A fast, free-form rubber-sheet algorithm for contiguous area cartograms

Shipeng Sun*

School of Natural Resources and Environment, University of Michigan, Ann Arbor, MI, USA

(Received 26 June 2011; final version received 22 June 2012)

This article presents a fast, free-form rubber-sheet (Carto3F) algorithm for the construction of contiguous area cartograms. Rubber-sheet algorithms are influential and popular because of their conceptual simplicity. Existing rubber-sheet algorithms, however, tend to be algorithmically inefficient and computationally slow. More critically, they cannot completely preserve topology. Carto3F specifically improves rubber-sheet algorithms in these two aspects. First, with a spatial structure of quadtree and a mathematical condition for topological equivalence, Carto3F can prevent topological errors and guarantees topological integrity. Second, Carto3F is designed with efficiency as a priority. Its efficiency is primarily gained through using the auxiliary quadtree to reduce the number of points to be transformed. Furthermore, Carto3F allows parallel computation and can fully take advantage of the increasingly common multi-core, multi-thread processors. Based on the mathematical analysis, Carto3F also mitigates the effect of force cancellation that is inherent in rubber-sheet algorithms. On a computer equipped with a 2.4-GHz quad-core CPU, Carto3F can produce quality population cartograms of the United States, China, and the world within 1 second, 18 seconds, and 8 minutes, respectively. Quantitative measures show that Carto3F outperforms the optimized rubber-sheet algorithm and the diffusion algorithm in both transformation effectiveness and computational efficiency.

Keywords: Carto3F; fast, free-form transformation; contiguous area cartogram; rubber-sheet algorithm

1. Introduction

Cartograms, including area and distance cartograms, are transformed maps in which sizes or distances represent non-geometric values such as population or travel costs (Friis 1974, Dorling 1994, Tobler 2004, Sun and Manson 2007, Shimizu and Inoue 2009). Area cartograms have two categories, contiguous and non-contiguous (alternatively called continuous and non-continuous) (Olson 1976, Sun and Li 2010). Contiguous or continuous area cartograms, hereafter cartograms unless otherwise noted, preserve map topologies and are arguably the most difficult to construct yet the most appealing and efficacious cartograms in terms of graphically representing the non-geometric data (Tobler 2004, Sun and Li 2010).

In the past 40 years, GIScientists and researchers in the fields of mathematics, computer science, and physics have accomplished remarkable progress in computer cartograms.

*Email: shipengs@umich.edu

© 2013 Taylor & Francis
(Tobler 2004). A series of efficient cartogram algorithms have been developed (e.g., Dougenik et al. 1985, Gusein-Zade and Tikunov 1993, Gastner and Newman 2004, Keim et al. 2004, Inoue and Shimizu 2006, Henriques et al. 2009, Sun forthcoming). Moreover, cartograms on various themes, especially on the US presidential elections, drew much attention of the boarder academia as well as the public (Dorling et al. 2006, Dorling 2007, Dorling et al. 2008). The success of cartograms, however, is not limited to mapping alternative conceptions of space and geography in social domains; researchers also explored their possible application to real-world planning like the design of mobile sensor networks (Lekien and Leonard 2009).

The family of rubber-sheet algorithms is among the earliest computer cartogram algorithms (Tobler 1973, Dougenik et al. 1985, Tobler 1986, Tobler 2004). The one proposed by Dougenik, Chrisman, and Niemeyer (DCN) algorithm is particularly popular because of its simplicity and ability to preserve shapes (Dougenik et al. 1985, Tobler 2004). DCN is an iterative algorithm simulating springs exerting forces on a rubber sheet. It calculates the radial transforming forces from polygon centroids, whose magnitudes are positively related to polygon size errors but inversely proportional to distance. These forces then form a global vector field to displace polygon vertices. One issue of this algorithm, however, is the possibility of topological errors (Gastner and Newman 2004, Tobler 2004). In an optimization to DCN, this topology issue is mathematically tackled by applying a sufficient condition for topological equivalence (Sun forthcoming). This optimized DCN (Opti-DCN) algorithm, although theoretically well-grounded, leaves one critical parameter loosely specified. If its value is not properly chosen, the resultant cartograms may still contain topological errors (Sun forthcoming). Apart from potential topology violation, these rubber-sheet algorithms, Opti-DCN in particular, do not utilize any spatial structure, and their computational efficiency is quite low compared with some of the best such as the diffusion algorithm (Gastner and Newman 2004).

To improve the computational efficiency of Opti-DCN and to solve its topological problems, this article proposes a new, fast, free-form rubber-sheet cartogram (named Carto3F, read Carto-Triple-F) algorithm. Although it partially shares the mathematical foundation with Opti-DCN algorithm, Carto3F has three distinctive features that are different from Opti-DCN. (1) Carto3F combines a spatial structure of quadtree with the mathematics of topology. Such a quadtree can practically eliminate line crossing and polygon overlapping that would otherwise cause topological errors. More importantly, the quadtree structure, with its ability to index and compress spatial data, can promote computational efficiency by reducing the number of points that need to be transformed. (2) Carto3F is designed with parallel computation capability. Although a parallel program is harder to develop than a sequential one, it can fully take advantage of the increasingly common multi-core, multi-thread CPUs. (3) Carto3F specifically uses a force cancellation mitigation strategy to improve its transformation efficiency for small but high-density areas. This mitigation strategy is firmly based on a mathematical analysis of the underlying transformation equations. Taken together, Carto3F algorithm, in contrast to Opti-DCN, practically solves topology problems, significantly improves efficiency, and is able to process complex maps.

The rest of the article is structured as follows. Section 2 briefly reviews the major algorithms of contiguous area cartograms with more focus on rubber-sheet-based ones. Section 3 presents a mathematical and computational foundation for Carto3F algorithm. Section 4 describes Carto3F algorithm and three key implementation issues. Section 5 presents the United States, China, and the world population cartograms produced by
Carto3F, in which its algorithmic performance is also quantitatively evaluated. This article concludes with the value of this new fast, free-form rubber-sheet algorithm as well as directions for future research.

2. Contiguous area cartogram algorithms

Three categories of cartogram construction algorithms can be identified from a technical viewpoint: cartographic and computational geometry methods, artificial intelligence, and mathematical–physical process simulation. These algorithms vary in conceptual simplicity, computational performance, cartographic quality, as well as in their ability to retain topology and to preserve shapes (Henriques et al. 2009). As many have already elaborated, there is no silver bullet solution for cartograms so far; each algorithm, although having certain advantages, has specific limitations (Henriques et al. 2009). Among others, Dorling (1996), Skupin and Fabrikant (2003), Tobler (2004), and Henriques et al. (2009) have provided broad and/or detailed reviews on cartogram and related transformation algorithms. To avoid repetition, this article only briefly summarizes some features of these three categories, particularly their advantages and limitations, followed by more detailed discussion on rubber-sheet algorithms.

