

## CHAPTER X

### DISCUSSION

#### 10.1 Overview

In this chapter, we compare continuous cartograms produced using our *Constraint-Based Method* with those of the previous methods reviewed in Chapter III. We also discuss directions for improvement of the algorithm.

#### 10.2 Comparison of Results with Existing Methods

This section is ordered in chronological progression, by decade, of population cartograms of the United States. The cartograms have been placed adjacent to each other and sized to be at the same scale, as much as possible, to provide fair comparisons. An original map of the United States is provided in Figure 10.1 for the viewer as a reference.

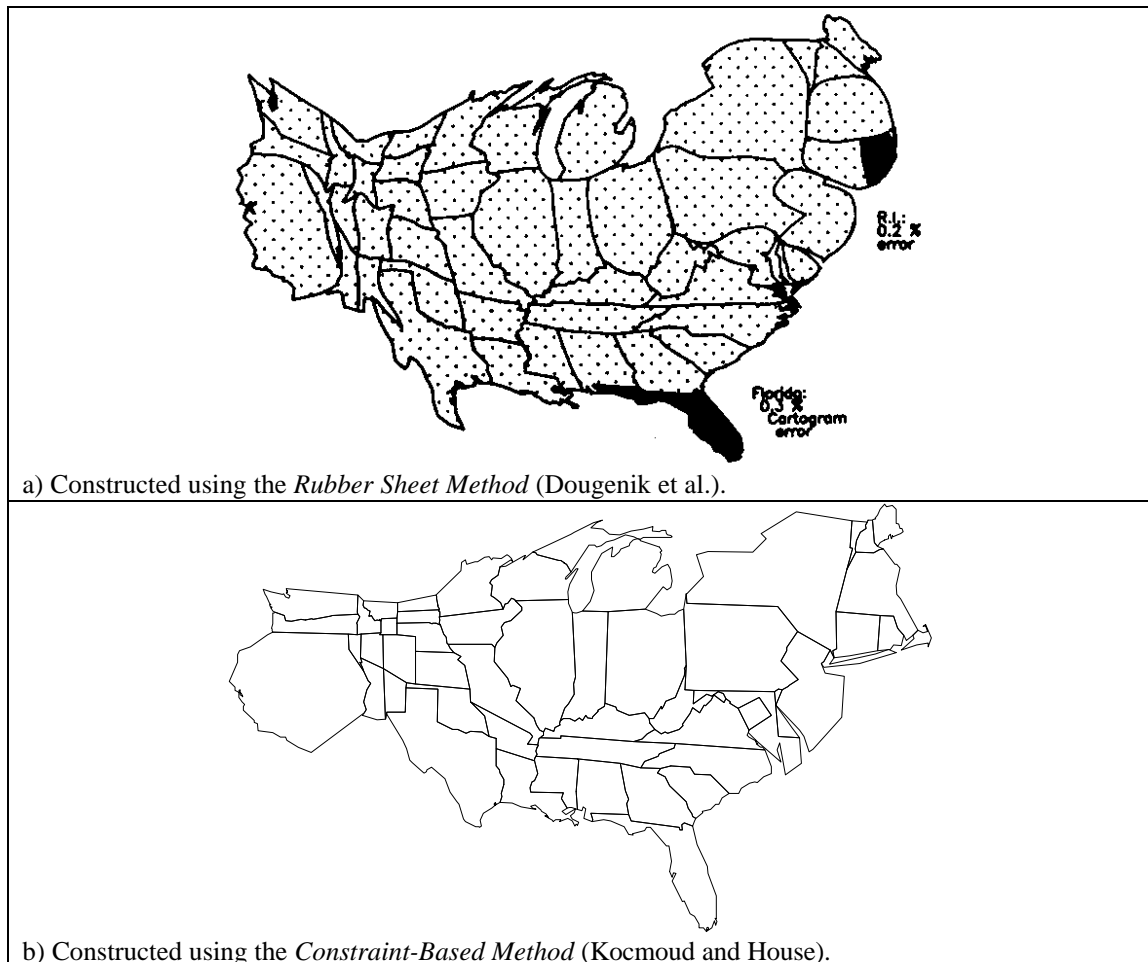


**Figure 10.1:** Map of the United States.

##### 10.2.1 1960 U.S. Population Cartogram

The 1960 population cartogram constructed using the *Rubber Sheet Distortion Method*, by Dougenik et al., is shown in Figure 10.2 alongside our cartogram. There is a lot of similarity between the two cartograms in the warping of the central and southeastern states. Their method retains a better likeness of the west coast, but produces a severe pinched distortion of the non-coastal western states that inhibits recognition. In

