis-based transformations retains the input map's topology.**

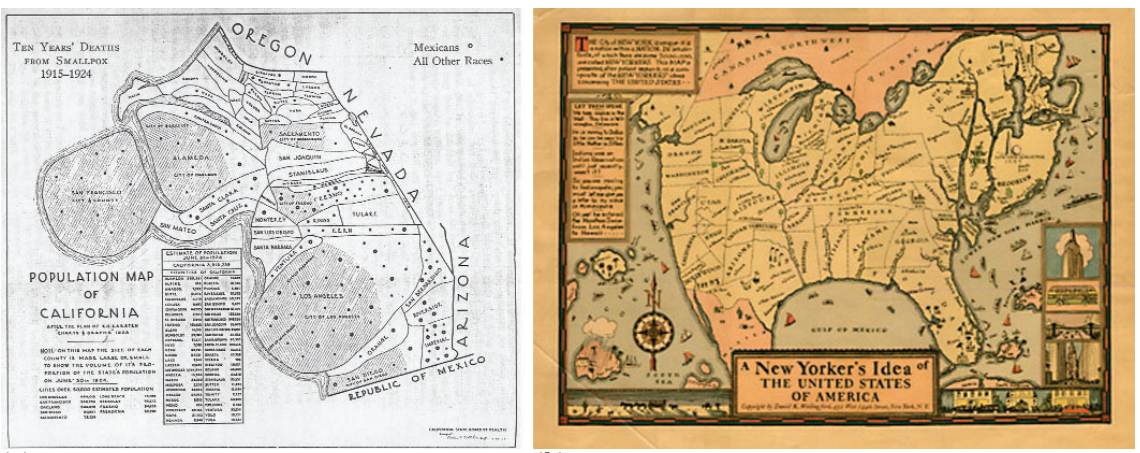
Cartographers and geographers were making cartograms for centuries before digital computers and displays became available. One of the oldest known cartograms, dating to about 300 C.E., depicts the extent of the Roman Empire. In data visualization, an area cartogram distorts a map by resizing its regions according to some external geography-related parameter, such as population or epidemiological data.

Figure 1 shows two historical examples. Figure 1a, from a 1927 medical journal, depicts the incidence of smallpox in the western US. It was made by an analog computer: modeling clay and a baker's rolling pin. Daniel Wallingford drew the cartogram in Figure 1b in 1932 and distributed it at the New York World's Fair (note Florida's size).

Cartograms are difficult to draw by hand because it's difficult to simultaneously optimize shape and area error while preserving the original map's topology. Automated methods for drawing cartograms have therefore received considerable interest (see the "Previous Work" sidebar).

Population-based cartograms can give a different overall impression from the original map. In a conventional choropleth map, for example, where parameter vectors are encoded by coloring the regions, important values in small, densely populated areas can be barely visible, while less important values spread out over large, sparsely populated areas seem emphasized. Such maps, therefore, tend to highlight patterns in areas with few people! Because cartograms can show geographic data in proportion to an additional parameter, such as population, they can display patterns in proportion to the number of people involved. Beyond the typical applications, a key motivation for studying computer cartograms is to define a general framework for trading shape and area adjustments in information visualization.

For a cartogram to be effective, an observer should be able to quickly relate the displayed data to the original map. Intuitive recognition depends on preserving basic properties such as shape, orientation, and contiguity. This, however, is difficult to achieve, because it's impossible, in the general case, to retain even the original map's topology. Even if we accept some error in shape and area in the cartogram, what remains is still a hard simultaneous optimization problem for which known algorithms are prohibitively time consuming. Further, because we're interested in the applicability of fine-grained and ani-




(a) (b)

**1** Historical examples of cartograms: (a) Gillihan's cartogram made with plasticine and a rolling pin, and (b) Wallingford's "A New Yorker's Idea of the United States." (Courtesy of the American Journal of Public Health)

## Previous Work

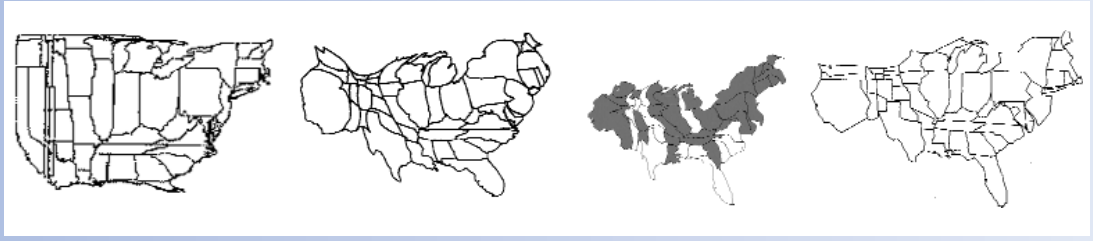
Few previous approaches to automated contiguous cartogram drawing yield results comparable in quality to good hand drawings. One reason, first identified by Dent,[1] is that straight lines, right angles, and other features that seem important in human recognition of maps are obliterated. Radial methods such as Tobler's conformal maps,[2] Selvin et al.'s radial expansion method,[3] Dougenik et al.'s rubber sheet method,[4] and Guseyn-Zade and Tikunov's line integral method[5] don't provide completely acceptable results, because they often heavily deform the polygons' shapes, as Figure A shows. Likewise, Tobler's pseudocartograms expand the longitude and latitude lines to achieve a least root mean square area error.[2]

Approaching the problem as distortion viewing by nonlinear magnification produces similar drawings. Jackel applied radial forces to change polygon size, moving the sides of each polygon relative to its centroid,[8] but an implementation of the solver runs slowly (taking 90 minutes to make eight iterations on a map of six New England states of the US). Nonconvex input polygons and potential self-intersections in the output map further complicate matters.

Another family of approaches operates on a grid or mesh imposed on the input map. The piezopleth method[9] transforms the grid using a physical pressure load model. Dorling's cellular automaton approach trades grid cells on a fine mesh until each region achieves the desired number of cells.[10] Edelsbrunner and Waupotitsch's combinatorial approach[6] computes a sequence of piecewise linear homeomorphisms of the mesh that preserve its topology. Whereas the first method effectively preserves polygons' shapes, the second method allows a good fit for area but doesn't account for shape preservation. Kocmoud and House synthesized the two approaches, proposing a force-based model and alternately optimizing shape and area objectives.[7] Although the results are better than most other methods, this optimization has prohibitively high execution time. Kocmoud and House report a running time of 18 hours for a reasonable-sized map with 744 vertices.


