Compact Clustering: A Solution to Gerrymandering?

Kyle Milligan
kmilligan@gatech.edu

Theodore J. LaGrow
tlagrow@gatech.edu

I. Introduction

The United States is a representative democracy; that is, citizens within a state, county, or other political district vote for individual politicians to represent them, rather than voting directly on legislation themselves. Every ten years, after the Census has been completed, state governments around the country are responsible for the job of redistricting: redrawing the boundaries of congressional districts to reflect changes in population and demographics. All states must comply with federal constitutional requirements related to population and anti-discrimination. For congressional redistricting, the Apportionment Clause of Article I, Section 2, of the U.S. Constitution requires that all districts be as nearly equal in population as practicable, while for state legislative districts, the Equal Protection Clause of the 14th Amendment to the U.S. Constitution requires that districts be substantially equal. Despite this constitutional basis for fair redistricting, savvy partisans in states around the country have been able to bias the process in favor of one party. This problem - the manipulation of district boundaries to favor one party or demographic group - is known as *gerrymandering*.

Recently, there has been a major push in states and at the federal level to abolish gerrymandering, either by mandating that representatives draw districts according to some definition of fairness, or by making the drawing of biased districts illegal. However, the difficulty is that concrete definitions of fair and biased districts do not exist. Generally speaking, experts and policymakers agree that fair districts must meet certain criteria:

1) *Compactness:* Districts must be as compact as possible. Intuitively, compactness refers to how close a districts boundaries are to its geographic center. Most states have their own interpretation of compactness, with some requiring districts to have smooth boundaries while others focus on the dispersion of the population within a district [1]. A common strategy for maximizing compactness is by making sure the district is convex.
2) *Demographics:* The populations of each district should be as close as possible. District boundaries should avoid splitting up defined communities as much as possible.
3) *Efficiency:* Districts should minimize wasted votes - the total number of votes for losing candidates plus the number of votes over the victory threshold for winners. Election winners in a state should also match the political affiliations of the voters in that state as closely as possible; if 70% of voters vote for a Democrat, about 70% of the winners should be Democrats.

In this report we frame gerrymandering as an optimization problem, and describe our work to solve it using variations on kmeans clustering. We also introduce a novel modification of the kmeans algorithm, which we call *balanced kmeans with gravity* (BKMG), which constrains the cardinality of the kmeans clusters and moves the cluster centers in a way that mimics the pull of gravity. We begin in Section II by providing a brief overview of the background literature in redistricting and optimization, and a summary of the main methods and results of the project. In Section III, we provide a thorough description of our algorithm and describe our methods in detail. Section IV describes our results. Section V features a discussion of our results and relates them to the real-world problem of fair redistricting.

II. Background

A. Literature Overview

Much work has been done recently to convexify the classical kmeans procedure, a non-convex, NP-hard problem. The main papers we drew inspiration from include the work of Li et al. [2], who discuss various convex relaxations of the kmeans problem including for the special case of equal cluster sizes; Lindsten et al. [3], who introduce the Sum-of-Norms (SON) algorithm for convex clustering; and Binder et al. [4], whose clustering algorithm includes a term based on the physical principle that there exists a gravitational force between objects with mass. Work focusing on gerrymandering itself has also been useful, particularly that of Guest et al. [5], who used a weighted l_2 distance metric to encourage balanced district sizes; and Whittle et al. [6], who take a similar approach but also take into account the number of iterations remaining, to encourage exploration early on, and rapid convergence to a stable setting as the number of iterations reaches the maximum number.

Li et al. [2] explore possible answers to a number of open questions in convex relaxation of kmeans, including how the number of clusters and dimensionality of the data affect the performance of a semidefinite programming relaxation for kmeans, the Peng-Wei relaxation [7]. They also test the special case of equal cluster sizes, and whether a tighter relaxation can improve on the Peng-Wei relaxation in this setting. The primary result of the balanced cluster case is that there is a