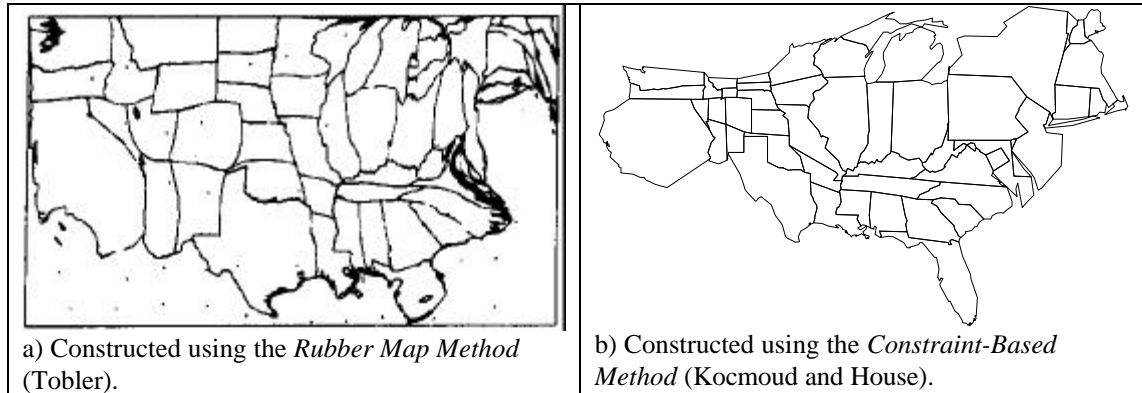
addition, the overall appearance of New England and the east coast is better represented in our cartogram when compared to the “ballooning” effect produced by their method.



**Figure 10.2:** 1960 U.S. population cartograms (a: Same as Figure 3.3).

### 10.2.2 1970 U.S. Population Cartogram

The 1970 U.S. population cartogram constructed using Tobler’s *Rubber Map Method* is shown in Figure 10.3 alongside our cartogram. Note that Tobler’s cartogram converged to only 68% of its intended accuracy, as is quite evident in the comparison. It would appear that this truly is “a cartogram of a different variable than the one intended” [6].



