Approaches to Map Anamorphosis

by

Colin Klipsch

Project submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science and Applications

©Colin Klipsch and VPI & SU 1993

APPROVED:

Dr. Clifford A. Shaffer, Chairman

Layne Watson

Dr. Layne Watson

Lenwood Heath

Dr. Lenwood Heath

December, 1993

Blacksburg, Virginia
Approaches to Map Anamorphosis

by

Colin Klipsch

Committee Chairman: Dr. Clifford A. Shaffer

Computer Science Department

(ABSTRACT)

Map anamorphosis is the distortion of a map to show graphically the variation of some quantity from region to region. The process of anamorphosis modifies the original map regions, keeping the inter-region topology, to produce new regions whose areas are proportional to their respective values of the relevant quantity. A typical example would be a distorted map of the United States, where each state's area is proportional to its population, yet the states still fit together in the correct way. Such maps, called "cartograms", can provide a good visual sense of where a quantity, such as population, is distributed. In this paper we look at five separate attempts to design a computer algorithm for generating cartograms, all of which use triangulation as a basis, and all of which, unfortunately, are unsuccessful. We also examine a working algorithm in the literature that uses similar ideas in its initial approach. While this algorithm produces aesthetically displeasing results, it may indicate a way to solve the map anamorphosis problem robustly using triangulation.
# TABLE OF CONTENTS

1 INTRODUCTION  
1.1 Problem Overview .................................................. 1  
1.2 Triangulation .......................................................... 5  
1.3 Number of Solutions .................................................. 7  
1.4 Aesthetics .............................................................. 7  
1.5 Outline ................................................................. 10  

2 A SIMPLE METHOD OF TRIANGULATION  
2.1 Introduction ............................................................ 11  
2.2 Method ................................................................. 12  

3 A WORKING ALGORITHM ...................................................... 16  

4 OWNERSHIP OF TRIANGLE PARTS  
4.1 Ownership of Edges .................................................. 21  
4.2 Ownership of Coordinates ......................................... 25  

5 TRANSFORMATION TO A LINEAR SYSTEM  
5.1 Approach ............................................................... 28  

6 NUMERICAL SOLUTION OF THE NONLINEAR SYSTEM  
6.1 Homotopy Method .................................................... 31  
6.2 Powell Hybrid Root Finder ......................................... 33  

7 CONCLUSION  

iii
LIST OF FIGURES

1.1 Population Cartogram of the United States ........................................... 2
1.2 Europe Before Triangulation .............................................................. 8
1.3 Europe After Triangulation ............................................................... 9
2.1 Finding Triangles ............................................................................. 14
3.1 Triangles Sharing One Vertex .......................................................... 18
4.1 Fixing Two Triangle Sides ................................................................. 22
4.2 Overlapping Triangles ................................................................. 25
4.3 Ownership of Vertex Coordinates .............................................. 27
5.1 Simple Triangulation ...................................................................... 28
6.1 Minimizing $f(z)$ ........................................................................ 34
6.2 Nevada and Utah ........................................................................... 36
6.3 Nevada and Utah After Powell ..................................................... 37
Chapter 1

INTRODUCTION

1.1 Problem Overview

Map anamorphosis refers to the distortion of a map to show graphically the variation of a quantity from region to region. A typical example might be a map of the United States where the area of each state has been forced to be proportional to its population, thus making New York and Florida relatively larger — and Alaska and Montana relatively smaller — than they would be on an ordinary equal-area projection map. (See Figure 1.1, which is taken from [9], and which was generated using an algorithm in the same article.) Such maps, called “cartograms”, can give a good visual sense of where a population is distributed. Cartograms might be useful, for example, in computer-assisted education software, such as that being developed under the Project GeoSim here at Virginia Tech. A student could view anamorphated maps to see more clearly where the population is dense or sparse. If anamorphosis can be done quickly, it might be possible for the student to change the population distribution and generate new cartograms dynamically, and thus understand better the significance of the changes.

Of course, quantities other than population are just as valid: annual income, rainfall, and so on. In general, the problem can be stated as follows:

*Given a set of simple, non-overlapping regions in the Euclidean plane, with each region possessing an associated positive real number, modify the shapes of the regions such that their new areas are proportional to their associated numbers by some global constant, while preserving the topology of inter-region adjacencies. Of all possible modified maps which satisfy this requirement, choose one which*
Figure 1.1: Population Cartogram of the United States
CHAPTER 1. INTRODUCTION

*best preserves the shapes of the regions according to some measurement criterion.*

In addition to the required properties of the cartogram, there are some desired properties we would like to achieve:

- The region shapes are preserved as best as possible.
- The anamorphosis is invariant to translation or rotation of the coordinate system.
- No matter how we partition the map regions (into triangles, grid cells, etc.), the final cartogram is the same.
- Substitutions of new values for the vertex coordinates (or whatever variables are being modified) should be simultaneous, not ordered.
- Convergence on a solution is guaranteed.
- The time and space costs are low.

We know of no fully satisfactory algorithms for solving this problem. In the past, various mechanical means have been used, such as projecting photographs through distorted lenses, measuring voltages in non-uniform electric fields, or enclosing ball bearings inside flexible barriers [16]. These methods tend to be cumbersome and inaccurate, and the necessary equipment is not commonly available.

In recent years, a variety of computer algorithms have emerged, most of which attempt to reach a global solution by iterating on local criteria. Many of these techniques are not guaranteed to converge, nor do they necessarily result in aesthetically-pleasing maps. Also, many do not work directly on maps represented by polygons, but rather require a grid overlay to partition the map into squares. Still, they represent the best solutions so far to the map anamorphosis problem, and one such computer algorithm will be discussed in chapter 3.

We studied five alternate approaches to designing a computer algorithm for generating cartograms, all of which are different in philosophy from approaches in the current literature.
CHAPTER 1. INTRODUCTION

Unfortunately, all five attempts were unsuccessful. The base assumptions of each approach were:

- The map regions are represented as non-overlapping simple polygons in the $xy$ plane.
- Each region has a population, which is the "associated positive real number" described in the problem statement above.
- The entire map is triangulated during initialization, for ease of processing.
- The desired areas of the resulting triangles can be deduced from the input population data, and from the sizes of the original triangles and regions.

Thus we have reduced the original problem to that of changing the areas of a triangulation’s triangles from their old values to a known set of new values, while maintaining the topology of the original triangulation. Since the new area of each region equals the sum of the new areas of its component triangles, the modified triangulation will constitute a new set of regions whose areas meet our proportionality requirements, and we will have generated our cartogram.

