k-means Approach to Legislative Redistricting

Miraziz Yusupov MIRAZIZ@VT.EDU
Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24060 USA

Abstract

Redistricting is a necessary and important process in any democratic society. Modern tools used by legislatures for apportionment of districts are heavily outdated and rely on archaic algorithms. Most use manual redrawing of lines, while others use simulators that simply switch neighborhoods between districts at random until constraints are met. In this document, I outline an attempt to provide a more direct and modern approach by using a modified k-means algorithm. Applying Lloyd's algorithm to redistricting results in mostly contiguous districts that meet two of the three main constraints for legislative districts in the US. However, LLoyd's algorithm does not account for population density and therefore creates districts with large disparities in population. My proposed modification adds a heuristic after the update step to account for the current size of the districts. By penalizing districts that already have large populations, the modified algorithm provides a better population distribution. The redistricting by the modified k-means algorithm converged quickly when there were fewer than 28 districts, but showed significant errors for higher numbers of districts.

1. Introduction

For a representative democracy to work, legislators must be elected to represent the people. Historically, populations were divided into districts which were then represented by legislators. The definition of a district varies, but districts tend to satisfy three constraints:

- **Geographic Contiguity** Every point in a district should be reachable by every other point in the district through some route that only goes through points in the district.
- **Population Equality** Every district should have roughly the same number of individuals within it. State legislatures in the U.S. did not follow this principle until a Supreme Court decision established the "one man, one vote" doctrine in the 1960s.
- **Compactness** The districts should be as compact as possible. Definitions of compactness vary, but the most popular include the perimeter to area ratio and the distance between the two farthest points in the district.

Until the mid-2000s, the most powerful methods for redistricting still relied on manual adjustments and focused more on providing humans with the tools necessary to redistrict themselves. By 2007, several attempts had been made to automate the process of redistricting. Most focused on optimizing only the three major constraints outlined above. Examples of different types of algorithms used today can be found in (Altman & McDonald, 2010).

Legislative redistricting has no known deterministic polynomial time solution, which is why all redistricting algorithms apply different methods of approximation. My initial approach was to use a k-means algorithm and see how well it performed. Once I confirmed that the algorithm failed at maintaing population equality, I began adding heuristics to improve the population distribution. Once I determined the efficacy of the modified k-means approach, the following step was to apply the Voronoi power diagram algorithm in (Fryer & Holden, 2011) and compare the results. Once both were acquired, I planned to modify the Voronoi algorithm to maximize the number of competitive districts. However, due to time limitations, this paper only touches on the modified k-means approach.

2. Input Data

The most accurate redistricting would require data on the locations of every individual in a region. However, such data is hard to come by. Instead, most methods rely on more concentrated measure of population, both to reduce complexity and to guarantee accuracy. Most past papers have relied on Census data, particularly populations by cen-

Virginia Tech CSx824/ECEx424 final project report.

sus tract, which are the small geographic regions the US is broken into for the purposes of the Census. My original plan was to use populations by zip code, since they were likely more permanent and could be used in actual redistricting. However, the populations in zip codes were in general larger and had more deviation between the smallest and biggest zip codes. Therefore, I relied on 2010 census tract population data found on the Census website. The dataset contains 74,002 census tracts with an average population of 4,223. The largest tract had 37,452 people.

3. k-means Approach

3.1. Lloyd's Algorithm

I first used Lloyd's algorithm to see what kind of districting I would get. The results seemed promising, as shown in Figures 1 and 2. Each circle represents a census tract; its color represents which district it was assigned to and it's radius is linearly proportional to its population.

