Fighting Gerrymandering by Automating Congressional Redistricting

Jacob Jenness
jenne077@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

Gerrymandering is a political problem that the United States has had for more than 200 years. Politicians have taken the dull and routine process of drawing congressional districts and turned it into a highly-partisan process. However, with recent improvements in redistricting algorithms, researchers Harry Levin and Sorelle Friedler have introduced their recursive Divide and Conquer Redistricting Algorithm. This algorithm has the potential to automate the process of congressional redistricting, thereby removing the potential for bias. By utilizing a set of partitioning and swapping algorithms, the Divide and Conquer Redistricting Algorithm achieves desirable goals, such as low population deviation, and high levels of compactness, as well as meeting all of the legal requirements needed for congressional districts.

Keywords: congressional redistricting, redistricting algorithms, gerrymandering, computational geometry

1 Introduction

For as long as politicians have been in positions of power, they have been trying to manipulate rules in their favor. Gerrymandering is one such example that has become more and more common, as well as controversial, in the United States throughout the last few decades. Many have argued that it gives politicians in power an unfair advantage, that it subverts the will of the people, and that citizens should choose their representatives, not the other way around [1].

To counteract gerrymandering, political scientists, computer scientists, and mathematicians have worked together to create algorithms that can automate the process of congressional redistricting. This is beneficial, as it removes nearly all human input from the redistricting process, essentially eliminating the potential for political mischief.

This paper will analyze one such algorithm, Levin and Friedler's recursive Divide and Conquer Algorithm [2]. First, we will take a closer look at gerrymandering and the current redistricting process in Sections 1.1 and 1.2. Next, we will consider some relevant background information and important metrics used to quantify whether a district is "good" or "bad" in Section 2. As we will see later in the paper, there is no consensus on what constitutes a "good" or "bad" district, so multiple measures are used. Further on, we will review the algorithm and its components in Section 3. And last, we will review the results of the algorithm, with some examples of the districts created in Section 4.

There is an important question that must be asked, however. How do we know that the algorithm we are using is not somehow biased itself? To answer this question, Levin and Friedler do two things. First, they have provided the source code for their redistricting algorithm, which can be found here[1]. This ensures that there is transparency in the redistricting process, and that their code can be independently reviewed. Second, their algorithm does not consider election or demographic data, which is discussed more in Section 2.7. Their algorithm only considers the census data, state shape, and number of districts provided to it.

1.1 What is Gerrymandering?

Gerrymandering is when legislative districts, such as congressional or state districts, are drawn to favor a specific group or political party. The term originated in 1812, when Governor Elbridge Gerry, a Democratic-Republican from Massachusetts, drew a state senate district somewhat in the shape of a salamander [5]. Ever since the term was coined, it has been used to describe a number of unfairly drawn legislative districts, particularly congressional districts.

In the modern day, most gerrymandering is done by analyzing election data, racial census data, and voter registration. Depending on how precisely a district will be drawn, lines can cut around counties, neighborhoods, and even around individual houses.

Gerrymandered districts can be categorized into two main types: packed districts and cracked districts [1]. Packing is when similar voters are tightly grouped together into as few districts as possible so that those similar voters are more concentrated in a few districts and are less likely to win other neighboring districts.

Cracking, on the other hand, is essentially the opposite of packing, as similar voters are spread out over many districts to become the minority in those districts. Figure 1 demonstrates the differences between the two types of gerrymandered districts.

[1] https://github.com/newspapercentral/automated-congressional-redistricting

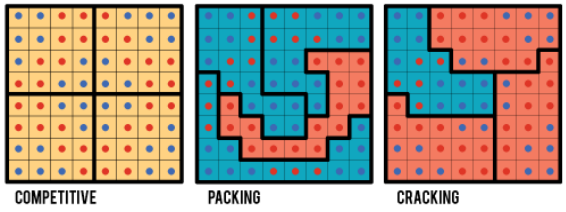

Figure 1. The leftmost diagram shows 4 districts where red voters and blue voters have an equal chance of winning in each district. The center diagram shows a district where the majority of red voters are too 'packed' to win other districts. The rightmost diagram shows blue voters being 'cracked' in most districts, leading to a majority of red districts [3].

