CartoDraw: A Fast Algorithm for Generating Contiguous Cartograms

Daniel A. Keim, *Member*, *IEEE Computer Society*, Stephen C. North, *Senior Member*, *IEEE*, and Christian Panse, *Member*, *IEEE Computer Society*

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 study, we formally define a family of cartogram drawing problems. We show that even simple variants are unsolvable in the general case. Because the feasible variants are NP-complete, heuristics are needed to solve the problem. Previously proposed solutions suffer from problems with the quality of the generated drawings. For a cartogram to be recognizable, it is important to preserve the global shape or outline of the input map, a requirement that has been overlooked in the past. To address this, our objective function for cartogram drawing includes both global and local shape preservation. To measure the degree of shape preservation, we propose a shape similarity function, which is based on a Fourier transformation of the polygons' curvatures. Also, our application is visualization of dynamic data, for which we need an algorithm that recalculates a cartogram in a few seconds. None of the previous algorithms provides adequate performance with an acceptable level of quality for this application. In this paper, we therefore propose an efficient iterative scanline algorithm to reposition edges while preserving local and global shapes. Scanlines may be generated automatically or entered interactively to guide the optimization process more closely. We apply our algorithm to several example data sets and provide a detailed comparison of the two variants of our algorithm and previous approaches.

Index Terms—Information visualization, visualization of geo-related information, continuous cartograms, value-by-area cartograms, visualization and cartography.

1 Introduction

Cartographers and geographers have used cartograms long before computers were available to make displays [1], [2], [3]. References date back as far as 1868.[1] A short historical overview can be found in [5]. The basic idea of a cartogram is to distort a map by resizing its regions by some geographically related parameter. Example applications include population demographics [6], election results [7], and epidemiology [8]. Because cartograms are difficult to make by hand, the study of programs to draw them is of interest.

A cartogram can be seen as a generalization of an ordinary map [9]. In this interpretation, an arbitrary parameter vector gives the intended sizes of the cartogram's regions, so an ordinary map is simply a cartogram with sizes proportional to land area. In addition to the classical applications mentioned above, a key motivation for cartograms as a general information visualization technique is to have a method for trading off shape and area adjustments. For example in a conventional choropleth map,[2] high values are often concentrated in highly populated areas and low values may be spread out across sparsely populated areas. Such maps therefore tend to highlight patterns in less dense areas where few people live. In contrast, cartograms display areas in relation to an additional parameter, such as population. Patterns may then be displayed in proportion to this parameter (e.g., the number of people involved) instead of the raw size of the area involved. In Fig. 1, a population-based cartogram is presented which shows that a cartogram can give a much different impression of overall trends as compared with the original map.

For a cartogram to be effective, a human must be able to quickly understand the displayed data and relate it to the original geographical model. Recognition, in turn, depends on preserving basic properties, such as shape, orientation, and contiguity. This, however, is difficult to achieve in the general case because—as we show in this paper—it is impossible even just to retain the original map's topology. Because the generation of contiguous cartograms by simultaneous optimization of these objectives is difficult, most currently available algorithms are very time-consuming.

1. See remarks on Levasseur in [4] on page 355.

2. Choropleth maps are divided into regions that are shaded according to the value of a variable for that region.

- *D.A. Keim and C. Panse are with the University of Constance, D-78457 Konstanz, Germany. E-mail: {keim, panse}@informatik.uni-konstanz.de.*
- *S.C. North is with AT&T Research Shannon Laboratory, Florham Park, NJ 07932. E-mail: north@research.att.com.*

Manuscript received 13 Nov. 2001; accepted 24 Jan. 2002.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 115361.

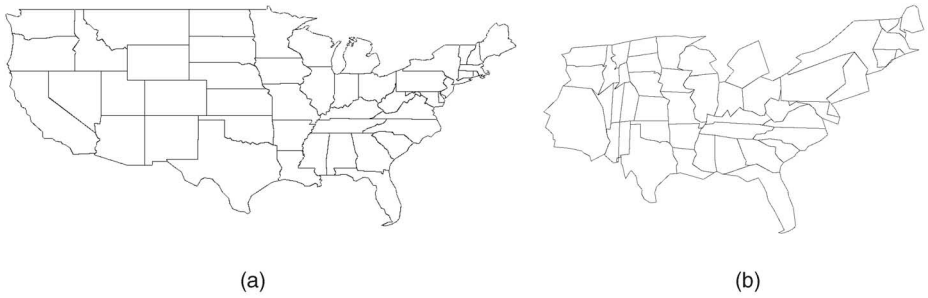


Fig. 1. Population-based cartogram (USA 1998). (a) Traditional map. (b) Population cartogram.

The ultimate goal of this work is to continuously display the behavior of an input parameter, particularly, its deviation from an expected value. Our application is to make dynamic cartograms for online data stream monitoring, such as display of network traffic or transaction event levels by country, state, and local regions. This requires cartogram generation on the fly and, to our knowledge, there is currently no competing algorithm with adequate speed for that.

The paper is organized as follows: Section 2 reviews previous work on cartogram drawing. Then, we define several variants of the problem and show that even simple ones are unsolvable in general. Because the feasible variants are NP-complete, heuristics are necessary to solve the problem. Based on some important observations, in Section 3, we develop a heuristic that uses scanline-based local repositioning of vertices with an explicit shape error control function to preserve both the global shape and the shape of interior polygons while providing sufficient speed for dynamic cartograms drawing. In Section 4, we present a number of application examples and provide a detailed comparison with previous approaches, showing the effectiveness and efficiency of our proposed algorithm. Section 5 summarizes our approach and discusses open issues.

2 Cartogram Drawing

In this section, we introduce a few basic concepts that underlie cartogram drawing. First, we formally define several variants of the problem. Then, we discuss the complexity and theoretical limitations of potential solutions and review the solutions which have been proposed in the literature. Finally, we outline some key observations that are the basis for a new, effective, and efficient solution.

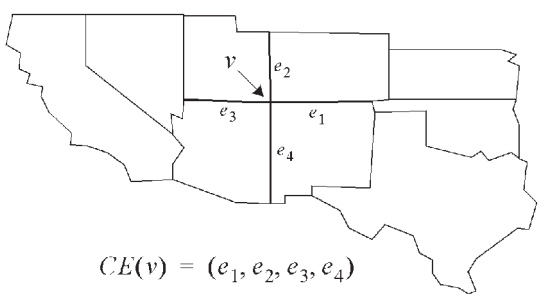


Fig 2. Cyclic order of edges.

2.1 Problem Definition

We assume that the input is a map defined by a set of connected simple polygons (a polygonal mesh) $\mathcal{P}$ and a parameter vector $\vec{X}$ that gives the desired values for the area of each polygon. Our goal is to generate contiguous cartograms and, therefore, the desired output also is a set of connected simple polygons $\overline{\mathcal{P}}$. Let $p \in \mathcal{P}$ $|p|$ denote the number of vertices, $A(p)$ the area, and $S(p)$ the shape of a polygon p, and $T(\mathcal{P})$ the topology of a set of polygons. Then, the ideal solution of the Contiguous Cartogram Drawing problem can be defined as:

Definition 1 (Contiguous Cartograms—Ideal Solution). *A contiguous Cartogram of a set of connected polygons $\mathcal{P} = \{p_1, \ldots, p_k\}$ with respect to a parameter vector $\vec{X} = \{x_1, \ldots, x_k\}$, $(\forall j \;\; x_j > 0)$, is a visualization of the transformed set of polygons $\overline{\mathcal{P}}$, where*

$$\begin{array}{lcll} T(\overline{\mathcal{P}}) & = & T(\mathcal{P}) & \text{(Topology Preservation)}, \\ S(\overline{p_j}) & = & S(p_j), \forall j = 1, \ldots, k & \text{(Shape Preservation)}, \\ A(\overline{p_j}) & = & \tilde{x_j}, \forall j = 1, \ldots, k & \text{(Area Resizing)}. \end{array}$$

The desired area $\tilde{x_j}$ of a polygon p_j is defined as

$$\tilde{x_j} = x_j \cdot \frac{\sum_{i=1}^{k} A(p_j)}{\sum_{j=1}^{k} x_j}.$$

To simplify the description, the following assumes that we have only one set of connected polygons (such as the continental United States) and not multiple unconnected sets (such as a world map).[3] Let v_j^i denote the ith vertex of polygon p_j, α_j^i the angle at the ith vertex, e_j^i the ith edge, $|e_j^i|$ the length of edge e_j^i, and $CE(v)$ the cyclic order of edges at vertex v (see Fig. 2).

If we assume that the transformed polygons have the same number of vertices (i.e., $|\overline{p_i}| = |p_i|$), then one way of formalizing the topology and shape preservation constraints is the following:

Definition 2 (Topology Preservation—Preservation of Connecting Vertices). *Topology preservation $T(\overline{\mathcal{P}}) = T(\mathcal{P})$ means that, for each vertex $v \in \overline{\mathcal{P}}$, the cyclic order of edges remains the same as in $\mathcal{P}$. More formally, $\forall v_j^i \in \mathcal{P}, j = 1, \ldots, k; i = 1, \ldots, |p_j| : \exists \overline{v_j^i} \in \overline{\mathcal{P}}, j = 1, \ldots, k; i = 1, \ldots, |\overline{p_j}|$*

3. These definitions may easily be extended to multiple polygonal meshes. The heuristic described in Section 2 operates on arbitrary maps.

