Computational Geometry in Public Policy

Jacob Spainhour; Lukas Bystricky and Dr. John Bu

ABSTRACT

Every ten years, the current members of state legislatures undergo a process of redistricting, in which congressional districts are drawn to accurately reflect changing demographics. However, this process has become entangled with partisan politics, leading to divisions that intentionally leave entire populations unrepresented. Where these politicians are prone to manipulating the system in their favor, a computer will be objective in providing an accurate representation. This project involves the creation of software to perform this task of dividing a given state into areas with regular shapes and similar populations. The program is written in Python code, and uses an algorithm based on the mathematical concept of a centroidal Voronoi tessellation (CVT).

BACKGROUND

This project uses Voronoi diagrams, a geometric method of dividing a region into distinct sub-regions. Each sub-region is drawn around a particular "generator" point, with the condition that every point inside a region is closer to its own generator than every other. A centroidal Voronoi tessellation adds the condition that each region's generator is also the centroid for the region itself. As a consequence, each sub-region has a roughly similar shape and size, as well as an evenly spaced distribution within the initial region. Both ordinary Voronoi diagrams and CVTs can be computed under a density function that affect the sizes of resulting sub-regions. These concepts can be easily combined into an objective method of drawing congressional districts.

OBJECTIVES

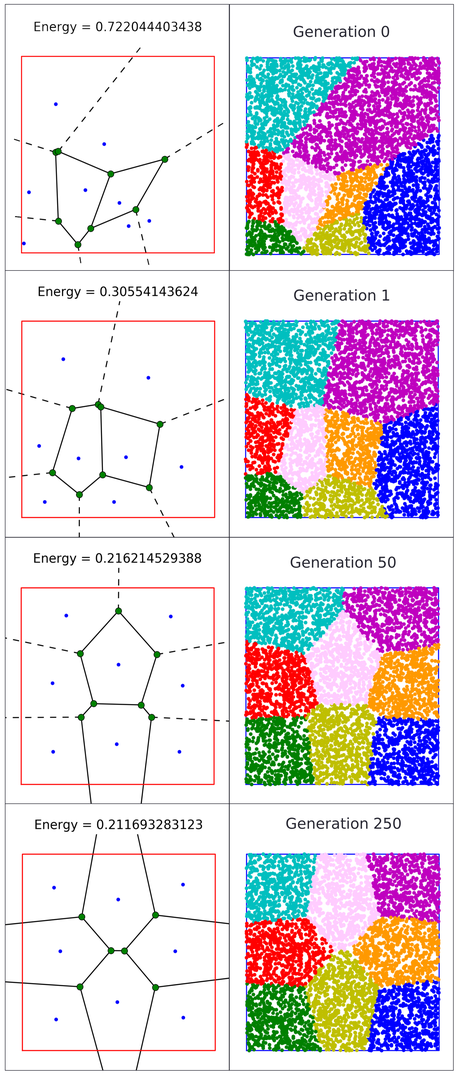
I am writing and testing Python code to perform the following tasks:

- Divide a two-dimensional region into areas of roughly equal size and shape.
- Compute energy for a Voronoi diagram to measure its efficiency.
- Read in Florida census data as a basis for density function in computing a CVT.

The code should be malleable, being capable of adapting to other uses of geometric CVTs. This includes drawing district lines for other states, given census data, as well creating exceptions for federal voting laws and existing geographies. The core of the CVT algorithm can also be used for non-geometric purposes, such as image manipulation software, discussed below.

CVT ALGORITHM

Left Column: Generators and their sub-regions
Right Column: Sample points colored by cluster



The adjacent diagrams represent the creation of a generic CVT diagram for a given shape. Every CVT algorithm cycles through the four steps:

- **Identify Generator Points**

Initial generator points are created randomly; each subsequent iteration uses the previous iteration's centroids.

- **Create Sample Points**

Sample points are generated randomly within the shape according to a given density function. The adjacent example uses 5000 sample points and a uniform density.

- **Assign Sample Points**

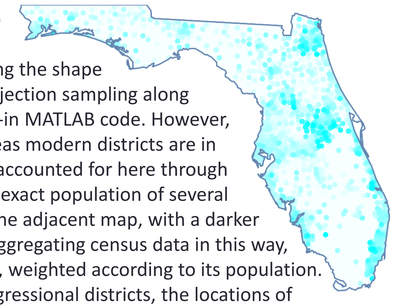
For each sample point, the distance between itself and each generator is calculated, and becomes "assigned" to that which it is closest, forming a cluster. Here, assignment is demonstrated through color.

- **Calculate Centroids**

The sample points of each cluster are averaged, and the resulting point becomes the generator for the following iteration.