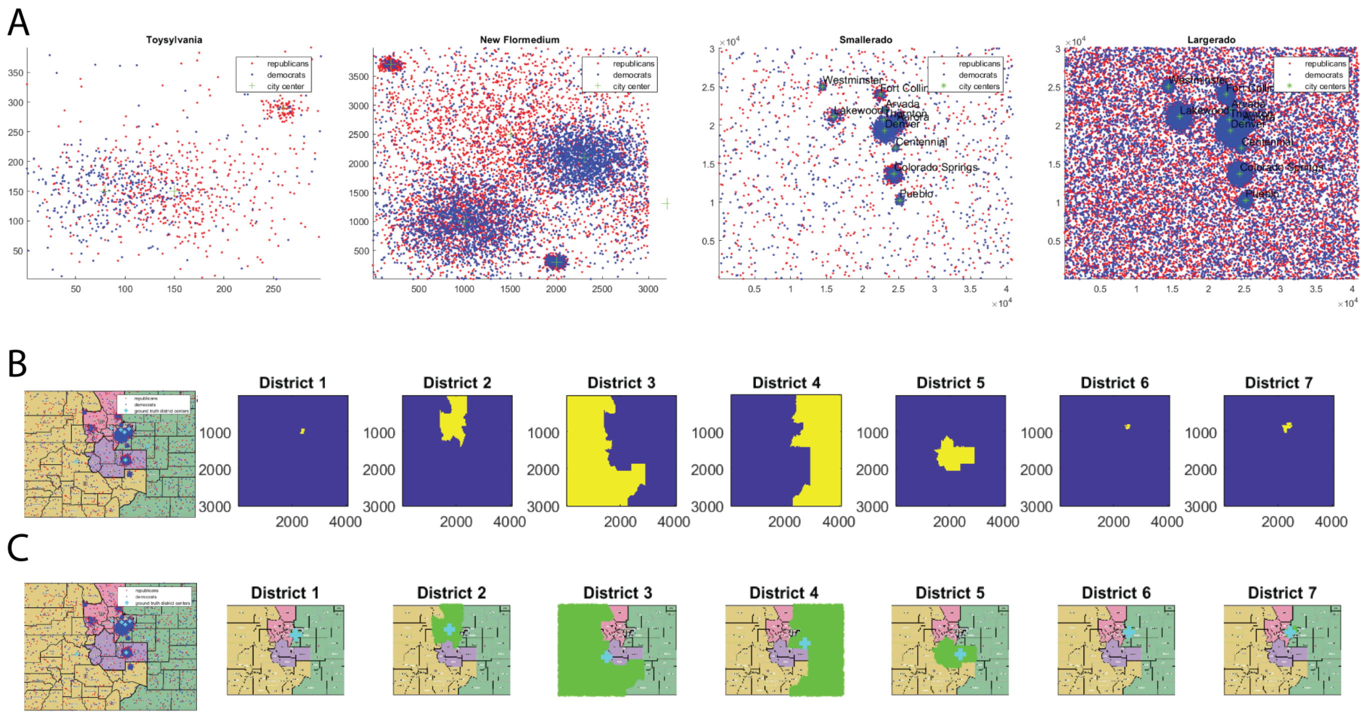

Fig. 1: *Demonstration of synthetic data* (A) Synthetic examples of state populations where cities have a majority of the state population. From left to right (in increasing order of clustering difficulty): Toysylvania (4 cities), New Flormedium (5 cities), Smallerado (10 cities), Largerado (10 cities). (B) Demonstration of ground truth labeling for Colorado, with the binary map of each district displayed. (C) The popularion from Largerado mapped onto Colorado for each respective district (marked with a green dot) and the centroids of each district (marked with a cyan cross).

unique minimizer to the relaxed form of the balanced kmeans clustering problem. We were unable to implement an algorithm to solve the convex relaxation of kmeans, but this paper drove our thinking when framing the problem and designing our analyses. Lindsten et al. [3] introduce a convex relaxation of kmeans clustering based on a l_0-regularized least-squares formulation of the clustering problem. We took inspiration from their objective function to define our own, balanced statement of the kmeans problem. Finally, Binder et al. [4] introduce the idea of implementing a gravitational term; we took this in a different direction from their original proposal, by encouraging cluster centers to not only move to the mean of the assigned points, but also to move towards the region of highest data density, as will be explained below.