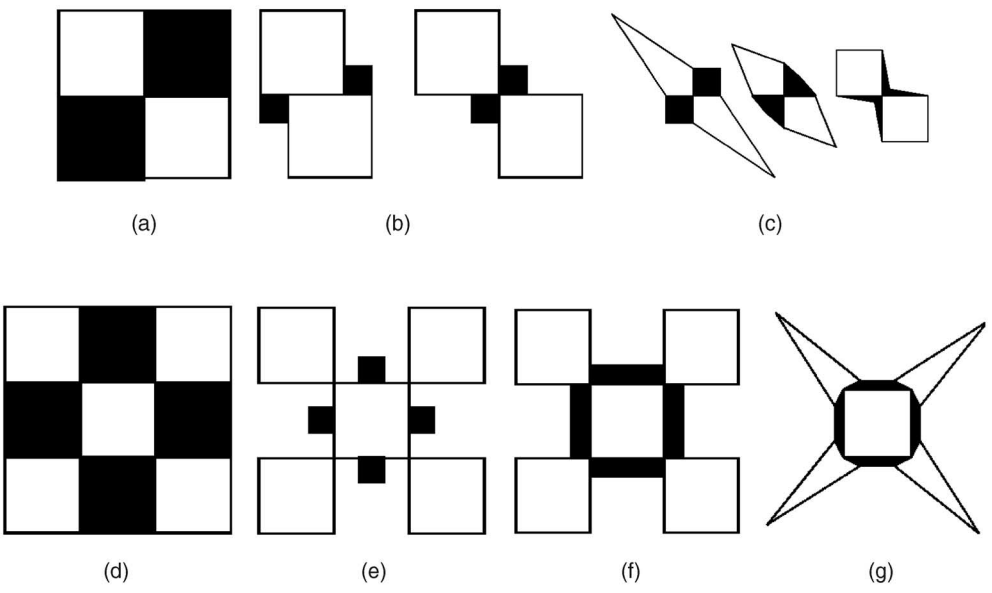


Fig. 3. Checker board example. (a) 2×2. (b) Relaxed topology. (c) Relaxed shape. (d) 3×3. (e) Relaxed topology. (f) Relaxed topology and shape. (g) Relaxed shape.

$$CE(v_i^j) = CE(\overline{v_i^j}).$$

If the cartogram construction algorithm does not provide a mapping to the original polygon set, topology preservation is difficult to test because, as a first step, the isomorphism problem between the two corresponding graphs must be solved. Graph isomorphism is a difficult problem and, therefore, efficient solutions have to maintain the topology of the original polygon mesh or provide a mapping to the original polygon mesh.

Definition 3 (Shape Preservation—Preservation of Edge Length Ratios and Angles). *Shape preservation $S(\overline{p_i}) = S(p_i)$ means that the edge length ratios of the polygons and the angles are preserved*

1.
$$\forall j = 1, \ldots, k \; \exists c_j \in \mathbb{R} : |\overline{e_j^i}| = c_j |e_j^i|,$$
$$i = 1, \ldots, |p_j|, e_j^i \in \mathcal{P}, \overline{e_j^i} \in \overline{\mathcal{P}},$$

2. $\forall j = 1, \ldots, k, \forall i = 1, \ldots, |p_j| : \overline{\alpha_j^i} = \alpha_j^i.$

Now, let us consider a simple example. Assume that we have a map with the topology of a checker board (cf. Fig. 3) and that we want to resize the map according to the color of the fields, scaling white fields by a factor of 1.5 and black fields by a factor of 0.5. This rescaling is impossible without changing the topology or shapes. So, in general, it is impossible to achieve the ideal solution. We state this observation in the following lemma.

Lemma 1 (Impossibility of the Ideal Solution). *The cartogram drawing problem of Definition 1 is unsolvable in the general case, i.e., there exist sets of polygons and parameter vectors such that it is impossible to obtain an ideal solution.*

Proof. Fig. 3 provides examples of sets of polygons which do not have ideal cartogram solutions according to Definition 1. □

To derive feasible variants of the problem, we need to relax some of the feature preservation conditions. If topology is the most important property to maintain, the only other conditions left to relax are the shape and area constraints. But, there are many ways to go about this. We can explore that in terms of two distance functions—an area distance function (which measures the distance of the area of a polygon from the desired size, typically, difference in area in the Euclidean plane) and a shape distance function (which measures the similarity of two shapes). Table 1 is an enumeration of possible constraints. The first column lists constraints that require a maximum distance for each polygon, the second column lists constraints that require a maximum distance for the sum of the distances of all polygons, and the third column lists minimum constraints for the sum of distances. By combining the different area and shape constraints in Table 1, we can construct variants of the cartogram drawing problem. A useful combination

TABLE 1
Possible Constraints for Cartogram Drawing

Constraints	single-polygon	all-polygon	minimum
Area	$\forall j : d_A(A(\overline{p_j}), \tilde{x_j}) \leq \varepsilon$	$\sum d_A(A(\overline{p_j}), \tilde{x_j}) \leq \varepsilon$	$\sum d_A(A(\overline{p_j}), \tilde{x_j}) \stackrel{!}{\rightarrow} \min$
Shape	$\forall j : d_S(S(\overline{p_j}), S(p_j)) \leq \varepsilon$	$\sum d_S(S(\overline{p_j}), S(p_j)) \leq \varepsilon$	$\sum d_S(S(\overline{p_j}), S(p_j)) \stackrel{!}{\rightarrow} \min$

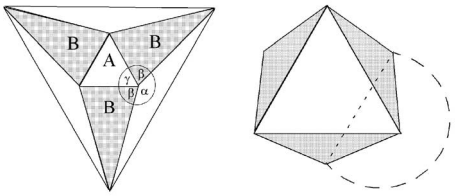


Fig. 4. Impossible cartogram drawing problem.

would be, for example, a restriction of the solution space to solutions where the shape of each polygon has at least a certain similarity to its original shape and the sum of all area differences is minimal. In the following, we discuss the different variants of the problem and their complexity.

2.2 Solvability and Complexity of the Problem

As shown by Lemma 1, in general it is impossible to find an ideal solution to the cartogram drawing problem. If we now consider the variants that may be constructed by a combination of the constraints in Table 1, it turns out that many of these are also unsolvable in the general case.

Lemma 2 (Impossibility of the Solution of Problem Variants). *Any variant of the cartogram drawing problem that involves the single-polygon area constraint or the all-polygon area constraint is unsolvable in the general case, i.e., there exist sets of polygons $\mathcal{P}$ and parameter vectors $\vec{X}$ such that, for any ε, the problem variants do not have a valid, topology-preserving solution.*

Proof. In Fig. 4a, we show an example of a symmetric cartogram consisting of seven polygons. If the parameter vector for scaling the polygons requires the white polygons to become larger and the black ones to become smaller, we can easily construct an impossible case. Due to the symmetric construction of the polygons, without loss of generality, we can assume that one angle $\gamma \leq \frac{\pi}{3}$. Thus,

$$\alpha = 2\pi - 2\beta - \gamma \geq 2\pi - 2\beta - \frac{\pi}{3}.$$

For the above-mentioned resize requirements (triangle A very large and triangles B very small), $\beta \to 0$ and, therefore,

$$\alpha \geq 2\pi - \frac{\pi}{3} = \frac{5}{3}\pi \quad \Longrightarrow \quad \alpha > \pi$$

and, thus, the topology cannot be preserved, as shown in Fig. 4b. □

This means that only variants of the problem that use the minimum-area condition are solvable and this is true for any combination with a shape constraint. The solvability is trivial to see since there is at least the identity solution, which has a perfect shape preservation but a rather bad value for the area difference. As the following lemma shows, the determination of the actual solution with the minimum area difference, however, is a computationally hard problem.

Lemma 3 (NP-Completeness of the Minimum-Area Problem). *Any variant of the cartogram drawing problem that involves the minimum-area condition is NP-complete.*

Proof. The proof depends on a constrained, simplified version of the cartogram problem called the *integer cartogram problem*. A proof sketch shows a reduction from integer cartograms to planar 3-SAT which is known to be NP-hard. The details are beyond the scope of this paper. □

In using this variant of the problem, one easily observes that there is little freedom to improve the second important parameter, namely, the shape. In most cases, the minimum area condition will provide some solution which is best optimized according to the area condition, but does not take the shape similarity into account. There might be, for example, a solution which much better preserves the shape, but is a little bit worse in area. To allow the shape constraint to have an impact on the solution, we have to adapt our constraints. In principle, there are two possibilities. The first is to determine the minimum area difference possible and then allow a certain maximum deviation from this minimum difference for finding the best shape. More formally, this may be defined as follows.

Definition 4 (Variant 1 of the Contiguous Cartogram Problem). *Given a set of polygons $\mathcal{P}$, a parameter vector $\vec{X}$, and an error value ε, the Contiguous Cartogram problem may be defined as a transformed set of polygons $\overline{\mathcal{P}}$ for which the following two conditions hold:*

$$(1) \sum_{j=1}^{k} d_A(\tilde{x_j}, A(\overline{p_j})) \leq \underset{\overline{\mathcal{P}}}{MIN} \left(d_A(\tilde{x_j}, A(\overline{p_j})) + \varepsilon\right.$$

$$(2) \sum_{j=1}^{k} d_S(S(p_j), S(\overline{p_j})) \stackrel{!}{\to} \min.$$

A second possibility is to normalize the area and shape distances and to use a weighted mean of the normalized distances as a combined optimization criterion.

Definition 5 (Variant 2 of the Contiguous Cartogram Problem). *Given a set of polygons $\mathcal{P}$, a parameter vector $\vec{X}$, and importance factors for the area and shape distances, the Contiguous Cartogram problem may be defined as the transformed set of polygons $\overline{\mathcal{P}}$ for which*

$$a \cdot \sum_{j=1}^{k} d_A(\tilde{x_j}, A(\overline{p_j})) + b \cdot \sum_{j=1}^{n} d_S(S(p_j), S(\overline{p_j})) \stackrel{!}{\to} \min_{a,b \geq 0}.$$

There are other meaningful and solvable variants of the problem which, for example, also include the single-polygon constraints (see Table 1). Most currently available algorithms try to solve the problem according to Definition 4 or Definition 5. This seems sufficient for some applications, but there are others where additional constraints seem necessary. In the following, we discuss some important observations which are the basis for our final definition and also the key to an efficient solution of the problem.

