Red vs. Blue: An Exploration of Mathematical Gerrymandering Using Simulated Annealing

Bryce Tham, Nicholas Choa, and Julia Medina

Introduction

In the United States of America, it is a common practice to have political parties manipulate the boundaries of electoral districts to favor a specific party. The party with the majority of members in the House of Representatives can redraw the political boundaries of their state after a decennial census. When the redrawing of the districts is intentionally used to benefit the minority party by creating as many districts that will elect members of their party and the least for the opposing party is called gerrymandering. This process includes drawing odd shaped districts to maximize the supporters votes and minimize the opponents' votes.

There are various constraints and criteria that redistricting plans must follow: all districts must be roughly equal in population and the shape of district must be contiguous. These same constraints imposed to the political aspect of gerrymandering will be used as the constraints in the code. Other constraints exist in the real world (such as not intentionally weaken the voting strength of a minority group) that are not explored in this report.

Through computer simulation, the boundaries of districts can be redrawn to optimize the electoral count of the intended political party while minimizing the opponents' count. In order to achieve this, various definitions and function were created to simulate the optimization and manipulation of a fictitious population. Using Python with scientific computing library NumPy, we stored our data in a matrices and vectors to create a color graph to represent political parties and show a clear graph of how the districts are separated/redrawn, allowing for a better visual understanding of how gerrymandering works and to see how easily the outcomes can be changed with the same given inputted data.

At first we created a matrix to represent a state with ones and zeros, each representing the political party of each member of the house. Then, in order to keep true to the political constraints, a function definition was created to assure that when a district is being redrawn it is contiguous and roughly equal in size. Since the whole purpose of this optimization problem is to minimize the opponent's political votes, which can be clearly achieved through simulated annealing.

With simulating annealing it makes it easier to check the neighbors state, which in this case is the political affiliation of the member, and determines whether it should be combined with the district depending on the probabilistic decision, the majority political party.

Mathematical Specification

While gerrymandering as a political concept is relatively simple to understand, the difficult part comes when the need to mathematically model the problem specification is required. In order to do this, we need to introduce several variables to represent the population points and their respective neighbors, party affiliation, and district membership.

Consider a graphical model in which the vertices are population points and the edges represent adjacency between two neighboring vertices. If each vertex is enumerated with an integer in the range $[0,\ p-1]$ (where p is the number of population points), we can choose to model our graph using an adjacency matrix A in which $A_{ij} = 1$ means population points i and j are neighbors and $A_{ij} = 0$ means they are not. Because the graph is undirected, A is necessarily symmetric (i.e. $A_{ij} = A_{ji}$).

To model party affiliation, we will use two vectors, one for each party. Let R be the vector in which R_i is the number of votes that political party R receives at population point i. Likewise, let B be the vector in which B_i is the number of votes that political party B receives at population point i. We can generalize this model to include more than 2 political parties; however, to simplify representation, we will assume that only these two political parties exist and that any vote for any other party is insignificant.

District membership can be modelled similarly to party affiliation. Let D be the vector in which D_i is the district number of which population point i belongs to. For example, if $D_5 = 2$ then population point 5 belongs to district 2. Like population points, each district is enumerated with an integer in the range $[0,\ n-1]$ (where n is the number of districts).

As an optimization problem, the goal is to maximize the margin of victory of one party, which is equivalent to minimizing the margin of victory of the other party. Suppose we wish to maximize the margin of victory for political party R in a region where the overall majority of population points lean towards political party B. Thus, we wish to minimize the margin of victory of B using the following objective function M:

$$M = \sum_{i=0}^{n} sign\left(\sum_{j=0}^{p} (B[j] - R[j])(D[j] = i)\right)$$

Recall that n is the number of districts and p is the number of population points. $D[j] = i$ returns 1 if the district of j is equivalent to i, and it returns 0 otherwise. $sign(x)$ returns 1 if

$x > 0$, -1 if $x < 0$, and 0 if $x = 0$. The objective function M returns the margin of victory for political party B; remember, we are trying to minimize this.

There are two constraints that need to be modelled. The first is the equal population constraint E which returns the difference in size between the district with the most population points and the district with the least:

$$E = max\,(s_i) \;- min\,(s_i)$$

The variable s_i is an element in the size vector S of magnitude n in which the integer at index i is the number of population points in district i. The vector S can be determined as follows:

$$S = [\sum_{j=0}^{p}(D[j] = 0),\ \ldots,\ \sum_{j=0}^{p}(D[j] = n)]$$

Recall that we wish E to be smaller than some value. Therefore, we can set a tolerable variance variable var such that the objective function is constrained to $E \leq var$.