In the realm of political science, fair and mathematical approaches to redistricting have recently been the focus of heated debate and much public attention. In this domain, we found elements of work by Guest, Kanayet, and Love [5] to be of importance when laying out the problem and implementing our algorithm. They made use of a modified l_2 distance metric to weight the distance from the data to each cluster center by the current cardinality of the cluster. In this way, clusters with greater population are considered by the algorithm to be ”further away” from the data than smaller clusters. Whittle et al. [6] adopt a similar approach, and weight their clusters according to population size and number of iterations remaining in the algorithm. This weighting encourages not only balanced clusters, but allows the algorithm to explore the space of potential clusters while guaranteeing convergence to a stable setting. Bayesian approaches, including Markov Chain Monte Carlo (MCMC) methods, have also found applications in this context, with some success [8], though this work is outside the scope of our current report.

B. *Project Summary*

In this project, we build on the works described above to implement a number of modified kmeans clustering algorithms in the context of fair redistricting to combat gerrymandering. Our algorithms utilize concepts from convexified kmeans, and are designed to encourage convexity, compactness, and balanced cluster cardinality. Our balanced kmeans with gravity (BKMG) includes a balancing step, which swaps data between clusters to encourage balanced cluster size, and a gravity step, which pulls the cluster centers of the smallest (K - 1) clusters towards the cluster center of the largest cluster to encourage compact clusters and faster convergence. We constrain our objective function such that the cardinality of each cluster is within some tolerance of the target, defined as N/K, where N is the total population and K is the number of clusters/districts.

III. Methods

In this section we lay out our problem formulation and describe our algorithms and data generation in detail.