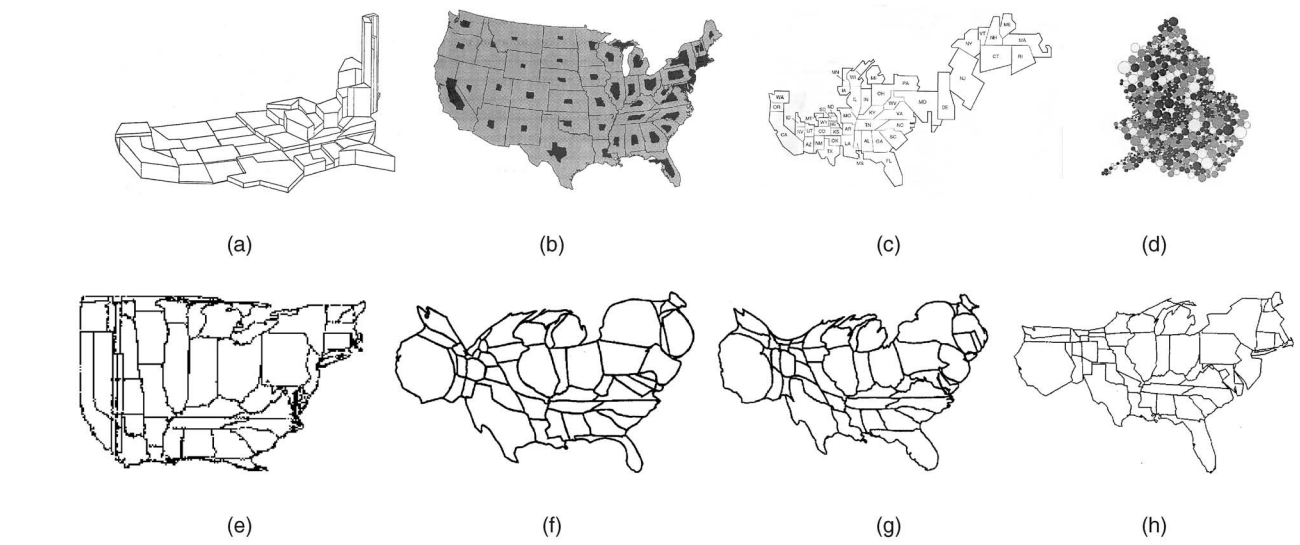


Fig. 5. Cartogram drawing methods—figures were first published in the given references, used by permission; [11] and [13] are official publications of the American Congress on Surveying and Mapping (ACSM). (a) 3D map [6]. (b) Noncontiguous cartogram [11]. (c) Non-topology-preserving cartogram designed by Bernard J. vanHamond [5]. (d) Circle cartogram [12]. (e) Tobler [13]. (f) Selvia [14]. (g) Zade and Tikunov [15]. (h) Kocmoud and House [7].

2.3 Related Work

Cartograms can made by contiguous or noncontiguous distortions. The noncontiguous case is much easier since the input map topology does not have to be preserved. As seen in Fig. 5, handmade noncontiguous cartograms have been made with overlapping or touching circles, by eliminating some of the original map's adjacencies, or even by drawing disconnected shapes over the original regions [10], [11].

Most previous attacks on automated drawing of contiguous cartograms do not yield results comparable to good handmade drawings. One reason, first pointed out by Dent [15], [16], is that straight lines, right angles, and other features considered important in human recognition of cartograms are obliterated. Methods that are radial in nature such as the conformal maps proposed by Tobler [6], the radial expansion method of Selvin et. al. [13], and the line integral method of Guseyn-Zade and Tikunov [14] do not provide the best possible results, since the shapes of the polygons are heavily deformed (see Fig. 5). Likewise, the pseudocartograms of Tobler expand the lines of longitude and latitude to achieve a least root mean square area error [12]. Very similar drawings are made by approaching the problem as distortion viewing by nonlinear magnification [17], [18], [19], [20]. Jackel [10] applies radial forces to change the size of polygons, moving the sides of each polygon relative to its centroid, but the solver runs slowly (taking 90 minutes to perform eight iterations on a map of six New England states of the US) and seems to have problems with nonconvex input polygons and with self-intersections in the output, which is consistent with our early experiments with a similar approach.

Another family of approaches operates on a grid or mesh imposed on the input map. The "piezopleth" method of Cauvin et al. transforms the grid by a physical pressure load model [21]. Dorling's cellular automaton approach trades grid cells until each region achieves the desired number of cells [22]. The combinatorial approach of Edelsbrunner and Waupotitsch [23] computes a sequence of piecewise linear homeomorphisms of the mesh that preserve its topology. While the first method is good at preserving the shape of the polygons, the other two methods allow a very good fit for area, but only poor shape preservation.

A synthesis of both approaches was recently described by Kocmoud and House, who propose a force-based model and alternately optimize the shape and the area error [7]. Although the results are better than most other methods, the complex optimization algorithm has a prohibitively high execution time. Kocmoud and House report a time of 18 hours for a medium-sized map with 744 vertices. In Fig. 5, we present population cartograms generated by several of the methods we have mentioned.

2.4 Important Observations

The current solutions have two major problems: First, the high time complexity of the algorithms restricts their use to static applications with a small number of polygons and vertices. Second, they have very limited shape preservation. Although the recent work by Kocmoud and House provides nice results, some effectiveness problems remain. One problem is the significant deformation of the global shape. In evaluating the different heuristic solutions which have been proposed so far, we found that insufficient preservation of the global shape is one of their major problems. In our experience, however, the global shape is one of the most important factors for cartograms to be effective and it is certainly at least as important as the preservation of interior polygon shapes. In our definition of cartogram drawing, besides the shape and area constraints of Table 1, we therefore explicitly include a global shape constraint which may be again either a single-polygon, all-polygon, or minimum constraint for the global shape(s).[4] If $G_r(\mathcal{P})(r = 1 \ldots l, l \leq k)$

4. Note that there may be multiple global shapes as they occur, for example, on a world map.

TABLE 2
Global Polygon Constraints for Cartogram Drawing

single-polygon	all-polygon	minimum
$\forall r : d_S(S(G_r(\overline{\mathcal{P}})), S(G_r(\mathcal{P}))) \leq \varepsilon$	$\sum d_S(S(G_r(\overline{\mathcal{P}})), S(G_r(\mathcal{P}))) \leq \varepsilon$	$\sum d_S(S(G_r(\overline{\mathcal{P}})), S(G_r(\mathcal{P}))) \stackrel{!}{\rightarrow} \min$

denotes the set of global polygons which may be derived from the set of polygons $\mathcal{P}$. The global shape constraints may formally be described as given in Table 2. Our final definition of the cartogram drawing problem uses a weighted minimum of area, shape, and global shape constraints.

Definition 6 (Contiguous Cartogram Problem). *Given a set of polygons $\mathcal{P}$, a parameter vector $\vec{X}$, and importance factors for the area, shape, and global shape constraints a, b, and c, the Contiguous Cartogram problem may be defined as a transformed set of polygons $\overline{\mathcal{P}}$ for which*

$$a \cdot \sum_{j=1}^{k} d_A(\tilde{x_j}, A(\overline{p_j})) + b \cdot \sum_{j=1}^{k} d_S(S(p_j), S(\overline{p_j})) + c \cdot \sum_{r} d_S(S(G_r(\mathcal{P})), S(G_r(\overline{\mathcal{P}}))) \stackrel{!}{\rightarrow} \min_{a,b,c \geq 0}.$$

Let us now focus on some important observations which are crucial for an efficient solution of the problem. One important observation is that, in practice, only very few vertices are actually important for defining the shapes of the polygons. In considering the US map, for example, we found that, in addition to a restricted number of outer vertices, only a limited number of interior vertices are actually relevant. Note also that the importance of polygons and their vertices largely depends on their size (which is directly related to the parameter vector) and on the length of the edges and the angles between them. In our proposed algorithm, we give special consideration to these facts and determine the importance of vertices based on these observations. A second observation is that—in order to obtain good results—the shape error has to be controlled explicitly, which is not done sufficiently in previous approaches. A last observation is that the high time complexity of most algorithms proposed previously is due to a complex and time-consuming optimization. In most cases, however, it is possible to locally reposition vertices and improve the area error while retaining the shape. To obtain good solutions, our algorithm iteratively repositions vertices based on scanline-defined locality measures with an explicit shape error control function.

3 The Cartogram Drawing Algorithm

The main objective of our proposed Cartogram Drawing Algorithm is fast generation of cartograms of acceptable quality. Because input maps often have far more vertices than are needed to compute good cartograms, the first step is an intelligent decimation. This is followed by the central heuristic, scanline-based repositioning of vertices. We first reposition vertices of the global polygon(s) and then interior vertices. Scanlines can be restricted to vertical and horizontal lines determined automatically or may be arbitrarily positioned line segments of any length, entered interactively. This follows the human-guided local search paradigm proposed by Anderson et al. [24]. In each step, the shape of the modified polygon mesh is controlled by the shape error function. The last step is fitting the undecimated polygons to the decimated mesh to obtain the output cartogram. By exploiting the potential for precomputation and fast local optimization, our algorithm runs quickly enough to support dynamic displays with high update rates on maps having dozens of polygonal regions.

3.1 Reduction Algorithms

Edge reduction algorithms have been used previously in solving the cartogram problem. Kocmoud and House, for example, use a simplification of the polygons to speed up cartogram generation. Our algorithm, however, is different. As mentioned in Section 2.4, preserving the global shape is very important in making recognizable cartograms. Our decimation algorithm takes this into account by simplifying the global and inner polygons differently.

3.1.1 Reduction of Global Polygon

A key observation is that the importance of the vertices of a polygon can vary greatly. Vertices on angles near 180 degrees and those with short edges make almost no noticeable difference in the shape of a polygon, while others with sharp angles or long edges have a significant effect. The basic idea of the global polygon reduction algorithm is to rate the importance of each vertex according to these criteria. Then, iteratively, the least important vertices are removed. To maintain the topology, only vertices that do not belong to multiple polygons are removed. To formalize the global reduction algorithm, we first define the notion of a vertex's importance as

$$I(v) = Sig(\alpha^v) \cdot |e_1^v| \cdot |e_2^v|, \tag{1}$$

where e_1^v and e_2^v are the two edges of vertex v and $Sig(\alpha^v)$ is a function denoting the significance of the angle α^v at vertex v. The significance function $Sig(\alpha)$ is important because different angles have a specific impact on the shape of the polygons. Sharp angles and angles close to 90° are more important than obtuse angles (c.f. [15], [16]) and the significance function therefore assigns higher values to sharp angles and lower values for obtuse angles. For our algorithm, we use

$$Sig(\alpha) = \sum_{\mu \in \{0^\circ, 90^\circ, 270^\circ, 360^\circ\}} \exp^{\frac{(\alpha-\mu)^2}{2\sigma^2}} \tag{2}$$

as the significance function. This function has peaks for $\alpha = 0^\circ, 90^\circ, 270^\circ, 360^\circ$ and is close to zero for $\alpha = 180^\circ$. The

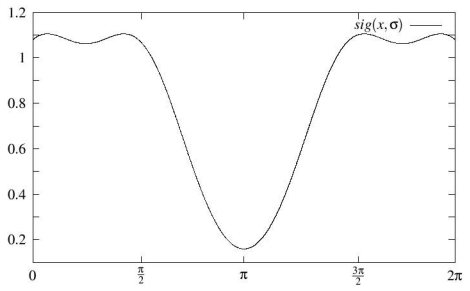


Fig. 6. Significance function.

function is defined for $\alpha \in [0°, 360°]$ and σ is chosen to be $0.2 \cdot \pi$. Fig. 6 shows a plot of this function.