### References

1. B.D. Dent, *Cartography: Thematic Map Design*, 4th ed., William C. Brown, 1996
2. W.R. Tobler, "Pseudo-Cartograms," *American Cartographer 13*, vol. 1, 1986, pp. 43-40.
3. S. Selvin et al., *Transformations of Maps to Investigate Clusters of Disease*, tech. report LBL-18550, Lawrence Berkeley Laboratory, Univ. of Calif., 1984.
4. J.A. Dougenik, N.R. Chrisman, and D.R. Niemeyer, "An Algorithm to Construct Continuous Area Cartograms," *The Professional Geographer*, vol. 37, no. 1, 1985, pp. 75-81.
5. S. Gusein-Zade and V. Tikunov, "A New Technique for Constructing Continuous Cartograms," *Cartography and Geographic Information Systems*, vol. 20, no. 3, 1993, pp. 66-85.
6. H. Edelsbrunner and R. Waupotitsch, "A Combinatorial Approach to Cartograms," *Computational Geometry*, vol. 7, nos. 5-6, 1997, pp. 343-360.
7. C.J. Kocmoud and D.H. House, "Continuous Cartogram Construction," *Proc. IEEE Visualization*, IEEE CS Press, 1998, pp. 197-204.
8. C.B. Jackel, "Using Arcview to Create Contiguous and Noncontiguous Area Cartograms," *Cartography and Geographic Information Systems*, vol. 24, no. 2, 1997, pp. 101-109.
9. C. Cauvin, C. Schneider, and G. Cherrier, "Cartographic Transformations and the Piezopleth Method," *Cartographic J.*, vol. 26, no. 2, 1989, pp. 96-104.
10. D. Dorling, *Area Cartograms: Their Use and Creation*, 1st ed., Dept. of Geography, Univ. of Bristol, England, 1996.

**A Cartogram drawing methods: (1) conformal maps,[2] (2) line integral method,[5] (3) combinatorial approach,[6] and (4) force-based model.[7]**

mated cartograms, we need more efficient algorithms.

We've developed an approach to computing continuous cartograms based on the medial axis of the input mesh. (The "Variants of the Cartogram Problem" sidebar describes other approaches.) Other researchers have suggested using the medial axis as a compact representation of an object's shape, making it a natural choice for our use. We combine this with the idea of using scanlines to guide cartogram generation.[1] In essence, the scanline approach interprets a line segment drawn through the map as a hint to the direction in which to expand or contract polygons. A series of such local

## Variants of the Cartogram Problem

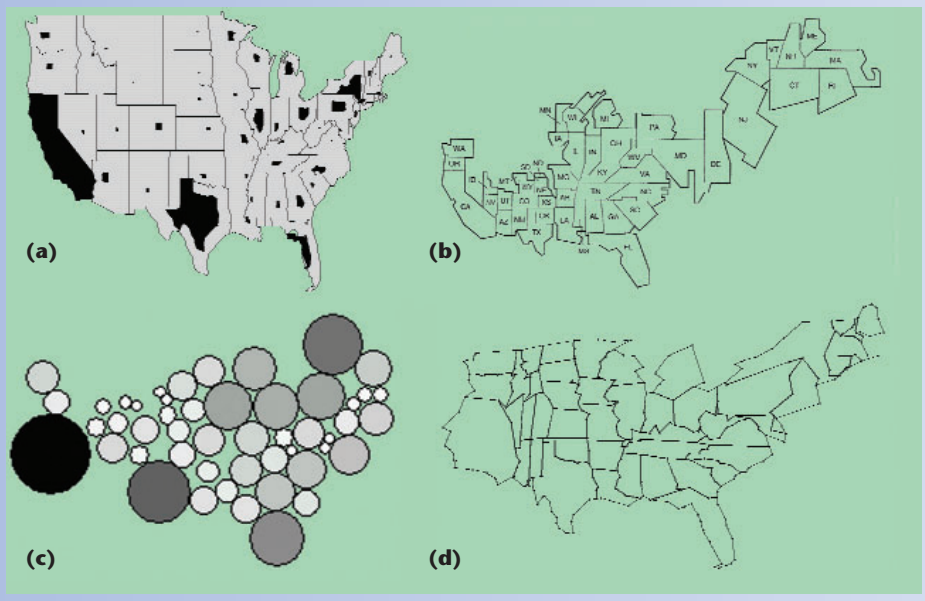
Several categories of cartogram problems exist.

*Noncontinuous cartograms*, shown in Figure A1, can exactly satisfy area and shape constraints but don't preserve the input map's topology. Because the scaled polygons are drawn inside the original regions, the loss of topology doesn't cause perceptual problems. More critical is that the polygons' original size restricts their final size. Consequently, you can't make small polygons arbitrarily large without scaling the entire map, so important areas can be difficult to see and screen usage can be poor.

*Noncontiguous cartograms*, shown in Figure A2, scale all polygons to their target sizes, perfectly satisfying the area objectives. Shapes can be slightly relaxed so polygons touch without overlapping, and the map's topology is also highly relaxed because polygons don't retain their adjacency relationships. Noncontiguous cartograms provide perfect area adjustment with good shape preservation. However, they lose the map's global shape and topology, which can make perceiving the generated visualization as a map difficult.

*Circular cartograms*, shown in Figure A3, completely ignore the input polygons' shape, representing them as circles in the output. In many cases, area and topology constraints are also relaxed, so circular cartograms have some of the same problems as noncontiguous cartograms.

The final category is *continuous* (and *contiguous*) *cartograms*, shown in Figure A4, which we generated using an algorithm presented elsewhere.[1] Unlike the other categories, contiguous cartograms retain a map's topology perfectly, but relax the given area and shape constraints. In general, cartograms can't fully satisfy shape or area objectives, so cartogram generation involves a complex optimization problem in searching for a good compromise between shape and area preservation. Although continuous cartograms are difficult to generate, the resulting polygonal meshes resemble the original map more than those generated by other cartogram variants. Our study therefore focuses on continuous cartograms.