Cartographers and artists originally created cartograms to represent non-geometric measures or virtual perceptions using cartographic techniques (Raisz 1934, Raisz 1936, Tobler 1961, Friis 1974). New computer technologies, especially Geographic Information Systems (GIS), allow cartographers to replicate and improve the core procedure of manual cartogram creation by referring to concepts in cartography and computational geometry. Medial axis and triangulation, for instance, have been introduced to help make cartograms (House and Kocmoud 1998, Keim et al. 2005, Inoue and Shimizu 2006). One general problem of many cartographic and computational geometry methods is the presence of exceptions, that is, there are always some special situations that require additional consideration and processing. It appears very difficult, if not impossible, to detect and handle all exceptions that would otherwise break topology or unevenly impact particular polygons. It is therefore quite common that a computational geometry algorithm calibrated for one region might not work well for another, thus impairing its generality.

Artificial intelligence (AI) is a promising direction for the development of cartogram algorithms and has received attention in recent years. Cellular automata (CA) and self-organizing maps (SOMs), for instance, have been applied to generate cartograms (Dorling 1996, Henriques et al. 2009). A CA machine can be calibrated to make cartograms, in which a region is divided into regular grid cells and each cell adjusts itself to approach the desired size without breaking topological relationships with its neighbors (Dorling 1996). A more recent AI algorithm is Carto-SOM (Henriques et al. 2009). This algorithm is based on SOMs, a neural network technology. The essential of Carto-SOM is to train a set of input points controlled by population density to mimic a set of output points controlled by a regular grid. Carto-SOM turns out to be efficient and the size error of cartograms produced by this algorithm is quite low. Despite many favorable features of AI techniques, these algorithms are either conceptually or computationally complex and often require deliberate calibration to achieve the best results.

Simulating mathematical and physical processes helped foster some of the earliest computer cartograms. Gusein-Zade and Tikunov (1993), for example, deduce mathematical equations for cartograms using line integral. Along with pure mathematics, intuitive and well-researched physical processes are also used to develop cartogram algorithms, of which diffusion is the most successful one. Diffusion algorithm by Gastner and
Newman can produce cartograms with efficiency and topological integrity (Gastner and Newman 2004, Gastner et al. 2005, Dorling 2007). Although these two methods, particularly the diffusion algorithm, can produce quality cartograms free from topological errors, they have few procedures that are not controlled by the algorithms. Cartogram makers, therefore, have little freedom to change or even impact their outputs. Furthermore, the diffusion algorithm would likely produce cartograms similar to circles in theory if the number of steps that the algorithm runs is large enough to indefinitely approach ‘equal density’ maps. Such ‘ballooning’ effect, however, could potentially impair shape preservation and reduce map readability (House and Kocmoud 1998).

Another sub-category of mathematical–physical process simulation is rubber-sheet algorithms. One primary requirement for a good cartogram algorithm is to preserve topology. In mathematical terms, preserving topology means maintaining topological equivalence between a cartogram and its source map. In the discipline of topology, which is often referred as ‘rubber-sheet geometry’, a rubber sheet is the most common metaphor (Johnson and Glenn 1960, Sauvy and Sauvy 1974). Some pioneers on computer cartograms utilized such an analog. For example, Tobler (1973, 1986) developed rubber map and pseudo-cartograms by simulating rubber sheet on a regular grid. DCN algorithm also simulates a rubber sheet and is one of the most popular cartogram algorithms so far (Dougenik et al. 1985, Sun forthcoming).

DCN algorithm simulates a system of springs that exert transforming forces on an elastic surface. These vector forces are generated using pairs of origin and destination points that are created from polygons. Their magnitudes are usually specified using a continuous, differentiable function with distance decay and their directions are parallel to the line from polygon centroids to polygon vertices (Dougenik et al. 1985). DCN is an iterative and approximate solution of cartograms and requires multiple steps. In each step, transforming forces mobilize polygon vertices toward their desired position. As polygons gradually approach their desired sizes, the forces become weaker and finally insignificant. In the end, the sizes of the transformed polygons will be very close to what are desired (Dougenik et al. 1985).

Despite its simplicity and popularity, rubber-sheet-based DCN algorithm cannot mathematically guarantee topological integrity. Furthermore, rubber-sheet algorithms are not optimized to converge as fast as mathematically allowed. These limitations negatively impact their applicability to maps with complex spatial patterns. For example, as up to date, no world population cartograms are ever produced using rubber-sheet algorithms. When DCN algorithm is applied to maps with complex patterns like China, the results have obvious topological errors (see Sun and Li 2010). Nevertheless, rubber-sheet algorithms are conceptually accessible and have relatively good performance in shape preservation.

Improving existing rubber-sheet algorithms, therefore, seems necessary and valuable, particularly increasing their computational efficiency and revising their mathematical foundation to eliminate topological errors. Sun (forthcoming) proposes an optimization to the original DCN algorithm, named Opti-DCN, and has mathematically solved its topological integrity problem by designing a transforming force function that meets a sufficient condition for topological equivalence. Such a mathematical optimization, however, hinges on an extra point densification procedure, that is, inserting more points on long polygon arcs to ensure smooth transformation. More critically, the key parameter of point density for this procedure is derived through trial-and-error experiments without a theoretically or computationally valid foundation. Furthermore, Opti-DCN is primarily based on mathematics, does not utilize any spatial structure, and is still quite slow. Extending Opti-DCN,
Carto3F presented here combines the mathematics of topology, a quadtree structure, parallel computation, and mitigation of force cancellation effects to computationally prevent topological errors and to boost its efficiency.

3. A mathematical and computational foundation

Like many simulation-based methods, rubber-sheet algorithms need multiple iterations to approach ideal ‘equal density’ cartograms. During each iteration, polygons either enlarge or shrink toward their ideal sizes, which is determined by statistical values like population. The positional differences between the current and ideal coordinates of polygon vertices generate transforming forces, which aggregately form a vector force field. This force field then exerts forces on all polygon vertices and moves them toward new coordinates that would gradually reduce transforming forces. Carto3F proposed here also follows this procedure, but with an auxiliary quadtree structure to pragmatically eliminate potential topological errors and to reduce the number of points to be transformed.

3.1. Force generation with topology preservation

Many previous discussions on area cartograms use polygons or sub-regions as basic transformation units (Keim et al. 2004, Henriques et al. 2009, Lekien and Leonard 2009, Sun and Li 2010). By contrast, Carto3F uses point displacements. Define a set of transforming point pairs that generate forces as

\[ T_I^o \rightarrow T_I^d : P_i^o (x_i^o, y_i^o) \rightarrow P_i^d (x_i^d, y_i^d), \quad i = 1, 2, \ldots, n \]

where \( P_i^o \) is the desired position for the origin point \( P_i^o \). Then, define a set of points to be transformed in the space \( T_D = \{ P_k(x_k, y_k), \quad k = 1, 2, \ldots, m \} \). The key of Carto3F, as any other rubber-sheet algorithms, is to find a mathematical transformation \( T \) that maps \( T_I^o \) to \( T_I^d \) and then uses \( T \) to transform \( T_D \). Separating transforming point pairs from the points to be transformed offers flexibility because it allows using simplified shapes to approximate polygons and using a quadtree structure to index and compress polygon vertices (for more details, see below).