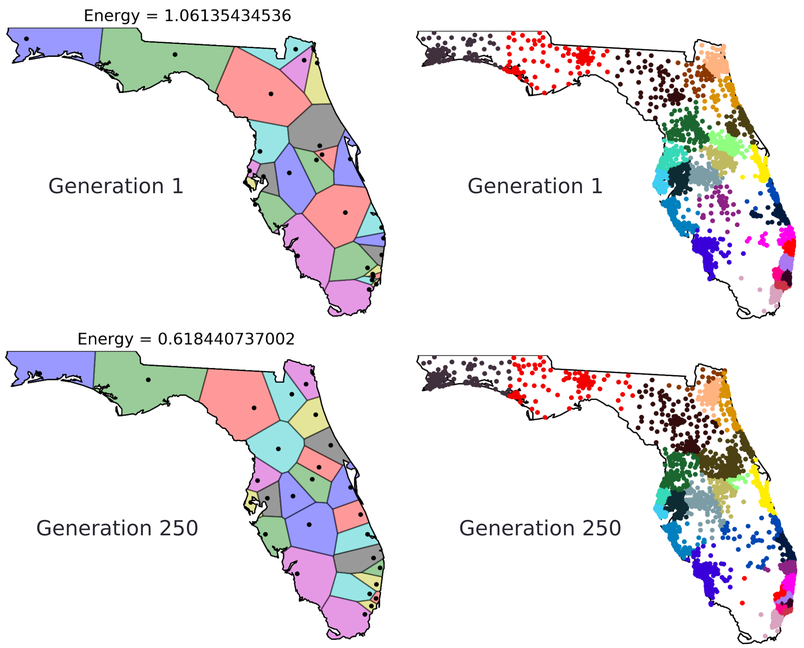
The efficiency of the CVT is quantified through "energy," calculated as the sum of the squares of each sample point to its assigned generator. Initial iterations yield a large decrease in energy, while later generations show little numerical difference. Generally, physical shape is independent of change in energy, as small changes in energy can occur with large changes in symmetry and regularity. Repeated iterations will always result in a rest point, at which both shape and energy remain constant. Occasionally this rest is less than ideal, called a local minimum. Even at an ideal position, a CVT is not guaranteed to be unique for a particular region.

ADAPTING FOR DISTRICTING

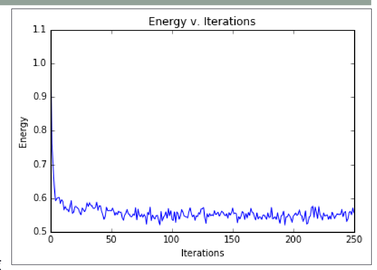


The basic CVT code can be easily applied to Florida districting with only minor changes, mostly of sampling methods. Sampling along the shape of Florida is more complicated, involving rejection sampling along a Florida-shaped polygon provided by built-in MATLAB code. However, this results in a uniform distribution, whereas modern districts are in reality largely based on population. This is accounted for here through the use of census data, which provides the exact population of several thousand physical coordinates, shown on the adjacent map, with a darker shade indicating a greater population. By aggregating census data in this way, each census point becomes a sample point, weighted according to its population. To maintain some similarity to existing congressional districts, the locations of twenty-seven most populous Florida cities are used as initial generator points. Otherwise, the CVT algorithm works identically to that of the uniform square.

RESULTS – FLORIDA CENSUS CVT



DISCUSSION AND APPLICATIONS

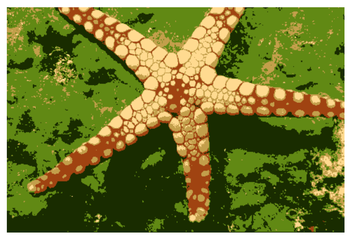


Ultimately, the code is successful, being able to divide the state of Florida into twenty-seven regions of regular size, population, and shape, without bias.

However, the code itself is not ideal. The adjacent graph shows that past as few as ten iterations, further iterations have an inconclusive effect on energy due to the randomness of the sampling process. Any CVT can be more precise, either by adding sample points or extra iterations, both of which are limited only by time and computing power. Additionally, the CVT produced below is only a single possibility, where others may be more accurate. Furthermore, the code in its current state does take into account many practical aspects of redistricting, such as minority voting laws and geographical boundaries, in particular those for coastal regions. Any of these issues can be resolved with further exceptions to the core of the CVT algorithm.

Most significant in the research is the application of the CVT algorithm to other areas of study. Image manipulation and compression in particular can be easily handled with only minor modifications to the CVT software. In brief, the program performs a CVT algorithm on the RGB color value of each pixel in an image, evenly distributing generator colors along the image. This way, the same information can be conveyed with fewer colors, one consequence being decreased storage on a computer. In the below example, the image is compressed to only five colors, decreasing the file storage size by a factor of ten, while still appearing visually recognizable.

Original Image: 281 KB



CVT Compressed Image with 5 Generators: 28 KB

REFERENCES

- Burns, Jared. "Centroidal Voronoi Tessellations." (2009).
- Supplemental code by: John Burkardt, Lukas Bystricky, Pauli Virtanen