1.2 How are Congressional Districts Drawn?

Every 10 years, the Census Bureau surveys the United States to determine the nation's population. Once the Bureau has completed this process, it sends a report of its findings to Congress. Congress then goes through the apportionment process, which is the process of determining how many representatives a state has based on the changes in that state's population. Once this apportionment process has been completed, every state with more than 1 representative must go through the redistricting process.[2]

In most cases, congressional districts are drawn by state legislatures directly. Some states, though, have advisory committees that are tasked with the redistricting process and can recommend maps to the legislature. However, the legislature can still reject these maps if it wishes. Other states have independent committees, which have the exclusive ability to redistrict the state.

2 Background and Definitions

One of the biggest challenges of trying to solve the problems that come with redistricting, is that there is no universal consensus on what constitutes a "fair" district. While it is somewhat easy to point to a district and claim that it is unfair or gerrymandered, it is much harder to do the opposite and claim that a district is not unfair. As we will see, there are many different metrics that can be considered when drawing legislative districts.

2.1 Communities of Interest

Communities of interest are perhaps one of the most difficult factors to consider when going through the redistricting process. This is largely due to the fact that there is no objective or standard definition of what a community of interest is.

A community of interest may refer to a group of people that share demographic traits, such as race, religion, sexual orientation, or culture, and live in close proximity. But, a community of interest could also refer to people that go to church together, or shop at the same grocery store. Due to the fact that these groups are not reliably and universally quantifiable, Levin and Friedler have excluded communities of interest from their analysis.

However, the Voting Rights Act of 1965 requires that certain states consider the racial composition[3] of a state to create a majority-minority district [1]. A majority-minority district is a congressional district where a particular racial minority group or groups makes up the majority of that district's population. In order to comply with this legal requirement, Levin and Friedler suggest merging population units (census blocks/tracts) together to reflect certain communities before the algorithm begins the redistricting process. We will discuss population units later in Section 2.7.

2.2 Competitiveness

When some people think of creating fair congressional districts, they consider whether or not Democrats and Republicans have a somewhat equal chance of winning that district in an election. This is called the competitiveness of a district.

However, most congressional districts are not competitive. Just 17 percent of districts were considered competitive in the 2016 election [1]. One of the reasons that so many districts are considered solidly red or blue is due to geography. Republicans are more likely to live in rural areas that take up a large amount of land, while Democrats are more likely to cluster in large urban areas. Because of this "self-sorting" phenomenon, a problem exists in the redistricting process where competitive districts may have strange shapes. While competitiveness could be considered desirable in some circumstances, Levin and Friedler do not consider it in their paper.

2.3 Proportionality

Proportionality is the idea that the percentage of votes a political party gets in an election determines the percentage of seats that that party receives in the legislature. If, for example, the Democratic party were to receive 40 percent of the total votes cast in an election, then the Democrats would control roughly 40 percent of the seats in the legislative body. While this seems like a fair system, it simply does not work with the current model of district-based representation.

In Massachusetts, for example, Republicans win roughly 35% of the votes in Presidential and Senate elections [5] and there are 9 congressional districts. Therefore, if Massachusetts had a system of proportional representation, then Republicans would win 3 seats in Congress. However, it would be nearly impossible to create 3 congressional districts that all have a Republican majority in them, as Republicans are widely spread out throughout the state.

[2]Every state has at least 1 representative

[3]This is the only circumstance where it is legal to consider race during the redistricting process [1].

2.4 Compactness

Compactness focuses on the shape of a district and prioritizes districts that are closer together than districts that branch off into specific and narrow parts. While there is no officially agreed upon definition of compactness, Levin and Friedler use three separate definitions in their paper [2]. Each measure of compactness outputs a score between 0 and 1, with scores closer to 1 being more desirable. Because each measure produces different results when calculating a district, Levin and Friedler choose to include multiple scores. For example, a district may have a high scores for one measure and low scores in other measures, or vice versa.

For each example below, D refers to the district being computed, and $p(D)$ refers to the perimeter of a district.