To formalize the global reduction algorithm, we first define the global polygon as a subset of the vertices of $\mathcal{P}$. For each polygon p_j, $j = 1 \ldots k$, the portion gp_j of the global polygon $\mathcal{GP}$ can be defined as

$$gp_j = \{v \in p_j : |edges(v)| > |polygons(v)|\}.$$

The global polygon is defined as $\mathcal{GP} = \bigcup_{j=1\ldots k} gp_j$. The algorithm for the reduction of the global polygon is shown in Fig. 7. Note that vertices are only considered for removal if they do not belong to multiple polygons (see initialization of V in Fig. 7) and they are only removed if the induced area difference is smaller than a given constant $MaxAreaDiff$. Note also that the area $A_s(p)$ of a polygon p is determined as if the polygon is perfectly scaled according to the parameter vector X and the area difference $\|A_s(p_1) - A_s(p_2)\|$—the subscript s of A_s stands for scaled—is defined as

$$\|A_s(p_1) - A_s(p_2)\| = (A_s(p_1) \cup A_s(p_2)) \setminus (A_s(p_1) \cap A_s(p_2)).$$

3.1.2 Reduction of Inner Polygons

To position interior vertices, we can again use iterative vertex removal. A more efficient alternative is based on the observation that, for most maps, only the connecting interior vertices are important. Instead of iteratively removing unimportant vertices, we therefore take a more direct approach and remove all vertices not common to more than two polygons (nonconnecting vertices). In some cases, the shape deformation and area error introduced by this reduction is unacceptably high. We therefore reintroduce a few additional vertices. See Fig. 8 for the complete algorithm.

Fig. 9 shows an example polygon (cf. Fig. 9a), a reduced polygon of its interior vertices common to more than two polygons (cf. Fig. 9b), and the final polygon after reintroducing a few additional vertices (cf. Fig. 9c). In practice, only few polygons need the additional vertices,

ReduceGlobalVertices($\mathcal{P}$, $\mathcal{GP}$, *MaxAreaDiff*) {
  $V = \{v \in gp_j \mid v \notin gp_m \text{ for } m \neq j\}$ // Only consider vertices if they are not part of multiple polygons
  do {
    $\bar{v} = \{v \in V \mid \overset{MIN}{v \in V} I(v)\}$ // Determine the least important vertex
    $j = \{j \in \{1 \ldots k\} \mid \bar{v} \in p_j\}$ // Determine the polygon containing the least important vertex
    if $(||A_s(p_j \setminus \{\bar{v}\}) - A_s(p_j)|| \leq MaxAreaDiff)$
      $p_j = p_j \setminus \{\bar{v}\}$;
    $V = V \setminus \{\bar{v}\}$
  } **while** $V \neq \{\}$
}

Fig. 7. Reduction of global vertices.

ReduceInnerVertices($\mathcal{P}$, $\mathcal{GP}$, *MinAreaImpr*) {
  $I\mathcal{V} = \bigcup_{j=1\ldots k}\{v \mid v \in p_j\} \setminus \bigcup_{j=1\ldots k}\{v \mid v \in gp_j\}$ // all interior vertices
  $\mathcal{CV} = \{v \in I\mathcal{V} : |edges(v)| > 2\}$ // connecting interior vertices
  forall $p_j \in \mathcal{P}$ $p_j = \{c \in p_j \mid c \notin (I\mathcal{V} \setminus \mathcal{CV})\}$; // remove all non-connecting vertices
  forall $p_j \in \mathcal{P}$ { // reintroduce important non-connecting vertices
    // Assume that the adjacent inner vertices $\{v_1, \ldots, v_m\} \in (I\mathcal{V} \setminus \mathcal{CV})$ have been removed between
    // vertex v_s and v_e of polygon p_j in step 1.
    forall $((v_s, v_e)$ removed in step 1) {
      $\overline{n} = \overset{MIN}{n \in 1\ldots m}\big(\overset{MAX}{(v_{i_1},\ldots,v_{i_n}) \in \{v_1,\ldots,v_m\}, v_{i_l} \neq v_{i_k}} ||A_s(p_j \cup \{v_{i_1}, \ldots, v_{i_j}\}) - A_s(p_j)||\big)$
      $p_{tmp} = p_j \cup \{v_{i_1}, \ldots, v_{i_{\overline{n}}} : \overset{MAX}{(v_{i_1},\ldots,v_{i_{\overline{n}}}) \in \{v_1,\ldots,v_m\}, v_{i_l} \neq v_{i_k}} ||A_s(p_j \cup \{v_{i_1}, \ldots, v_{i_j}\}) - A_s(p_j)||\}$;
      if $(||A_s(p_{tmp}) - A_s(p_j)|| \geq MinAreaImpr)$ $p_j = p_{tmp}$;
    }
  }
}

Fig. 8. Reduction of interior vertices.

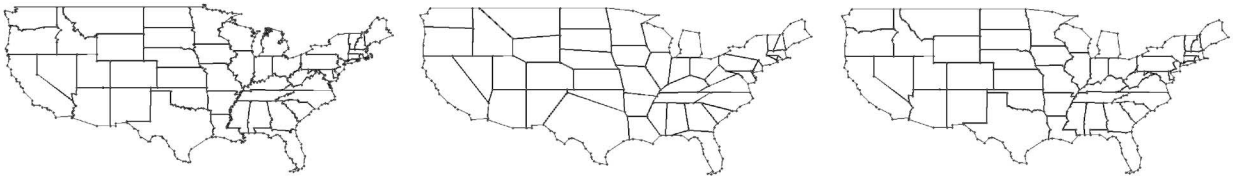


Fig. 9. Map reduction (US state map). (a) Original map (743 vertices). (b) Without nonconnecting vertices (204 vertices). (c) Reduced map (343 vertices).

so the likelihood of reintroducing vertices that were removed is low (see Fig. 9).

3.2 The CartoDraw Algorithm

The main idea of the *CartoDraw* algorithm is to incrementally reposition the vertices along a series of scanlines. A scanline is a line segment of arbitrary length and position. Each scanline defines a scan section, orthogonal to the scanline. All points within a scan section are repositioned in a single step. For each section on a scanline, a target scaling factor for each of its polygons is determined according to their area error factors. Vertices are then repostioned according to the polygon scaling factors and distances to the scanline. The repositioning may be parallel or orthogonal to the scanlines. If the shape error introduced by applying a scanline exceeds some threshold, its candidate vertex repositionings are discarded.

Scanlines should be applied to parts of the map where the area error is large and there is still potential for improvement. A simple approach to scanline generation is to use horizontal and vertical line segments positioned on a regular grid. Significantly better results can be obtained by a manual scanline placement, guided by the shape of the input polygons and the local potential for improvement. Note that the incremental repositioning of vertices per scanline application is intentionally small compared to the expected change in area. This means the same scanline may need to be applied many times to make large adjustments in an area.

Before we describe the main CartoDraw algorithm, we first introduce its three main components—the *area error* function, the *shape similarity* function, and the *scanline algorithm.*

3.2.1 Area Error Function

The objective of cartogram generation is to obtain a set of polygons where the area of the polygons corresponds to values given in a data vector X. In each step of the algorithm, the area error function is needed to determine the reduction of the area error achieved by applying a given scanline. The relative area error E^j_{rel} of a polygon p_j can be computed as:

$$E^j_{rel} = \frac{\left|A^j_{desired} - A^j_{actual}\right|}{A^j_{desired} + A^j_{actual}}. \tag{3}$$

Hence, the area error for the set of polygons $\mathcal{P}$ is defined as

$$E^{\mathcal{P}}_{rel} = \sum_{j=1}^{k}\left(E^j_{rel}\cdot\frac{A^j_{desired}}{\sum_{j=1}^{k}A^j_{desired}}\right). \tag{4}$$

3.2.2 Shape Error Function

In addition to reducing area error, the cartogram generation process also aims at retaining the original shapes. To assess shape preservation, we need a shape similarity function that compares the new shape of a polygon with its original shape. Defining a useful shape similarity function is in itself a difficult problem since the similarity measure should be translation-invariant, scale-invariant, and at least partially rotation-invariant. From CAD research, it is known that the Euclidean distance in Fourier space is useful for measuring shape similarity [25], [26]. To gain invariance against translation, rotation, and scaling, we use the Fourier transformation of the differential geometric curvature of the polygons, instead of the polygons themselves, and normalize the arc length of the polygons to 2π. Using the curvature guarantees translation and rotation-invariance and normalizing the arc length guarantees scale-invariance.

In the following, we assume that the polygons are transformed into a normalized parameterized polygon contour function $p : [0, 2\pi] \rightarrow \Re^2$. Then, we can define the curvature C of the polygons as

$$C : (\Re \rightarrow \Re^2) \longrightarrow (\Re \rightarrow \Re^2).$$

The Fourier transformation F is a transformation

$$F : (\Re \rightarrow \Re^2) \longrightarrow \Re^d,$$

determining the Fourier coefficients for a given curvature function in d-dimensional Fourier space. The shape similarity of two polygons p and $\overline{p}$ can then be defined as

$$d_S(S(p), S(\overline{p})) = d_{Euclid}(F(C(p)), F(C(\overline{p}))).$$

In the following, we describe the curvature transformation C and the Fourier transformation F in more detail.

Determining the Curvature of a Polygon. In general, the curvature of a polygon defined as a parameterized function is mathematically undefined because the second derivative is not continuous. We can avoid this problem by approximating the polygon by replacing each vertex with very small circular arcs, as shown in Fig. 10. This yields a new geometric object of which the first derivative is continuous. The curvature of this structure is defined in sections; concatenating these sections, we obtain the curvature as a square wave function.

To describe the curvature transformation in more detail, let us focus on two consecutive edges, e_{i-1} and e_i. These

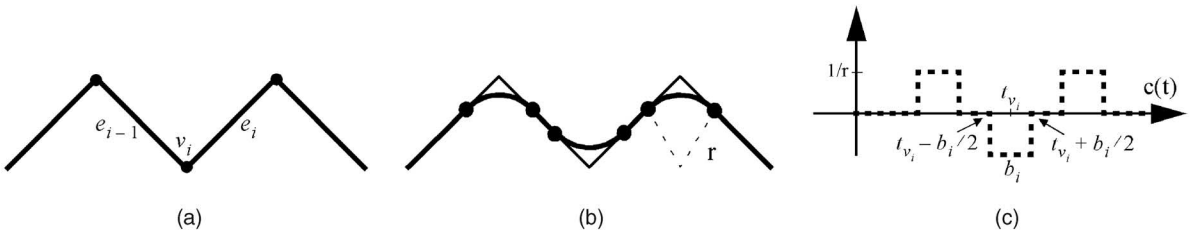


Fig. 10. Curvature approximation of the polygon. (a) Original polygon. (b) Approximated polygon. (c) Curvature of approximated polygon.