One fundamental requirement for cartogram algorithms is to preserve topology. According to theories in the discipline of topology, bi-continuity is a sufficient and necessary condition for topological equivalence (Sauvy and Sauvy 1974, Binmore 1980). Carto3F uses a bi-continuous transformation function that guarantees topological equivalence by ensuring that the determinant of its Jacobian matrix is greater than 0. The transforming force generated by \( T_I = \{ P_i^o (x_i^o, y_i^o) \rightarrow P_i^d (x_i^d, y_i^d), \quad i = 1, 2, \ldots, n \} \) at \( P_k(x_k, y_k) \) can be calculated as follows:

\[ F_k(x_k, y_k) = F_{kx}(x_k, y_k) \vec{u}_x + F_{ky}(x_k, y_k) \vec{u}_y \]

\[ F_{kx} (x_k, y_k) = \sum_{i=1}^{n} F_{ikx} = \sum_{i=1}^{n} (x_i^d - x_i^o) e^{-\frac{d_{iok}}{d_{iol}}} \]

\[ F_{ky} (x_k, y_k) = \sum_{i=1}^{n} F_{iky} = \sum_{i=1}^{n} (y_i^d - y_i^o) e^{-\frac{d_{iok}}{d_{iold}}} \]

where \( \vec{u}_x \) and \( \vec{u}_y \) are unit vectors, \( d_{iok} \) is the distance between \( P_k(x_k, y_k) \) and \( P_i^o(x_i^o, y_i^o) \), and \( d_{iold} \) is the distance between \( P_i^o(x_i^o, y_i^o) \) and \( P_i^d(x_i^d, y_i^d) \) (Figure 1). For a more detailed explanation of these equations, see Sun (forthcoming).
Figure 1. Transforming force generated by point pair \( \{ P^o_i \rightarrow P^d_i \} \) at point \( P_k \).

With these force functions, the transformation equation and the determinant of its Jacobian matrix are defined as follows, in which \( c_k \) is a local elasticity coefficient.

\[
\begin{pmatrix}
  x_k^d \\
  y_k^d
\end{pmatrix} = T
\begin{pmatrix}
  x_k \\
  y_k
\end{pmatrix} = \begin{pmatrix}
  x_k + c_k F_{kx}(x_k, y_k) \\
  y_k + c_k F_{ky}(x_k, y_k)
\end{pmatrix} \quad (4)
\]

\[
\det J_T = \left( \frac{\partial T_{kx}}{\partial x} \frac{\partial T_{ky}}{\partial y} - \frac{\partial T_{kx}}{\partial y} \frac{\partial T_{ky}}{\partial x} \right)
\]

\[
= \left( 1 + c_k \frac{\partial F_{kx}}{\partial x} \right) \left( 1 + c_k \frac{\partial F_{ky}}{\partial y} \right) - \left( c_k \frac{\partial F_{kx}}{\partial y} \times c_k \frac{\partial F_{ky}}{\partial x} \right) \quad (5)
\]

The determinant can be reformatted as a quadratic form.

\[
\det J_T = \left( \frac{\partial F_{kx}}{\partial x} \frac{\partial F_{ky}}{\partial y} - \frac{\partial F_{kx}}{\partial y} \frac{\partial F_{ky}}{\partial x} \right) c_k^2 + \left( \frac{\partial F_{kx}}{\partial x} + \frac{\partial F_{ky}}{\partial y} \right) c_k + 1 \quad (6)
\]

If \( a = \left( \frac{\partial F_{kx}}{\partial x} \frac{\partial F_{ky}}{\partial y} - \frac{\partial F_{kx}}{\partial y} \frac{\partial F_{ky}}{\partial x} \right) \) and \( b = \left( \frac{\partial F_{kx}}{\partial x} + \frac{\partial F_{ky}}{\partial y} \right) \), it becomes \( \det J_T = ac_k^2 + bc_k + 1 \), with a constraint of \( 0 < c_k \leq 1 \). This quadratic equation has seven different situations depending on the values of \( a \) and \( b \) (Figure 2). Because point \((0, 1)\) must be on the parabola defined by \( \det J_T = 0 \), a value of \( c_k \) must exist on \((0, 1]\), so that \( \det J_T = \varepsilon > 0 \), where \( \varepsilon \) is a positive infinitely small number. In addition, to ensure a smaller \( c_k \) still makes \( \det J_T \) positive, \( \det J_T \) must always be greater than \( 0 \) on interval \([0, c_k]\). From Figure 2a, it is quite straightforward that when \( a \) is greater than \( 0 \), \( c_k \) should take the smaller solution of \( \det J_T = \varepsilon > 0 \) if there are two solutions; if the solutions are not on \((0, 1]\), however, \( c_k \) should be \( 1 \). From Figure 2b, when \( a \) is smaller than \( 0 \), \( c_k \) has no more than \( 1 \) solution on \((0, 1]\). Similarly, \( c_k \) takes the value of \( 1 \) when no solutions exist. Taken together, the local elasticity coefficient \( c_k \) could be calculated as follows:

\[
c_k = \begin{cases} 
  \varepsilon - 1/b, & \text{if } a = 0 \text{ and } b < 0 \\
  -b - \sqrt{b^2 - 4a(1 - \varepsilon)}/2a, & \text{if } a \neq 0 \text{ and } b^2 > 4a(1 - \varepsilon) \\
  1, & \text{all others or } c_k < 0 \text{ or } c_k > 1 \text{ from above}
\end{cases}
\]
Sun (forthcoming) also introduced a gradient-based method for $c_k$. Although that method is easier to calculate, it is not recommended for complex maps because it is based on a relaxed condition for topological equivalence. With all $c_k$, define the global elasticity coefficient $c = \min(c_k)$, where $k = 1, 2, 3, \ldots$. Using this global elasticity, the destination of point
\( P_k^d(x, y) \) can be set as \( \begin{pmatrix} x_k^d \\ y_k^d \end{pmatrix} = \begin{pmatrix} x_k + c F_{k,x}(x, y) \\ y_k + c F_{k,y}(x, y) \end{pmatrix} \). Because the global \( c \) is no greater than any local elasticity, the determinant of the Jacobian matrix at any point must be greater than 0.

Overall, these equations mathematically utilize the internal relationships between the transforming forces, their partial derivatives, and the Jacobian matrix to guarantee the existence of a local inverse function at all points to be transformed. Combined together, they are able to theoretically maintain topological equivalence over the whole map region (Sun forthcoming).

### 3.2. Auxiliary quadtree

Even with a sufficient condition for topological equivalence, the topology of vector polygons may not be entirely preserved in a discrete space. There are situations where irregularly or sparsely distributed points may break topology in spite of the mathematically guaranteed topological equivalence. In Figure 3, the space is transformed with topological integrity as illustrated by the grid. The direct line, however, still intersects with the other line because it does not ‘bend’ after transformation. Inserting more points to polygon arcs can mitigate such issue (Du and Liu 1999, Sun forthcoming); however, it is hard to mathematically derive an appropriate point density value that leads to a smooth enough transformation without significantly increasing the number of points.