**A Variants of the cartogram problem: (a) noncontinuous cartogram, (b) noncontiguous cartogram, (c) circular cartogram, and (d) continuous cartogram.**

### Reference

1. D.A. Keim et al., "Efficient Cartogram Generation: A Comparison," *IEEE Symp. Information Visualization* (InfoVis), IEEE CS Press, 2002, pp. 33-36.

improvements using different scanlines leads to the final cartogram. Our approach runs fast enough to handle large maps, and preserves shapes accurately while reflecting the associated geospatial data set.

## Cartogram drawing

We pose cartogram generation as a map-deformation problem. The input map is a planar polygon mesh and a value associated with each region (face) that's a desired fraction of the total cartogram area. The goal is to deform the map so that each region's area is close or equal to the target, while preserving the map's connectivity and the shapes of its faces, including the overall shape.

**Definition:** The Cartogram problem

*Input.* A planar polygon mesh $\mathcal{P}$ consisting of polygons $p_1, \ldots, p_k$, values $\mathcal{V} = (v_i)_{i=1,\ldots,k}$ with $v_i > 0$, $\Sigma v_i = 1$, and $\forall i, j \in \{1, \ldots, k\} \wedge i \leq j \rightarrow v_i \geq v_j$. That is, the elements of are sorted in nonincreasing order. Let $A(p_i)$ denote the normalized area of polygon $p_i$ with $A(p_i) > 0, \Sigma A(p_i) = 1$.

*Output.* A topology-preserving polygon mesh $\overline{\mathcal{P}}$ consisting of polygons $\overline{p}_1, \ldots, \overline{p}_k$ such that the function $f: \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}$ is minimized:

$f(\overline{S}, \overline{A}) \rightarrow \min$

where

$\overline{S} = (s_1, \ldots, s_k)$ with $s_i = d_s(p_i, \overline{p}_i)$ is the shape error and

$\overline{A} = (a_1, \ldots, a_k)$ with $a_i = d_A(v_i, A(\overline{p}_i))$ is the area error.

Topology preservation means that $\mathcal{P}$ and $\overline{\mathcal{P}}$ must be homeomorphic. We can define that in terms of Betti numbers[2] or by explicitly testing that there's a one-to-one mapping of faces from $\mathcal{P}$ to faces of $\overline{\mathcal{P}}$ that preserves adjacencies.

### *Area error*

Because it's generally impossible to exactly satisfy

area and shape constraints simultaneously, we seek approximate solutions. The functions $f$, $d_S$, and $d_A$ model error in the output cartogram. In the simplest case, the single polygon area error function $d_A$ is the norm of the difference of the desired and actual area.

If we normalize $v_i$ and $A(p_i)$, $(v_i > 0 \wedge \sum v_i = 1;\ A(p_i) > 0 \wedge \sum A(p_i) = 1)$, and $w_i$ is a weight between 0 and 1, we can express the area error $d_A$ as

$$d_A\left(v_i, A\left(\bar{p}_i\right)\right) = w_i \cdot \left| v_i - A\left(p_i\right)\right| \quad \forall i = 1, \ldots, k$$

The weight function $w_i$ is important for generating informative cartograms. In meshes of thousands of polygons, all faces will likely have equal importance. The area error in regions of high interest (for example, with high $v_i$ values) should therefore have more influence on the overall error than the area error in regions of low interest.

In applications where only a few polygons are of interest (as given by the factor $\xi (0 \le \xi \le 1)$), we can ignore the others' area errors, giving us the weights:

$$w_i = \begin{cases} 1 & \text{if} \sum_{j=1}^{i} v_j \le \xi \\ 0 & \text{otherwise.} \end{cases}$$

We use this type of weight function in some of the application experiments we describe later.

### *Shape error*

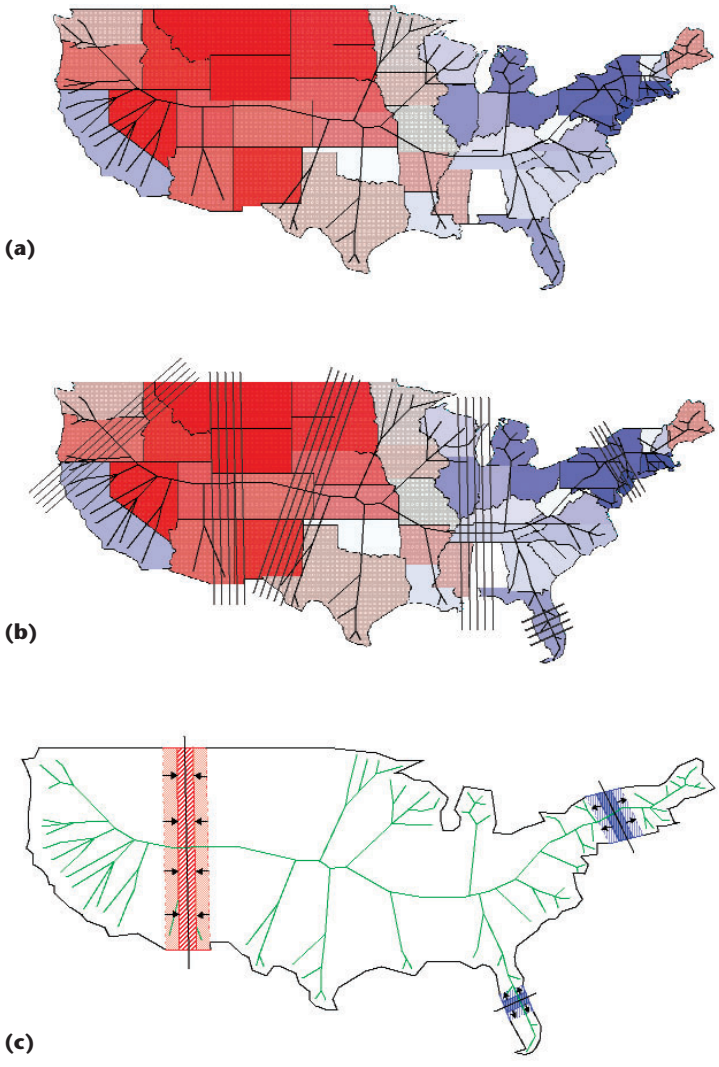
We can compare the shapes of two polygons in various ways. For example, we can approximate the polygons' curvature using a turning angle algorithm, curvature plots such as the centroidal profile, super segments, geometric hashing, or Fourier approximations. Our implementation incorporates a Fourier transformation of the polygons' curvatures. If $C(p)$ denotes the curvature of a polygon $p$ and $F(C(p))$ denotes its Fourier transformation, we define the shape error $d_S$ as

$$d_s\left(p_i, \bar{p}_i\right) = \left\| F\left(C\left(p_i\right)\right), F\left(C\left(\bar{p}_i\right)\right)\right\| \quad \forall i = 1, \ldots, k$$

where $\| \cdot \|$ denotes the Euclidean $L_2$ norm. Note that we can determine the Fourier transformation of the polygons' curvature analytically.[3]