1. Convex Hull: $area(D)/convexHull(D)$
2. Polsby-Popper: $(4\pi \cdot area(D))/p(D)^2$
3. Modified Schwartzberg: $\left(2\pi\sqrt{area(D)/\pi}\right)/p(D)$

A Convex Hull is a shape that contains all of the points within a district and has the minimal distance between the outermost points. An example of a Convex Hull would be a random set of nails hammered onto a board, with a rubber band surrounding all of the nails.

Polsby-Popper and Modified Schwartzberg scores, while technically different, are essentially just area divided by perimeter. This simplification will come in handy when determining compactness scores in a later section.

2.5 Population Deviation

Population equality is a prerequisite for congressional districts that requires the populations of each district to be nearly equal to each other. Population deviation is a metric used to measure population equality.

To find the population deviation, ϵ, of a state, we need two things. The first is the ideal district population, L, which can be found by dividing the state's total population by the number of districts for that state. The second is a list of the populations of each district, G_i, for district i. Then we find the biggest difference between district populations and the ideal population, and divide it by the ideal population.

More formally, we can put it like this.

$$\epsilon = (max(|G_i - L|)/L$$

For the rest of the paper, we will define the optimal population deviation to be less than or equal to 0.5%.

2.6 Contiguity

Contiguity is a very important requirement for legislative districts, as every district in the country has to abide by this rule. The formal definition of contiguity says that, for any two points within a district there must be a path between those points that does not cross a district line. Therefore, every part of a district must connect with every other part of a district.

Contiguity also contains a requirement that district lines cannot go through individual houses or apartment buildings. This issue can be avoided in the redistricting process by using census blocks, which will be further explained in the next section.

2.7 Census Data

The Census Bureau categorizes every geographic part of the United States into components of various size. At the smallest level are census blocks, which are drawn in relation to visible lines such as roads, railroad tracks, and streams as well as invisible boundaries like property lines, city limits, and school districts [4].

In urban areas, census blocks are roughly equal in size to city blocks, while in rural areas they tend to be larger and more irregular in shape. Census blocks are grouped into block groups for naming purposes, and block groups are grouped into census tracts.

Census tracts are generally used to encompass a neighborhood, and contain anywhere between 2,500 to 8,000 people [4]. Census tracts are also used to divide individual counties into multiple parts, and counties are used in the same fashion at the state level.

3 The Algorithm

Levin and Friedler have introduced their recursive Divide and Conquer (D&C) Redistricting Algorithm as a potential solution to partisan gerrymandering [2]. The D&C Algorithm utilizes a Voronoi component and swapping components to divide a state into multiple districts.

Let us start with an overview of what the D&C Algorithm will be doing in the following sections. The D&C Algorithm starts with the entire state and treats it like one district by encompassing it in a rectangle. It then divides the state into two districts by growing a district from one of the corners of the rectangle, as shown in Figure 2. We refer to these two districts as a two-district partition for the majority of this section.

Then, we ensure that the two-district partition is contiguous, and begin swapping population units. Once we are no longer able to make population swaps that are beneficial, we start over with a new two-district partition at a different corner of the rectangle.

Once we have made a two-district partition for each corner of the rectangle, we compare the two-district partitions to each other to find the best one in terms of population deviation and/or compactness. When we have found the best initial two-district partition, we repeat the entire process recursively on both districts. This process of division and sub-division is done until we reach the base case, where there are no more districts to create.

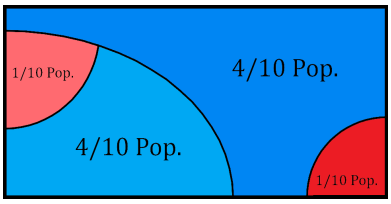

Figure 2. A state that needs 10 districts. The first division yields two blue districts that have a population of 5/10 each. The second division yields another red district in each previous district with a population of 1/10 each. The blue districts with populations of 4/10 will be further sub-divided until we reach the base case.

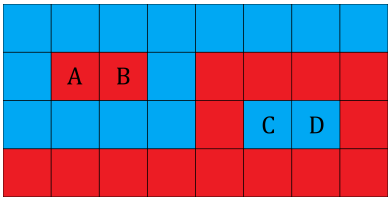

Figure 3. An example of two districts, marked as red and blue, with non-contiguous components. Units A,B,C, and D are not contiguous with their respective districts.