To solve this problem, an auxiliary quadtree structure is introduced to systematically eliminate such errors and to improve the performance of the algorithm as well. Carto3F algorithm first creates a quadtree structure to cover all polygon vertices with a given tree depth. And then the quadtree nodes, instead of the original polygon vertices, are taken as the points to be transformed \( TD \). In other words, Carto3F does not directly transform polygon vertices; instead, it only transforms quadtree leaf nodes. After the quadtree is transformed, the original polygons could be interpolated using simple affine transformation. To do that, a quadtree node has to be split into two triangles (Figure 4). For the sake of shape preservation, the maximum angle in the two triangles should be minimized, which implies that the two adjacent sides forming the biggest angle should not be in the same triangle. Because each triangle is defined by three points, say \( A, B, \) and \( D \), in Figure 4, the affine transformation matrix \( M \) can be solved as

\[
\begin{bmatrix}
  x_d^a & x_d^b & x_d^c \\
  y_d^a & y_d^b & y_d^c \\
  1 & 1 & 1
\end{bmatrix} = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  0 & 0 & 1
\end{bmatrix} \times \begin{bmatrix}
  x_o^a & x_o^b & x_o^c \\
  y_o^a & y_o^b & y_o^c \\
  1 & 1 & 1
\end{bmatrix}
\]

or \( X^d = M \times X^o \)

![Figure 3. Topology preservation with a full quadtree.](image-url)
where \( x \) and \( y \) are coordinates of quadtree nodes and \( o \) and \( d \) are original and transformed coordinates, respectively. Solve these linear equations as \( M = X^d \times (X^o)^{-1} \). For any polygon vertex \( P^p_k(x, y) \) located in the triangle \( ABD \), its destination point after the affine transformation is

\[
P^d_k(x^d, y^d) = M \times \begin{pmatrix} x^o \\ y^o \\ 1 \end{pmatrix}
\]

The benefits of using a quadtree structure are twofold. First, the quadtree structure can systematically eliminate the negative impacts of long arcs without significantly increasing the computational cost. More significantly, quadtree itself is a spatial indexing and compression structure (Samet 1984). Using a quadtree helps reduce the number of points to be transformed because there are more vertices in polygons than in the quadtree in general. Quadtree can adapt to various point densities, which provides unmatched flexibility and efficiency for point query and insertion. Moreover, when some leaf nodes in the quadtree are not small enough to maintain topology, they could be further divided into smaller ones to decrease the length of targeted arcs without influencing other nodes. Such situations are not encountered in the three cases presented in this article; nevertheless, a simple line cross-checking routine and the standard quadtree node splitting routine can accomplish these operations when needed.

Second, a quadtree structure helps capture the minimum elasticity coefficient and makes the global elasticity coefficient more close to what is needed to preserve topology. The elasticity coefficient calculated from the Jacobian matrix only guarantees topological equivalence for a local region. The local region in the mathematical sense could be much smaller than even the minimum unit represented by the underlying vector data. Using a quadtree enables the algorithm to systematically calculate the local elasticity coefficients and is less likely to leave out some critical points where the coefficients are much lower than their neighbors.

Practically speaking, a regular grid structure might be as good as a quadtree when the original map is simple. For complex shapes, however, a quadtree can considerably improve computational efficiency because of its capability to compress and index spatial data. Furthermore, a quadtree is more flexible and adaptive than a regular grid. Balancing the cost of implementing a slightly more complex quadtree structure and the benefit of its flexibility and efficiency is a matter of implementation. Nevertheless, an auxiliary spatial
structure is necessary to ‘eliminate’ long arcs, to ensure a globally valid elasticity coefficient, and to improve the algorithmic efficiency of Carto3F. The quadtree structure proves to be a proper choice for these purposes.

4. Fast, free-form rubber-sheet algorithm

Like other rubber-sheet algorithms, Carto3F is an iterative and approximate solution for contiguous area cartograms. The flowchart of Carto3F is illustrated in Figure 5. It first explicitly creates the set of transforming points $T_I$ from polygons. In contrast to DCN algorithm, which directly uses the original polygon vertices, Carto3F uses simplified shapes, that is, circles and squares, to approximate polygons to construct $T_I$ (see Figure 5). After creating $T_I$, a quadtree structure is built to contain and index all polygon vertices. The vertices of the quadtree leaf nodes are then taken as the points to be transformed $T_D$. The aggregate transforming force $\vec{F}_k(x_k, y_k)$ and local elasticity coefficient $c_k$ at each point $P_k = (x_k, y_k)$ in $T_D$ are calculated using Equations (1)–(7). The minimum local elasticity

![Carto3F algorithm flowchart.](image-url)
coefficient is set as the global elasticity coefficient \( c \), which can guarantee topological integrity of the transformed quadtree. Using this transformed quadtree, the original polygons can be processed with interpolation and simple affine transformation. At the end of each step, the algorithm recalculates the global size error and would stop if the error is smaller than a predefined criterion or the error goes up. Otherwise, Carto3F would proceed to the next iteration using the newly transformed polygons.

### 4.1. Parallel computation capability

In recent years, computer performance has been increasingly driven by integrating more cores on a single chip other than by boosting the frequency of a single-core (Geer 2005, Ernst and Stevenson 2008). Industrial manufactures of desktop computers and mobile devices are switching to multi-core CPUs with multi-threading techniques. To get the maximum computational power out of multi-core, multi-thread chips, however, computer programs have to support parallel computation. Parallel computing is not possible if an algorithm is a chain of dependent calculations, in which every calculation depends on the previous one. Although Carto3F is executed in sequential steps, the most intensive computations in each step are independent from each other and thus able to utilize parallel computation.

The most time-consuming sub-routines in Carto3F are (1) calculating transforming forces and local elasticity coefficients for all points in \( TD \) and (2) interpolating polygons using the leaf quadtree nodes. Every point \( P_k \) in \( TD \) is independent, and the force and elasticity at \( P_k \) only rely on \( TI \). As a result, calculating transforming forces and elasticity coefficients can be paralleled and be carried out on multiple threads. Similarly, when interpolating polygons using the quadtree, every polygon vertex is also independent from each other and only relies on the quadtree leaf node in which it is located. Parallel computation, therefore, can be achieved by simply dividing the points in \( TD \), that is, the quadtree nodes, or the polygons to be interpolated, that is, the original polygon vertices, into multiple groups. And these groups can be evenly distributed to multiple threads to utilize the computational capacity of multi-core, multi-thread processors. This ‘divide-and-conquer’ method not only promotes the computational efficiency of Carto3F but also helps avoid synchronization problems like deadlocks that are common to multi-thread programs.

One reason why the proposed algorithm is named free-form is that Carto3F could largely maintain effectiveness, efficiency, and topological integrity regardless of some implementation details. As long as the global elasticity coefficient and the quadtree structure are included, the way of generating transforming point pairs and the form of force generation equations can be substantially varied to improve the algorithm. Here I present two such variations that turn out to be beneficial to the efficiency of Carto3F. One is using simple circles and squares to approximate polygons and to generate transforming point pairs (also see Sun forthcoming) and the other is a modification to the underlying force equations, which is designed as a remedy for force cancellation effects. Force cancellation, if not properly handled, would pose severe limitation on Carto3F. It should be understood that these two variations are not core components of Carto3F algorithm. Replacing any of them would not change the nature of Carto3F algorithm regarding topological integrity and algorithmic effectiveness.

