01:198:445
11/21/22

# Redistricting and Gerrymandering NJ State Assembly

Group: R

**Task**

The goal of this project was twofold: first to create a fair map based on a set amount of standards, and second to create a more unfair, biased map, based on a gerrymandering method. To do this, we took a list of precincts and their information and created code with the idea of assigning them in order to create contiguous districts with relatively equal populations, deviating no more than ten percent on either side and not favoring a single party.

**Approach**

The method we chose for this undertaking is based entirely around integer linear programming. By using this, we simply had to establish a number of constraints for the algorithm to follow, and then run it through a solver. This approach seems simple in theory, it has two main flaws: the amount of time it takes to produce a solution, as well as the way it goes about creating assignments. The former made it more difficult to test our code, and the latter caused the issue where, while it assigned chunks of contiguous precincts to a district, the end result was not one where every precinct in said district was contiguous. Unfortunately, this was not a problem we were able to rectify given the time constraints.

Our approach to designing the actual algorithm involved a discussion of the algorithm's constraints we wanted to commit to implementing. Some of these constraints were to have a contiguous district with equal populations. The algorithm takes into account the total Republican and Democratic population. If there is n% Republican population, then they should get a fair representation and so (n%*total districts) should be the amount of districts with Republican majority. Once we decided on our hard constraints, we began to discuss what a fair map might look like. We define a fair map as being a map algorithmically created by factoring in the algorithm itself. This means that our strategy of constructing the districts in a sequential manner, ensuring that each new district added does not vary the population differences by more than +/- 10% would be fair. We decided that this is fair because the districts are decided solely based on the most recent population density, rather than being decided on something that is relative and subject to change such as party distribution and other demographics. The population itself encompasses many different demographic factors, which are always subject to change because people are constantly moving from between districts. Rather than focus on predicting the best districts based on changing demographics, we focus on the population itself.

Moreover, this is a preliminary algorithm that can be changed and modified to better fit the ever-changing demographics of New Jersey. Our algorithm is committed to properly districting the New Jersey districts based on their population density and contiguity, and can be further implemented to factor in other items such as party density, age demographics, etc. Future

iterations of this algorithm can improve this algorithm to fit other needs, but this map effectively treats all districts the same based on their population, and thus is a fair map.

**Objective Function & Constraints**

Our objective function was to minimize the number of precincts assigned (in other words the most optimal solution would be one where each precinct is assigned to a single district)

In our actual algorithm, we had the following constraints implemented:

- Every precinct must be assigned to a population
- In order to allocate the population of a precinct to a district, the precinct must be assigned first
- Every precinct in a district must be contiguous*
- The population of all districts must be within plus or minus ten percent of the average

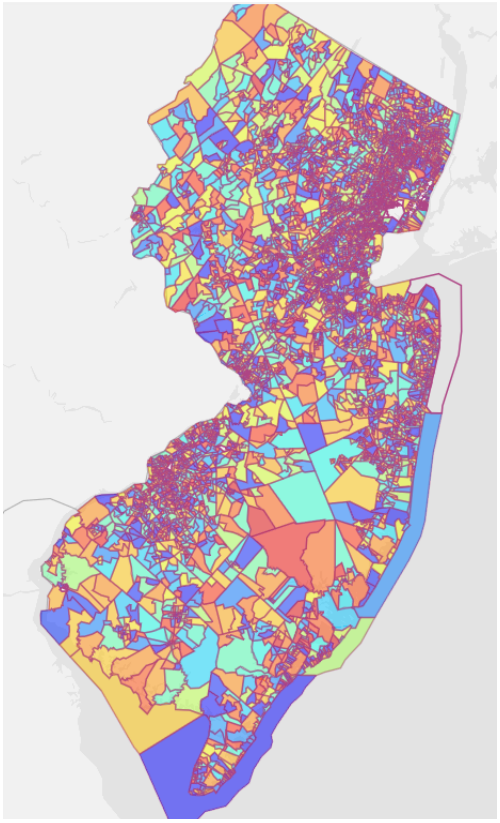
*This constraint was done by going precinct by precinct and seeing which ones were contiguous, and thus could possibly be assigned to the same district. Since ILP determines assignment in "islands" and not incrementally, however, this did not end up working.