Defining cartogram generation this way means that feasible solutions always exist, although finding good ones might be difficult. In contrast, variants that allow absolute area constraints can be infeasible.[1]

**2** Medial axis: (a) rectangle with circles, (b) triangle, and (c) polygon.




**3** Cartogram algorithm: (a) map with area error and medial axis, (b) stretching and contracting the global polygon, and (c) example cutting lines.

## M-CartoDraw

The M-CartoDraw algorithm incrementally repositions the vertices of the polygon mesh using medial axis segments as scanlines. The medial axis, or skeleton, of a 2D region is the loci of the centers of its maximal inscribed circles, as Figure 2a illustrates. Many equivalent definitions of the medial axis transform exist. For example, some researchers define the transform as the centers of the circumcircles of the polygon's Delaunay triangulation. The prairie fire transformation[3] is a more intuitive definition. Imagine the polygon's interior is dry grass and the exterior is unburnable wet grass. Suppose we set a fire simultaneously at all points on the polygon's boundary. The fire propagates at uniform speed toward the figure's middle. At some points, however, different fire fronts meet and extinguish each other. These points are the fire's *quench points*; the set of quench points defines the figure's skeleton.

Figure 3a illustrates the application of the medial axis to cartograms. We use census population data for the target area vector $\mathcal{V}$. We encode area error using a bipolar red and blue color map. Blue regions should be larger; red regions smaller. Color intensity indicates magnitude.

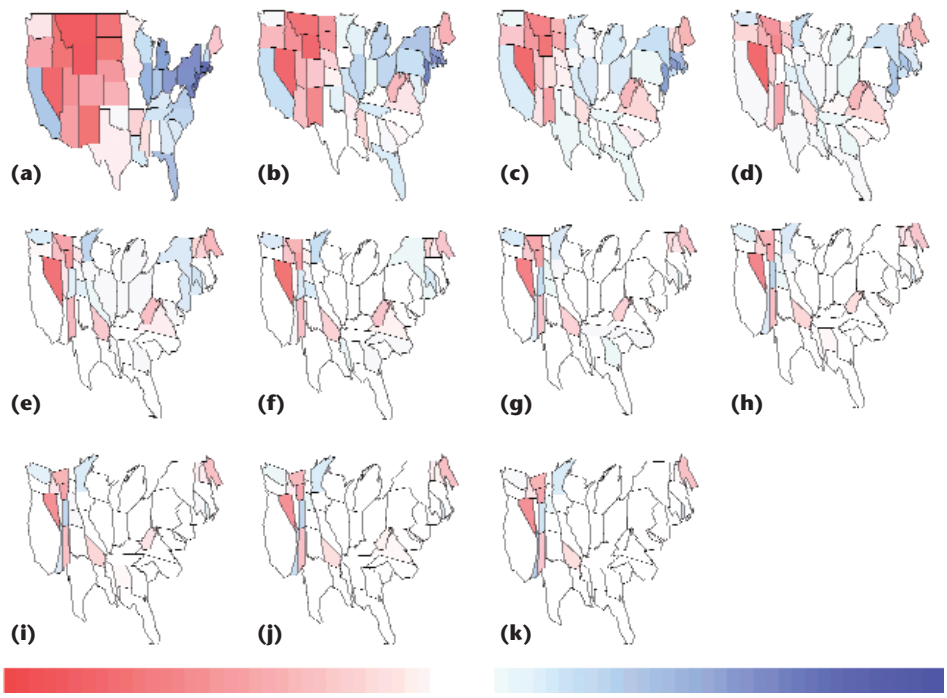
Assume we draw a scanline inside a polygon. Our

```
processMASegement(P, V, MASeg, XY);
/* for all cutting lines on medial axis segments MASeg */;
for cl ∈ CuttingLines(MASeg) do
    /* determine the aggregated scaling factor of all p_i */;
1   sf =computeScalingFactor({p_i | p_i ∈ P ∧ p_i ∩ cl ≠ 0}, V);
    /* for each vertex of the mesh */;
    for v ∈ P do
2       XY[v] = XY[v]+ sf · sgn(ori(cl.v)) · MÃSeg / |MÃSeg| ;
```

**4** Algorithm 1. Processing a single medial axis segment.

```
M-CartoDraw(P, V);
/* initialize the node array which stores the candidate transformation */;
XY .init(P, (0.0; 0.0));
1   while ( ||E_area − AE(P, X)|| > ε_a) do
        E_area = AE( , X);
        /* compute the medial axis MA of the global polygon */;
        MA = computeMedialAxes(GP(P));
2       for MASeg ∈ MA do
            processMASegment(P, V, MASeg, XY );
3           if (T(P, XY) ∧ (SE(P, XY) ≤ ε_s) ∧ (AE(P,
                X, XY) < E_area)) then
                makePersistent(P, XY);
```

**5** M-CartoDraw algorithm.




**6** M-CartoDraw construction series: (a) step 0, (b) step 1, (c) step 2, (d) step 3, (e) step 4, (f) step 5, (g) step 6, (h) step 7, (i) step 8, (j) step 9, and (k) step 10 (area error in step 10 is less than 3 percent).