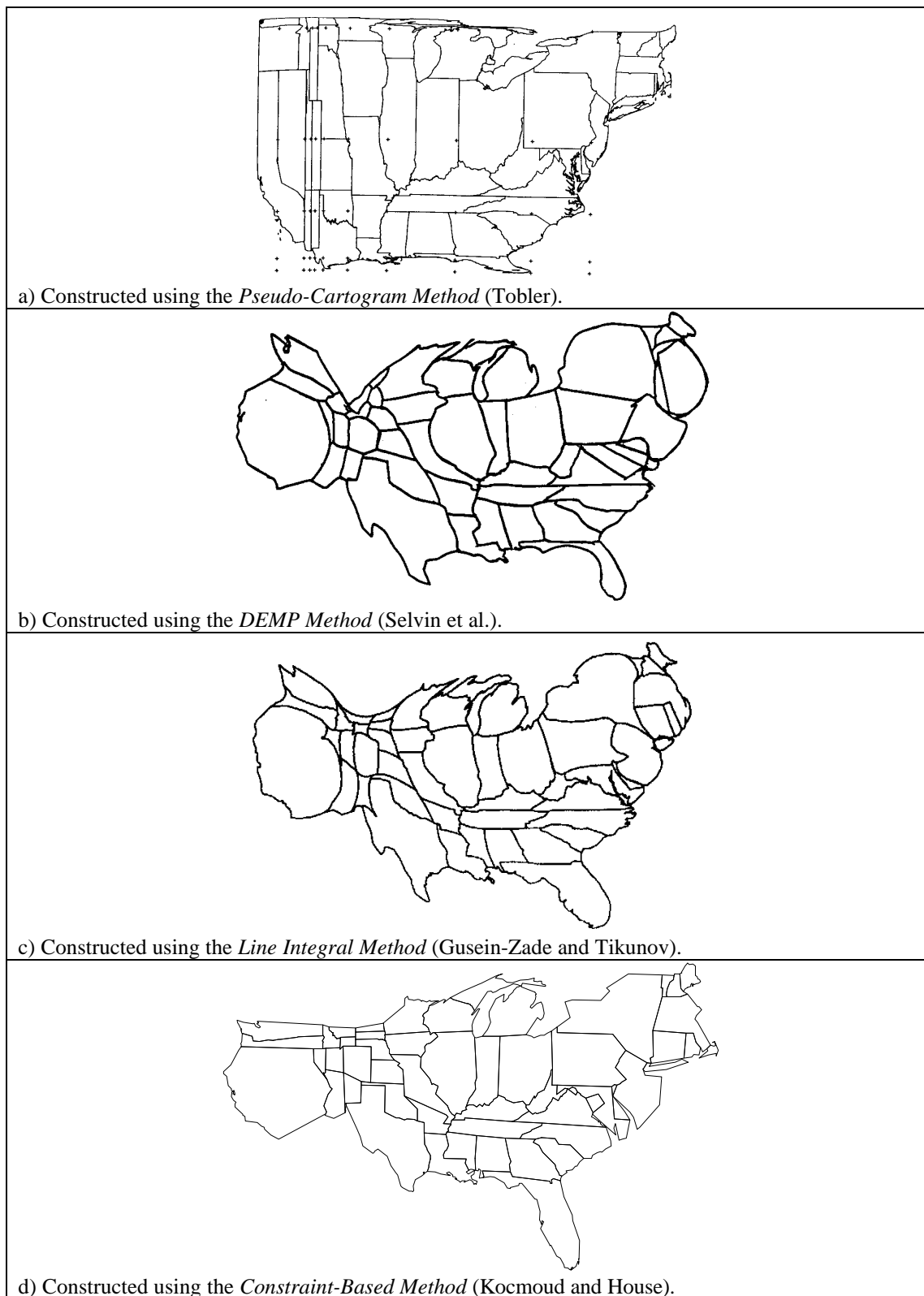
**Figure 10.3:** 1970 U.S. population cartograms (a: Adapted from Figure 3.1).