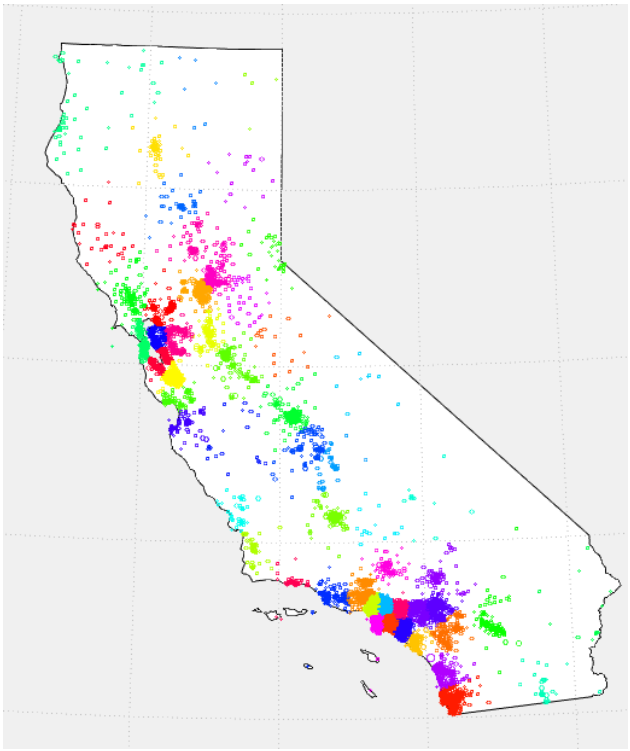

Figure 1. Results of redistricting in California using Lloyd's algorithm.

Cities were seperated into multiple districts, which suggested that they were also being divided evenly in population. However, further inspection revealed that even though the results were better than I expected, the sheer difference in density between rural and urban areas was not overcome.

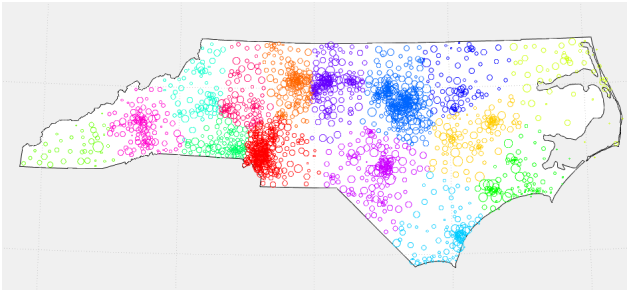

Figure 2. Results of redistricting in North Carolina using Lloyd's algorithm.

In the North Carolina, for example, some districts only had a few hundred thousand people, while others had millions. Although accuracy was low, the algorithm ran within a second on California, so performance was not an issue.

Conceptually, Lloyd's algorithm should always create contiguous districts. However, since I relied on geospatial locations and Euclidean distances, the presence of water sometimes resulted in non-contiguous districts. A prime example can be found in Northeastern Virginia. In some redistricting results, the district encompassing Northern Virginia would also contain some census tracts across the river. Since there tended to be a seperate district to the south of Northern Virginia, the few tracts across the river became disconnected from the main district. A simple way to avoid this problem is to use distances based on land routes, which could be precomputed. I did not have time to do that, so I ignored this case because it rarely occurred.

I also attempted to divide districts into equal parts, so that every district contained roughly 1,000 people. The hope was that as urban areas were more segmented, Lloyd's algorithm would more evenly distribute populations across districts. The results of the algorithm after this change were almost exactly the same as without the change, but the run time increased dramatically, so I decided against splitting the tracts for the remainder of the project.

3.2. Heuristic Modification

My next goal was to modify Lloyd's algorithm to account for the different populations of each census tract. The simplest approach would be to modify the maximization step of Lloyd's algorithm by only assigning a census tract to a district if the district's population was below the parity level. In mathematical terms, if the total population being considered is N and the number of districts is K, then we assign a census tract to a district d with population p_d if and only if $p_d < \frac{N}{K}$. This way, we guarantee that every census tract will be assigned to a district and that the pop-

ulation of any district will never be more than $\frac{N}{K} + M$, where M is the size of the largest census tract. However, this modification has a significant chance of producing non-contiguous districts. For example, if a tract at an edge of a state was surrounded by a district that was already full, then the tract would become a non-contiguous section of another district. Furthermore, the order in which census tracts were traversed would greatly alter the results, in addition to increasing run time.