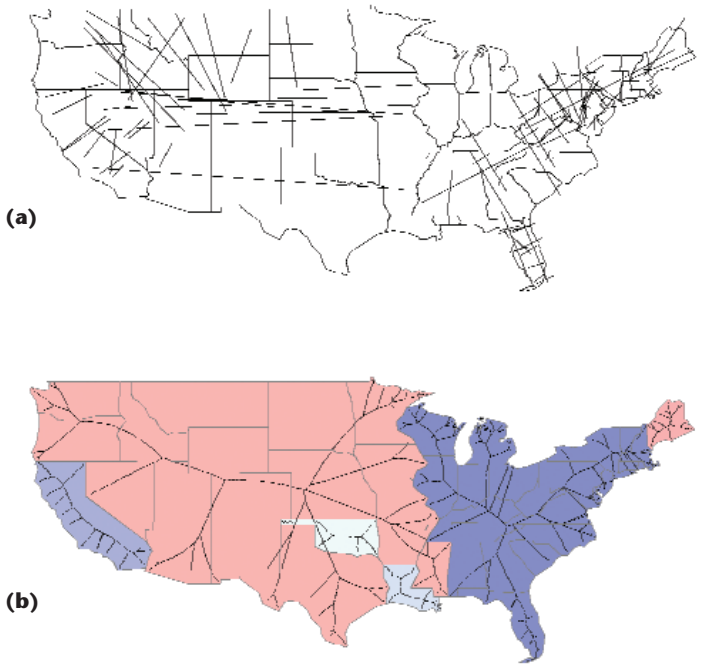
algorithm computes line segments (or *cutting lines*) perpendicular to the scanline at regular intervals. A scanline divides the polygon's boundary into two chains. To expand the polygon, the algorithm applies a translation parallel to the scanline to each vertex on the chains (in opposite directions). To contract it, the algorithm applies the translation in reverse. Our algorithm repeatedly applies medial axis segments as scanlines, thus using the polygon's shape to make local expansions or contractions. Figure 3b shows three examples of this process. In the midwest, a cutting line contracts the global shape, while in the northeast and Florida, the global shape is stretched. We insert the cutting lines in regular intervals on the medial axis. Figure 3c shows cutting lines at six different map locations.

### *Cartogram algorithm*

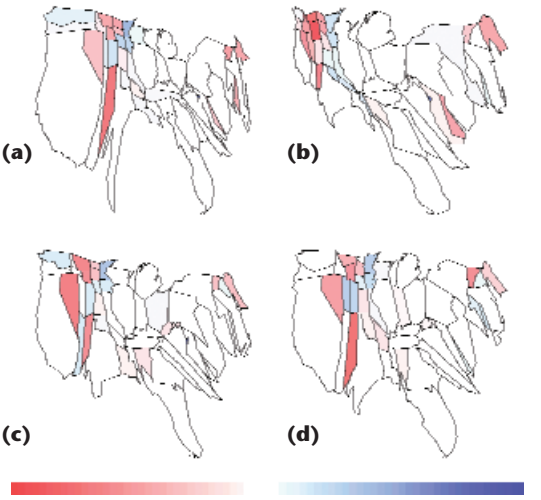
The algorithm in Figure 4 processes a single medial axis segment. The function `computeScalingFactor` determines whether to stretch or contract the global polygon and the amount of adjustment (line 1). We compute this function as a weighted average of the area errors of the polygons cut by the cutting line, weighted by their scale factors. The algorithm doesn't calculate new positions of all vertices for each cutting line. Instead, it aggregates the distortion vectors for each point and applies the aggregate vector after considering all cutting lines of a medial axis segment (line 2).

Figure 5 (next page) shows the main algorithm. The top-level loop iterates over all medial axis segments (line 2), computes a candidate transformation of the polygon, and checks for topology preservation and shape error (line 3). The order in which the algorithm processes medial axis sections depends on their potential for reducing area error. A shape error threshold $\varepsilon_s$ controls the maximum error that can be introduced in one step. If a candidate transformation passes these tests, it is made persistent, otherwise it is discarded. The algorithm continues the iteration until the total area error improvement is below a threshold $\varepsilon_a$ (line 1). In processing individual medial axis segments, we let the algorithm increase the area error if necessary to escape local minima. However, in each main loop iteration the area error decreases monotonically, guaranteeing termination.

Figure 6 shows what happens when we incrementally apply M-CartoDraw to a US population cartogram. As before, the blue polygons should be larger, and red ones smaller. The algorithm quickly provides




7 Extensions of the cartogram algorithm: (a) interactive scanlines, (b) cluster region medial axis, and (c) all-polygon medial axis.




8 US telephone call volume data over a 24-hour period (times are EST): (a) midnight, (b) 6:00 a.m., (c) noon, and (d) 6:00 p.m. Color represents the area error. White polygons are distorted with an area error close to 0; blue polygons should be made larger; and red polygons should be made smaller.

acceptable, informative results in areas that are well suited to the approach (that is, the Midwest, New England, California, New York, and Pennsylvania).

### *Robustness and stability*

Noise and error in the input data set can lead to problems in the medial axis transforms. Although treating these problems is outside our work's scope, practical solutions based on pruning strategies and simplification exist.[4] In addition, several authors have suggested generating cartograms from decimated maps, mitigating some of these issues.

### *Extensions of the algorithm*

One problem with the proposed algorithm is that the global polygon's medial axes might not let us locally adjust certain regions with high area error (see, for example, Figure 3a). Previous experiments with an interactive scanline-based cartogram algorithm suggest that manually placing additional scanlines in such regions can improve the resulting cartograms (see Figure 7a).

One approach to locally adjusting regions is to cluster regions with area errors in the same direction—that is, they all must expand or contract (see Figure 7b). We compute a medial axis for each such cluster and then apply the M-CartoDraw algorithm. We process the cluster regions in order of decreasing aggregate area error.

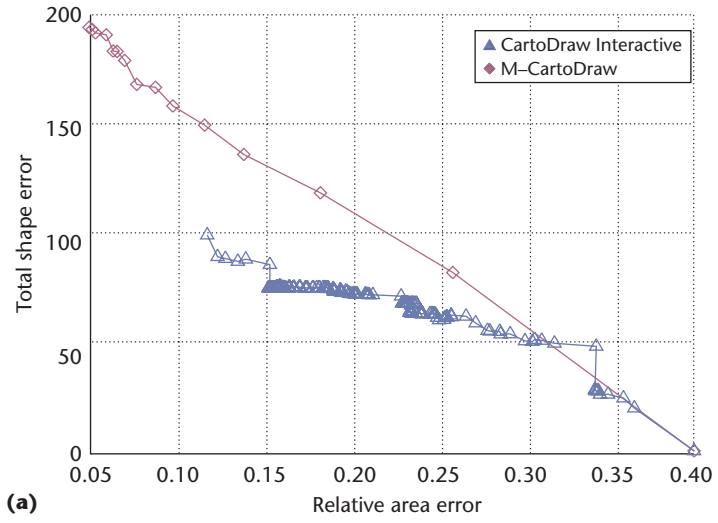
We can further extend the cluster-based approach by computing the medial axis of each polygon in the input map. Figure 7c shows the polygons and their associated scanlines. Again, we consider each polygon's scanlines in order of decreasing area error.