The rationale for the initial triangulation is to simplify the anamorphosis. Not only are triangles easier to calculate with than the original polygons, but we can also avoid some unwieldy algebra and potential pitfalls. For example, the area of an arbitrary polygon in terms of its edge lengths (and interior angles, which are also needed) would be complicated and seemingly unhelpful in deriving an algorithm to solve the anamorphosis problem. The same expression for the special case of a triangle is fairly simple, and might yield an elegant approach. The area of a polygon as a function of its $n$ vertices, namely

$$A = \frac{1}{2} \sum_{i=1}^{n} x_i y_j - x_j y_i; \quad j = (i \mod n) + 1$$

where $(x_i, y_i)$ are the Cartesian coordinates of the $i$th polygon vertex, can generate positive real values even for self-intersecting polygons. Since all of our polygons are taken to be simple polygons, and must remain that way, it would not be adequate to find a new set
CHAPTER 1. INTRODUCTION

of vertex coordinates which yielded the correct new areas; we would also have to check for self-intersection in every polygon, and figure out a way to eliminate it. Either that, or we would have to guarantee that polygon self-intersection never occurred in the first place. It is not obvious how either of these safeguards could be implemented, to ensure that the simple polygons stay simple after an arbitrary movement of their vertices.

1.2 Triangulation

An example of a map and its triangulation are shown in Figures 1.2 and 1.3. This map of Europe, comprising 27 polygons and 544 vertices, was triangulated using 1625 edges, and resulted in 1082 triangles. There are many choices for the method of triangulation [1, 2, 5, 6, 14, 15], but it must yield a constrained triangulation — that is, the algorithm must incorporate an initial set of edges as part of the final triangulation. In our case, the initial edges will be from the boundaries of the polygons that define the regions.

We are given populations for the regions, not the triangles; however, a simple choice for the population $P_t$ of triangle $t$ belonging to region $r$ with population $P_r$ would be:

$$P_t = \frac{A_t}{A_r} P_r$$

where $A_t$ and $A_r$ are the initial areas of the triangle and region respectively. Then the new area $A'_t$ of triangle $t$ is proportional to its population $P_t$ by some global constant $k$ that is yet to be determined.

$$A'_t = kP_t = k \frac{A_t P_r}{A_r}$$

We can require that the total area of the map remain unchanged after anamorphosis. Equating the total areas of the initial and final maps:

$$\sum A_t = \sum A'_t = \sum kP_t = k \sum P_t.$$  

Thus,

$$k = \frac{\sum A_t}{\sum P_t}.$$
CHAPTER 1. INTRODUCTION

Using the fact that \( \sum A_t = \sum A_r \) and \( \sum P_t = \sum P_r \),

\[
k = \frac{\sum A_r}{\sum P_r}
\]

This is just the reciprocal of the global average population density \( \bar{\rho} \). We have finally

\[
A'_t = \frac{A_t P_r}{\bar{\rho} A_r}; \quad \bar{\rho} \equiv \frac{\sum P_r}{\sum A_r}
\]  
(1.1)

as the new area required for each triangle \( t \).

In most maps, there will be some triangles which are not inside any region with a preset population, such as lakes or oceans. Since these triangles do not have any population, using equation 1.1 would give new areas of zero for such triangles. Instead, we should assign some non-zero areas, perhaps ones that allow us to preserve better the shapes of the region triangles. A simpler idea, however, is just to let an ocean triangle's new area equal its old area.

\[
A'_t = A_t
\]  
(1.2)

This is equivalent to pretending that the ocean's population density is equal to \( \rho \).

The signed area of a triangle in terms of its vertex coordinates is

\[
A = \frac{1}{2}(x_1y_2 - x_2y_1 + x_2y_3 - x_3y_2 + x_3y_1 - x_1y_3)
\]  
(1.3)

It is important to note that the above equation gives a positive area if the vertices are visited in counterclockwise order, and a negative area for clockwise order. For consistency, it is worthwhile to stipulate that a triangle’s signed area is always calculated by visiting the vertices in counterclockwise order as seen on the initial map. If we move the vertices of the map and then discover that a triangle’s signed area is now negative, we know that the triangle has become “backwards” or “flipped” from its original configuration; the vertices are now in clockwise order instead of counterclockwise. Likewise, if the new vertex coordinates yield positive areas for all the triangles (as is the case before anamorphosis), then we know the triangles have kept their original orientation.
CHAPTER 1. INTRODUCTION

1.3 Number of Solutions

One way of seeing that the map anamorphosis problem is underconstrained is to view it as a general transformation of coordinates. We wish to convert points in $xy$ space to points in $uv$ space:

$$u = U(x, y) \quad v = V(x, y)$$

We can consider the original map to have some density function $\rho(x, y)$ which defines where the population is distributed. We then require that the new density function on the $uv$ plane be uniformly constant. This is equivalent to requiring that the Jacobian of the functions $U$ and $V$ equal $\rho(x, y)$ [9]:

$$\frac{\partial U}{\partial x} \frac{\partial V}{\partial y} - \frac{\partial U}{\partial y} \frac{\partial V}{\partial x} = \rho(x, y)$$

Since we have two functions to specify, but only one constraining equation, the functions $U$ and $V$ do not have unique solutions.

If the map is defined by a total of $N_v$ unique vertices, then with Euler's formula we can expect the triangulation to contain at most $N_t = 2N_v - 5$ triangles and $N_e = 3N_v - 6$ edges [14]. If we add four vertices which define a bounding rectangle of all the vertices (as was done for the map in Figure 1.3), the convex hull is then a rectangle, and we can expect exactly $2N_v - 6$ triangles and $3N_v - 7$ edges.

The problem of finding new triangles with “corrected” areas can be viewed as solving a nonlinear system of $2N_v$ unknowns (the $x$ and $y$ coordinates of the vertices) and $N_t$ equations (each taking the form of Equation 1.3), where we know $N_t < 2N_v$. This last fact also reflects that the system is underdetermined, which gives us hope that some solution, perhaps even an “aesthetically pleasing” solution, can be found using our basic approach.

1.4 Aesthetics

Aside from finding a correct solution, the other main issue is that of aesthetics. Since there are a multiplicity of solutions [9], we would like to find one that minimizes some kind of “badness” criterion on the shapes of the regions or triangles. Ideally, each triangle
Figure 1.2: Europe Before Triangulation
Figure 1.3: Europe After Triangulation
CHAPTER 1. INTRODUCTION

(or region) in isolation would simply scale larger or smaller until it acquired its correct area. This way, its shape would be perfectly preserved. Of course, the triangles are not in isolation; they each share vertices and edges with their neighbors, and this is what makes the anamorphosis problem challenging.

It may not always be possible, but ideally some aesthetic concern would be an integral part of the algorithm design. Even after the trial cartogram has been generated, however, it may still be useful to transform the entire map to maintain, as closely as is possible, the shapes of the original triangles and regions. For example, it is easy enough, given a completed cartogram, to scale all vertex \( x \) coordinates by some factor \( a \), and also scale all \( y \) coordinates by the factor \( \frac{1}{a} \). This will not change the final areas of the regions, but it will change their shapes, and in a controlled way. If we can design an error function \( \epsilon \) which somehow captures the total deviation of the triangles from their “correct” shapes, and express it as a function of \( a \), then by solving for the \( a \) that minimizes \( \epsilon(a) \) and scaling the map appropriately by \( a \), we can generate a (presumably) more appealing cartogram. Alternatively, we can use this fact to maintain the bounding rectangle for the entire map.