### 4.2. Shape approximation and force exaggeration

One important feature of Carto3F, by guaranteeing topological equivalence and separating transforming points from points to be transformed, is the ability to approach polygons
Figure 6. Transforming point pairs generation using circle and square.

with simplified shapes. Sun (forthcoming) illustrates that the extremely simple shapes of circles and squares are sufficient to generate readable, quality cartograms. To approximate a polygon such as the state of New York in the United States (Figure 6), a circle is first created, whose center is the polygon centroid and its size equals to the polygon area. Then, a square is created around the circle. Carto3F uses points sampled from these circles and squares to substitute polygon vertices for force generation. It is proved that such simplification works well for most polygons like states in the United States. For complex maps such as China and the world, large polygons with low roundness could be divided into smaller ones so as to allow circles and squares to adequately delineate their shapes.

Such simplification of shapes, inevitably, has some drawbacks such as reducing transforming forces and producing irregular force distributions. An ‘exaggeration’ mechanism, therefore, can be applied to enlarge the intended transformation ratio (Sun forthcoming). This is a ‘shooting higher’ strategy and proves to be beneficial to transformation efficiency. Take Japan as an example. If the area of Japan needs to enlarge 50 times, the exaggeration equation would instead make it enlarge 70 times. Because it is well understood that rubber-sheet transformation cannot achieve ideal polygon sizes in one step, the resultant ratio would be like 30 even with 70 as a goal. If 50 was set as a goal, by contrast, the resultant ratio might be just 20, which is less than the exaggerated case. In other words, the exaggeration could accelerate transformation when appropriately used.
Circles and squares, although simple, are certainly not good approximations for many polygons. As polygons are transformed into equal-density areas, their shapes, even of those that are initially similar to circles and/or squares, are gradually shifting from these approximates. As a result, the simplified shapes will likely generate irregular force distribution and make transformed polygons hard to recognize. This situation usually happens when over-transformation occurs, where the density of an originally high-density area is below the average and the density of an originally low-density area is above average. Such cases can be mathematically detected by monitoring the global size errors (for more details, see below). When all size errors are bouncing back, the program should stop because the simple circles and squares, to a large extent, cannot describe the shapes of many polygons any more.

4.3. Force cancellation effects

Through experiments, it is found that two types of transformation, if poorly handled, could seriously impact the efficiency of a rubber-sheet-based cartogram algorithm in general and Carto3F algorithm in particular. One type is the transformation of large areas with low-density, for example, Montana and Idaho in the United States and Canada, Russia, and Australia in the world. The other is the opposite, that is, small areas with high density, such as Beijing and Hong Kong in China and Singapore and Netherlands in the world. These two are actually manifestations of two kinds of force cancellation effects in rubber-sheet algorithms, one from neighboring polygons and the other within the same polygon. When a polygon needs to shrink, for example, the forces pointing toward its centroid would be greatly canceled by the forces generated by its neighboring polygons if all these neighbors also try to shrink. This type of force canceling effect, however, is integral to the cartogram problem and seems only able to be dealt with through multiple iterations. The force canceling effect within a polygon, on the contrary, can be mathematically analyzed and mitigated, particularly for the exponential distance–decay force formula used in Carto3F algorithm.

The effectiveness of the exponential distance–decay force generation function, when applied to simple circles and squares, could be significantly limited for small regions with high statistical values, that is, the second type of force cancellation. To simplify the discussion, I use a one-dimensional case to illustrate such effect (Figure 7). For a circle with a radius of $x$, the point pair $x \rightarrow (x + d)$ on the right side generates a force $F^+(x)$ and the pair $-x \rightarrow (-x - d)$ on the left side generates a force $F^-(x)$. These two forces intend to enlarge the circle by stretching it in opposite directions. The aggregate force on the circle boundary $x$ is $F(x) = F^+(x) - F^-(x)$. This situation is similar to Figure 6 but with only two opposite transforming forces.

![Figure 7. Force cancellation effects.](image)
Using the force generation formula specified earlier,

\[ F(x) = F^+(x) - F^-(x) = d e^{-\frac{d}{2x}} - d e^{-\frac{x}{2}} = d\left(1 - e^{-\frac{x}{2}}\right) = 2x \frac{d}{2x} \left(1 - e^{-\frac{x}{2}}\right) \]

Let \( z = \frac{d}{2x} \), then \( F(x) = 2xz \left(1 - e^{-\frac{x}{z}}\right) \). Because \( \lim_{z \to \infty} z \left(1 - e^{-\frac{1}{z}}\right) = 1 \),

\[ \lim_{d \to \infty} F(x) = \lim_{d \to \infty} 2xz \left(1 - e^{-\frac{x}{z}}\right) = 2x \lim_{z \to \infty} z \left(1 - e^{-\frac{1}{z}}\right) = 2x \tag{8} \]

This formula implies that the maximum transformation that an exponential distance-decay force equation can possibly achieve is two times of the radius of the original circle no matter how large the transforming forces are. For small states with high population density like Singapore, which needs to be enlarged over 150 times, such a maximum transformation limit would significantly reduce the transforming power of Carto3F. To relax such limitation, a size correction factor, \( \gamma = \bar{r} / r_i \), is introduced, where \( \bar{r} \) is the average radius of all original circles and \( r_i \) is the radius of the \( i \)th circle. For those circles whose radii are smaller than the average radius, the exponential part could be multiplied by a correction term, that is, change \( e^{-d_{ix}/d_{iod}} \) to \( e^{-\left(d_{ox}/d_{iod}\right)\gamma} \) in Equations (2) and (3) and accordingly the partial derivatives in Equation (5). The effect of these corrected equations is equivalent to increasing \( d_{ox} \) and therefore allows greater maximum transformation rate for small regions. In other words, these corrected equations could relax the transformation limit imposed by Equation (8). Computationally, they make transforming forces decay faster and mitigate the force cancellation effect, thus casting more local forces and enlarging small regions more efficiently. The figures in the next section will illustrate their impacts on the transformation of Singapore. It should be noted, however, that this size correction factor should be no more aggressive than suggested because sharp distance-decay force could significantly reduce the elasticity coefficient solved from the Jacobian matrix and would negatively impact the overall convergence rate. In addition, the optimal form of the force cancellation mitigation equation might not be the same for different maps.