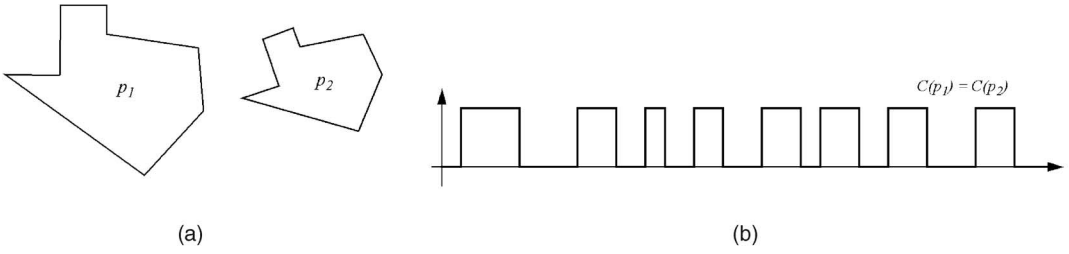


Fig. 11. Curvature transformation. (a) Two polygons. (b) Curvature of the two polygons.

edges coincide in vertex v_i with an angle α_i. For the polygon containing v_i, we may easily compute the curvature function $c_i(t)$, describing the differential geometric curvature of the approximated polygon because the curvature of a circle segment with radius r is a constant function $\frac{1}{r}$ and the curvature of a straight line is a constant zero function. We may calculate the arc length of the circle segment substituting vertex v_i by $b_i = |\alpha_i| \cdot r$. For $c_i(t)$, we therefore obtain

$$c_i(t) = \begin{cases} \frac{1}{r} & \text{if } (t_{v_i} - b_i/2 > t > t_{v_i} + b_i/2) \\ 0 & \text{otherwise.} \end{cases}$$

The curvature of an arbitrary polygon p is $c(t) = \sum_{k=0}^{|p|-1} c_k(t)$. Fig. 10c shows the graph of the curvature function $c(t)$ for the approximation of the polygon section in Fig. 10a. Fig. 11 shows the curvature functions for two polygons which are identical under translation-invariance, rotation-invariance, and scale-invariance.

The approximation of the original polygon and, in particular, the choice of r, influences the curvature function. If we reduce the radius r of the circle segment, $\frac{1}{r}$ will be increased while b_i will be decreased. This causes $c(t)$ to become more narrow and the amplitude of square waves to become higher, while the approximation of the polygon converges against the polygon itself. On the other hand, $c(t)$ becomes difficult to handle numerically. An adequate value for r which has proven useful for our application is $\frac{\pi}{50}$ for polygons with a normalized length of 2π. As our experiments show, the similarity function is quite robust against a suboptimal choice of r as long as r is smaller than half of the length of the shortest edge since, otherwise, individual square wave functions may overlap.

Fourier Transformation. The next step is computing the Fourier transformation F of the curvature. The principle of the Fourier transformation is to approximate a function by summing up sine and cosine functions with certain parameters. The quality of the approximation is improved by increasing the degree d of the Fourier approximation, which means to successively sum up $\cos(x), \sin(x), \cos(2x), \sin(2x), \ldots, \cos(kx), \sin(kx)$. More formally, the Fourier approximation of a function f with a period of 2π is defined as

$$F(x) = \frac{a_0}{2} \sum_{k=1}^{d} (a_k \cos(kx) + b_k \sin(kx)),$$

where the coefficients a_k and b_k are defined as

$$a_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(kx) dx \quad \text{and} \quad b_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(kx) dx.$$

In general, integrals of the form $\int f(x) \sin(x) dx$ are difficult to solve analytically. For the special case where $f(x)$ is a square wave function, however, the integral can be easily determined. Let us assume that $f(x)$ has a value of $\frac{1}{r}$ in the interval $[u, v]$ and is zero elsewhere. Since the value of the integral is zero outside of $[u, v]$, we just have to integrate from u to v. Therefore, we are able to calculate a_k and b_k as:

$$a_k = \frac{1}{\pi k r} (\sin(kv) - \sin(ku)) \text{ and } b_k = \frac{1}{\pi k} (\cos(kv) - \cos(ku)).$$

To determine the Fourier coefficients of the curvature function $c(t)$ of the whole polygon p, we only have to sum up the above formula $c_i(t)$ for all vertices v_i of the polygon. We obtain the following formulas for the Fourier coefficients:

$$a_k = \frac{1}{\pi k r} \sum_{i=0}^{|p|-1} \frac{\alpha_i}{|\alpha_i|} \left(\sin\left(k \left(t_i + \frac{|\alpha_i r|}{2} \right) \right) - \sin\left(k \left(t_i - \frac{|\alpha_i r|}{2} \right) \right) \right)$$

$$b_k = -\frac{1}{\pi k r} \sum_{i=0}^{|p|-1} \frac{\alpha_i}{|\alpha_i|} \left(\cos\left(k \left(t_i + \frac{|\alpha_i r|}{2} \right) \right) - \cos\left(k \left(t_i - \frac{|\alpha_i r|}{2} \right) \right) \right).$$

The calculation of a_k and b_k can be done in $O(|p|)$ time, and the calculation of all coefficients can be done in $O(|p| \cdot d)$, where d is the degree of the Fourier sum. Note

Scanline$(\mathcal{P}, \tilde{X}, sl)$ {
  foreach $sp \in \{sp_i \mid sp_i = s + \frac{i}{n}(t-s), i = (0 \ldots n)\}$ { // set of scanline section points which lie on sl
    $l_i = sp \perp \vec{sl}$ // section line
    $V_i = \{v \in V \mid |l_i - v| \leq \xi\}$ // vertices v in scanline section i, i.e. closer than ξ to l_i
    $SP_i = \{j \in \mathbf{N} \mid \exists v : v \in V_i \wedge v \in p_j\}$ // numbers of polygons which contain at least one vertex from V_i
    $SF_i = const \cdot \sum_{r \in S_i} \left(\frac{\tilde{x}_r - A(p_r)}{\tilde{x}_r + A(p_r)} \cdot \frac{\tilde{x}_r}{\sum_{l \in S_i}^{n} \tilde{x}_l} \right)$ // scaling factor
    foreach $v \in V_i$ {
      if $(direction = scanline)$
        $o(v) = \frac{\overrightarrow{v \perp l_i}}{|\overrightarrow{v \perp l_i}|} \cdot \frac{sl}{|sl|}$
      else
        $o(v) = \frac{\overrightarrow{v \perp sl}}{|\overrightarrow{v \perp sl}|} \cdot \frac{l_i}{|l_i|}$; // direction = section line l_i
      $v = v + SF_i \cdot o(v)$
    }
  }
}

Fig. 12. Scanline algorithm.

Fig. 13. Scanline algorithm example. (a) Scanline section. (b) Scanline section with limited range.

that we are able to compute the coefficients of the Fourier sum analytically and therefore do not run into numerical problems such as finding the right sample rate. Experimental results show that the Fourier transformation provides a good approximation of the polygons and their curvature function even for rather small d. For a detailed discussion of Fourier theory, see [27].

3.2.3 Scanline Algorithm

The key to the *CartoDraw* algorithm is the scanline heuristic, which incrementally repositions vertices along scanlines. A scanline sl is a line segment of arbitrary position and length and is partitioned into n portions of length $\frac{|sl|}{n}$. The scanline section points sp_i $(i = 0 \ldots n)$ define $n+1$ sections of the polygon mesh, which are orthogonal to the scanline (see Fig. 13a). In one step of the scanline algorithm, all vertices $v \in V_i$ within a certain distance $(\xi = \frac{|sl|}{2n})$ of l_i are considered for incremental repositioning (see Fig. 13a). Let SP_i be the set of polygons (by index number) which have at least one vertex in scanline section i $(i = 0 \ldots n)$. Then, the scaling factor SF_i is determined according to the area error of all polygons p in section i:

$$SF_i = const \cdot \sum_{r \in S_i} \left(\frac{\tilde{x_r} - A(p_r)}{\tilde{x_r} + A(p_r)} \cdot \frac{\tilde{x_r}}{\sum_{l \in S_i}^{n} \tilde{x_l}} \right).$$

Next, we have to determine the direction $o(v)$ of a vertex v and apply the scaling factor SF_i to reposition the vertex. The repositioning can be done either in the direction of the scanline $(direction = scanline)$ or in the direction of the section line l_i. The algorithm is shown in Fig. 12. Note that the scanline sections always span the full range orthogonal to the scanline of the polygon net. If we want to restrict the changes to be local in both directions, we can optionally limit the polygons considered to those close to the scanline (see Fig. 13b). This option is not reflected in the algorithm shown in Fig. 12.

3.2.4 CartoDraw Main Algorithm

Having defined the components of the CartoDraw algorithm, we can now describe its main procedure. The algorithm assumes as input a set of polygons $\mathcal{P}$, a scaling vector of the desired statistical parameter $\tilde{X}$, and a set of scanlines SL, which can be determined automatically or manually (see Section 3.2.5). Output is the modified set of polygons $\mathcal{P}$ which describes the cartogram. The algorithm works as follows (see Fig. 14). For each scanline, the algorithm applies the scanline transformation and checks the results. If the area difference E_{rel} introduced by the scanline transformation is below a certain threshold ε_A and the shape distortion is below a certain threshold ε_s, then the changes are retained and, otherwise, discarded. Then, the

```
CartoDraw(P, X̃, SL){
  do{
    AreaError = AreaDist(P, X̃);
    foreach (sl ∈ SL){
      P̄ = scanline(P, X̃, sl);
      if (ShapeDist(P, P̄) < ε_s and |AreaError − AreaDist(P̄, X̃)| < ε_A)
        P = P̄;
    }
  }while (0 ≤ AreaError−AreaDist(P, X̃) ≤ ε);
}
```

Fig. 14. CartoDraw algorithm.

algorithm proceeds with the next scanline until all scanlines are applied in the same way. At this point, the algorithm checks whether, in applying all scanlines, an improvement of the area error has been obtained. If this is the case, the algorithm applies all scanlines again and repeats the entire procedure until no further improvement is reached (area improvement below ε). Since the area error improvement must be positive and above the threshold ε in each iteration, the area error is monotonously decreasing and termination of the algorithm is guaranteed. Note that, in applying an individual scanline, we allow the algorithm to potentially increase the area error to allow escaping local optima. Also, notice that, after applying a scanline, all the other ones remaining to be processed must be transformed as well, so that they correspond properly to the transformed map.