3.1 Divide and Conquer Component

This is the main function in the algorithm. Initially, it starts off with the population of the district being divided, n, the district we wish to divide (which is initially the entire state), and the number of districts that must be created, k.

It also takes as input the maximization function of choice, which we will explore more in Section 3.6. The primary purpose of this component is to create a two-district partition that is recursively subdivided into other districts. We do this by redefining the ideal population, L, every time we must divide.

In Figure 2, we can see an example of sub-division on a hypothetical rectangular state. In the first division, the state needs 10 districts. 10 is an even number, so we can cut the state in half for our two-district partition. Therefore, the algorithm creates a district from the bottom left with a population that is roughly half of that state's population ($L = 1/2 \cdot Population$). In Figure 2, this results in two blue districts.

In the next sub-division, we need 5 districts in both of the previous districts. 5 is an odd number, so we must create one district with the ideal population before we cut in half again. Therefore, we create a district where $L = n/k$ or $L = (1/2)/5$ which equals $1/10$. In Figure 2, this results in two red districts.

3.2 Redistrict Two Districts Component

This is the function that is called by the Divide and Conquer function after L has been determined. First, we define the rectangle that will encompass the district we intend to divide. Second, we initialize a collection of two-district partitions that will be used after all two-district partitions have been generated, so that we can compare population deviation or compactness scores.

Next, we call the Voronoi, Contiguity Swapping, and Population Swapping components on each of our two-district partitions. And finally, we call our Maximization Function on the two-district partitions to determine which one scores the best on the population deviation and compactness metrics.

3.3 Voronoi Component

The D&C Algorithm uses a modified Voronoi component as the second step of its two-district partition process. Essentially, this means that the algorithm starts with a single point that grows in every direction it can until it becomes bigger than L.

The component begins by placing this starting point, which is called a 'seed', in one of the four corners of a rectangle that encompasses a district we want to divide. This seed will grow to become the first district. Then, it looks for the geographically closest population units, such as census blocks, and adds those units to the first district. It continues to do this until the first district's population becomes bigger than L.

Every other population unit not assigned to the first district is then given to the second district. Once the other components are executed, this process will be repeated from the beginning for the other three corners of the rectangle to determine which pair of districts produces the best population deviation and compactness.

3.4 Contiguity Swapping

Because of the way that the Voronoi component creates the two-district partition, the districts may not be contiguous. Therefore, a contiguity swapping component exists to ensure that both of the created districts become contiguous before they are modified further.

Because of the structure of the algorithm, the Voronoi component takes a two-district partition as input. This means that the Voronoi component will only ever process exactly two districts at a time. Levin and Friedler take advantage of this, as all of the non-contiguous components for one district, must be contiguous with the other district, and vice versa.

First, all of the contiguous components of the first district are identified. Second, we find the largest contiguous component in the first district. Third, all of the non-contiguous components in the first district are swapped to the second district. The above steps are then applied to the second district. Now, both the first and second districts are contiguous. Figures 3 and 4 demonstrate contiguity swapping.

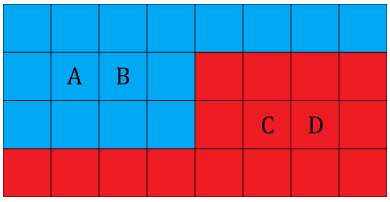

Figure 4. The result of the Contiguity Swapping Component working on the non-contiguous districts. A and B were swapped with the blue district, and C and D were swapped with the red district.

3.5 Population Swapping

After the contiguity swapping component completes, the D&C Algorithm moves on to population swapping. A set of swappable components is found and created between the two districts. These components are found by determining whether a population unit (like a census block) shares a boundary with a population unit in the other district. If the population unit meets this criteria, then it is added to the set of swappable components.

Then, the algorithm determines whether swapping the component is beneficial or not. If it finds that the population deviation is lowered as a result of swapping, the component is swapped and is then removed from the set. This is done until there are no more swappable components.

For most states, this initial population swapping achieves the optimal population deviation [4] after the first round of swapping. However, some states require more than one round of population swapping due to a large number of population units. This is only notable due to the fact that more than one round of population swapping can lead to long compute times, potentially even leading to the algorithm running for more than a day to complete.[5] While this can be significant, it is certainly more efficient than doing the redistricting process by hand.