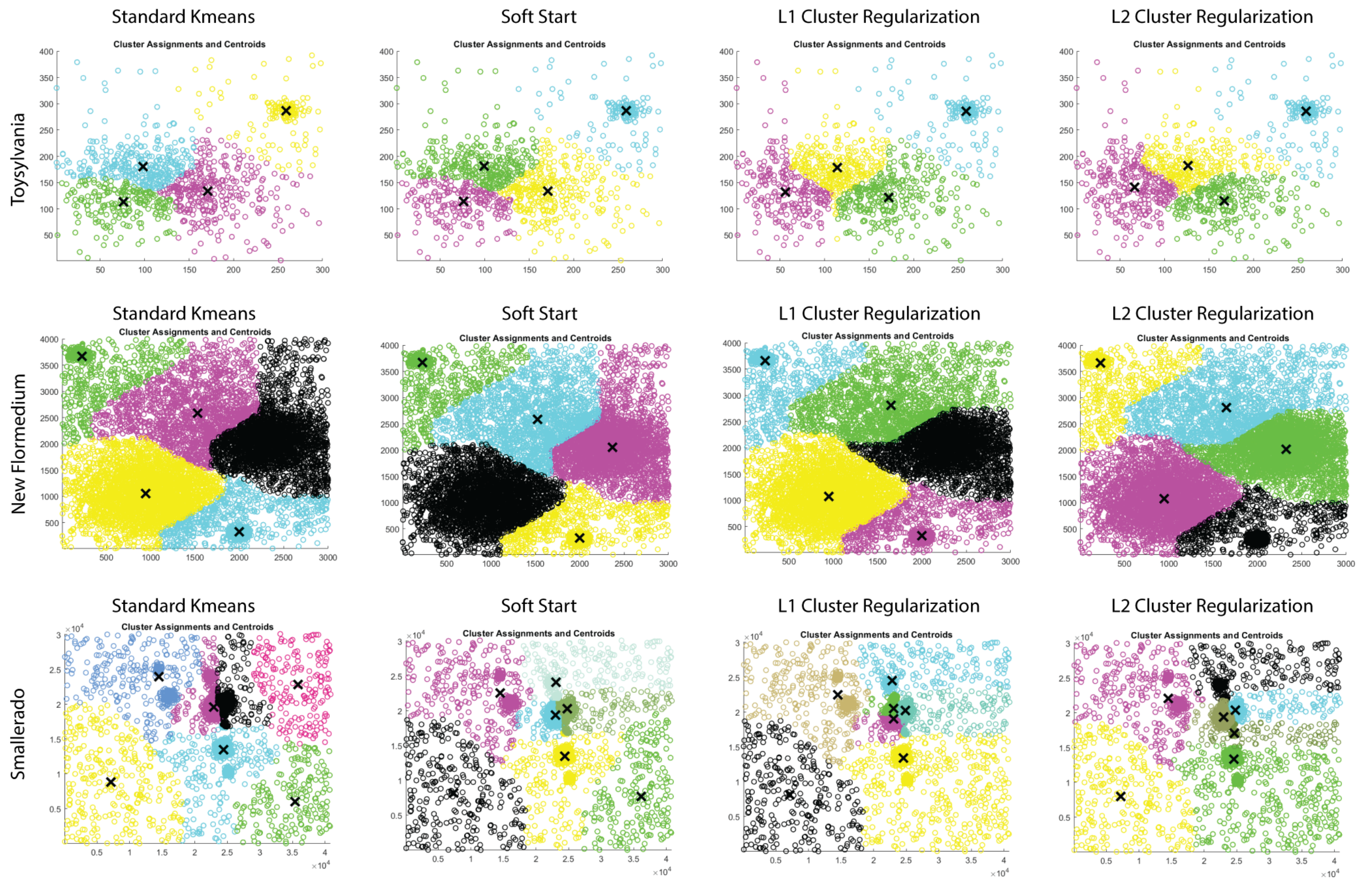

Fig. 2: *Demonstration of Kmeans methods* For each variation of the kmeans (Standard, Soft Start, L1 Cluster Regularization, and L2 Cluster Regularization) each of Toysylvania, New Flormedium, and Smallerado are shown. Each algorithm demonstrates varying borders. Note that Largerado is not shown due to the number of points in the dataset and Matlab will not reasonably show the plot after calculation.

.

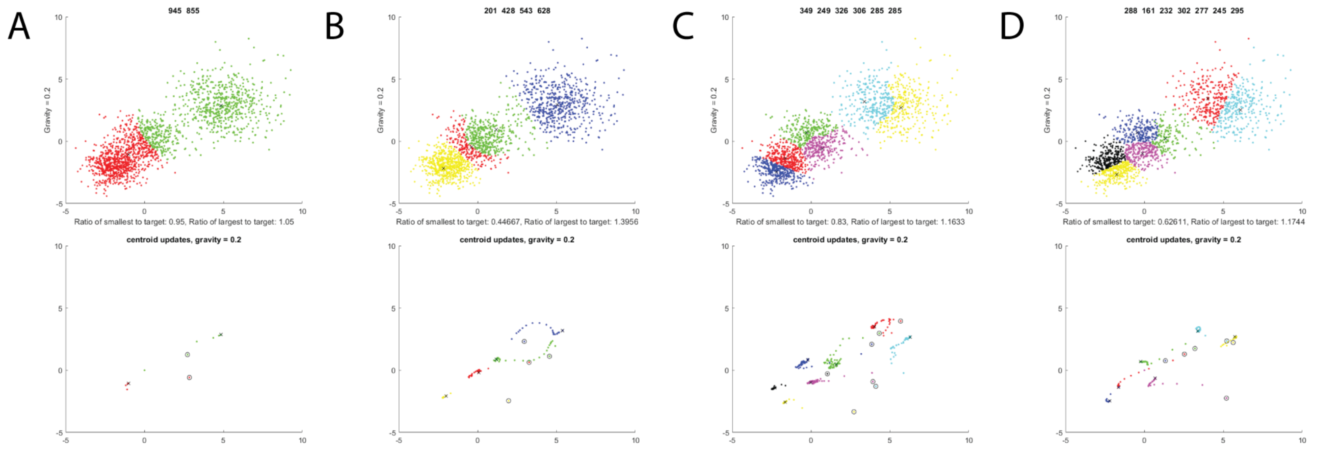

Fig. 3: *Demonstration of BKMG method* The output clustering (top row) and cluster paths (bottom row) are shown for different values of k, k = 2, 4, 6, 7 (A-D, respectively). The target cluster size for each value of k was N/K (900, 450, 300, 257) and the error tolerance was 5 percent.

.