1.5 Outline

Chapter 2 describes the triangulation algorithm used to generate the map in Figure 1.3, and which was used as the basis of our attempts at anamorphosis. Chapter 3 describes a working algorithm by Petrov et. al. [13] which already exists in the geography literature, and which is also based on a triangulation of the initial map. Chapter 4 shows two ideas for solving the anamorphosis problem by assigning the calculation of individual triangle parts to particular triangles. Chapter 5 explores the idea of transforming the original nonlinear system of area equations to a linear one. Chapter 6 describes two attempts to solve directly the nonlinear system. Finally, Chapter 7 summarizes the insights that have been gained with these unsuccessful approaches, and indicates possible paths for finding a better algorithm.
Chapter 2

A SIMPLE METHOD OF TRIANGULATION

2.1 Introduction

Triangulation is the connecting of a set of vertices with line segments (or edges), resulting in a set of contiguous triangles. A constrained triangulation requires that a given set of initial edges (which in our case will be the polygon boundaries) be a part of the final triangulation. Reasonably simple unconstrained triangulation methods requiring $\Theta(N \log N)$ time exist, such as the Delaunay triangulation [14]. Unfortunately, fast constrained triangulation algorithms are generally difficult to implement. This chapter describes the triangulation algorithm used for this project, which, though slow, is easy to code. For small sets of vertices, this algorithm should prove adequately fast; on a DECstation 5000, 100 vertices can be triangulated in under a second, and 500 vertices in under a minute. We use this triangulation method for this and other projects (such as the GeoSim module “Mental Maps”), where speed is not so important. The anamorphosis algorithm itself does not depend on the efficiency of the triangulation step. The map needs to be triangulated only once before computing cartograms for different variables or distributions.

We are given a set of $N = N_n$ vertices on the Euclidean plane, and an initial set of edges which we require to be part of the triangulation. The algorithm then generates additional edges as necessary to complete the triangulation, and accumulates an array of the triangles thus formed. The entire process consumes temporary storage space of $\Theta(N)$, and has a time complexity of $\Theta(N^3)$. Having revealed this drawback, we note that the advantages of this approach include ease of implementation and later customization by the programmer.
CHAPTER 2. A SIMPLE METHOD OF TRIANGULATION

2.2 Method

The caller passes an array of $N$ vertices, an array of initial edges, and an empty triangle array. The caller must also provide adequate space in these arrays for the triangulation: space for at most $3N - 6$ edges, and at most $2N - 4$ triangles. The initial vertices need not be unique for the algorithm to work, but time will be wasted testing duplicate edges if there are duplicate vertices.

We maintain an initially empty array of edges which have been accepted into the triangulation. (Actually, these accepted edges reside in the caller’s passed edge array.) In general, new trial edges will be accepted if they do not intersect any currently accepted edge. This test is essentially a brute-force comparison against every edge in the array, which takes $O(N)$ time for each trial edge. Gilbert [6] gives a method for reducing this to $O(\log N)$, but it requires maintaining an auxiliary data structure of size $\Theta(N^2)$, and is non-trivial to implement. (See also [14].)

Starting with our empty array of accepted edges, we attempt to add each of the initial edges given by the caller. No pair of these edges may intersect, so our algorithm tests each initial edge against the others. If, at this stage, any pair of edges cross, we abort the triangulation and return an error; such an edge crossing means that the initial map is inconsistent.

We now attempt to add each of the $\binom{N}{2} = \frac{1}{2}N(N - 1)$ possible edges. Since the acceptance test described above takes $O(N)$, this is where the $O(N^3)$ behavior of the entire algorithm arises — an $O(N)$ test for each of these $\Theta(N^2)$ trial edges. For these edges, however, it is not an error to intersect a previously accepted edge; they are simply ignored. (In practice, the algorithm cost turned out to be more like $O(N^{2.5})$, perhaps because crossing edges tended to be discovered more quickly, and were generated more and more frequently.)

The order in which we generate the $\binom{N}{2}$ possible edges requires some thought. A simple double loop over the vertex indices tends to yield clusters of triangles radiating from a common vertex, as well as many long skinny triangles. This is equivalent to traversing by
CHAPTER 2. A SIMPLE METHOD OF TRIANGULATION

rows the lower triangular matrix defined by placing edge \((i, j)\) in row \(i\) and column \(j\), for all \(i > j\). A slightly better no-cost approach is to traverse the diagonals, instead of the rows, as is done in our implementation. Another variation we tried in both cases was to randomly permute the vertex indices before generating the edges. This gave marginally better results over an array of vertices sorted by \(x\) coordinate, but not good enough, in our judgement, to be worth retaining.

An altogether different approach, and one which generates a more aesthetic triangulation, is to add the \(\binom{N}{2}\) edges in ascending order by length. However, this requires allocating \(\Theta(N^2)\) storage to hold a temporary array, which might be an unacceptable cost. Either this method of generating the edges, or the one described in the previous paragraph, may be used; the choice is not crucial to the algorithm. In any case, we eventually will have an array of accepted edges, which will be all the edges for the triangulation. It now remains to deduce the triangles.

Given that an edge connects a vertex \(p\) to a vertex \(q\), where \(p\) and \(q\) are indices into the array of vertices, we make the convention that index \(p\) is less than index \(q\) for every edge. This is arranged when an edge is added to the array. We sort the edges by the array indices of their endpoints: primarily by \(p\), and secondarily by \(q\) to break ties in \(p\). Doing this allows us to quickly find an edge in the array by its endpoints, or to discover that the edge does not exist. Also, we can efficiently accumulate the neighbors of a given vertex, which will be needed shortly.

We move through the (now sorted) edge array, accumulating into a temporary buffer a set of vertices which all have the same neighbor \(p\). For example, suppose we encounter edges \((20-23), (20-28), (20-32), (20-35), (20-37), (21-6)\), and so on. We would accumulate the indices \(23, 28, 32, 35,\) and \(37\) into the buffer, all of which are neighbors of vertex \(20\). The edge \((21-26)\) marks the starting place for the next round of neighbor accumulation.

With our buffer full of “higher” neighbors of vertex \(p\), we sort the neighbors by the direction angles they make from vertex \(p\). We move through the buffer, adding triangles that consist of \(p\) and a pair of adjacent neighbors in the buffer, if an edge exists between
CHAPTER 2. A SIMPLE METHOD OF TRIANGULATION

the pair of neighbors. Referring to Figure 2.1, the vertices 37, 23, 32, 28, and 35 have been sorted in anti-clockwise order around vertex 20. We will then add triangles (20, 37, 23) and (20, 32, 28), because the edges (23–37) and (28–32) exist. We will not add triangle (20, 23, 32), for example, because there is no edge (23–32); vertex 14 lies in the way.