## Evaluation and applications

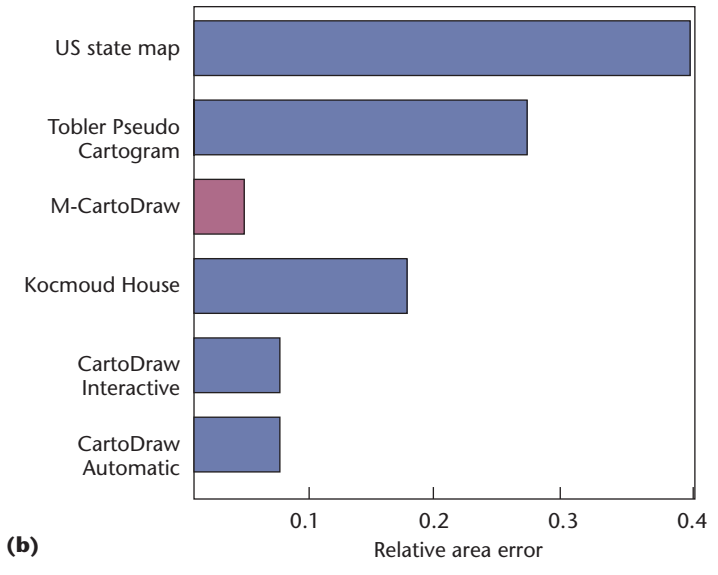
We implemented our algorithm in C++ using the LEDA library[5] on Microsoft Windows and Linux. We performed our tests on a 1.5-GHz Intel Xeon server with 4 GBytes of main memory (although we only needed 15 MBytes) under Linux. On the whole, our cartograms are competitive with those from previous approaches (see the "Previous Work" sidebar).

### *Effectiveness*

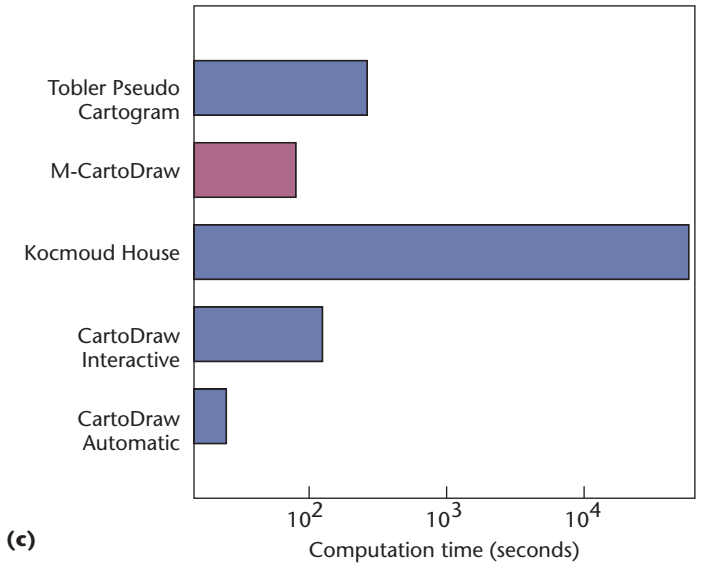
Figure 8 shows M-CartoDraw's output for the usage of a telecommunication service in the US. The figure shows cartograms of volume for this service at four points in time in one day (midnight, 6 a.m., noon, and 6 p.m. EST). All of the cartograms are of satisfactory quality in the sense that the US geography is clearly recognizable, while the area error for each cartogram is less than 5 percent. A polygon's color represents its area error. White polygons are perfectly distorted with an area error close to 0; blue polygons should be larger; and red should be smaller. The visualizations show interesting




(a)




(b)




(c)

**9 Effectiveness and efficiency comparison using the US state map: (a) shape versus area error, (b) area error comparison, and (c) computation time comparison.**

service usage patterns reflecting the different time zones in the US.