3.2.5 Automatic versus Interactive Scanline Placement

So far, we assumed that the set of scanlines SL used by the CartoDraw algorithm are given. In this section, we discuss how the scanlines can be obtained. Our implementation allows them to be defined automatically or interactively. The *automatic generation of scanlines* uses a fixed grid of horizontal and vertical scanlines (see Fig. 15a). The grid's resolution can be varied, but, within reason, this has only a minor influence on the result. Because only those scanlines that do not induce a higher shape and area error are applied, generating many useless scanlines causes a potential loss in efficiency, but does not affect the quality of the result.

The best cartograms seem to be obtained when the scanlines are well adapted to the shape of the input polygons and are placed in areas with a high potential for improvement. Automatic placement based on these criteria is difficult to achieve, so we allow the user to *interactively position the scanlines* depending on the result of the previous steps. The user usually starts with scanlines in regions with a high area error. The scanlines seem to work best if they are positioned such that they are either parallel or orthogonal to the contour of the global polygon (see Fig. 15b).

Once the scanlines are specified for a given polygon mesh, we can store and reapply them later to different data on the same map. This makes it practical to generate a continuous time series of cartograms, without user interaction in each step. While the generated cartogram may not be as good as if the scanlines were specified anew, the results seem sufficient for many applications. In our experience, manual positioning of scanlines is not difficult and can be done quickly. Fig. 15b shows an example of a set of manually placed scanlines. It took about 5 minutes to enter these scanlines. Note that parts of the map that need large changes have many scanlines of varying lengths, while other parts have hardly any scanlines.

Fig. 16 shows a few intermediate steps of incrementally applying the automatic scanlines shown in Fig. 15a to the US population cartogram problem. The area error is encoded in red for polygons that should be smaller and blue for polygons that should be larger. The algorithm quickly provides nice results in some areas which are well adapted to horizontal and vertical scanlines (e.g., the midwestern states and New England states). In other areas, the improvement that can be reached by global horizontal and vertical scanlines seems limited (e.g., California or New York and Pennsylvania).

Fig. 17 shows a similar sequence applying the interactive scanlines shown in Fig. 15b to the US population cartogram problem. The local nonorthogonal scanlines specified interactively allow a better adaptation of the algorithm to the shape and area error of the polygons and, therefore, provide better results. The residual area error obtained from interactive scanline placement is much lower than that obtained by automatic placement, with both having about the same shape error. A detailed comparison of shape and area error of the automatic versus interactive scanline placements is provided in Section 4.2.

4 Application and Evaluation of the Algorithm

The algorithm as described in the previous section has been implemented in C using the LEDA library [28] and run on several different example applications. Unless noted otherwise, the tests were performed on a 1 GHz Pentium computer with 128 Mbytes of main memory. In this section, we report and discuss the results and compare the effectiveness and efficiency of the different approaches. Although our focus is on efficiency, the examples show that the proposed *CartoDraw* algorithm also provides results of good quality. For most of the examples, we will continue to use a state map of the continental US as a running example.

4.1 Comparison with Previous Methods

Fig. 18 shows population cartograms generated by our algorithm and by the technique proposed by Tobler [12] and by Kocmoud and House [7]. A visual comparison shows that our approach offers comparable if not better visual results, with the geography of the United States being clearly perceivable.

To evaluate the results analytically, Fig. 19a shows the total area error E_{rel} for all three approaches. Fig. 19a shows that our proposal can provide better results than the complex optimization-based approach by Kocmoud and House [7]. Since the total area error is basically an average over the state-wise area error, in Fig. 19b, we show the area error state by state, sorted according to the area error. Fig. 19b reveals that, for most states, our approach provides significantly less area error than the Tobler cartogram[5] and

5. Note that the Tobler cartogram was not optimized according to our error measure, which puts higher weights on polygons that should become large. Since many of the polygons with large weights still have a large area error in the Tobler cartogram, the overall improvement of E_{rel} by the Tobler cartogram is low.

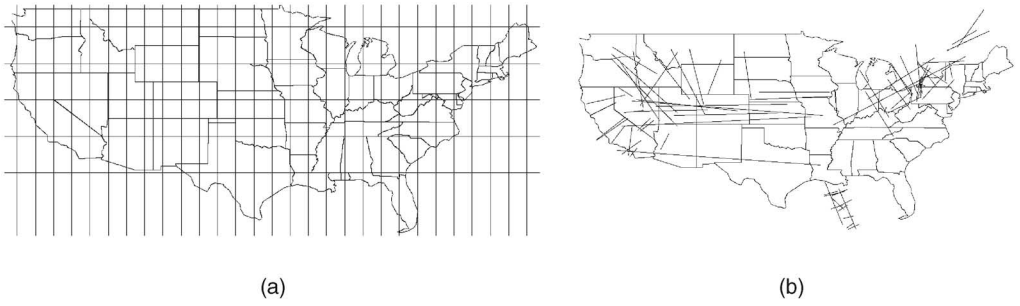


Fig. 15. Automatically versus interactively placed scanlines. (a) Automatic scanlines. (b) Interactive scanlines.

Fig. 16. Cartogram construction steps with automatically placed scanlines. (a) Step 0. (b) Step 12. (c) Step 21. (d) Step 30. (e) Step 36.

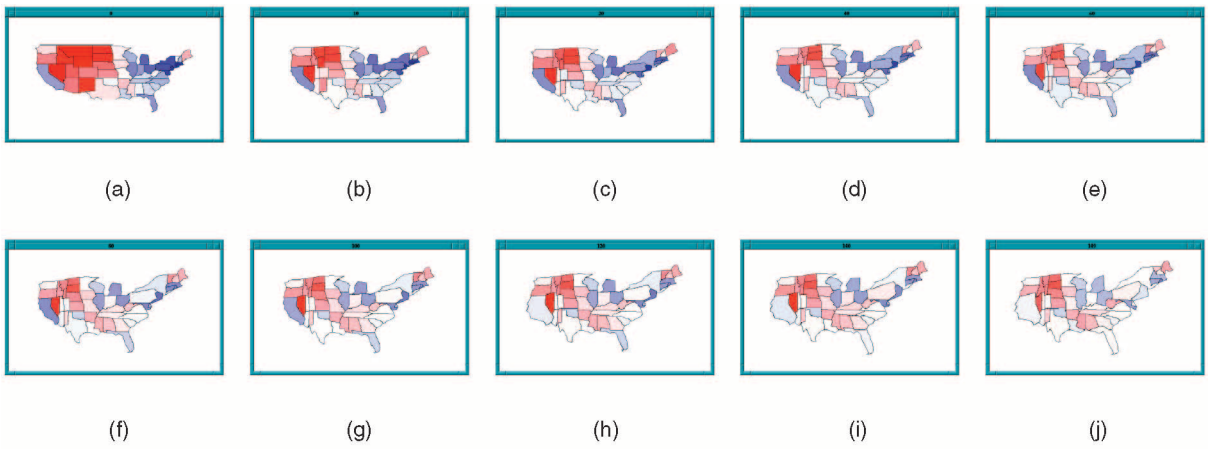


Fig. 17. Cartogram construction steps with interactively placed scanlines. (a) Step 0. (b) Step 10, (c) Step 20. (d) Step 40. (e) Step 60. (f) Step 80. (g) Step 100. (h) Step 120. (i) Step 140. (j) Step 149.

a slightly less area error than the Kocmoud and House cartogram, with few exceptions.

In terms of efficiency, our approach is much faster than existing techniques. While previous approaches that attempt to preserve shape and topology need hours or even days to compute a solution, our implementation runs in only seconds. Fig. 19c shows that our scanline-based heuristic needs about 25 seconds, while the Kocmoud and House approach needs about 16 hours, making our approach about 2,000 times faster.[6]

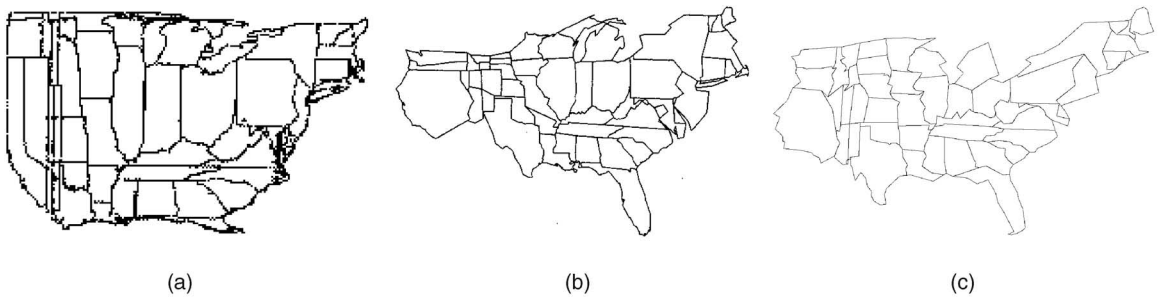
4.2 Comparison of the *CartoDraw* Variants

One important aspect of the *CartoDraw* algorithm is the specification of the scanlines. As mentioned previously, we allow scanlines to be determined automatically or interactively. In this section, we compare these two approaches with respect to effectiveness (quality of the results) and efficiency (time needed to produce the results).