![Figure 2.1: Finding Triangles](image)

That we are only considering the higher neighbors is not important; in fact, we are eliminating unnecessary work. Vertex 20 will be a higher neighbor of previous vertices that we have already encountered, hence “lower” triangles will already have been considered. In general, supposing there is a triangle \((a, b, c)\) out there to be found, and supposing that \(a < b < c\), then the triangle will be added when we consider vertex \(a\) and its higher neighbors, among which will be \(b\) and \(c\); there is no need to add the triangle again when we get to vertices \(b\) and \(c\).

Having dealt with vertex \(p\) and its higher neighbors, we then go back to the list of edges to process the higher neighbors of the next vertex \(p\), and so on until the edge array is
CHAPTER 2. A SIMPLE METHOD OF TRANGULATION

exhausted. (Vertices which have fewer than two higher neighbors obviously do not generate any triangles.) We will then have completed the triangulation, which will contain at most $3N - 6$ edges and $2N - 4$ triangles in general, or $3N - 7$ edges and $2N - 6$ triangles if the map is confined within a bounding rectangle.

For the anamorphosis, we must also know the polygon in which each triangle resides. A triangle is inside a particular polygon if: (1) all three triangle vertices are also vertices of the polygon, and (2) traversing the polygon vertices in counterclockwise order would imply that we also visit the triangle vertices in counterclockwise order. Triangles which fail either of these two criteria for all polygons are “ocean” triangles, and are not part of any polygon. In any case, we can then use Equations 1.1 or 1.2 to assign new areas to each triangle. The anamorphosis problem then remains: to modify the triangulation, keeping the same topology but yielding triangles with their new “corrected” areas.
Chapter 3

A WORKING ALGORITHM

Here we will discuss an algorithm for map anamorphosis, already published in the scientific literature, from Petrov, Serbenyuk, and Tikunov [13]. This algorithm appears to work, at least for the examples shown in the article. Though it is not proven to converge to a correct solution, it may be instructive to examine this algorithm as we attempt to design other algorithms that work with triangulated maps.

Petrov et al. take the same initial approach that we do—namely, an initial partitioning of the map into triangles, assigning populations to triangles based on the original map regions. The algorithm then attempts to move vertices to minimize differences in population density among triangles that share a vertex, until all the densities are the same to within some error tolerance. Making all the population densities the same is equivalent to making the new triangle areas proportional to their populations.

The Petrov et al. algorithm works with a triangulation as we do. The authors recommend that the triangles be as close to equilateral in shape as possible. For one example map, they begin with a grid of squares and produce four right triangles from each square by partitioning along the square’s two diagonals. For another, they make a triangulation of near-equilateral triangles by adding new vertices inside the map regions. It is certainly reasonable to believe that long thin triangles could lead to numerical calculation problems at some later stage. The handling of “ocean” triangles is also the same as ours; ocean triangles are taken to have a population density of $p$.

Having created the triangles and assigned populations to them, we then visit each vertex to change its coordinates. For a particular vertex $j$, as pictured in Figure 3.1, there will be
CHAPTER 3. A WORKING ALGORITHM

$m_j$ triangles which share the vertex in common. We define a function

$$f_j(\rho_1, \ldots, \rho_m)$$

where

$$\rho_i = \frac{P_i}{A_i},$$

the current population density of triangle $i$ adjacent to vertex $j$. The function $f_j$ represents the error of the triangle population densities from the local average density $\bar{\rho}_j$; the goal will be to minimize this difference. Petrov et. al. offer three choices for $f_j$:

$$f_j = \sum_{i=1}^{m_j} (\rho_i - \rho_j)^2$$  \hspace{1cm} (3.1)

$$f_j = \sum_{i=1}^{m_j} |\rho_i - \rho_j|$$  \hspace{1cm} (3.2)

$$f_j = \max_{1 \leq i < k \leq m_j} |\rho_i - \rho_k|$$  \hspace{1cm} (3.3)

where

$$\bar{\rho}_j = \frac{\sum_{i=1}^{m_j} P_i}{\sum_{i=1}^{m_j} A_i}.$$

If we move the coordinates $(x_j, y_j)$ of vertex $j$, while keeping the neighboring vertices fixed, the triangle densities will change, and so will $f_j$. Let us define the function $g_j(x'_j, y'_j)$ to be the value of $f_j$ when $(x_j, y_j)$ is replaced by $(x'_j, y'_j)$, keeping all the other local vertex coordinates constant.

We wish to find $x'_j$ and $y'_j$ that minimize $g_j(x'_j, y'_j)$, but we cannot look arbitrarily far away for these coordinates. To maintain the “integrity” of the triangles around vertex $j$, we must not let vertex $j$ move outside the polygon defined by $j$’s neighboring vertices. Thus we have some maximum region of extent $G$ over which we can seek a minimum to $g$. The authors use a circle of radius $R_j$ centered at $(x_j, y_j)$ as the maximum extent, where

$$R_j = \frac{\min_{1 \leq i \leq N} d_i (\max_{1 \leq i \leq k \leq m_j} |\rho_{ij} - \rho_k|) \max_{1 \leq i \leq N} \max_{1 \leq i \leq k \leq m_i} |\rho_{ij} - \rho_k|}{\max_{1 \leq i \leq N} \max_{1 \leq i \leq k \leq m_i} |\rho_{ij} - \rho_k|}$$

where $d_i$ is the shortest distance of vertex $i$ from the line segments of the polygon surrounding vertex $l$. This complex expression prevents vertex $j$ from creating flipped triangles by
crossing its surrounding-polygon boundary, and it also scales the radius of the search circle $G$ as a function of how different the triangle densities are: very different population densities will allow a larger circle, more uniform densities will confine the search to a smaller circle.

After choosing the size and shape of $G$, we then proceed to find the local minimum $g_j(x_j', y_j')$ within that maximum extent. The algorithm used by the authors is based on a random walk, and goes as follows:

1. Let $(x_j', y_j')$ equal $(x_j, y_j)$, the starting coordinates of vertex $j$.

2. Generate uniform random numbers $\xi$ and $\eta$ in the range of 0 to 1. Calculate

$$\tilde{x} = x_j' + h \cos(2\pi \xi) \quad \tilde{y} = y_j' + h \sin(2\pi \eta)$$

If $(\tilde{x}, \tilde{y})$ is outside the search region, repeat this step.

3. If $g(x_j', y_j') > g(\tilde{x}, \tilde{y})$, then set $x_j' = \tilde{x}$, $y_j' = \tilde{y}$, and go to step 2.
CHAPTER 3. A WORKING ALGORITHM

4. Otherwise, if the number of calculations of \((\hat{x}, \hat{y})\) is still below some maximum threshold, go to step 2.