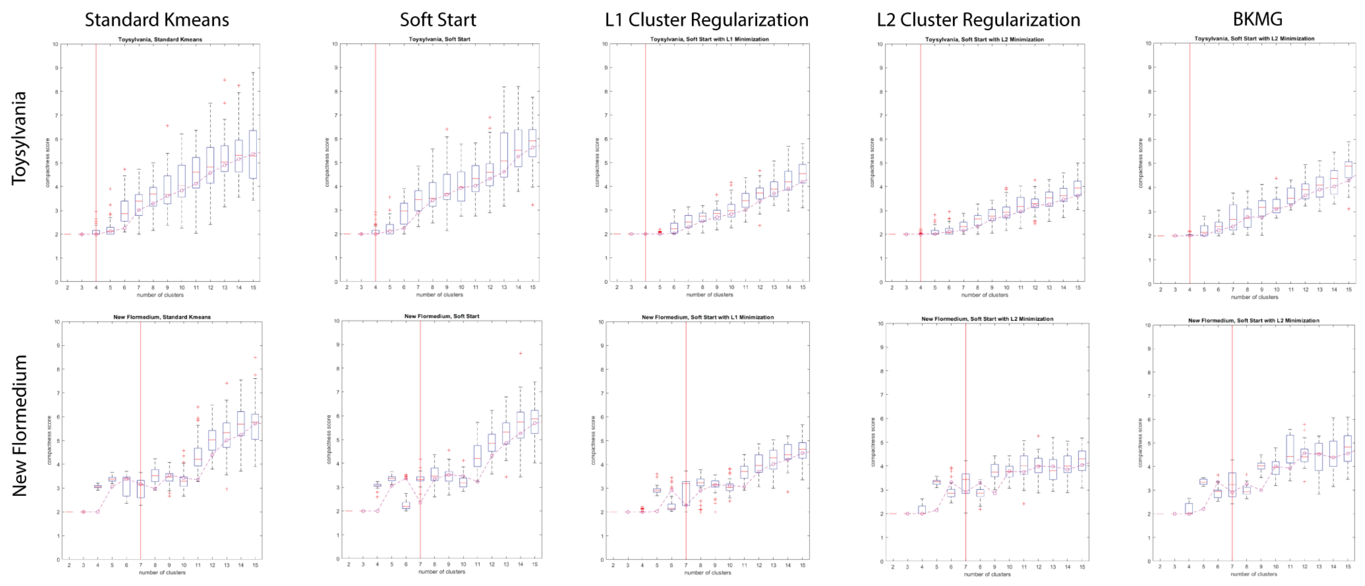

Fig. 4: *Evaluation of Compactness* The graphs depict the compactness across a number of assigned clusters. Each algorithm was ran on both Toysylvania and New Flormedium each randomly initialized 50 times for each cluster. Each cluster depicts a box plot of the 50 trials with a pink line indicating the mean of the trails. The vertical red line indicates how many cities were initialized for the data (Toysylvania has 4 cities and New Flormedium has 7 cities).
.

A. *Problem Formulation*

The standard kmeans objective function is:

$$\min_{\mathbf{C},\mathbf{N}} \sum_{k=1}^{K} \sum_{n=1}^{N_k} \|\mathbf{x}_n^{(k)} - \mathbf{c}_k\|_2^2 \tag{1}$$

The objective function for balanced kmeans that we attempt to solve can be written as:

$$\min_{\mathbf{C},\mathbf{N}} \sum_{k=1}^{K} \sum_{n=1}^{N_k} \|\mathbf{x}_n^{(k)} - \mathbf{c}_k\|_2^2 + \lambda \sum_{k=1}^{K} |N_k - \frac{N_{tot}}{K}| \tag{2}$$

where:

$\mathbf{C} \in \mathbb{R}^{M \times K}$ = a matrix of K cluster centers, with columns $c_k \in \mathbb{R}^M$
$\mathbf{X} \in \mathbb{R}^{M \times N_{tot}}$ = a matrix of N data points, with columns $x_n \in \mathbb{R}^M$
$\mathbf{N} \in \mathbb{R}^K$ = a vector of cluster sizes, N_k , such that $\sum_{k=1}^{K} N_k = N_{tot}$
N_{tot} = total number of data points
K = number of clusters
$\lambda > 0$ = a parameter that penalizes the difference between the size of each cluster and the target size, $\frac{N_{tot}}{K}$