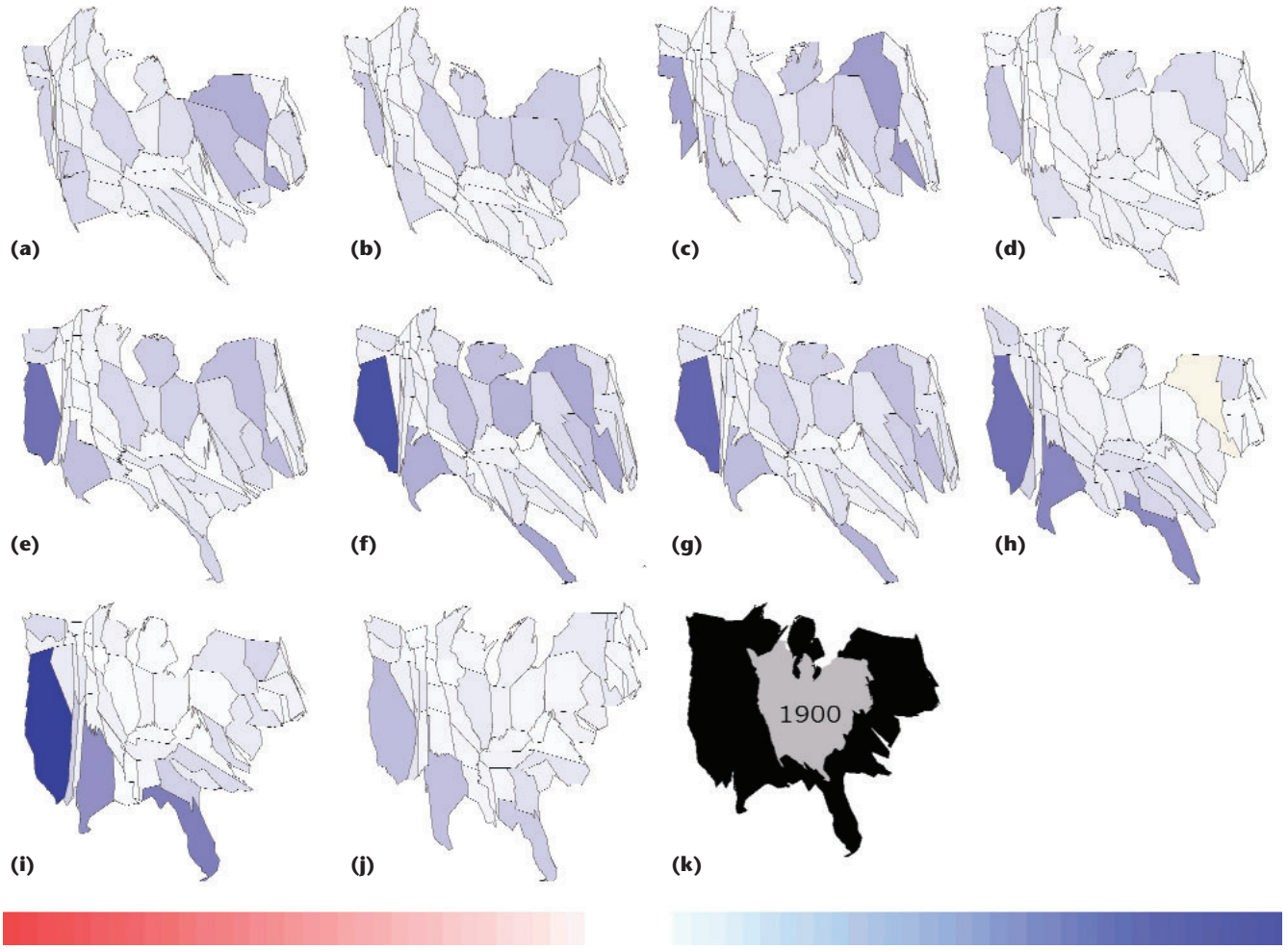
As Figure 9a illustrates, M-CartoDraw yields better results than the interactive scanline approach[1] with respect to the area and shape error trade-off. To measure shape distortion, we used the Fourier-based method as discussed earlier.[1] In Figure 9a, each point corresponds to an intermediate solution found in one M-CartoDraw step. At the beginning, the area error is larger than 36 percent. With more iterations, the area error decreases and the shape error grows because of distortions that are introduced. As expected, the curve traces from the lower right corner to the upper left corner until the area error is small enough, the area error difference is less than its threshold, or the shape distortion is larger than a given threshold. In most cases, shape and area error have an inverse relationship. Figure 9a also shows that the final shape error depends on the area error at the beginning. This is because maps that start with a high area error must be more heavily distorted than maps with lower area error. The image also implies that the slope of the curve for M-CartoDraw is more constant than that of the interactive scanline approach,[1] which we attribute to the irregularity of human interactions. Figure 9b shows the total area error for M-CartoDraw (with 3-percent area error). This figure demonstrates that the proposed approach is preferable to both our earlier non-medial-axis-based work[1] and the hybrid optimization-based approach.[6]

### *Efficiency*

We performed experiments to evaluate the proposed algorithm's efficiency. For this study we didn't include the computation time needed to simplify (decimate) the input map, because we treat this as an external one-time precomputation. The main advantage of our approach is its low running time: times range from 6 seconds for the US state map to 5 minutes for the US country map (with about 3,000 polygons). Finding the medial axis takes about half the total running time. This compares favorably with the prior best-known approach,[6] which (adjusted for current CPUs) takes about two orders of magnitude longer to compute a cartogram, as Figure 9c demonstrates. The figure compares M-CartoDraw's running time with that of Kocmoud and House's approach.[6] The test assumed that the algorithm runs on a 120-MHz computer with 32-MByte RAM. Note that the *Y* scale is logarithmic.

Figures 10a through 10j illustrate trends in the US population by decade during the 20th century. Each decade is colored according to population increase or decrease: the scale from dark to light blue, through white to red corresponds to the range in census values from high immigration to (low) emigration. The maps are normalized so they don't reflect changes in the total population by year. The global polygon of the US in 2000 with 270 million inhabitants is colored black, and the global polygon of the population in 1900 with 76 million is colored gray.

A key goal for M-CartoDraw is to handle large maps. Figure 11 shows the population of Germany on a map with about 400 polygons. We used a unipolar color map



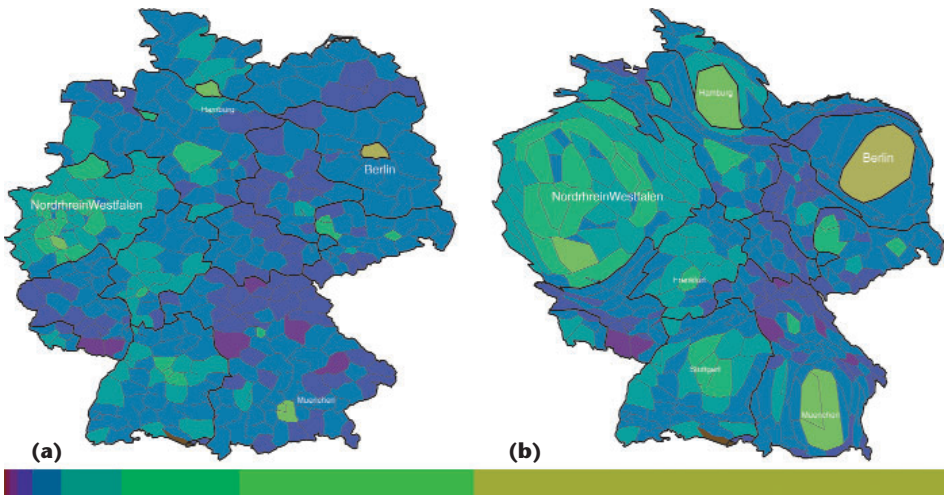

**10** **Immigration trends in the US over the last 100 years: (a) 1910, (b) 1920, (c) 1930, (d) 1940, (e) 1950, (f) 1960, (g) 1970, (h) 1980, (i) 1990, (j) 2000, and (k) population comparison, 1900 and 2000. Colors correspond to increasing or decreasing population. Dark blue indicates a high rate of immigration, light blue indicates low immigration, white indicates no change, and light red indicates low emigration.**

to visualize this population data. The cartogram helps focus attention on highly populated regions. The area errors of the three largest polygons and the yellow polygons in the northwest are almost zero, with cities and other highly populated regions clearly visible.