5. Otherwise, divide \(h\) by 2. If \(h > h_0\), go to step 2.

6. The new coordinates \((x'_j, y'_j)\) of vertex \(j\) have been calculated for this iteration.

The parameters \(h\) and \(h_0\) are defined only vaguely, but the initial value of \(h\) is based on the size of \(G\) (and therefore \(R_j\)), and \(h_0\) is an error tolerance for the vertex positions. The above procedure is executed for each vertex, and a complete iteration over the entire set of vertices is executed until the vertex coordinates no longer change. The end result will be a modified triangulation in which each triangle has a population density of \(\bar{\rho}\); in other words, the triangle areas will be proportional to their populations. For the examples shown in the article, the number of iterations that yielded convergence was always less than 20.

The authors of this algorithm offer the following significant advantages to their approach: simplicity, free movement of map boundaries, and independence from the exact choice of the Cartesian coordinate system — i.e., translations or rotations of the initial map will not generate different cartograms. This last property is probably the most important, since there is nothing special about either the location of the origin of the map’s coordinate system, nor about the directions or scales of the \(x\) and \(y\) axes. Ideally, a transformed initial map should just generate an identically transformed cartogram.

There are also some stated disadvantages. The final cartogram depends heavily on the initial choice of triangles, and to the choice of \(G\), the region over which local minimization is calculated. Petrov et. al. recommend a triangulation with large triangles in regions of low population density, and small triangles in regions of high density. This will tend to keep the triangle populations approximately equal, and thus yield roughly equal-size triangles in the final cartogram.

In addition, the authors fail to show that convergence on a solution is guaranteed, or even rapid in most cases. Using a random walk to find the local minimum seems unnecessary; a simple linear search along the negative gradient direction, \(-\nabla g_j(x'_j, y'_j)\), stopping at the...
CHAPTER 3. A WORKING ALGORITHM

circle boundary, will yield the constrained minimum much more easily. Also, the issue of aesthetics is not addressed explicitly, either in the design of the basic algorithm, or in any post-processing step on the final map. The shapes of the output triangles are in no way tied to the shapes of the input triangles, and thus the regions have a corresponding freedom to distort arbitrarily. Still, if the empirical data in the article is to be believed, this algorithm may be a good starting point for robustly solving the map anamorphosis problem, given that we are interested in using a triangulation for the basic approach. Also, the choice of a triangulation of near-equilateral triangles is probably a good idea to incorporate into our own approaches.
Chapter 4

OWNERSHIP OF TRIANGLE PARTS

4.1 Ownership of Edges

The goal of our map anamorphosis is to give new areas to the triangles in the triangulation. The area of a triangle can be expressed in terms of its edge lengths, or its vertex coordinates. In general, if some “part” of the triangle can be expressed as a function of the other parts and the triangle’s new area, then perhaps we can design an algorithm that lets each triangle set one (or more) of its parts, given that the other parts have already been determined. Each triangle part in the map will then be calculated by one triangle based on its area equation, though other triangles generally will be sharing that same part, so conflicts may arise. However, as long as we can guarantee each triangle at least one “degree of freedom”, we might be able to solve the entire system of edges or coordinates, and thus we will have achieved the new triangulation.

Our first approach to distorting the mesh of triangles is to solve for the edge lengths instead of the vertex coordinates. If we allow each edge to be exclusively “owned” by one of its two adjacent triangles, it might be possible for each triangle to calculate the lengths of its owned edges, given the other edge lengths and the triangle’s new area.

Clearly, if a triangle owns none of its edges, meaning that its three neighbors will set the edge lengths instead, it will be impossible in general for the triangle to meet its required area. We must let each triangle own at least one of its edges. This can be done with a depth-first traversal of the triangles, which effectively generates a tree. Each triangle will have exactly one parent triangle in the tree, except for the first, which will have no parent. We stipulate that each triangle owns at least the edge that comes between it and its parent.
CHAPTER 4. OWNERSHIP OF TRIANGLE PARTS

Every triangle will have a parent except the root triangle, the one with which we began the traversal. We can still ensure that it owns one of its edges by choosing it to be a triangle on the border of the map — that is, a triangle with at least one edge facing the exterior.

Ultimately we will have given each triangle ownership of at least one of its three edges, and the other edges will be owned by the triangle’s neighbors. There are then three cases for each triangle: the triangle owns one, two, or three of its edges. The last two cases are easy to handle. If a triangle owns all three of its edges, then it can easily choose lengths to satisfy its new area. If a triangle owns two of its edges, the third edge may take on any positive value; we can still always form a triangle with the correct area. Conceptually, we must put the third vertex somewhere on a line parallel to the base (the fixed edge) at a distance $h = 2A'/b$ from the base. (See Figure 4.1.) This is the only constraint, so we have great freedom in choosing the lengths of the two owned edges of such a triangle.

![Figure 4.1: Fixing Two Triangle Sides](image)

The remaining case, however, could cause difficulties. If a triangle owns just one of its edges, it may not be possible in general to satisfy the triangle’s new area requirement. Consider a triangle with two edges owned by its neighbors. If the required area of the triangle is 100, but the edge lengths set by the neighbors are 2 and 5, then there will be no way to set the third edge length to make a triangle of area 100. Thus, although we will eventually give ownership of each edge to one of its two triangles, we cannot let a triangle set its owned edges arbitrarily.

To handle the difficult case then, we will allow a “surrendering” triangle to specify minimum values for the lengths of the edges it is surrendering. The owning triangle has the freedom to set an edge length to anything above this minimum. Each triangle must now
CHAPTER 4. OWNERSHIP OF TRIANGLE PARTS

not only satisfy its area requirement, but also return sufficiently large values for the edges that it owns, and provide minimum values for the edges it does not own.

The details of setting minimum and final edge lengths will now be described. We can consider each of the three cases for a given triangle in the tree:

One Edge Owned. Given the new triangle area $A'$ and the minimum length $p_m$ for the one owned edge, we must provide minimum lengths $q_m$ and $r_m$ for the two unowned edges. Let us assume a simple isosceles triangle of area $A'$ and base $p_m$. Then we let $q_m$ and $r_m$ be the lengths of the two equal sides:

$$q_m = r_m = \sqrt{\left(\frac{p_m}{2}\right)^2 + h^2} = \sqrt{\frac{p_m^2}{4} + \frac{4A'^2}{p_m^2}}; \quad A' = \frac{1}{2}p_mh$$

If the returned values $q$ and $r$ are larger than the minimums $q_m$ and $r_m$, as they generally will be, then this just means $p > p_m$, which is allowed. Using the area of a triangle as a function of its side lengths,

$$A' = \sqrt{s(s-p)(s-q)(s-r)}; \quad s = \frac{1}{2}(p + q + r)$$

and solving for the larger of the two values for $p$, we get

$$p = \sqrt{q^2 + r^2 + 2\sqrt{q^2r^2 - 4A'^2}}$$

which is the final length for the owned edge. The value inside the innermost radical is guaranteed to be non-negative, since $A' \leq \frac{1}{2}q_mr_m \leq \frac{1}{2}qr$.