The approach I decided to use was to penalize districts for populations that strayed from parity. To simplify calculations and to avoid extra computations, I multiplied the distances from a district center to all the census tracts by the district's penalty, which was based on the population of the district after the previous maximization step. If the district had a larger population than parity ($\frac{N}{K}$), then the penalty would be greater than one. Otherwise, the penalty would be less than one. In effect, the algorithm pushed away all the census tracts from the districts with large populations, and pulled in census tracts towards districts with low populations.

My first penalty function was simply the population of the district divided by the parity population. However, the results were lackluster. The dense districts lost some population, but not a lot. The main problem was that even if a dense district pushed the census tracts far away, the resultant distance was still shorter than the distance between those tracts and the adjacent districts. Those adjacent districts tended to already be large, and hence their centers were too far away to absorb more census tracts.

To make up for this imbalance, I provided different penalties for districts with populations above and below the parity level. The districts with low populations were multiplied by a power of the ratio between the district population and the parity population. The districts with higher populations were incremented by a small fraction of the square of the ratio. This way, the sparse districts could eventually start taking tracts far from their centers, while the dense districts wouldn't push all their tracts away too soon. This method worked occasionally, but generally failed to converge. When sparse districts crossed the parity line, they instantously pushed back all the census tracts, which reset their progress. The mirror result occured with dense districts.

Instead of relying on such volatile penalties, I created a cumulative score for each district. Every iteration that a district's population was below the parity level, the district's score was incremented by "-1". Every time the population was above the parity, the score was incremented by "+1". The score was then used in place of the parity ratio in the penalty function described before. This way, if a district's population crossed the parity level, it would only go back one step, instead of resetting to the beginning.

My current algorithm uses this approach and produced the following results.

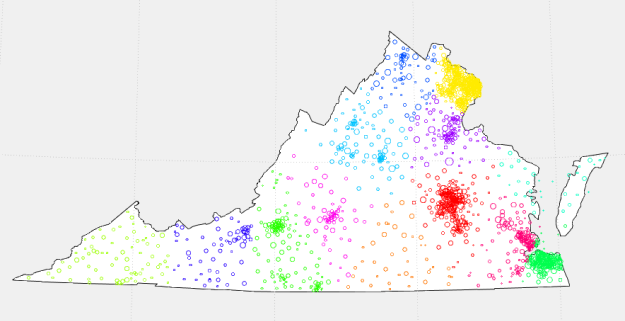

Figure 3. Results of redistricting in Virginia using Lloyd's algorithm.

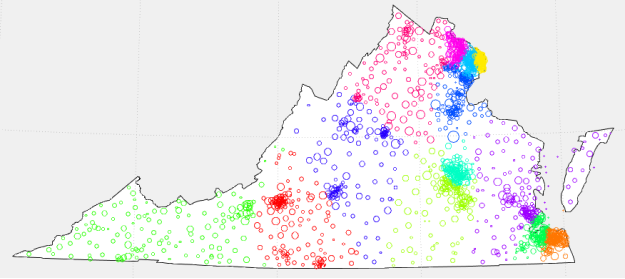

Figure 4. Results of redistricting in Virginia using the modified version of Lloyd's algorithm.

The differences between Figures 3 and 4 are most pronounced in the extremes of density. The western districts in the original Lloyd's algorithm were combined in the new version, while the urban centers of northern and central Virginia were split into multiple districts. In the modified k-means districting for Virginia, the greatest percentage difference between any district's population and the parity population was 4.8%. The algorithm got the result in less than a second.

3.3. Issues

The greatest drawback to the modified k-means is that it did not converge for larger states. After multiple attempts to district California (55 districts), Texas (38 districts), and Florida (29 districts), the algorithm never went below 10% error. In many cases, it never went below 50% error. The crude penalty functions I created simply did not scale. What's more, the algorithm started to run very slowly once the number of districts was in the upper 20s, so any additional modifications I was considering would likely be too

slow for large states. However, the algorithm did remarkably well even with states that had 20 districts. In Figure 5, the modified k-means approach redistricted with less than 5% error.