### 10.2.3 1980 U.S. Population Cartogram

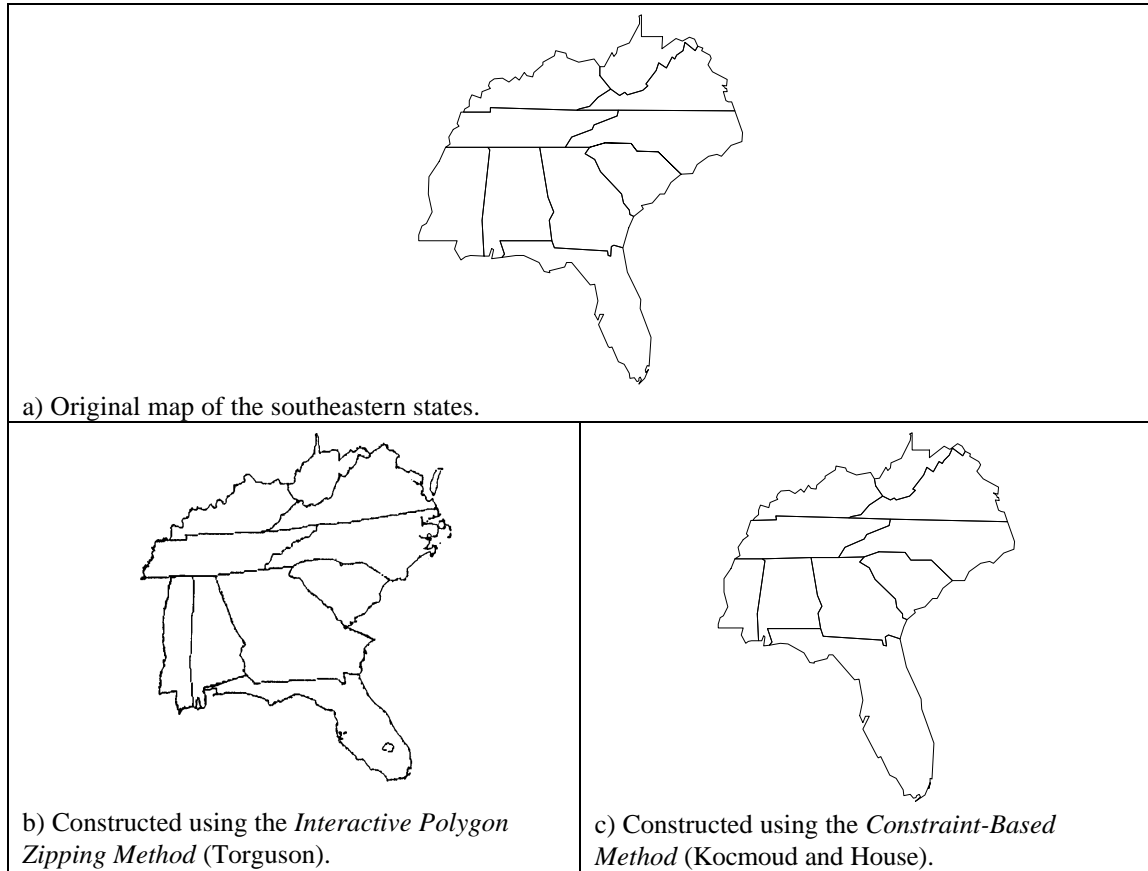
Tobler's *Pseudo-Cartogram Method* is demonstrated in Figure 10.4a. The amount of error contained in his approximation is made quite obvious when viewed next to other cartograms. However, recognition of the states is quite good since the detailed borders of the states have not been sacrificed or generalized.

The 1980 U.S. population cartogram in Figure 10.4b was constructed using the *DEMP Method* by Selvin et al. It is worth noting some similarity in warping of the southeastern states when compared to our cartogram in Figure 10.4d. However, state recognition is a challenge in the *DEMP* cartogram due to the severe distortion of the non-coastal western states, in particular, as well as the overall loss of detail due to the radially ballooning effect produced by the algorithm. For example, it is especially difficult to determine what state is represented by the “bubble” to the right of New York.

The 1980 U.S. cartogram created by Gusein-Zade and Tikunov is shown in Figure 10.4c. There are many similarities in appearance between this *Line Integral* cartogram and the previous *DEMP* cartogram due to the common radial nature of the algorithms. While sharing the same ballooning and X-shape tendencies, this method is much better in its handling of the northwestern and New England states. However, it lacks some of the clarity and precise shape preservation that is obtained with our method, as shown in Figure 10.4d.



**Figure 10.4:** 1980 U.S. population cartograms (a: Same as Figure 3.4b, b: Same as Figure 3.2b, c: Same as Figure 3.7b).