Two Edges Owned. Here, we again have a target area $A'$ for the triangle, but we have two minimum edge lengths to satisfy, namely $p_m$ and $q_m$. We must calculate $r_m$ as, and, after the triangle's successors have finished fixing edges, we also must set $p$ and $q$. Observing Figure 4.1, we note that it does not matter what value we supply for $r_m$; given any returned length for $r$, we can always make a triangle with the correct area. Moreover, by "sliding the third vertex" to an arbitrary distance along the dashed line, we can satisfy any minimums for the other two sides. Thus, there are essentially no constraints for this case, though aesthetic issues might guide more restrictive choices for the calculations.
CHAPTER 4. OWNERSHIP OF TRIANGLE PARTS

Three Edges Owned. In this case, we have control over all three sides of the triangle, subject to the minimums $p_m$, $q_m$, and $r_m$. By fixing $p = p_m$, we can reduce this case to the case of two owned edges. Again, to generate better looking maps, there may be more intelligent means of setting the three sides to try to preserve triangle shapes.

Now that we can handle all three cases, we can form the complete algorithm as follows:

1. Start with all triangles unmarked, and all edge length minimums equal to 0.

2. Choose a triangle with fewer than three neighbors as the start (root) triangle.

3. Mark the current triangle.

4. Examine this triangle's neighbors. The unmarked neighbors are this triangle's successors in the traversal, and the corresponding edges are owned by the successors. All remaining edges are owned by this triangle.

5. Calculate minimum lengths for unowned edges, based on this triangle's required area and the already-specified minimum lengths for the owned edges.

6. Using recursion, let each unmarked neighbor (child) in turn become the current triangle, starting execution for each from step 3.

7. After processing all the children of this triangle, set the lengths of all owned edges, based on this triangle's required area, the lengths of previously set edges, and the minimum edge lengths that have to be satisfied for the ancestor triangles.

8. After all triangles have been processed, traverse the triangles again, setting vertex coordinates to match the edge lengths.

Unfortunately, even though we have correctly calculated a new set of edge lengths to match the target areas for the triangles, there is nothing to prevent the scenario pictured in Figure 4.2. Here, the new triangle areas have been realized, but the vertices have been rearranged to give a clockwise orientation to triangle (1, 4, 3). This triangle has been
CHAPTER 4. OWNERSHIP OF TRIANGLE PARTS

flipped, and now overlaps its neighbors. On a more complicated triangulation, the final map could have many flipped triangles mixed with correct triangles, and the appearance of the map will be unacceptable; the integrity of the triangulation's topology will have been lost.

![Original and Final Triangles](image)

Figure 4.2: Overlapping Triangles

The problem ultimately lies in calculating with edge lengths instead of vertex coordinates. Correct edge lengths do not guarantee an acceptable final map, since triangles can end up with any orientation. Maintaining the orientation of the triangles is very important; it lets the new triangles fit together with the same topology as the original triangles. This suggests that if the map anamorphosis problem can be solved using triangulation, it is the vertices that must drive the design of the algorithm, not the edge lengths.

4.2 Ownership of Coordinates

If owning and setting edge lengths is not the correct approach, then perhaps triangles can own one or more of their vertex coordinates. Our second approach is to calculate and assign ownership of vertex coordinates for each triangle. From Equation 1.3, we note that
CHAPTER 4. OWNERSHIP OF TRIANGLE PARTS

for a particular triangle, the area constraint can be met as long as we have at least one of the six vertex coordinates free to specify. As an example, we can solve for coordinate $x_1$:

$$x_1 = \frac{1}{y_2 - y_3}(2A' + x_2y_1 - x_2y_3 + x_3y_2 - x_3y_1)$$

Since there are $2N_v$ coordinates and $2N_v - 6$ triangles, it might be possible to arrange each triangle to “own” at least one of its coordinates, letting the other coordinates be set by neighboring triangles beforehand.

However, even if some means could be devised to guarantee, for each triangle, ownership of some of its vertex coordinates (which is by no means obvious), we will still have problems with precedence. If we have triangle $a$ owning some of triangle $b$’s coordinates, and triangle $b$ owning some of triangle $a$’s coordinates, then we must have $a$ set all of its coordinates before $b$ does, and likewise $b$ must set all of its coordinates before $a$ does, which is clearly impossible. For example, in Figure 4.3, we have vertex 1 at the center with coordinates $(x_1, y_1)$. If triangle $a$ owns $x_1$ and triangle $b$ owns $y_1$, then $y_1$ is needed in the calculation of $x_1$, and $x_1$ is needed in the calculation of $y_1$, which cannot be satisfied.

If, contrary to Figure 4.2, $a$ and $b$ were the same triangle, then this would cause no conflict, since the triangle could set both coordinates in infinitely many ways to satisfy its area requirement. This cannot be the case for all vertices, however. That would imply each triangle owns at least one entire vertex (one coordinate pair), not just a single coordinate, and there are simply not enough vertices to distribute among the triangles; there are only $N_v$ vertices for approximately $2N_v$ triangles. Thus, we must split at least some coordinate pairs between two separate triangles, which will lead to the contradiction already described. It is therefore impossible to solve the map anamorphosis problem using this idea.
Figure 4.3: Ownership of Vertex Coordinates
Chapter 5

TRANSFORMATION TO A LINEAR SYSTEM

5.1 Approach

As was noted in Chapter 1, we can view the anamorphosis problem as that of solving a system of equations, with all the vertex coordinates as our variables, and each equation specifying a constraint on a triangle’s signed area. For example, the simple triangulation in Figure 5.1 yields the following system to solve:

\[ x_3 y_4 - x_4 y_3 + x_4 y_5 - x_5 y_4 + x_5 y_3 - x_3 y_5 = 2A'_1 \]
\[ x_2 y_4 - x_4 y_2 + x_4 y_3 - x_3 y_4 + x_3 y_2 - x_2 y_3 = 2A'_2 \]
\[ x_1 y_2 - x_2 y_1 + x_2 y_3 - x_3 y_2 + x_3 y_1 - x_1 y_3 = 2A'_3 \]

Figure 5.1: Simple Triangulation

There is a lot of redundancy in the above equations. For example, the term \( x_3 y_4 - x_4 y_3 \) appears in the \( A'_1 \) equation, and again, negated, in the \( A'_2 \) equation. The appearance of
CHAPTER 5. TRANSFORMATION TO A LINEAR SYSTEM

this term in the two triangle equations reflects the fact that the two triangles share the edge connecting vertices 3 and 4. Since we are always traversing a triangle’s vertices in counterclockwise order when we calculate its area, we “traverse” the edge in one direction for one of the triangles, and in the other direction for the other triangle; hence the term will appear in one triangle’s equation, and its negation will appear in the other.