3.6 Maximization Function

As previously stated, the Voronoi component generates four variations of the two-district partition. The maximization function determines which variation is the best by looking at the scores for population deviation and compactness.

At this point, different maximization functions can be run to determine which two-district partition is the best. This is done at the end of the Redistrict Two Districts component. Levin and Friedler use three different maximization functions in their analysis,

- *MinPop*: Selects the two-district partition with the lowest population deviation.
- *MaxCompact*: Selects the two-district partition with the largest average Modified Schwartzberg compactness.[6]
- *ValidCompact*: Same as *MaxCompact*, but two-district partitions must have optimal population deviation ($\leq 0.5\%$) if possible.

4 Results

The D&C Algorithm's results have been measured with population deviation and various compactness scores. In this section, we will explore these results and look at one specific version of the algorithm. The full results of the algorithm can be found in Levin and Friedler's paper [2], which includes maps for other versions of the algorithm, graphs about population deviation and compactness scores, and more.

4.1 Population Deviation

When using census blocks, population deviation scores across all versions of the algorithm were valid and extremely close to 0.0%.

When using census tracts instead of census blocks, population deviation scores get worse. Figure 6 shows the algorithm being run on census tracts with population swapping enabled. All the results in Figure 6 shows the median scores with the 25th and 75th percentile error bars for their respective measurements. Remember that valid population deviation scores must be less than 0.5%.

Population deviation scores also decrease when the algorithm does not use the population swapping component. When the D&C Algorithm is run on census tracts without population swapping, we achieve even worse results than what is seen in Figure 6. Population deviation scores are almost never considered valid, and can become as high as 35% on MaxCompact.

4.2 Compactness Scores

Recall the simplification we made in Section 2.4. The Polsby-Popper and Modified Schwartzberg scores are essentially just area divided by perimeter. This means that in order to get a high compactness score, a district must have a large area with a relatively small perimeter. This can become a problem when we consider how districts are drawn with the D&C Algorithm.

If you look at states such as California or Texas in Figure 5, you may notice some districts that appear to be long, thin strips of land. These districts generate low compactness scores due to the fact that they have very little area, and a large perimeter. Figure 5 depicts ValidCompact, which

[4] Remember that the optimal population deviation is less than or equal to 0.5%

[5] When running the MinPop version of the D&C Algorithm on the census blocks of Texas, it took 1 Day, 8 hours, 4 minutes, and 8 seconds to complete on a system with 128GB of RAM.

[6] Levin and Friedler do not provide an explanation for why they chose this compactness measure in their algorithm, but do provide multiple compactness scores in their analysis of the algorithm.

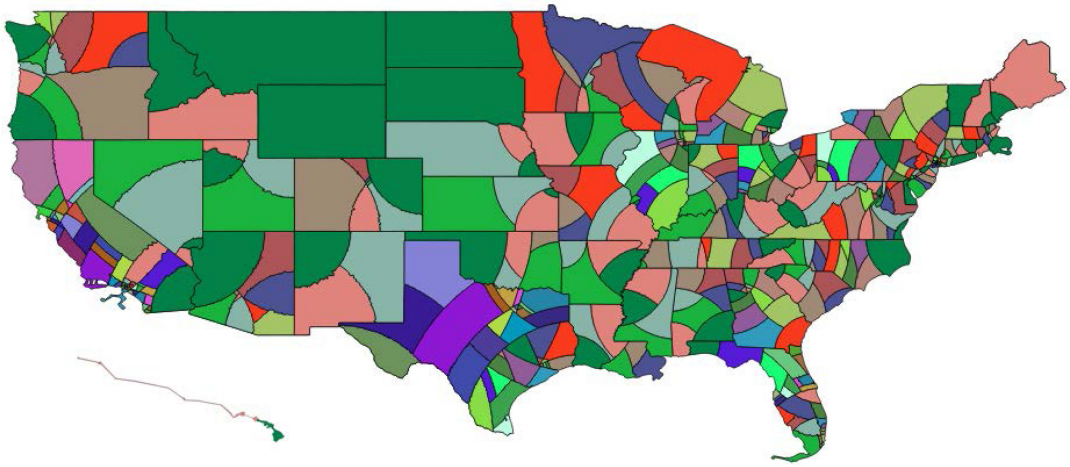

Figure 5. ValidCompact version of the D&C Algorithm run on census blocks. Each color represents a different district of equal or nearly equal population size on a per state basis. Notice how the edges of each district are not perfect circles, and are fit to census blocks. Maps for other versions of the algorithm are included in Levin and Friedler's paper [2].