**Justification**

The main factor behind our choice in using Integer Linear Programming for our code was the fact it is relatively simple to other methods, only requiring the establishment of the problem and constraints to generate a solution. While the solver's method of assignment was an oversight, if given more time, we could work out a different constraint that checked district contiguity instead of just for individual precincts. While the amount of time it takes may seem troubling, we decided that in exchange for ILP's accessibility, it was well worth it. Our choice of a gerrymandering method in packing, however, was mainly focused on effectivity over all else. It is one of the best ways of ensuring one party comes out beating another, despite potential differences in population from one party to the other.

**Analysis**

Fair Map:

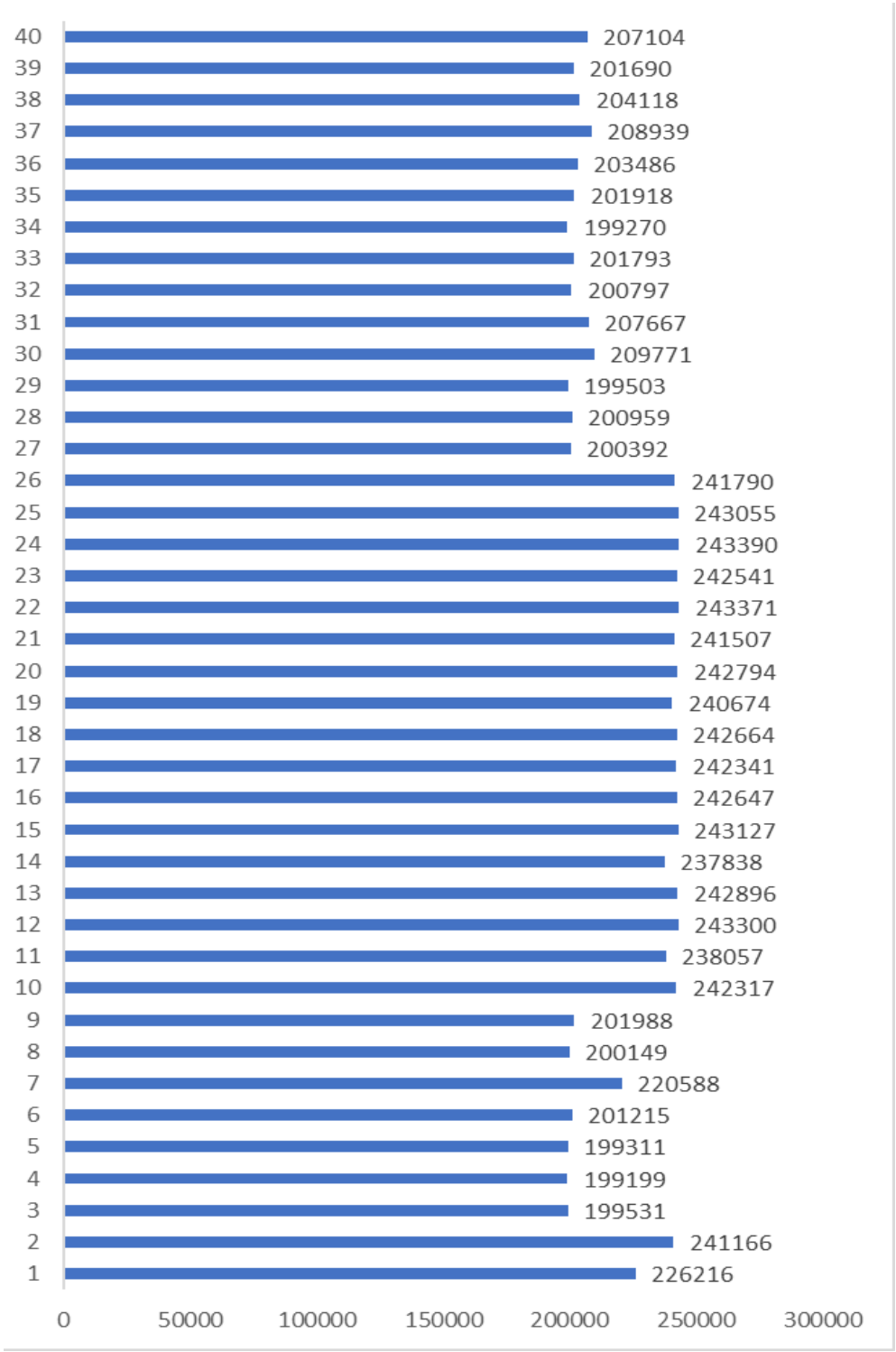


While the lack of contiguity makes it difficult for DRA to determine some of the metrics such as equal population and proportionality, we have shown them through actual operations performed on the CSV file. Using this, we can observe that, although the precincts may not be contiguous, our algorithm at least strives to maintain a level of fairness through its constraints.