We can define a new variable \( u \) for an edge connecting vertices \( i \) and \( j \):

\[
u_{ij} \equiv \frac{1}{2}(x_i y_j - x_j y_i); \quad i < j.
\] (5.1)

A geometric interpretation of \( u_{ij} \) is the signed area of the triangle that the edge \( ij \) makes with the origin. By defining this set of \( u \) variables, we can convert the original non-linear system of equations into a linear one:

\[
\begin{align*}
    u_{34} + u_{45} - u_{35} &= A_1' \\
    u_{24} - u_{34} - u_{23} &= A_2' \\
    u_{12} + u_{23} - u_{14} &= A_3'
\end{align*}
\]

The coefficient of \( u_{ij} \) in a particular triangle’s equation will be +1 if vertex \( i \) precedes vertex \( j \) in counterclockwise order around the triangle, and −1 if \( j \) precedes \( i \). Note that the linear system is underdetermined, since the number of \( u \) variables (edges) is greater than the number of equations (triangles).

After solving this linear system for all the \( u \) variables, we must then recover the \( x \) and \( y \) coordinate values. For a particular triangle with vertices \( i, j, \) and \( k \), if we have the values for \( u_{ij}, u_{jk}, u_{ki}, \) along with the coordinates of vertices \( i \) and \( j \), we can solve for the coordinates of vertex \( k \):

\[
\begin{align*}
x_k &= \frac{x_i u_{jk} + x_j u_{ki}}{-u_{ij}}, \\
y_k &= \frac{y_i u_{jk} + y_j u_{ki}}{-u_{ij}}.
\end{align*}
\] (5.2)

If we arbitrarily set two vertices of the first triangle, we have determined the third by the above equations. By fixing all three vertices, we have also fixed two vertices of a neighboring triangle, and likewise determined its third vertex by these equations. Continuing in this manner, we can eventually fix the coordinates of all the vertices. A depth-first traversal of
CHAPTER 5. TRANSFORMATION TO A LINEAR SYSTEM

the triangles will guarantee that when we visit a triangle, at least two of its vertices will have already been fixed. If all three have been fixed, there is nothing to do; if two are fixed, we calculate the third using the above equations, the coordinates of the two fixed vertices, and the relevant $u$ values for the triangle.

Unfortunately, although the system of $u$ variables is linear and underdetermined, the system which transforms back to coordinates is overdetermined: the number of $u$ variables (one for each edge) is greater than the number of coordinates (two for each vertex). Recalling the exact expressions, $N_e = 3N_v - 7 > 2N_v$. We actually have more variables than equations, and hence in general there will be no solution to the system. Attempting the post-processing step of setting the vertices will result in an inconsistent placement of vertices, with triangles disagreeing where shared vertices should be, causing many triangles to be flipped.

Recalling the geometric meaning of $u_{ij}$, a small or zero value for $u_{ij}$ indicates an edge aligned with the origin, which might be common in general. The appearance of a $u$ variable in the denominator of equations 5.2 imply that low values of $u$ variables will lead to bad numerical results. Since there is nothing to stop a given $u$ from taking on any real value, it is difficult to imagine controlling the solution well enough to avoid this difficulty.
Chapter 6

NUMERICAL SOLUTION OF THE NONLINEAR SYSTEM

6.1 Homotopy Method

To generate the new triangulation, we can attempt to directly solve the nonlinear system for the $2N_v$ coordinates. Again, each triangle $t$ generates an equation of the form

$$A_t = \frac{1}{2}(x_i y_j - x_j y_i + x_j y_k - x_k y_j + x_k y_i - x_i y_k).$$  

(6.1)

The equations are in fact second degree polynomials in the vertex coordinates. We can attack the problem using various numerical methods. We first attempt to solve the problem using a numerical technique called a homotopy method, specially designed to solve nonlinear systems of equations.

In general, a homotopy is formed by adding a new variable $\lambda$ to the original system. If we begin with a system of equations $A(z) = 0$, where $z$ is a vector of the variables, we then embed the function $A(z)$ in a family of functions $h(z, \lambda)$, known as a homotopy map. The new variable $\lambda$ will range from 0 to 1, and will take us from a known solution of the simple problem $h(z, 0) = 0$ to a solution of the given problem $h(z, 1) = 0$.

In our case, we can create the homotopy from the system of equations

$$A(z) = (1 - \lambda)A_0 + \lambda A_1,$$

where $z$ is a vector of our $2N_v$ coordinates, each function of $A(z)$ is of the form in Equation 6.1, and $A_0$ and $A_1$ are the initial and final values, respectively, of the triangle areas. This gives us a homotopy:

$$h(z, \lambda) = A(z) - (1 - \lambda)A_0 - \lambda A_1.$$
CHAPTER 6. NUMERICAL SOLUTION OF THE NONLINEAR SYSTEM

Hence as \( \lambda \) goes from 0 to 1, we are interpolating the triangle areas from their old values to their new values, and solving the interpolated system at each step. We are in fact following a smooth curve in the \((z, \lambda)\) variable space whose dimension is \(2N_n + 1\).

There are several possible outcomes of the homotopy approach. The curve may proceed, after however indirect a route, from \( \lambda = 0 \) to \( \lambda = 1 \), at which point a solution has been found. However, it is also possible for the curve to soar away to infinity, the vector of variables getting larger and larger, and with \( \lambda \) never reaching 1. Also, there are other curves in the multidimensional space of the variables, connecting other initial solutions to other final solutions. The curve being tracked may intersect another such curve, at least within the computer's numerical precision, and the path tracking part of the algorithm will start following the wrong curve. (This can be detected in some cases, but it still does not help in finding a proper solution to the original problem.) Even if none of these problems exists, the presence of singular solutions can cause troubles, since singular solutions are difficult to compute accurately. (The singularity of a solution can usually be detected afterward by examining the condition number of the Jacobian matrix at the solution. Singular solutions can sometimes be improved by Newton iteration.) Finally, if the solution of \( h(z, \lambda) = 0 \) at \( \lambda = 0 \) is not unique, it is possible for the curve to bend back; \( \lambda \) will reach some maximum less than 1 before heading back to 0 again.

For all the sample maps given to the homotopy software [19], \( \lambda \) never reached 1, but instead always returned to a final value of 0. The lengths of the tracked curves, however, were positive, which implies that the curves fell into the last category described above. Thus there are multiple solutions at \( \lambda = 0 \); i.e., there are multiple sets of vertex coordinates which satisfy the equations for the initial areas. If it were possible to constrain the variables, or perhaps to add some key equations, such that we could guarantee the uniqueness of the initial triangulation, then the curve tracker might reach \( \lambda = 1 \) and hence find a legitimate solution. However, given any initial triangulation, there is always a continuum of other triangle meshes having the same areas but different vertex coordinates; simply take the original map and translate it, or rotate it, or shear it, or scale it by an arbitrary factor \( a \).
CHAPTER 6. NUMERICAL SOLUTION OF THE NONLINEAR SYSTEM

along the $x$ axis and by $\frac{1}{a}$ along the $y$ axis. Any of these transformations yields triangles with identical areas, but different vertex coordinates. This is the obstacle to using homotopy continuation as an approach to this problem; it is unclear how to eliminate the plethora of solutions at $\lambda = 0$. If there is no means of doing so, then both the initial and final solution sets are manifolds, not isolated points, and the problem is ill-posed for the homotopy method.