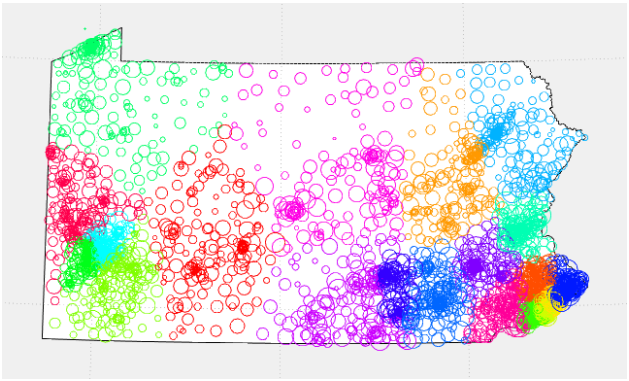

Figure 5. Results of redistricting in Pennsylvania using the modified version of Lloyd's algorithm.

The fewer the number of districts, the faster and more accurate the algorithm became. In Nebraska, as seen in Figure 6, the algorithm ran within a second and produced an error of less than 2%.

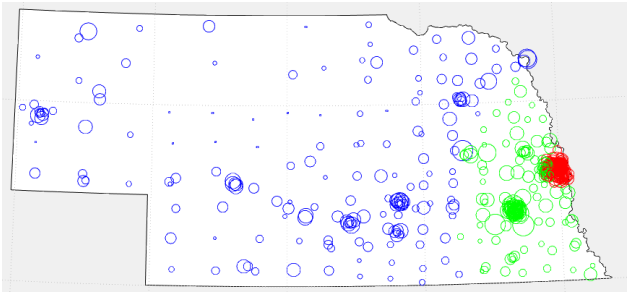

Figure 6. Results of redistricting in Nebraska using the modified version of Lloyd's algorithm.

3.4. Other Considerations

There are many other modifications I did not have time to attempt. For example, changing the expectation step so that the district's center is a population-weighted average of its census tracts might have improved convergence; population density tends to be proportional to the distance to the closest urban center, so having a population-weighted centroid would push the centers of sparse districts towards urban areas and allow them to pick up more census tracts. I could have also made the penalty values functions of the distances themselves. In other words, the farther away a census tract is from a district center, the more effect the district's penalty would have on its distance. This way, the farther districts would be the most affected by penalty changes, which would likely increase convergance rates and reduce run times.

4. Voronoi Approach

The modifications I specified above turn out to be very similar to the ideas used for the power diagram algorithm. The Voronoi approach is extensively explained in (Fryer & Holden, 2011). The basic idea is: districting with only the three constraints mentioned at the start of this paper can be reduced to a power diagram problem. With this simplification, we can assign weights to each district, λ_d, that correspond to its radius in a power diagram. We then run a gradient descent on the vector of lambdas, minimizing the error (the largest percentage difference between any district's population and the parity population). We will then end up with a roughly equal distribution of the population. We then repeat the process as we would in a normal k-means. The key difference in this approach is that the objective function is well-defined and a standard optimization approach is used. However, my attempts to emulate this method did not succeed.

5. Future Work

I hope to implement the Voronoi approach and then maximize on the number of competitive districts, which are defined as those with a 50-50 split of Democrats and Republicans. I would use data from the 2012 presidential election that shows the number of votes for each party by zip code. The goal is to maximize the number of people who have a real choice in their representative, instead of having entrenched incumbents. The paper that highlighted the power diagram approach showed how it incorporated different constraints into its algorithm, so I would try to emulate that process when adding the competitive district optimization.

6. Conclusion

I implemented a functional modification to the k-means algorithm that works well on states with large populations and at most 20 districts. The algorithm, as it stands, slows down significantly and fails when there are at least 28 districts. Better penalty functions could improve the modification further, but a more promising route would be to instead implement the power diagram method. The question I set out to answer still remains: What is the maximum number of competitive districts we can have in a state? I hope to address it in the future once I manage to implement the power diagram method.

References

Altman, Micah and McDonald, Michael P. The promise and perils of computers in redistricting. *Duke J Const Law Pub Poly*, 5:69–159, 2010. URL http://www.law.duke.edu/journals/djclpp/?action=downloadarticle&id=167.

Fryer, Roland and Holden, R. Measuring the compactness of political districting plans. *Journal of Law and Economics*, 2011.