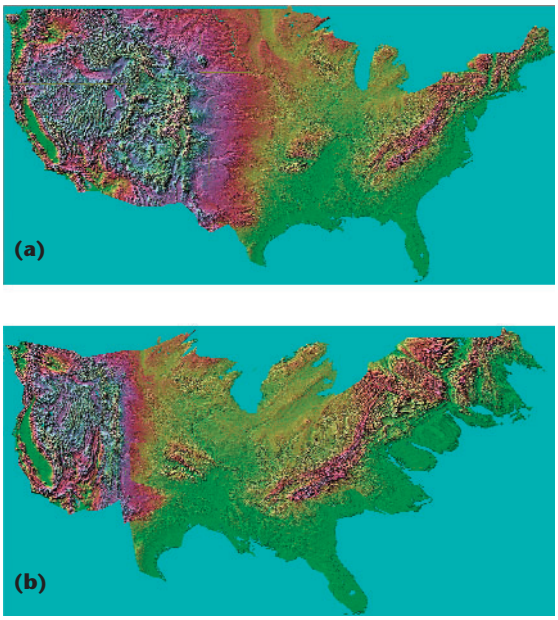
Because the underlying polygons don't change dramatically in every step (especially when the algorithm is near termination), to save time we don't recalculate the medial axis on every iteration.

M-CartoDraw took less than five minutes to compute the cartogram in Figure 11b. This computation time is only one magnitude greater than that needed to compute the US state population cartogram.

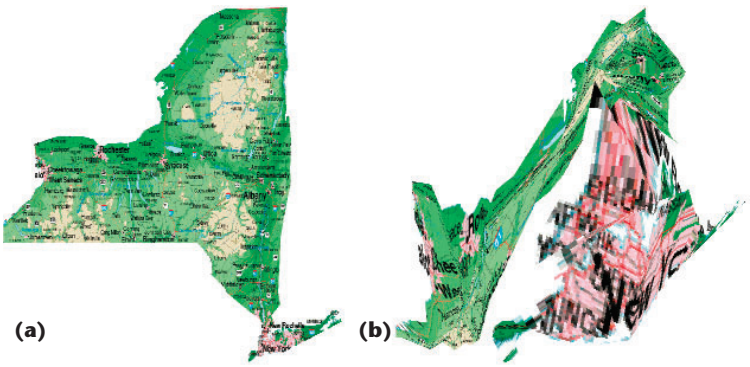
Figure 12 also shows the result of using texture maps to visualize geo-




**11** **County-level map of Germany: (a) original population cartogram and (b) distorted map using M-CartoDraw. The thicker black lines indicate the German states. A unicolor map encodes population density. The yellow-green regions (for example, Berlin, Hamburg, and Munich) are densest.**




**12** US population cartogram with texture: (a) original map and (b) M-CartoDraw distorted map. Map images provided by Ray Sterner of the Johns Hopkins University Applied Physics Laboratory, licensed by North Star Science and Technology (http://www.landforms.biz/).




**13** Population cartograms of New York state: (a) original map and (b) M-CartoDraw distorted map with almost zero area error. We drew map (b) using a texture extracted from map (a). M-CartoDraw enlarges polygons where the population density is high.

graphically related data. Figure 12a is an undistorted map of the US. Figure 12b shows the US map after we apply M-CartoDraw. In this population texture cartogram, the relief is distorted in proportion to the number of inhabitants.

Figure 13 shows a texture cartogram of New York state using an almost zero area error transformation. The cartogram emphasizes the proportion of New York City in the total population.

## Conclusions and future work

Although our algorithm is a significant step toward fast, effective cartogram generation, open problems remain. It would be interesting to study general methods for computing a morph (homotopy) with specified boundary properties that optimizes some function of the interior. We can also further generalize our method of using medial axis segments as scanlines (for example, we can apply vectors in the direction of flow from the medial axis to the boundary). ■

## References


1. D.A. Keim, S.C. North, and C. Panse, "CartoDraw: A Fast Algorithm for Generating Contiguous Cartograms," *Trans. Visualization and Computer Graphics*, vol. 10, no. 1, 2004, pp. 95-110.
2. S.S. Cairn, *Introductory Topology*, 1st ed., Ronald Press Company, 1961.
3. J. O'Rourke, *Computational Geometry in C*, 1st ed. Cambridge Univ. Press, 1994.
4. M. Foskey, M.C. Lin, and D. Manocha, "Efficient Computation of a Simplified Medial Axis," *Proc. ACM Symp. Solid Modeling*, ACM Press, 2003, pp. 96-107.
5. K. Mehlhorn and S. Näher, *LEDA Platform of Combinatorial and Geometric Computing*, 1st ed. Cambridge Univ. Press, 1999.
6. C.J. Kocmoud and D.H. House, "Continuous Cartogram Construction," *Proc. IEEE Visualization*, IEEE CS Press, 1998, pp. 197-204.



***Christian Panse*** *is a doctoral student in the Data Mining and Visualization Group at the University of Constance, Germany. His research interests include visual data mining on large spatial data and cartogram drawing. Panse has an MS in computer science from the Martin-Luther-University Halle-Wittenberg, Germany. He is a member of the IEEE Computer Society. Contact him at panse@dbvis.inf.uni-konstanz.de.*



***Daniel A. Keim*** *is a full professor in the Computer Science Department of the University of Constance. His research interests include information visualization and data mining. Keim has a PhD in computer science from the University of Munich. He is an editor of the* IEEE Transactions on Visualization and Computer Graphics, IEEE Transactions on Knowledge and Data Engineering, *and the* Palgrave Information Visualization Journal. *Contact him at keim@dbvis.inf.uni-konstanz.de.*



***Stephen C. North*** *is head of Information Visualization Research at AT&T Labs, a group in the AT&T Infolab that studies interactive, high-performance visualization of large, complex relational and semistructured data sets. His research interests include graph layout and spatial data visualization. North has a PhD in computer science from Princeton University. He is a senior member of the IEEE and a member of the ACM. Contact him at north@research.att.com.*