5. Results

In this section, I present three population cartograms created by Carto3F algorithm: the United States, China, and the world. Although a thorough comparison between Carto3F and other cartogram algorithms would shed more light on its advantages and shortcomings, it is quite difficult to directly compare running times of different cartogram-generating programs. This is because existing cartogram programs are implemented using different languages (Visual Basic®, C/C++, Java™, Python®, etc.), run in different environments (Unix/Linux® vs. Windows™ and standalone vs. ArcMap™ extension), and require various pre- and post-processes (e.g., simplification, generalization, and format conversion). Furthermore, these programs might utilize various existing, well-tuned computational packages such as fast Fourier transform. As a result, results from these programs are likely influenced by factors other than algorithms per se. This article mainly focuses on the performance of this new Carto3F algorithm. It briefly compares the effectiveness of Carto3F with other published cartogram algorithms. In addition, it includes a more detailed quantitative comparison of effectiveness and efficiency between Carto3F and Opti-DCN as well as the diffusion algorithm.
5.1. Cartogram algorithm evaluation

Two aspects of cartogram algorithm assessment are particularly informative, effectiveness and efficiency. The effectiveness of a rubber-sheet algorithm is its ability to produce equal-density maps with shape preservation but without topological errors. Its efficiency includes the number of iterations and/or the absolute time required to converge to acceptable cartograms. Both effectiveness and efficiency have to be considered in order to evaluate and compare cartogram algorithms. Because not a single algorithm could achieve ideal area cartograms so far, it is not possible to measure efficiency without referring to a specific level of effectiveness. As a result, cartogram efficiency is better presented as a time-error curve that plots effectiveness on one axis (as cartogram size error) and efficiency on the other axis (as running steps or time).

Although measuring the number of iterations or running time of a cartogram algorithm is relatively straightforward, evaluating its effectiveness requires some complex spatial metrics. There are two routes to evaluate the effectiveness of cartogram. One uses the standard measure of polygon size error defined as $e_i = \frac{|S_i - S_i^d|}{S_i^d}$, where $S_i^d$ is the desired area of polygon $i$ calculated from population, and $S_i$ is its area after cartogram transformation (Inoue and Shimizu 2006). The overall cartogram size error can then be calculated as the widely accepted root-mean-square error (RMSE) and area-weighted RMSE (termed root-weighted-mean-square error or RWSE). Their definition is as follows, where $N$ is the number of polygons.

$$I_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2}, \quad I_{RWSE} = \sqrt{\frac{\sum_{i=1}^{N} w_i e_i^2}{\sum_{i=1}^{N} S_i^d}}$$  \hspace{1cm} (9)

The other route uses the Keim size error (Keim et al. 2005, Henriques et al. 2009). Keim et al. (2002, 2004, 2005) defined the size error of a single polygon $i$ as $e_i = \frac{|S_i^r - S_i^d|}{S_i^r + S_i^d}$. They use mean quadratic error (MSE), weighted mean error (WME), and simple mean error (SME) to measure the overall effectiveness of a cartogram algorithm.

$$I_{MQE} = \frac{1}{N} \left[ \sum_{i=1}^{N} e_i^2 \right]^{\frac{1}{2}}, \quad I_{WME} = \frac{\sum_{i=1}^{N} (e_i S_i^r)}{\sum_{i=1}^{N} S_i^r}, \quad I_{SME} = \frac{1}{N} \sum_{i=1}^{N} e_i$$  \hspace{1cm} (10)

All these five error indicators measure the polygons’ size differences between those transformed by a cartogram algorithm and those determined by population. A smaller indicator means less difference and higher effectiveness. Because the Keim error is always lower than the standard error, it tends to overestimate the effectiveness of a cartogram algorithm. Despite such difference in definition, these error indicators follow the same trend in most cases and are rather consistent when comparing cartogram algorithms. In this article, Keim size-error indicators are only used for comparison purposes. Discussion of cartogram effectiveness is mainly based on the area-weighted RWSE because area-weighted errors are more indicative as they take polygon sizes into account.
Besides these quantitative metrics, topology preservation can be examined using generic GIS software. In addition, it is easy to visually and mathematically check the quadtree produced by Carto3F in each step and to verify its topological correctness. Keim et al. (2004) proposed shape error metrics based on edge length ratios and angles; however, they are not quite intuitive in terms of measuring and comparing the degree of shape preservation and being recognizable, especially when the maps are complex and not intentionally simplified. This article, therefore, simply juxtaposes cartograms generated by Carto3F and other algorithms to visually evaluate their shape preservation. This practice is widely used in similar research works (e.g., Gastner and Newman 2004, Keim et al. 2004, Henriques et al. 2009).

5.2. Population cartograms of the United States, China, and the world

Among these three cartograms produced by Carto3F, the US population cartogram is the easiest one because US states, as far as sizes, shapes, and population density are concerned, are not as diverse as provinces of China or world countries. As a result, the United States is the most commonly used example in cartogram algorithm development. To this end, this article presents the first three steps of the transformed continental US population map as well as the quadtree used to interpolate polygons (Figure 8). For better visual effect, the quadtree is intentionally made full and thus resembles a regular grid. In each step, Carto3F updates the quadtree and also uses the newly transformed quadtree to interpolate the polygons produced in the previous step (Figure 8a–c). If these quadtrees are aggregated, the overall transformation effects could be represented as accumulatively transformed quadtrees (Figure 9). These accumulative quadtrees could then be used as skeletons to interpolate any spatial data within the map boundary. Figure 9a combines the transformations represented in Figure 8a and b; Figure 9b combines Figure 8a–c.

One drawback of using simple circles and squares to approximate polygons is their inaccuracy and insufficiency in approximating complex shapes with low roundness like Inner Mongolia of China and Canada. One remedy for this problem is to divide a map into smaller, more evenly sized regions by overlaying a fishnet on the original polygons. Because Carto3F can completely preserve topology, a standard spatial union operation can merge different parts of the transformed polygons back together. Maps of China and the world are processed using this method. The cartogram of China and the non-full quadtree after the first-step transformation are plotted together to illustrate how the quadtree structure adaptively captures the distribution of polygon vertices (Figure 10). Each polygon vertex is contained by a quadtree leaf node and each node could index multiple vertices. Figure 11 maps China population cartogram produced by Carto3F after steps 3 and 7. For the world population cartogram, the intermediate step 5 is presented in Figure 12 and the final one is presented in Figure 15.

5.3. Algorithm performance

To make a more informative and more convincing statement of the effectiveness and efficiency of Carto3F, key performance metrics are calculated for the United States, China, and the world population cartograms (Table 1). These quantitative measures include step number, time, global elasticity coefficient, and five global size-error indicators specified above. The computation was conducted on a computer with an Intel® Core 2® Q6600 (a 2.4 GHz quad-core CPU released in 2007) and a 4 GB memory. The operating system is Windows 7 Home Premium®. Carto3F algorithm is implemented with C++ and MFC®.
Figure 8. Iterative transformation of the continental United States: (a)–(c) steps 1–3 (upper part is transformed polygons; and lower part is quadrees used to interpolate polygons).
using Visual Studio 2008®. It must be pointed out that the computational platform only influences the time for each step and has no impacts on the global elasticity coefficient and the size-error indicators.