An attempt was made to “leapfrog” from $\lambda = 0$ to 1 by successive runs of the homotopy algorithm. If the curve tracker reaches a maximum $\lambda$ value of 0.2, say, then at that point we have discovered a triangulation which is one-fifth of the way to our complete solution. By re-initializing the vertex coordinates with the values achieved at the largest $\lambda$ on the previous run, it is conceivable that we could eventually reach $\lambda = 1$, and thus sidestep the problem. Unfortunately, in all the tests that were tried, the highest reached value of $\lambda$ quickly converged to zero, which rapidly eroded the progress made toward the solution at $\lambda = 1$.

6.2 Powell Hybrid Root Finder

The second numerical approach that we tried was the Powell Hybrid Method, which is designed to directly solve a system of nonlinear equations. The system we have is the set of $N_l$ equations $A(z) = A_1$, where $A(z)$ and $A_1$ are defined as in the previous section. The algorithm then attempts to minimize the magnitude (the norm) of the vector $F$ of function values, and thus reach a solution of the system $F(z) = A(z) - A_1 = 0$. It is called a “hybrid” algorithm because it combines two methods: a global method to find the vicinity of a solution, and a form of Newton’s method to converge rapidly on that solution.

Minimizing $\| F \|_2$, the Euclidean 2-norm of the vector $F$, is equivalent to minimizing the following function [3]:

$$f(z) = \frac{1}{2} F^T(z) F(z).$$

Roots of the original system $F(z) = 0$ will be global minima for the function $f(z)$. At any
CHAPTER 6. NUMERICAL SOLUTION OF THE NONLINEAR SYSTEM

particular moment, we have a current guess $z_c$ for the minimizing vector of $f(z)$. Since we are potentially calculating with arbitrarily complicated nonlinear equations, the surface described by $f(z)$ in the variable space could be exceedingly complex. The algorithm uses the Jacobian matrix of $F$ to derive a quadratic polynomial approximation to $f$ at the point $z_c$. This polynomial function $m_c(z)$ is an approximation to the function $f(z)$ in the neighborhood of $z_c$, and can be used to calculate approximate gradients to take us toward a minimum. The function $m_c(z)$ is said to “model” the function $f(z)$. (See Figure 6.1.)

![Diagram showing $z_c$, $z_+$, and $f(z)$, with $m_c(z)$ approximating $f(z)$ within the trust region.]

Figure 6.1: Minimizing $f(z)$

Also associated with the point $z_c$ is a “trust” region—a sphere centered on $z_c$ within which $m_c(z)$ can be “trusted” to model $f(z)$. The initial radius of this region can be chosen using various strategies, either automatically or by allowing the user to specify a value; in practice, the algorithm can be made robust enough to recover from bad choices of an initial radius, though usually with a cost of extra iterations.

The next point in our iteration,

$$z_+ = z_c - J(z_c)^{-1}F(z_c),$$

where $J(z_c)$ is the Jacobian matrix of $F$ evaluated at $z_c$, must reduce the value of $f$ and lie inside the trust region to be an acceptable next step. The trust region will grow or shrink
CHAPTER 6. NUMERICAL SOLUTION OF THE NONLINEAR SYSTEM

as we iterate, depending on the local behavior of $f(z)$, and whether our modeling function $m_z(z)$ is good or poor. If the tentative new point is unacceptable, we must shrink the trust region and use a new modeling function in the neighborhood of $z_c$.

Eventually this process will converge to a point $z_*$, which locally minimizes the function $f$, and therefore minimizes $\| F \|_2$. If $z_*$ is a root of the system $F(z) = 0$, then we will have converged to a global minimum of $f(z)$. The converse, unfortunately, is not true; local minima of $f$ will not always be roots of $F$. This is the primary drawback of this approach. The algorithm can get trapped in a local minimum of $f$ and thus fail to solve the real problem of $F(z) = 0$. To quote [3]: "There is not much one can do in such a case, except report what has happened and advise the user to try to restart nearer to a root of $F(z)$ if possible." It is difficult to see how we could supply a closer root in our situation; generating an "approximate" cartogram is potentially as difficult as generating a correct one.

An almost successful result is shown, in Figure 6.3, for the simple map in Figure 6.2. The numerical software calculated new vertices which resulted in approximately correct areas (within 20 percent) for most, though not all, of the triangles. The populations used for this run were 20 for Nevada, and 50 for Utah; and indeed, the final areas of the two regions are roughly in that ratio.

One might be encouraged by the result shown for this map. However, when the algorithm was applied to the map of Europe in Figure 1.3, the resulting map had negative triangle areas, which indicates flipped and overlapping triangles. Given that we are finding a local minimum of $\| F \|_2$, and not a true root, it is not surprising that we could end up with negative triangle areas. After all, the algorithm does not know, nor can it be made to recognize, that we require each area to be positive. An approach using constrained optimization would allow us to stipulate this requirement, but we still have the same basic problems of: 1) arriving at local minima instead of global minima, and 2) finding a solution from a continuous manifold of solutions, rather than from a set of isolated points. Thus, the optimization approach is unreliable at best, sometimes yielding close results if one is lucky, but unacceptable results in general.

35
Figure 6.2: Nevada and Utah
Figure 6.3: Nevada and Utah After Powell
Chapter 7

CONCLUSION

This paper has examined five new approaches to the problem of generating cartograms from polygonal maps. Each approach, though initially promising, proved ultimately to be unsuccessful. It may be useful to recall these approaches and the ideas on which they depend when designing a correct algorithm, so that obstacles may be avoided. For example, it was shown that solving for the correct edge lengths can lead to fundamentally unworkable “solutions”, and that therefore we should seek a method which changes the vertex coordinates to satisfy the triangle area requirements.

All the approaches discussed have several elements in common. The input map is assumed to be specified by a set of polygonal regions, a common format for data in geographical information systems. The map is to be modified, yielding new polygons whose sizes are proportional to their respective populations, while keeping the polygons simple and non-overlapping. To simplify the meeting of this requirement, the map is further assumed to be partitioned into triangles, which are much easier to control and calculate with than polygons with arbitrary numbers of vertices. The final goal then, for all the approaches, is to modify the initial triangulation to produce a new triangulation of identical topology, but with different triangle areas.

This idea cannot be too far afield, since an apparently viable algorithm, already published in the literature, works on a triangulation as well. Though it comes with no guarantees for convergence or aesthetics, this algorithm from Petrov et. al. [13] may provide a good basis for a more rigorous solution to the map anamorphosis problem. Of all the five failed approaches we explored, the numerical solutions of the system of triangle area equations may offer the most promise for a working algorithm. If a modified problem with a unique
CHAPTER 7. CONCLUSION

initial solution could be found, then the homotopy method might be successful. Or, if a
guaranteed means could be discovered for finding a good approximate root of the system
of equations, then the Powell Hybrid method might be guaranteed to converge on a correct
solution.
REFERENCES


REFERENCES