The second constraint is the contiguous district constraint C which returns 1 if all districts are contiguous and 0 otherwise:

$$C = \prod_{i=0}^{p}(\ \| conn(i) \| \ = \sum_{j=0}^{p}(D[i] = D[j]))$$

The function $conn(i)$ returns a set of population points that are both connected to population point i and are part of the same district as i. $\| x \|$ returns the magnitude (or size) of a set x; therefore, $\| conn(i) \|$ returns the number of population points that are connected to and are of the same district as i. $D[i] = D[j]$ returns 1 if the district of i is the same as the district of j. Thus, $\| conn(i) \| \ = \sum_{j=0}^{p}(D[i] = D[j])$ will return 1 if the number of points in $conn(i)$ is the same as the total number of points in district $D[i]$ and 0 otherwise; if for any i this expression returns 0, then $C = 0$, ensuring contiguity if $C = 1$.

Simulated Annealing

The main approach to this problem involved using simulated annealing, a method often used for both constrained and unconstrained optimization problems, which helps in approximating some global optimum. The nomenclature behind this technique was very much inspired by annealing in metal-works, where one has a control over the heating and cooling of a material to increase the size of its crystals, minimize its defects, and manipulate it into a more precise form. Our source code does exactly this; it takes an initialized map of unweighted population points, that have been put into districts, and uses some global variable as the “temperature” that slowly cools down as the map is constantly being transformed and manipulated (Kirkpatrick, 1) depending on which color we would like to win. A win for us meant maximizing a margin of winning “votes” or population points.

In more detail, the temperature would start at some initial state; we used the value of 1000 at first. Then we would input it into a function for simulated annealing. The map of districts would then undergo a process that would rearrange the points into different districts. While being rearranged, we would constantly check and make sure that it was getting closer to a minimized loss for the opposing party. Not only that, but we also implemented the various constraints mentioned beforehand that help us simulate more realistic form of gerrymandering (Levitt, 2).

In order to ensure the validity of our model, we needed to make sure that all points in the district were adjacent to one another. We achieved this by checking the adjacency matrix relative to the population points on the map. We could not have districts spotted all over the map and say that these separated areas were being correctly represented. A constraint function we have implemented is one that makes sure that all the districts have roughly the same amount of people. In smaller overall population counts, the variance between district populations is very minimal if not non-existent. We decided to go with a rough equality constraint because much like in real life, where America’s congressional districts are not all completely equal. Also, this helps avoid stalemates when there is an even number of population and assures that all points are included, which brings us to the third constraint. The other constraint we have implemented into a function makes sure that all the districts are connected to each other, which also ensures that no points are left out and that the entire population on the map is being represented. All of these constraints provide a more realistic approach to our mathematical perspective towards gerrymandering.

Every time the program would disobey one of these constraints, a very large penalty would be added to the global temperature constant. The penalty can range from 10 to 1000 in order to achieve better districts. Having the temperature set back so far away from 0 is important because we found it very common that if one of the rules had been broken, further manipulation of the board often made things worse from that point onwards.

Once the “temperature” value gets close enough to 0, the resulting array provides perfectly manipulated matrix that has all the points included in (roughly) equally populated districts. Utilizing simulated annealing, the average amount of iterations and wins amongst a batch of simulated gerrymandering sessions prove how effective our program is:

batch size	**average iteration count**	**wins/losses/draws (blue)**
50	134	32/2/9
100	149	68/5/28
200	148	141/32/27

(We set the blue party to be the winner of 16 points, with a majority of them being red.)

Quantitative Results

The data below shows the results of performing gerrymandering via the simulated annealing algorithm. Recall that n is the number of districts, p is the number of population points, and r is the percentage of population points that are blue. Each variable setting is run 50 times and the margin of victory is averaged to give the results for each row. Political party affiliation is chosen randomly for each population point adhering to the percentage defined by r.

The table below shows the resulting opponent margins of victory for the blue party when the number of districts is varied in the set $n \in \{2, 4, 6, 8, 10, 12, 14\}$ with $p = 16$ and $r = 0.6$.

n	**p**	**r**	**blue margin of victory**	**blue wins**	**red wins**	**draws**
2	16	0.6	0.02 ± 0.13	10	6	34
4	16	0.6	-1.56 ± 0.20	4	36	10
6	16	0.6	-1.98 ± 0.26	2	37	11
8	16	0.6	-2.06 ± 0.29	4	38	8
10	16	0.6	-2.0 ± 0.51	13	28	9
12	16	0.6	0.82 ± 0.59	27	18	5
14	16	0.6	2.1 ± 0.48	36	13	1

The table below shows the resulting opponent margins of victory for the blue party when the number of population points is varied in the set $p \in \{9, 16, 25, 36, 49\}$ with $n = 8$ and $r = 0.6$.

n	**p**	**r**	**blue margin of victory**	**blue wins**	**red wins**	**draws**
8	9	0.6	0.84 ± 0.44	24	13	13
8	16	0.6	-2.84 ± 0.26	2	41	7
8	25	0.6	-3.06 ± 0.25	0	43	7
8	36	0.6	-3.34 ± 0.24	1	46	3