6. The comparison assumes that both algorithms run on a 120 MHz computer with 32 MByte RAM.

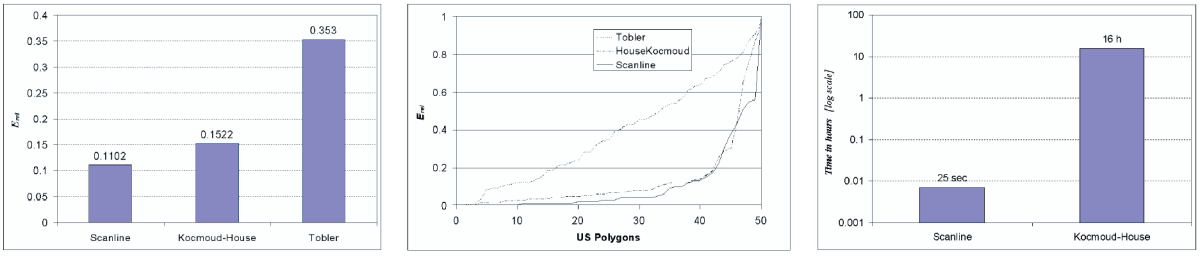
4.2.1 Quality

In Fig. 20, we show the original US map (Fig. 20a) with the results of the *CartoDraw* algorithm using automatically generated scanlines (Fig. 20b) and interactively generated scanlines (Fig. 20c). Both approaches provide good quality cartograms. Fig. 20 shows that the area error (E_{rel}) is much lower for the interactive scanlines, but shape distortion seems to be higher. To measure shape distortion, we use a Fourier-based shape similarity function (see Section 3.2.2). In Fig. 21, we compare the trade off between area and shape error for each incremental step of the algorithm. Each point in Fig. 21a and Fig. 21b corresponds to one intermediate result of the *CartoDraw* algorithm (with interactive scanlines). The result shows the trade off between area error and shape distortion: In the beginning, there is a large area error $E_{rel} = 0.36$. By applying a scanline, the area error is improved, but the shape becomes more distorted. It is therefore not surprising that the curve goes from the lower right to the upper left until the area error is small enough or the shape distortion reaches some threshold. A similar behavior can be observed for the global shape. There is,



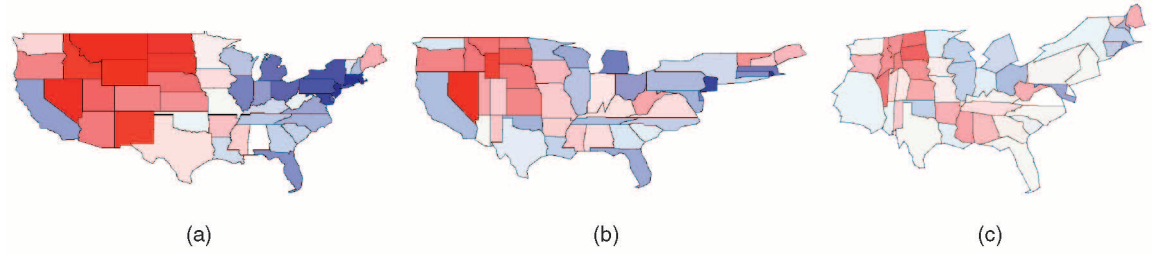
(a) (b) (c)

Fig. 18. Comparison of cartogram drawing algorithms. (a) Tobler [13]. (b) Kocmoud and House [7], Scanline-based algorithm.



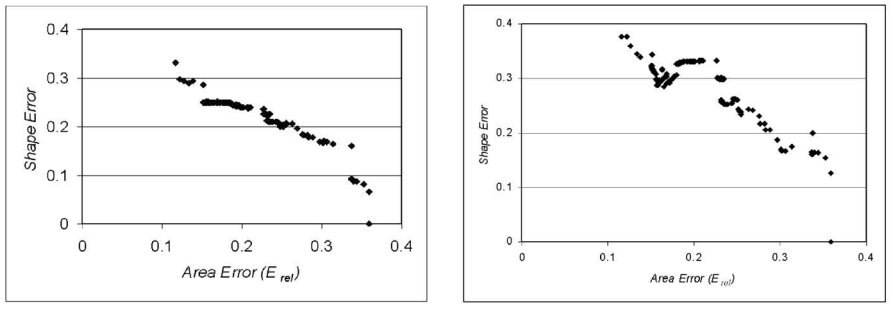
(a) (b) (c)

Fig. 19. Area error and efficiency comparison (1980 US population cartogram). (a) Total area error. (b) Area error state-wise sorted. (c) Efficiency comparison.



(a) (b) (c)

Fig. 20. Population cartogram with automatically and interactively placed scanlines. (a) Original map ($E_{rel} = 0.36$). (b) Automatic scanlines ($E_{rel} = 0.21$). (c) Int. scanlines ($E_{rel} = 0.1$).



(a) (b)

Fig. 21. Shape versus area error comparison (interactive scanlines). (a) State polygons. (b) Global polygons.

however, a slight difference: While the area error still improves from one step to the next, the distortion of the global shape in some cases actually improves.

Comparing the area-shape error trade off of interactive versus automatic scanlines reveals some interesting properties of our algorithm (see Fig. 22). In the beginning, both approaches have a similar trend in shape-area error trade off. At a certain point, however, the automatically generated scanlines lead to a deterioration in area error that subsequent scanlines are not able to improve. In the case of interactively generated scanlines, the area error continues to improve by smaller and smaller increments. Note the jump in shape error for an area error of about $E_{rel} = 0.15$. At this point, we switched the direction from *scanline* to *section line* (see scanline algorithm in Section 3.2.3), which yields a continued improvement of the area error, but with considerable deterioration of the shape error.

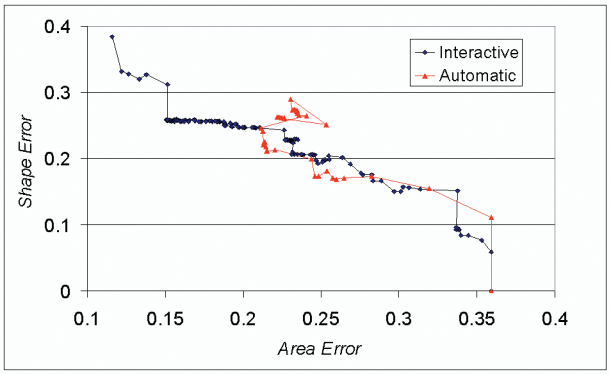


Fig. 22. Comparison of automatic and interactive scanlines.

4.2.2 Efficiency

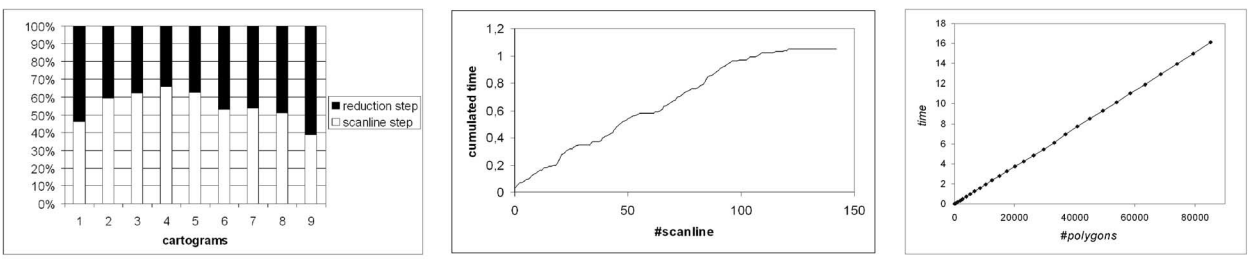
We also performed extensive experiments to evaluate the efficiency of the *CartoDraw* algorithm. The time needed to run the algorithm on the US population data is about 2 seconds. If we change the parameter vector, the time needed for the reduction step versus the scanline execution varies slightly between 40 percent and 60 percent. Fig. 23a shows the percentages needed for the two steps of the algorithm for nine different parameter vectors, namely, long-distance telephone call volume data by state for nine time steps during a day. Note that the reduction step can be precomputed so that it does not have to be rerun each time the algorithm is executed.

We also analyzed the effect of changing the length of scanlines. Fig. 23b shows the results for the 144 interactively defined scanlines for the US population data. The time needed to process a scanline depends only on the number of scanline sections, which in turn depends only on the length of the scanlines. This means that a steep increase corresponds to long scanlines and a shallow increase corresponds to short scanlines. Fig. 23b reveals that shorter scanlines are more likely toward the end of the process and are used for fine tuning portions of some polygons. Nevertheless, some shorter scanlines are applied regularly in the process as indicated by the irregularities in the curve.

Our final efficiency analysis was aimed at testing how the performance of the *CartoDraw* algorithm varies with the number of polygons. Since we do not have many different real data sets with a widely varying number of polygons, we generated synthetic data sets, namely, checker boards with an increasing number of rectangular polygons. We then used random numbers to initialize the parameter vectors. Fig. 23c shows the results of these tests, revealing a clear linear dependency on the number of polygons. The algorithm needs about 16 seconds for a net of 90,000 polygons. Note, however, that, in this case, the number of vertices per polygon is very low (four) and a reduction of vertices is not necessary.

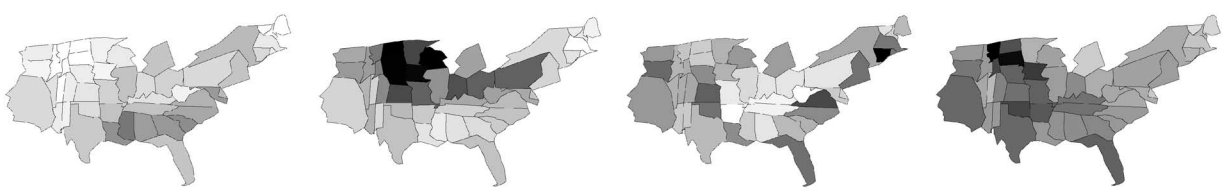
4.2.3 Application Examples

We ran the *CartoDraw* algorithm on several example data sets. First, we used a US population cartogram to show some statistical data obtained from the US census database. In Fig. 24a, we show the population cartogram with color corresponding to the area error in the cartogram. Blue colored regions should be larger, while red regions should be smaller. In Fig. 24b, Fig. 24c, and Fig. 24d, the color shows different statistical parameters on top of the population cartogram: Fig. 24b shows the percentage of people with German ancestry, Fig. 24c the median household income in 1989, and Fig. 24d the percentage of unpaid family workers. Since the area of the states corresponds to their population, the shaded areas in the figures directly correspond to the number of people with those properties. It is interesting that the highest percentage of people with German ancestry is in the northern midwest (with some



(a) (b) (c)

Fig. 23. Efficiency tests. (a) Reduction versus scanline step. (b) Number of scanlines (US states). (c) Number of polygons.



(a) (b) (c) (d)

Fig. 24. Population cartogram showing census data statistics ($E_{rel} = 0.10$). (a) Area error. (b) Percentage of people with German ancestry. (c) Median household income in 1989. (d) Unpaid family workers.

Fig. 25. Population cartogram of Europe.