Carto3F generates a quality cartogram for the continental United States almost instantly (0.64 seconds) using a quadtree with a depth of 6, which is equivalent to a $64 \times 64$ grid. Four out of the five error indicators for the US cartogram are below 4%, with the RMSE being 6.4%. Although Carto3F takes more steps (9 vs. 7) and longer time (18 seconds vs. 0.64 seconds) to generate the China population cartogram than the US one, it reduces all errors below 4%. The performance of Carto3F obviously drops when maps become more complex. It takes about 3 minutes to decrease the weighted mean Keim error of the world population cartogram below 5% when the quadtree depth is 10 (equivalent to $1024 \times 1024$ grid). After 11 steps and 7 minutes, the RMSE of the world population cartogram is still as high as 16.7, that is, 1670%, and the weighted RWSE is 30%. Nonetheless, it is the first world population cartogram produced by a rubber-sheet algorithm. It should be noted that these results are still better than the 4036% RMSE and 65% RWSE achieved by the diffusion algorithm (for more details, see below).

To compare the effectiveness of Carto3F with other cartogram algorithms, the Keim size-error indicators of the US cartogram are assembled from published works (Table 2). These algorithms include the original rubber-sheet algorithm (DCN), Carto-SOM, the diffusion algorithm, and Opti-DCN. Other prominent algorithms are not included because they either did not report the Keim size error or they used highly generalized US maps. From these results, Carto3F is able to achieve the smallest size error on almost each of the three Keim errors. Only Carto-SOM is better than Carto3F on the weighted mean error metric. Although Opti-DCN and Carto3F use the same set of transformation equations, Carto3F achieves better results than Opti-DCN because it accomplished more iterations within shorter time (for more details, see below).

Although comparison with published results sheds light on the effectiveness of Carto3F, it provides no information on efficiency, that is, how fast cartograms converge to equal density status. To answer this question, Carto3F is further compared against Opti-DCN and the diffusion algorithm, arguably the best cartogram algorithm so far. The diffusion algorithm is implemented in ScapeToad program (http://scapetoad.choros.ch/). Results from Opti-DCN, Carto3F, and ScapeToad were generated on the same computer. All programs take

Figure 9. Cumulative transformation quadtrees: (a) sum of quadtrees in 8a and b and (b) sum of quadtrees in 8–c.
Figure 10. Transformed polygons and accompanying quadtrees for step 1: (a) all provinces of China and (b) Guangdong province.
Shapefiles as inputs and use quadtree or grid at the same resolution. Because Opti-DCN is very slow to produce cartogram for the world population, Opti-DCN is only compared using China population cartogram. For Carto3F and ScapeToad, both China and the world cartograms are used for comparison.

For China, Opti-DCN and Carto3F approximately achieve the same level of effectiveness for each step because they use the same mathematics of topology. Regarding speed, however, Carto3F is about nine times faster than Opti-DCN on the quad-core computer (Figure 13a). The first iteration of Opti-DCN takes 39 seconds and Carto3F takes 2.5 seconds. Although the time used for later iteration gradually decreases to about 13 seconds, one step of Opti-DCN consumes 18 seconds on average. By contrast, Carto3F only needs about 2 seconds for each step, which is nine times less. It is clear that Carto3F is faster and has much better performance than Opti-DCN on computational efficiency.

For both China and the world, the diffusion algorithm has a higher initial efficiency than Carto3F (Figure 13b and c). In particular, the first iteration of the diffusion algorithm is more efficient, although later iterations achieve much less than the first one as the size-error curve quickly becomes level. Carto3F, on the contrary, requires more iterations to match the diffusion algorithm. For example, for the China cartogram, the first iteration of the diffusion method achieves roughly the same result as the first three iterations of Carto3F (Figure 13b). For the world cartogram, the second iteration of the diffusion is close to the sixth iteration of Carto3F (Figure 13c).

When compared using absolute time, however, Carto3F is faster and more efficient than the diffusion algorithm. This is particularly true for the simpler China cartogram. Carto3F’s time-error curve is lower than the diffusion algorithm, which suggests that with the same amount of time, Carto3F reduces more size error and is more efficient (Figure 13b). Or in another way, Carto3F is faster than the diffusion algorithm to reach the same level of size error. With about 9 seconds, Carto3F decreases RWSE to 0.05; the diffusion algorithm reduces this error only to 0.13 in 17 seconds. For the case of the world population cartogram, the diffusion algorithm is more efficient in the beginning. After about 4 minutes, however, Carto3F outperforms the diffusion method (Figure 13c). For example, the diffusion algorithm reduces RWSE to 0.66 after 6 minutes; with the same time, Carto3F achieves a smaller error of 0.37. Overall, although Carto3F takes more algorithmic iterations, it consumes less time than the diffusion algorithm to generate quality cartograms.
Figure 12. World population cartogram, step 5.
Table 1. Carto3F algorithm performance.

<table>
<thead>
<tr>
<th>Step</th>
<th>( t ) (seconds)</th>
<th>( c )</th>
<th>( I_{\text{RMSE}} )</th>
<th>( I_{\text{RWSE}} )</th>
<th>( I_{\text{MQE}} )</th>
<th>( I_{\text{WME}} )</th>
<th>( I_{\text{SME}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>0</td>
<td>−</td>
<td>3.9849</td>
<td>1.5798</td>
<td>0.0713</td>
<td>0.3748</td>
<td>0.4159</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.13</td>
<td>0.043</td>
<td>0.9939</td>
<td>0.5560</td>
<td>0.0414</td>
<td>0.1924</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.12</td>
<td>0.084</td>
<td>0.3068</td>
<td>0.1987</td>
<td>0.0171</td>
<td>0.0596</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.08</td>
<td>0.157</td>
<td>0.1693</td>
<td>0.1052</td>
<td>0.0131</td>
<td>0.0328</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.07</td>
<td>0.354</td>
<td>0.1878</td>
<td>0.1265</td>
<td>0.0119</td>
<td>0.0329</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.08</td>
<td>0.291</td>
<td>0.1170</td>
<td>0.0694</td>
<td>0.0081</td>
<td>0.0219</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.08</td>
<td>0.590</td>
<td>0.0967</td>
<td>0.0585</td>
<td>0.0068</td>
<td>0.0186</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.08</td>
<td>0.718</td>
<td>0.0641</td>
<td>0.0375</td>
<td>0.0043</td>
<td>0.0149</td>
</tr>
<tr>
<td>China</td>
<td>0</td>
<td>−</td>
<td>10.8862</td>
<td>3.4172</td>
<td>0.0974</td>
<td>0.4407</td>
<td>0.4887</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.49</td>
<td>0.028</td>
<td>6.0146</td>
<td>1.8192</td>
<td>0.0723</td>
<td>0.1899</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.40</td>
<td>0.043</td>
<td>1.6101</td>
<td>0.4729</td>
<td>0.0504</td>
<td>0.0654</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2.18</td>
<td>0.042</td>
<td>0.4306</td>
<td>0.1386</td>
<td>0.0371</td>
<td>0.0301</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.77</td>
<td>0.056</td>
<td>0.1870</td>
<td>0.0536</td>
<td>0.0276</td>
<td>0.0167</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.65</td>
<td>0.101</td>
<td>0.1986</td>
<td>0.0575</td>
<td>0.0251</td>
<td>0.0136</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1.73</td>
<td>0.086</td>
<td>0.1482</td>
<td>0.0376</td>
<td>0.0198</td>
<td>0.0098</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1.81</td>
<td>0.079</td>
<td>0.1191</td>
<td>0.0344</td>
<td>0.0131</td>
<td>0.0103</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1.93</td>
<td>0.068</td>
<td>0.0585</td>
<td>0.0252</td>
<td>0.0055</td>
<td>0.0088</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>2.04</td>
<td>0.085</td>
<td>0.0354</td>
<td>0.0204</td>
<td>0.0029</td>
<td>0.0080</td>
</tr>
<tr>
<td>World</td>
<td>0</td>
<td>−</td>
<td>123.133</td>
<td>5.581</td>
<td>0.038</td>
<td>0.509</td>
<td>0.492</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>57.0</td>
<td>0.020</td>
<td>103.323</td>
<td>4.324</td>
<td>0.031</td>
<td>0.236</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>49.7</td>
<td>0.027</td>
<td>79.240</td>
<td>3.025</td>
<td>0.026</td>
<td>0.089</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>40.7</td>
<td>0.026</td>
<td>63.957</td>
<td>2.265</td>
<td>0.024</td>
<td>0.050</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>36.9</td>
<td>0.038</td>
<td>49.534</td>
<td>1.555</td>
<td>0.021</td>
<td>0.027</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>34.8</td>
<td>0.046</td>
<td>38.992</td>
<td>1.056</td>
<td>0.019</td>
<td>0.017</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>33.3</td>
<td>0.040</td>
<td>33.780</td>
<td>0.828</td>
<td>0.018</td>
<td>0.014</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>32.6</td>
<td>0.075</td>
<td>26.995</td>
<td>0.559</td>
<td>0.017</td>
<td>0.012</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>32.5</td>
<td>0.077</td>
<td>22.711</td>
<td>0.434</td>
<td>0.015</td>
<td>0.011</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>32.8</td>
<td>0.070</td>
<td>19.757</td>
<td>0.366</td>
<td>0.015</td>
<td>0.013</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>33.0</td>
<td>0.039</td>
<td>18.541</td>
<td>0.339</td>
<td>0.014</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>32.7</td>
<td>0.063</td>
<td>16.694</td>
<td>0.303</td>
<td>0.014</td>
<td>0.010</td>
</tr>
</tbody>
</table>