**Figure 10.5:** Cartograms of the 1980 population of the southeastern U.S. (b: Same as Figure 3.5b).

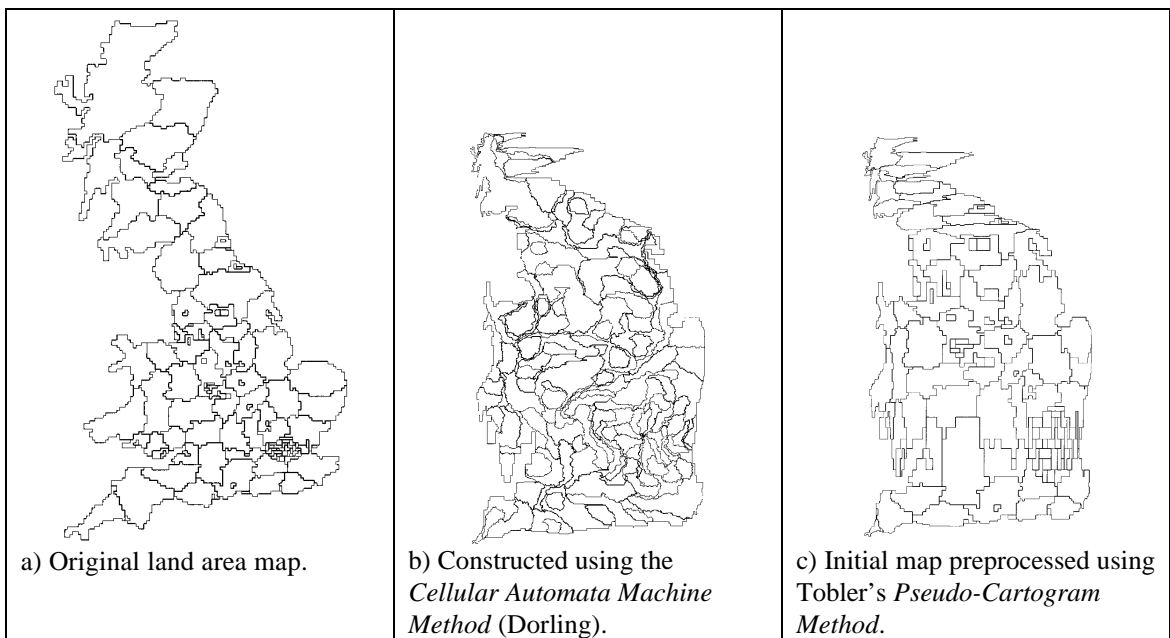
Torguson used a subset of the United States in his 1980 population cartogram, as shown in Figure 10.5b. While Torguson's cartogram correctly maintains topology, it is slightly deceptive in that the top-left corner of Mississippi is not attached to the lower-left corner of Tennessee. In addition, there appears to be a large discrepancy in the size of Georgia when compared to our cartogram in Figure 10.4c, which contains under 1% average error.

#### 10.2.4 1981 Britain Population Cartogram

This section contains comparative results from Dorling's *Cellular Automata Machine Method* and our *Constraint-Based Method*. It is important to note, however, that Dorling has mapped not only the British counties, but also the major cities.

Therefore, we will compare the initial map and cartogram of each method and separately evaluate how well each has preserved region recognition.

A map of British counties and major cities is shown in Figure 10.6a. Recall that Dorling’s cartogram method is based on the “Game of Life,” which drastically generalizes region shapes. Surprisingly, Dorling’s cartogram in Figure 10.6b does a fair job at preserving some of the region shapes along the coast. However, it must be noted that this image was pre-processed using Tobler’s *Pseudo-Cartogram Method*, producing the result in Figure 10.6c prior to the start of Dorling’s method. It is very evident that any shape preservation was purely the result of the pre-processing, and not Dorling’s algorithm.

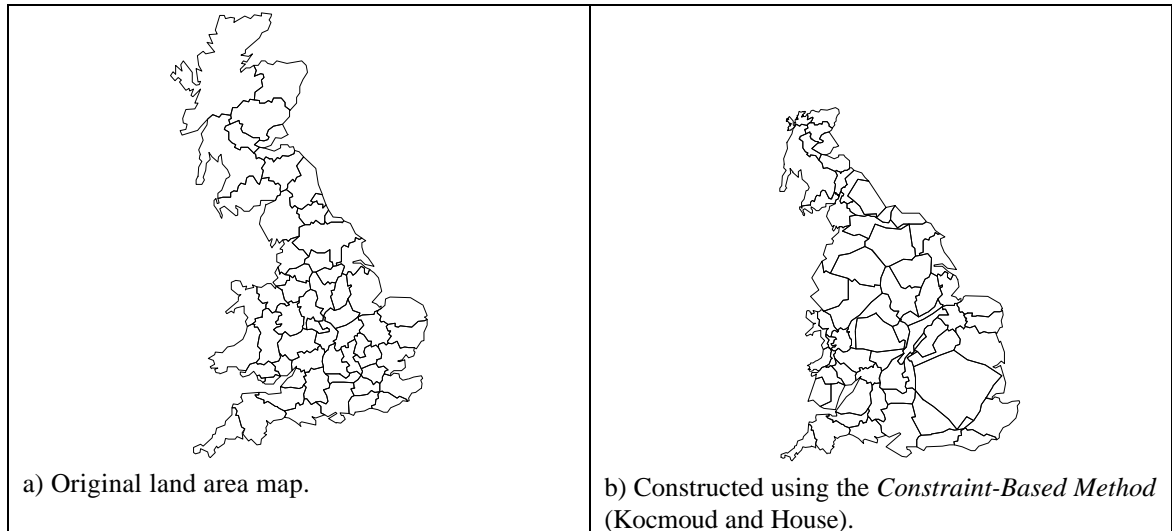


**Figure 10.6:** 1981 equal population cartogram of British counties and major cities (a, b: Same as Figure 3.6, c: Reproduced with permission from [4], page 22, figure 14d).