B. *Algorithms*

The balanced kmeans with gravity algorithm (Algorithm 1) is an extension of the classic kmeans algorithm (aka "Lloyd's algorithm"), with additional balance and gravitational steps. Inputs to the algorithm are the dataset, the number of clusters (K), the tolerance on the difference between cluster size and target (sizetol), the max number of iterations (maxiter), and the "force of gravity" parameter (β). β determines the force with which the cluster centers of the smaller clusters will move towards the largest cluster center, as well as the force with which the largest cluster center will move away from its own mean and the mean of the other clusters. This means that, effectively, for each setting of sizetol, there is one parameter, β, to optimize for a given dataset and number of clusters.

Steps 5 - 9 of Algorithm 1 represent the classic kmeans algorithm steps, which consist of iterating between assigning data to clusters and recalculating the cluster centers. Steps 10 - 19 are the balance steps, which move data between clusters to maintain an equal number of points in each cluster, within some tolerance. For each $\mathbf{c}_k$, the balance steps sort the data in the other clusters by nearest to farthest from $\mathbf{c}_k$, and then take the closest data points until the minimum cluster size is reached. Steps 20 - 23 are the gravitational steps, which move each cluster center towards the center of the largest cluster, and then move the largest cluster center away from the mean of the other cluster centers and the mean of the data in the largest cluster, by some amount determined by the gravitational constant and the difference in population between the largest cluster and the others.

C. *Data Generation*

To model the problem effectively, we need to generate synthetic examples of states to run our algorithms. Attempting to take into consideration the three criteria from policymakers (compactness, demographics, efficiency), synthetic examples were made with varying number of cities, political affiliations, populations, and city radii. For each city, we assumed that the population will be evenly distributed from the center of the city. For this project, the two two initial states we created are as follows (see Fig 1A for visual example):

Algorithm 1 Balanced K-Means with Gravity (BKMG)

```
1: Input: X, K, sizetol, maxiter, $\beta$
2: Initialize: C, $i = 0$
3: while not converged, i < maxiter do
4:     $i = i + 1$
5:     for $n = 1, ..., N_{tot}$ do
6:         $\mathbf{x}_n \rightarrow$ nearest $\{\mathbf{c}_k\}$
7:     end for
8:     for $k = 1, ..., K$ do
9:         $\mathbf{c}_k = mean(\mathbf{x} \in \{\mathbf{c}_k\})$
10:        $N_k = |\{\mathbf{c}_k\}|$
11:        if $|N_k - \frac{N_{tot}}{K}| > sizetol$ then
12:            $\mathbf{X}^{sort} \leftarrow$ Sort data in $\mathbf{c}_{j \neq k}$ by distance to $\mathbf{c}_k$
13:            $j = 0$
14:            while $|N_k - \frac{N_{tot}}{K}| > sizetol$ do
15:                $\{\mathbf{c}_k\} \leftarrow \mathbf{X}_j^{sort}$
16:                $N_k = N_k + 1$
17:                $j = j + 1$
18:            end while
19:        end if
20:        $\mathbf{c}_k = \mathbf{c}_k - \beta(1 - \frac{N_k}{N_{max}})(\mathbf{c}_k - \mathbf{c}_{max})$
21:        if $\mathbf{c}_k = \mathbf{c}_{max}$ then
22:            $\mathbf{c}_k = \mathbf{c}_k + \beta(1 - \frac{N_{tot}}{KN_k})(\mathbf{c}_k - mean(\mathbf{C}_{j \neq k}) - mean(\mathbf{x} \in \mathbf{c}_k)$
23:        end if
24:    end for
25: end while
26: return C, N
```