Notes: The quadtree depths for the United States, China, and the world are 6, 8, and 10, respectively. \( t \) (s) is running time in seconds and \( c \) is the global elasticity coefficient.

From these basic measures, it is clear that Carto3F algorithm, with auxiliary quadtree, parallel computation, and force cancellation mitigation, can achieve a higher level of efficiency than other well-performed algorithms. Carto3F’s speed advantage largely results from the use of quadtree and parallel computation. The force cancellation mitigation in Carto3F also contributes. For example, Singapore is minimally transformed by the
Figure 13. Efficiency comparisons: (a) Opti-DCN versus Carto3F on China, (b) Diffusion algorithm versus Carto3F on China, and (c) Diffusion algorithm versus Carto3F on world.

Note: Each point on the curves represents one step/iteration. The numbers labeling points are RWSE. The values of RWSE for Carto3F are given in Table 1.
diffusion algorithm while Carto3F enlarges it significantly (see insets of Figure 15). Carto3F seems particularly suitable for simple maps. The weighted error of US population cartogram is reduced below 5% within 1 second. This potentially makes it feasible to generate cartograms on the fly, either online or on mobile devices.

Unlike the computational efficiency and size-error measures, topological integrity and shape preservation are only visually examined here. From the transformed quadtree, it is easy to verify that Carto3F algorithm completely preserves topology (Figure 9). The quadtree in each step has no crossing or overlapping and all quadtree nodes retain their topology (Figure 9a and b). Carto3F algorithm has a quality of preserving shapes and cartograms generated by this algorithm are generally recognizable. Carto3F has smaller ‘ballooning’ effect in many cases. In the US population cartogram, for instance, the shapes of Cape Cod of Massachusetts, Long Island of New York, and the state of Nevada are better preserved by Carto3F than DCN or the diffusion algorithm (Figure 14). The shape-preserving ability of Carto3F algorithm is also evident in the world population cartogram. For example, India visually suffers less from the ballooning effect in Carto3F algorithm than in the diffusion algorithm (Figure 15).

6. Conclusion
Rubber-sheet algorithms are among the earliest computerized methods for cartogram construction. Despite their popularity and simplicity, existing rubber-sheet algorithms,
particularly Opti-DCN, work at less than full capacity due to possible topology violation and slow transformation speed. Through mathematical and computational optimization, Carto3F algorithm presented in this article completely excludes the possibility of topological errors. In addition, Carto3F is highly efficient and its transformation effectiveness and speed compare to the best. With an ordinary 2.40 GHz quad-core CPU, it can produce the US population cartogram within only 1 second, China cartogram within 18 seconds, and the world within 8 minutes. On the same computational platform, Carto3F is about nine times faster than Opti-DCN on China cartogram; it is also faster than the diffusion algorithm on China and on the more complex world population cartogram.

The transformational effectiveness and computational efficiency of Carto3F algorithm are mainly gained from combining an auxiliary quadtree structure with the mathematics of topology. As an indexing and compressing spatial structure, the quadtree is self-adaptive to map complexity and helps reduce the number of points to be transformed, thereby greatly improving the efficiency of Carto3F. Moreover, Carto3F algorithm supports parallel computing and can take advantage of the increasingly common multi-core, multi-thread processors. Its mathematically deduced force cancellation mitigation strategy also helps transform small but high-density areas more efficiently. Although Carto3F algorithm can produce cartograms with efficiency and topology preservation, it is also a generic cartogram solution and allows for alterations like force cancellation mitigation. Carto3F algorithm requires little calibration for parameters. One of the most important parameters in Carto3F algorithm, the global elasticity coefficient, is endogenously determined by the internal relationships between transforming forces and their partial derivatives. Combining these advantages, Carto3F algorithm provides a flexible, error-free, efficient, yet easy to understand framework for contiguous area cartogram construction.
Carto3F is still open for further improvement. Different forms of force equations, for example, polynomial equations, could be applied to achieve transformation that has different characteristics. The simple circles and squares could also be replaced by geometries that better describe the polygon shapes. Cartographic generalization routine, for instance, can be used to generate simplified shapes of polygons, to improve the geometric similarity between the polygons and their approximations, and to better preserve shapes. Additionally, the single global elasticity coefficient could be replaced by multiple local or regional coefficients that can capture force dynamics across the space and can further increase the transformation efficiency. These directions warrant further inquiry.

Acknowledgments
This research is supported in part by the University of Minnesota and the University of Waterloo. The author greatly acknowledges the efforts of Dr. May Yuan and five anonymous reviewers for their valuable comments and constructive suggestions. Prof. Zhilin Li and Ms. Hui Sun kindly shared the provincial boundary data of China. The author is solely responsible for any other errors.

References
Dorling, D., 1996. Area cartograms: their use and creation. Norwich: School of Environmental Sciences, University of East Anglia.