An original map of the British counties and a cartogram produced using our method is shown in Figure 10.7a and b, respectively. Overall, the average viewer would have little trouble recognizing cartogram regions from the original map.

### 10.3 Comparison of Characteristics with Existing Methods

It seems appropriate to apply the same critique toward our method as given in the method review in Chapter III. Therefore, Table 3.2 has been modified to include a column for our method and included as Table 10.1.



**Figure 10.7:** 1981 equal population cartogram of British counties.

Our method is traversal order independent since all area, shape, and constraint forces are distributed into force accumulators and then acted upon at once during the point dynamics procedure. Likewise, our method is independent of coordinate axes since all force mechanisms are based on relative error and not absolute. For example, an orientation spring attempts to restore an edge to its initial orientation, without regard to the chosen axes. However, our method contains the flexibility to apply stronger penalties to chosen orientations; to secure the horizontal and vertical edges, for example. We have not done this in any of the examples shown here.

Our area forces are not global in nature, although this could be easily be accomplished. Currently, each region exerts force only upon its vertices to achieve its desired area, producing a local affect upon the adjacent regions. Our shape restoration process, however, is very global in nature. The dynamic area constraints look at the big picture, incorporating the needs of all of the regions in its calculations. All fifty-two states of the U.S. are analyzed, for example, before constraint forces are calculated that

attempt to prevent region areas from varying. Thus, constraint forces are the result of a balanced, global assessment of needs and consequences.

Our method meets the conformal mapping property, since local angles in the detailed region boundaries are preserved without generalizations, despite the distortion induced upon the cartogram. Likewise, the difficult problem of region self-intersection has been addressed.

<b>Cartogram Characteristics</b>	Tobler (1973) <i>Rubber Map</i>	Selvin et al. (1984) <i>DEMP</i>	Dougenik et al. (1985) <i>Rubber Sheet Distortion</i>	Tobler (1986) <i>Pseudo-Cartogram</i>	Torguson (1990) <i>Inter: Polygon Zipping</i>	Dorling (1990) <i>Cellular Automata Machine</i>	Gusejin-Zade, Tikunov (1993) <i>Line Integral</i>	Kocmoud, House (1997) <i>Constraint-Based</i>
1. Independent of region traversal order	<b>YES</b>	No	<b>YES</b>	<b>YES</b> <sup>1</sup>	<b>YES</b> <sup>1</sup>	<b>YES</b> <sup>1</sup>	<b>YES</b>	<b>YES</b>
2. Independent of coordinate axes	No	<b>YES</b>	<b>YES</b>	No	No	<b>YES</b>	<b>YES</b>	<b>YES</b>
3. Global displacements per iteration	No	No <sup>3</sup>	<b>YES</b>	<b>YES</b>	<b>YES</b> <sup>1</sup>	No	<b>YES</b>	<b>YES</b> <sup>7</sup>
4. Conformal mapping	<b>YES</b>	<b>YES</b>	<b>YES</b>	<b>YES</b>	<b>YES</b>	No	<b>YES</b>	<b>YES</b>
5. Intersection prevention	<b>YES</b> <sup>2</sup>	No	No <sup>4</sup>	<b>YES</b> <sup>1</sup>	<b>YES</b> <sup>1</sup>	<b>YES</b> <sup>1</sup>	No <sup>6</sup>	<b>YES</b>
6. Ability to fix points	No	No	No	No	No	<b>YES</b>	No	<b>YES</b>
7. Incorporation of user controls on area vs. shape	No	No	No	No	<b>YES</b> <sup>5</sup>	No	No	<b>YES</b>
<p><sup>1</sup> Not applicable and, therefore, satisfies this characteristic.</p> <p><sup>2</sup> A time-intensive topological test prevents the vertices from crossing [23].</p> <p><sup>3</sup> Each transformation adjusts the area of only one region without concern for the other regions [10].</p> <p><sup>4</sup> Although placing extra vertices at regular intervals and splitting regions into convex sections effectively prevents self-intersection [6].</p> <p><sup>5</sup> Although the user has <i>too much control</i> and practically does <i>all</i> the work.</p> <p><sup>6</sup> Self-intersection can be avoided by adding more vertices or decreasing the time step, thus increasing computation time [15].</p> <p><sup>7</sup> During shape restoration only, not while achieving area.</p>								