1) *Toysylvania:* there are 3 cities; the state population is 1000 homes; the state size is 300 pixels by 400 pixels; the city radii are 40 pixels, 10 pixels, and 50 pixels; the city centers are placed at (80 150), (260 290), and (150 150); the city populations are 0.3, 0.1, and 0.5 of the full population (and the remaining of the population is randomly distributed); the city political affiliations for republicans/democrats are 0.9/0.1, 0.1/0.9, 0.1/0.9 and the remaining population is 0.5/0.5.
2) *New Flormedium:* there are 6 cities; the state population is 10,000 homes; the state size is 3000 pixels by 4000 pixels; the city radii are 400 pixels, 50 pixels, 500 pixels, 700 pixels, 300 pixels, and 50 pixels; the city centers are placed at (1000 1000), (2000 300), (3200 1300), (1500 2100), (2300 2100), and (200 3700); the city populations are 0.3, 0.1, 0.1, 0.2, 0.2 and 0.05 of the full population (and the remaining of the population is randomly distributed); the city political affiliations for republicans/democrats are 0.6/0.4, 0.3/0.7, 0.4/0.6, 0.1/0.9, 0.9/0.1 and the remaining population is 0.5/0.5.

To further test our algorithm, we aimed to implement our algorithms on as real of data as possible. Since Colorado is one of the more square states, we felt this to be an apt example. The state of Colorado is 380 miles long by 280 miles wide (or 2006400 ft by 1478400 ft) [9]. An average house size in Colorado is 2412 square feet (which is 49.1 feet by 49.1 feet) [10]. Thus, Colorado normalized to average the average house size to a single pixel is: 40860 pixels by 30110 pixels. From the 2010 census, Colorado's population is 5029196 [9]. The top 10 largest cities are [11]:

1) City: population, Democrat/Republican, radius (normalized to pixels)
2) Denver: 600158, 0.75/0.25, 645
3) Colorado Springs: 416427, 0.32/0.68, 521
4) Aurora: 325078, 0.53/0.47, 360
5) Fort Collins: 143986, 0.44/0.56, 410
6) Lakewood: 142980, 0.48/0.52, 613
7) Thornton: 118772, 0.59/0.41, 270
8) Pueblo: 106595, 0.67/0.33, 291
9) Arvada: 106433, 0.52/0.48, 210
10) Westminster: 106114, 0.52/0.48, 256
11) Centennial: 100377 0.5/0.5, 197
12) Total of rest: 2862276 (0.56 of population), 0.49/0.51

To reasonably operate within this information, we decided to make two example Colorados that are downsized from the true information about Colorado (see Fig 1A for visual example).

1) *Smallerado:* each data point represents 1000 houses making the total dataset 5029. We also incorporated 0.1 of the population into the 3 largest cities to have less noise throughout the rest of the state. The other parameters of Colorado are included.
2) *Largerado:* each data point represents 50 houses making the total dataset 100584. The other parameters of Colorado are included.

Taking into consideration the demographics criteria, by incorporating 1000 houses into one data point for Smallerado, we are effectively assuming the demographics of the city are conserved in local areas (an example would be that neighborhoods of people have the same political affiliation). Additionally, by incorporating 50 house into one data point for Largerado, we are subdividing these supposed neighborhoods further to incorporate the demographics criteria.

To test the efficacy of our algorithms, we used the last political cycle for the districts of Colorado and generated ground truth labels of each district (see Fig 1B for visual example). For data loading purposes, when labeling, we downsize an image of Colorado, did the labeling, and then up-scaled the annotations while in the Matlab environment. We included an example of each district's population for Largerado in Fig. 1C.

D. Compactness Evaluation

To evaluate the compactness of the our methods, we decided to incorporate the L1 of every point in a cluster to the centroid of that cluster as our standard evaluation throughout this project. The more compact a cluster the smaller this measure will be. This evaluation measure calls for the straw-man evaluations of both an L1-minimization of random initialization of centroids and an L2-minimization of random initialization of centroids for kmeans to test against our more robust methods with.

For ground truth, Smallerado had a measure of 4.26 and Largerado had a measure of 4.45.

E. Soft Start Method

In attempts to further control the compactness implemented, we wanted to included information that is known about the state beforehand. The assumption is: if cities hold the majority of a state's population, the initialization of the centroids should be at the city centers (which is know a priori in real life as well). So, for an initial start:

1) if the number of districts needed is less than or equal to the number of cities of a state, randomly assign the inital centroids to the cities.
2) else, assign centroids to all of the city centers and randomly initialize the remaining centroids throughout the rest of the state.