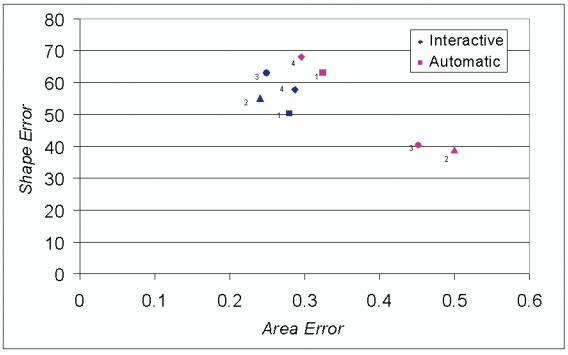


Fig. 27. Interactive versus automatic scanlines (call volume data).

higher numbers also in Philadelphia and East). In the median income cartogram, it is interesting that the areas with small income are mostly states with relatively small populations (the middle of the noncoastal east of the US) and, in the unpaid family workers cartogram, the numbers are high in the northern midwest, with some additional high numbers in Florida, Texas, and California. To show that our algorithm works with arbitrary polygon meshes, we also applied the algorithm to the population data of Europe, as shown in Fig. 25.

As mentioned at the outset, we developed the *CartoDraw* algorithm for displaying applications like telephone call volume data, which needs to be continuously monitored and visualized. Since the underlying polygon data does not change, the vertex reduction only needs to be run once. This provides a significant speed-up, making interactive display with an update rate of about one second possible. Fig. 26 shows the results of the normalized telephone call volume at different times during one day. 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. In Fig. 27, we compare the shape-area error trade off of the four call volume cartograms. The results clearly show that interactive scanlines always provide better shape-area error trade offs. Note, however, that, for frames 2 and 3, the interactive solution provides significantly lower area error at the cost of slightly greater shape error, while, for frames 1 and 4, it is superior in both shape and area error.

5 Conclusions and Future Work

In this study, we analyzed and discussed the problem of efficient cartogram drawing and proposed an optimistic algorithm that outperforms previous techniques by orders of magnitude and provides results that are at least as correct. Experiments show that the proposed algorithm offers good results for a variety of applications and scales to a large number of input polygons. For medium sized data sets, its performance is sufficient for interactive display of streaming data in telecom applications. Although the proposed algorithm is a significant step toward fast, reliable, and effective cartogram generation, there remain several promising directions for further research, including study of the dependence of the results on the selected scanlines and improvement of automatic scanline placement.

Acknowledgments

Stephen North thanks David Dobkin and Ilan Sender for collaborating on the initial cartogram experiments that helped to inspire this work. The authors also thank Rick Becker and Allan Wilks for their comments and suggestions.

References

[1] E. Raisz, *General Cartography*. New York: McGraw-Hill, 1948.
[2] E. Raisz, *Principles of Cartography*. New York: McGraw-Hill, 1962.
[3] J. Hunter and J.C. Young, "A Technique for the Construction of Quantitative Cartograms by Physical Accretion Models," *The Professional Geographer*, vol. 20, pp. 402-406, 1968.
[4] H. Gray Funkhouser, "Historical Development of the Geographical Representation of Statistical Data," *Osiris*, vol. 3, pp. 269-403, 1937.
[5] B.D. Dent, *Cartography: Thematic Map Design*, fourth ed., chapter 10. Dubuque, Iowa: William C. Brown, 1996.
[6] W.R. Tobler, "Cartograms and Cartosplines," *Proc. 1976 Workshop Automated Cartography and Epidemiology*, pp. 53-58, 1976.
[7] C.J. Kocmoud and D.H. House, "Continuous Cartogram Construction," *Proc. IEEE Visualization*, pp. 197-204, 1998.

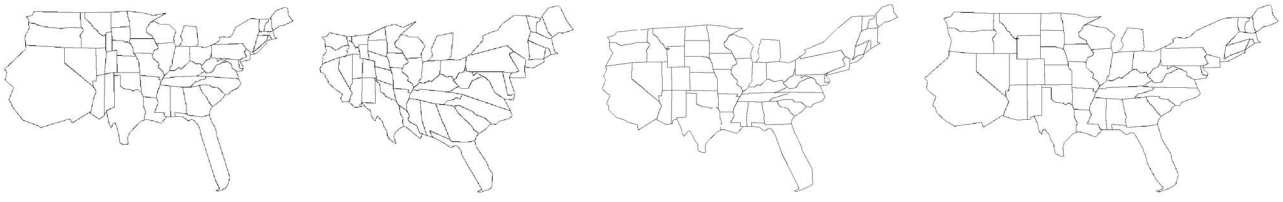


Fig. 26. Long distance call volume data ($E_{rel} < 0.20$). (a) 0:00 am (EST). (b) 6:00 am (EST). (c) 12:00 pm (EST). (d) 6:00 pm (EST).

[8] S. Gusein-Zade and V. Tikunov, "Map Transformations," *Geography Rev.,* vol. 9, no. 1, pp. 19-23, 1995.
[9] A.M. MacEachren, *How Maps Work: Presentation, Visualization, and Design.* New York: The Guilford Press, 1995.
[10] C.B. Jackel, "Using Arcview to Create Contiguous and Noncontiguous Area Cartograms," *Cartography and Geographic Information Systems,* vol. 24, no. 2, pp. 101-109, 1997.
[11] B. White, I. Gregory, and H. Southall, "Analysing and Visualising Long-Term Change," *GIS Research UK, Proc. Sixth Nat'l Conf.,* 1998.
[12] W.R. Tobler, "Pseudo-Cartograms," *The Am. Cartographer,* vol. 13, no. 1, pp. 43-40, 1986.
[13] S. Selvin, D. Merrill, J. Schulman, S. Sacks, L. Bedell, and L. Wong, "Transformations of Maps to Investigate Clusters of Disease," *Social Science and Medicine,* vol. 26, no. 2, pp. 215-221, 1988.
[14] S. Gusein-Zade and V. Tikunov, "A New Technique for Constructing Continuous Cartograms," *Cartography and Geographic Information Systems,* vol. 20, no. 3, pp. 66-85, 1993.
[15] B. Dent, "A Note on the Importance of Shape in Cartogram Communication," *The J. Geography,* vol. 71, no. 7, pp. 393-401, Oct. 1972.
[16] B. Dent, "Communication Aspects of Value-by-Area Cartograms," *The Am. Cartographer,* vol. 2, no. 2, pp. 154-168, Oct. 1975.
[17] T. Keahey and E. Robertson, "Nonlinear Magnification Fields," *Proc. IEEE Symp. Information Visualization,* pp. 51-58, 1997.
[18] T. Munzner, "Exploring Large Graphs in 3D Hyperbolic Space," *IEEE Computer Graphics and Applications,* vol. 18, no. 4, pp. 18-23, July/Aug. 1998.
[19] M.S.T. Carpendale, D.J. Cowperthwaite, M. Tigges, A. Fall, and F.D. Fracchia, "The TARDIS: A Visual Exploration Environment for Landscape Dynamics," *Visual Data Exploration and Analysis VI, Proc. SPIE,* vol. 3643, pp. 110-119, Jan. 1999.
[20] T.A. Keahey, "Area-Normalized Thematic Views," *Proc. Int'l Cartography Assembly,* Aug. 1999.
[21] C. Cauvin, C. Schneider, and G. Cherrier, "Cartographic Transformations and the Piezopleth Method," *The Cartographic J.,* vol. 26, no. 2, pp. 96-104, Dec. 1989.
[22] D. Dorling, *Area Cartograms: Their Use and Creation,* first ed. Dept. of Geography, Univ. of Bristol, England, 1996.
[23] H. Edelsbrunner and R. Waupotitsch, "A Combinatorial Approach to Cartograms," *Computational Geometry,* pp. 343-360, 1997.
[24] D. Anderson, E. Anderson, N. Lesh, J. Marks, K. Perlin, D. Ratajczak, and K. Ryall, "Human-Guided Greedy Search: Combining Information Visualization and Heuristic Search," *Proc. Workshop New Paradigms in Information Visualization and Manipulation (NPIVM '99),* pp. 21-25, 1999.
[25] L. Kehrer and C. Meinecke, *Perceptual Organization of Visual Patterns: The Segmentation of Textures,* chapter 2. London: Academic Press, 1995.
[26] S. Berchtold, D.A. Keim, and H.-P. Kriegel, "Using Extended Feature Objects for Partial Similarity Retrieval," *VLDB J.,* vol. 6, no. 4, pp. 333-348, 1997.
[27] N. Weisstein, *The Joy of Fourier Analysis.* Hillsdale, N.J.: Erlbaum, 1980.
[28] K. Mehlhorn and S. Näher, *The LEDA Platform of Combinatorial and Geometric Computing,* first ed. Cambridge Univ. Press, 1999, http://www.mpi-sb.mpg.de/mehlhorn/LEDAbook.html.

Daniel A. Keim received the PhD degree in computer science from the University of Munich in 1994. He is working in the area of information visualization and data mining. In the field of information visualization, he developed several novel techniques which use visualization technology to explore large databases. He has published extensively on information visualization and data mining; he has given tutorials on related issues at several large conferences including Visualization, SIGMOD, VLDB, and KDD; he was program cochair of the IEEE Information Visualization Symposia in 1999 and 2000; he was program cochair of the ACM SIGKDD conference in 2002; and he is an editor of the *IEEE Transactions on Visualization and Ccomputer Graphics* and the *Information Visualization Journal*. He has been an assistant professor in the Computer Science Department of the University of Munich, an associate professor in the Computer Science Department of the Martin-Luther-University Halle, and he is a full professor in the Computer Science Department of the University of Constance. Recently, he worked at AT&T Shannon Research Labs, Florham Park, New Jersey. He is a member of the IEEE Computer Society.

Stephen C. North received the PhD degree in computer science from Princeton University in 1986. He is head of Information Visualization Research at AT&T Labs, a group that studies novel interactive displays and high performance graphics for network visualization in the AT&T Infolab. His background is in software visualization, applied computational geometry, and the design of reusable software. He is one of the authors of *graphviz*, a widely used collection of open source programs for drawing and interacting with graph layouts. His other current technical interests include dynamic graph layout, and Internet mapping and measurement. He is a senior member of the IEEE and a member of the ACM.

Christian Panse received the master's degree from the Martin-Luther-University Halle-Wittenberg, Germany, in 2001. He is currently pursing the PhD degree in the Data Mining and Visualization Group at the University of Constance, Germany. He wrote his master's thesis on cartograms and has worked on related research issues since then. he is a member of the IEEE Computer Society.