**Table 10.1:** Summary of characteristics of the continuous cartogram methods.

Our *Constraint-Based Method* also permits vertices to be fixed upon the map. This feature would allow building an interface so the users could adjust the final

cartogram by “pinning down” aesthetic modifications and watching the other vertices adjust accordingly during a few additional iterations. However, we have not done this in any of our examples here, with the exception of the fixed perimeter example of the southeastern U.S. in Figure 9.7c.

Another important feature of our method is the weighing of area versus shape. This would allow building a simple slider bar interface to adjust the level of area accuracy with respect to sacrifice in shape. This can be accomplished via the force scaling parameters, as well by adjusting the threshold that, once achieved, signals a stagnation. In fact, the method is sufficiently flexible to allow individual shape or area springs to be strengthened or diminished, before or after the cartogram is constructed, allowing hand adjustment aesthetics. Again, we point out that all maps presented here were created completely automatically without any hand adjustment.

## **10.4 Recommendations for Improvement**

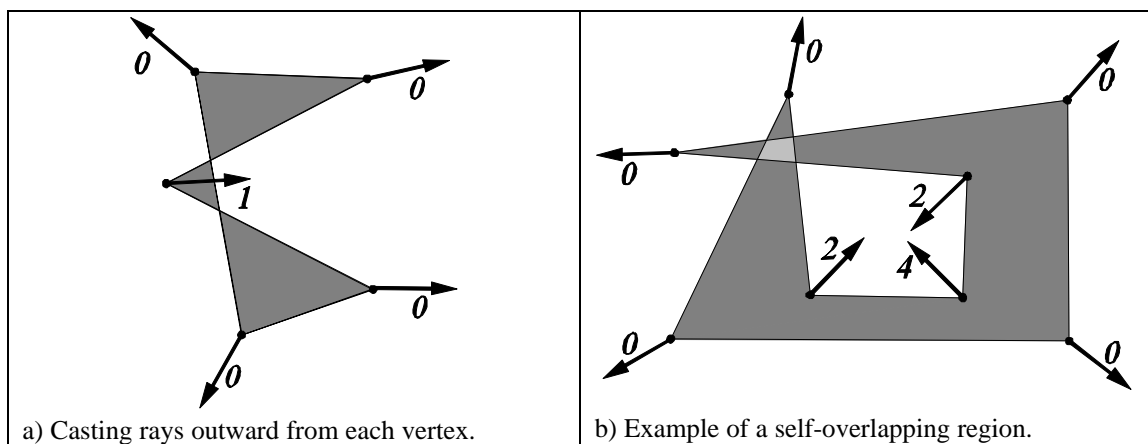
We believe that our *Constraint-Based* cartogram construction method produces results demonstrably superior to any of the other automatic methods we have found. However, there is always room for improvement. Throughout the development process, the algorithm has undergone many changes as we struggled to overcome hurdles. This section proposes additions or modifications to the method that would make important improvements.

### **10.4.1 Intersection Detection and Handling**

As mentioned in Section 7.3.1, our method of detecting intersections does not work in all cases. Our current method detects intersections by marching around the region boundary searching for intersection types “A” through “D,” (see Figure 7.8). A sometimes faulty assumption of our method is that the detection begins in a valid state (i.e. the first vertex we analyze is not currently penetrating a region). If detection does begin in an invalid state we get an effect opposite to what is desired, forcing the entire region to invert. The other case where our current method is not consistently correct is when multiple complex intersections occur adjacent to (or overlapping) each other.