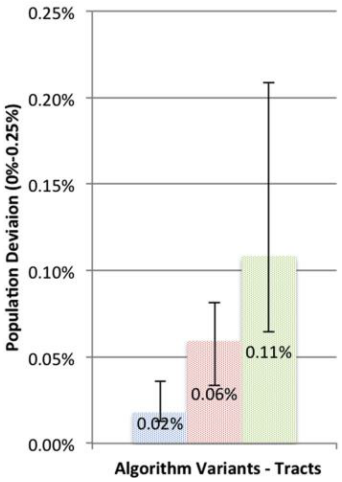

Figure 6. Population Deviation scores for different versions of the algorithm run with census tracts and population swapping. The leftmost bar refers to MinPop, the center bar refers to ValidCompact, and the rightmost bar refers to MaxCompact.

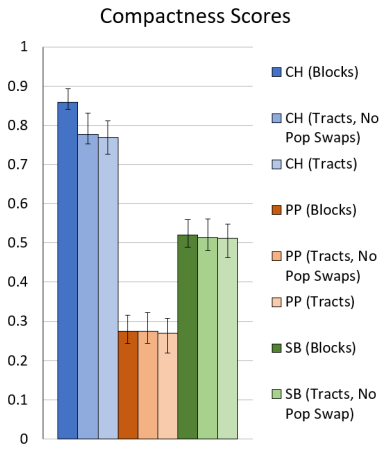

Figure 7. Compactness scores for ValidCompact. CH stands for Convex Hull, PP stands for Polsby-Popper, and SB stands for Modified Schwartzberg

had the best compactness scores for census blocks. However, MaxCompact and MinPop both had worse compactness scores for census blocks than census tracts.

Figure 7 shows the median compactness scores for ValidCompact with the 25th and 75th percentile error bars for their respective measurements.

4.3 Recommended Algorithm Version(s)

Ultimately, Levin and Friedler recommend the ValidCompact version of their algorithm being run on census blocks, which can be seen in Figure 5. Due to this version achieving high levels of compactness while maintaining low population deviation scores, it utilizes the algorithm to its fullest potential.

However, Levin and Friedler also recognize that a single redistricting algorithm may not satisfy the needs of every state in every situation. As we have seen through most of Section 2, there are a wide variety of metrics that can be used to consider whether a district is "good". Some states may value compactness over population deviation, while others may try and include communities of interest in their redistricting process.

Therefore, Levin and Friedler conclude that their algorithm is the most useful when used in conjunction with "a human understanding of the specific redistricting needs." [2]

Acknowledgments

Special thanks to both Nic McPhee and Elena Machkasova for their feedback and advising during the process of writing this paper. I'd also like to thank Harry Levin and Sorelle Friedler for their paper.

References

[1] Galen Druke, David Wasserman, Harry Enten, Aaron Bycoffe, Ella Koeze, and Julia Wolfe. 2017. The Gerrymandering Project. https://fivethirtyeight.com/tag/the-gerrymandering-project/

[2] Harry A. Levin and Sorelle A. Friedler. 2019. Automated Congressional Redistricting. *ACM J. Exp. Algorithmics* 24, Article 1.10 (April 2019), 24 pages. https://doi.org/10.1145/3316513

[3] Daniel McGlone. 2018. Exploring Pennsylvania's gerrymandered congressional districts. https://www.azavea.com/blog/2018/01/23/exploring-pennsylvanias-gerrymandered-congressional-districts/

[4] Katy Rossiter. 2011. What are census blocks? https://www.census.gov/newsroom/blogs/random-samplings/2011/07/what-are-census-blocks.html

[5] Zachary Schutzman. 2020. Trade-Offs in Fair Redistricting. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (New York, NY, USA) *(AIES '20)*. Association for Computing Machinery, New York, NY, USA, 159–165. https://doi.org/10.1145/3375627.3375802