Data:

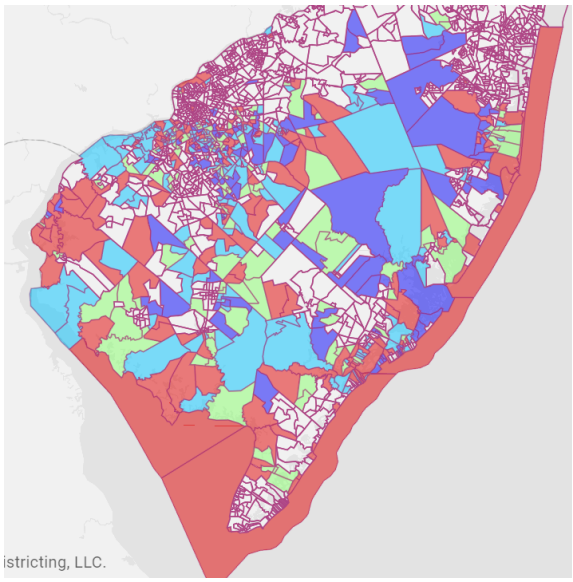
Average Population: 221277
Upper Bound: 243404.7
Lower Bound: 199149.3
Minimum: 199199
Maximum: 243390

From this data we can observe that the minimum population that a district has is greater than the lower bound (hence, within 10% of the average population) and the maximum population that a district has is less than the upper bound. The below page shows a graph for the population of each respective district. We also felt that it would not be so blatant as to get immediately thrown out in courts.


40 207104
39 201690
38 204118
37 208939
36 203486
35 201918
34 199270
33 201793
32 200797
31 207667
30 209771
29 199503
28 200959
27 200392
26 241790
25 243055
24 243390
23 242541
22 243371
21 241507
20 242794
19 240674
18 242664
17 242341
16 242647
15 243127
14 237838
13 242896
12 243300
11 238057
10 242317
9 201988
8 200149
7 220588
6 201215
5 199311
4 199199
3 199531
2 241166
1 226216
0
50000
100000
150000
200000
250000
300000

Unfair Map:

- The average map-wide Democratic two-party vote share is 56.04%, the Republican 43.96%.
- The number of Democratic seats closest to proportional is two. The likely number of Democratic seats is 1.47. The likely number of unexpected Democratic seats (won) lost is 0.53.
- The rating is zero ('Very Bad'), because this map is antimajoritarian. Even though they will probably receive roughly 56.04% of the total votes, Democrats will likely only win 36.78% of the seats.




For our unfair map, our goal was to create a skew towards one of the parties, in our case the republican party, through the use of gerrymandering tactics. The way we implemented this was through a method known as packing, in which a portion of districts are set up where the negatively impacted party has a clear majority in them. This causes them to win with an overwhelming majority in said areas, but also increases their efficiency gap. At the same time, the majority of the districts are established such that the positively impacted group barely beats the other party, minimizing their efficiency gap, maximizing their opponent's, and ending in their eventual win.

We applied this algorithm to our program by adding two additional decision variables for the democratic and republican party population in each precinct. On top of this, we used an extra constraint that for one-fifth of the districts made the ratio of democrats to republicans five to one, while for the rest simply making sure there were more republicans than democrats.

The above map shows a subset of the csv that our code would produce, as the actual program takes too long to run, given the time constraints of the project, although given a bit more time we could produce a full csv. The analysis of it shows that there, as intended, is a clear advantage given to the republican party, causing them to win even when they got a lower amount of overall votes. The only problem with this is that it may be a bit obvious, although the ratio of districts actually packed could be lowered in order to make this less so. The subset also showcases the earlier problem with the precinct contiguity constraint a bit better, as it clearly shows that some areas have contiguous precincts in the district, but only in spurts.

**Conclusion**

In conclusion, our Integer Linear Program algorithm works to assign one precinct to exactly one district, while maintaining population variance to be within 10% of the mean. With more familiarization with python linear programming, as well as a bit more time, we could also

modify our contiguity constraint to look on a district to district level, instead of precinct to precinct. At the same time, we ensured that in our fair map, there would be a proportional level of fairness when it comes to republican and democratic representation in the state. Through the gerrymandering method packing, we created an algorithm that essentially ensures that one group will have the highest possible efficiency gap while the other will have the lowest. We felt this choice would be the most effective, while not being completely blatant. Overall, our method of precinct assignment shows what fair redistricting can look like, while also showing how much the process can be abused and what impact it can have.