When this occurs, the algorithm can incorrectly pair up the crossover points, resulting in valid vertices mistakenly receiving intersection penalty forces. We have concluded that searching for specific intersection types does not appear to be an ideal method.

A better intersection detection method could be realized by casting rays outward from each vertex and counting the number of intersections with the other region edges. An odd number of intersections would signal a penetrating vertex. In the example region in Figure 10.8a, only vertex  $P$  results in an odd number of intersected rays. The rays should be cast in the direction that bisects the exterior angle of the two edges adjacent to the vertex, similar to the area spring force directions given in equation 4.4.



**Figure 10.8:** Examples of the proposed intersection detection method.

When a penetrating vertex is found, a search could be performed to locate the edge, or edges, that intersect the two edges adjacent to the vertex. The intersection preceding the vertex marks the crossover where the intersection begins, and the subsequent crossover marks the end of the intersection.

*Self-overlapping* regions, such as the example in Figure 10.8b, will not be detected by ray casting. Overlapping occurs when one or more vertices penetrate through two “layers” of edges. An adapted marching technique could be used to detect overlapping, in which we begin marching around the region perimeter from an initial valid state, detecting intersections between the current edge and all other edges in our region. The first crossover signals the beginning of an invalid state and the next

crossover toggles back to a valid state. Multiple crossovers along a single edge can be sorted by distance to ensure their correct order. Since there will always be two pairs of intersections, they can be intelligently grouped and resolved together.

Another shortcoming of our current method is that the application of forces is very dependent upon “type” classification. If an intersection starts as Type “B”, as demonstrated in Figure 7.4b, and later is partially corrected to a Type “A” intersection, it is treated as an entirely new intersection. All of the accumulated forces are lost, as the new Type “A” intersection begins its battle. The adverse outcome is that the penetrating vertex is given momentary leeway, due to the sudden reduction in penalty force, and could penetrate farther. This could result in an intersection tennis match, bouncing back and forth between a Types “A” and “B” without resolving the problem.

Therefore, in conjunction with the proposed ray-casting detection method, a single intersection model should be used that monitors the progression of each intersection until it is eventually resolved. In this manner, when a single complex intersection splits into several simple ones, the accrued force will be inherited.

#### **10.4.2 Convergence Optimization**

The length of time required for the cartogram to converge to a solution can be reduced in at least two ways. First, it would be desirable to use Tobler’s *Pseudo-Cartogram Method* to pre-process (i.e. optimize) the input map (see Figure 10.4a). In terms of the U.S. examples, this would provide a large performance increase since the noncoastal western states would already be condensed and the west coast states relocated eastward prior to the start of our method.

Performance time would also benefit from implementing *global* vertex displacements. Regions could project area resizing forces not only upon their vertices, but upon other region vertices at a magnitude inversely proportional to distance. In this manner, distant regions would begin preparing in advance for a dilating neighbor “one or two houses away.”

### 10.4.3 Comprehensive Application

Another improvement would be to implement this method as a standalone, platform-independent application. The foundational code already follows the standard C programming language and is compilable on any platform. The MicroStation graphics display calls can be replaced with OpenGL, a platform-independent library of graphics routines. An interface containing all the user controls necessary for creating custom cartograms could be developed and implemented on a windowing package for each platform.

The standalone application would require input conversions from various data file formats, supporting at least the DXF standard. The “playback” capability that exists in our current implementation should also be supported, enabling the user to replay the cartogram creation later in “real time.” Finally, the cartogram result should be exportable to various file and image formats to provide easy use within other applications.

A more suitable approach could be to complete the application for use within a geographical information system (GIS) such as MicroStation Modular GIS Environment (Intergraph Corporation), MicroStation GeoGraphics (Bentley Systems, Inc.), ARC/INFO (Environmental Systems Research Institute, Inc.), or MapInfo (MapInfo Corporation). This approach would simplify map input conversions, interface issues, and cartogram output formats since tools in the respective GIS application could be utilized. The map loading routine could easily be modified to read data linkages associated with the region elements, thereby enabling direct input and output of GIS linked elements. A capability to spatially transform a map to analyze data distribution could be a very useful GIS tool.