This 'Soft Start' method is incorporated with the L1- and L2-minimizations for kmeans.

F. Data Availability

For reproducibility, all code in this project is available at: https://github.com/tjlagrow/ece8823finalproject

IV. Results

The results for standard Kmeans, a Soft Start Kmeans, L1 Cluster Regularization, and L2 Cluster Regularization is shown in Fig. 2. Each method produced a variety of different clusters, mostly centering around the cities (even without the Soft Start).

The results of the BKMG method on a synthetic dataset of 1800 data points are shown in Fig. 3 for 4 values of K. For these experiments, a sizetol value of $\pm 5\%$ was used, with maxiter=100 and β=0.2. The method encouraged equality in cluster size, and the tolerance value was achieved for K=2 in every trial. For K>2, the problem remains sensitive to initialization of cluster centers, as with traditional kmeans, but near-equal cluster sizes were achieved with some initializations. The average runtime for each experiment was approximately 15 seconds, though this varied with K, β, and the initialization of $\mathbf{C}$. The paths of the cluster centers are plotted on the bottom row of Figure 3 to demonstrate the convergence properties of BKMG. Initial cluster centers are shown with a $\circ$ and final cluster centers are shown by $\times$.

As a means of evaluating how our methods hold up with a variety of desired districts (how many clusters to include), Fig. 4 depicts Toyslyvania and New Flormedium randomly initialized 50 times for 2 to 15 clusters for each method. The compactness measure seems to be smallest when the number of clusters is close to the number of cities in the given state. Additionally, compactness is better conserved with the additional regularization of L1 and L2 and performs well with BKMG. All algorithms perform better than the ground truth of Colorado's distracting for both Smallerado and Largerado.

V. Discussion

Although the BKMG method did not yield compact, convex districts for each setting of K, β, and $\mathbf{C}_{initial}$, it did achieve results that demonstrate the potential of a modified kmeans algorithm to address gerrymandering in general. With different values of β, the algorithm yields a variety of cluster shapes, including potentially non-contiguous clusters. That said, the method converges more quickly and to a qualitatively better result with the gravitational step than without it. We also developed a stochastic version of BKMG that adds a small amount of random noise to the cluster centers if they converge before finding a satisfactory solution; this algorithm achieved faster convergence and converged to a tighter tolerance interval than standard BKMG, but we were unable to generate results in time for this report.

However, we did achieve success using soft-start, L1-, and L2-regularized methods. Importantly, the clusters we obtained using these methods were convex, and were significantly more compact than the average congressional district in North Carolina.

VI. Individual Contributions

Kyle Milligan developed the BKMG algorithm, completed the literature review, and helped write the final report.

Theodore LaGrow developed the methods to generate the synthetic data (including information and labeling regarding Colorado); implemented standard kmeans, soft start kmeans, L1 and L2 cluster minimization regularization methods; compactness measure; and helped write the final report.

References

[1] J. Levitt, "Where are the lines drawn?," 2019.
[2] X. Li, Y. Li, S. Ling, T. Strohmer, and K. Wei, "When do birds of a feather flock together? k-means, proximity, and conic programming," 2018.
[3] F. Lindsten, H. Ohlsson, and L. Ljung, "Just relax and come clustering! a convexification of k-means clustering," 2011.
[4] P. Binder, M. Muma, and A. M. Zoubir, "Gravitational clustering: A simple, robust and adaptive approach for distributed networks," 2017.
[5] O. Guest, F. J. Kanayet, and B. C. Love, "Cognitive capacity limits and electoral redistricting," 2018.
[6] S. Whittle, W. Essig, and N. S. Bottman, "Why weight? a cluster-theoretic approach to policital districting," 2007.
[7] J. Peng and Y. Wei, "Approximating k-means-type clustering via semidefinite programming," 2007.
[8] G. Herschlag, H. S. Kang, J. Luo, C. V. Graves, S. Bangia, R. Ravier, and J. C. Mattingly, "Quantifying gerrymandering in north carolina," 2018.
[9] D. Access and D. S. (DADS), "American factfinder," Oct 2010.
[10] Person, "See how denver's average home size compares to other major cities," Oct 2018.
[11] O. S. Nag, "The largest cities in colorado," Jan 2019.