8	49	0.6	-3.26 ± 0.23	0	44	6

The table below shows the resulting opponent margins of victory for the blue party when the percentage of blue points is varied in the set $r \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ with $n = 8$ and $p = 16$.

n	p	r	blue margin of victory	blue wins	red wins	draws
8	16	0.5	-4.28 ± 0.22	0	48	2
8	16	0.6	-2.32 ± 0.32	5	38	7
8	16	0.7	-0.54 ± 0.37	14	25	11
8	16	0.8	2.26 ± 0.41	31	7	12
8	16	0.9	4.52 ± 0.32	46	0	4

Conclusions

The data suggests the following trends regarding gerrymandering via simulated annealing.

When varying the number of districts, the margin of victory for the blue party forms an inverse bell curve with its minimum when $n = p/2$. If the number of districts is too small, then there is less "wiggle" room when it comes to manipulating the district lines in a particular party's favor; if the number of districts is too large, then most of the districts will be made up of single population points, effectively taking the majority vote of the entire graph (in which case the majority party will win). Therefore, to ensure maximize the effects of gerrymandering, it is best to initialize the number of districts as half the total number of population points and adjust it accordingly (we will see later that this point is not always the global minimum).

When varying the number of population points, the margin of victory for the blue party decreases as the total number of population points in the graph increases until no further reductions can be made. For example, the margin of victory for blue sees the steepest decrease between $p = 9$ and $p = 16$, but averages out to a constant value after $p = 25$. This is because like with the number of districts, a larger number of population points allows for more "wiggle" room to maximize the effects of gerrymandering. Beware, however, of the caveat mentioned following the first conclusion; increasing the number of population points to $n << p$ may result in highly diminishing returns.

When varying the percentage of blue points, an obvious trend occurs: the higher the percentage of blue points in the graph, the higher the margin of victory is for the blue party. The interesting thing to note about this data is that even with a graph made up of 80% blue points, it is still possible for the red party to win in at least some scenarios. While seemingly shocking, this is consistent with some estimates that the United States presidential election can be won with only 22% of the popular vote (Kuzoian). Therefore, the main conclusion to be drawn is this: in most scenarios, so long as it maintains some measure of reasonability, it is very possible for the minority party to win an election through gerrymandering.

Gerrymandering New York

As mentioned earlier, after a state calculates its total population and number of representatives in the census, it then determines how many districts it will have. The majority party will redraw its districts. In order to test this optimization problem in a real life scenario, New York was chosen as an example. To represent New York within the code, an adjacency matrix was created with each node representing the state's counties, two arrays to represent the two political parties, and one array to represent the population counts with the indexing corresponding to the counties. Within the arrays corresponding to the political parties, there is either a 1 or 0 to represent whether that district was won or not with the indexing matching the latter. The array corresponding to the population count contained the number of population rounded to the closest tenth thousand. When the information on New York is run, it first determines the original victory margin of the blue party. It then goes through the simulated annealing to determine the new victory margin. Once it is finished it was seen that in one try the victory margin had decreased from 15 to 7 and in another one from 15 to 8, showing that the simulation is not perfect but falls within the margin of errors that could have occurred.

Difficulty & Individual Effort

The individual efforts of each member of this team is described below.

Bryce worked on the mathematical specification and the conversion of the objective function and constraint functions into code. The difficult part about this was deciding how to clearly define these functions in mathematical terms that could be easily and directly translated into Python. In addition, Bryce also offered his version of the simulated annealing function to run the optimization problem, thereby contributing to the data shown in the results above. He also wrote up the main conclusions found in the data including any trends worth mentioning in the report.

Nick worked on a few on the constraints functions, the adjacency matrix setup function, put together the poster, and covered the simulated annealing section of the paper. It was very challenging when it came to creating the adjacency matrix generator, as the previous functions took the matrix in a zig-zag/snake form and the amount of population points and rows varied.

Julia worked on simulating the state of New York on the functions, the introduction on Gerrymandering, and Gerrymandering New York section of the paper. The most challenging part was making sure the adjacency matrix represented New York correctly and assuring that the implementing was simulation was done correctly.

Works Cited

Kirkpatrick, C. D. Gelatt Jr.1, M. P. Vecchi 21 Research Staff Members at IBM Thomas J. Watson Research Center, Yorktown Heights, New York 105982 Instituto Venezolano De Investigaciones Cientificas, Caracas 1010A, Venezuela, S. "Optimization by Simulated Annealing." *Optimization by Simulated Annealing | Science*. N.p., n.d. Web. 04 Dec. 2016.

Kuzoian, Alex. "Turns out a Presidential Candidate Could Win the Election with Just 22% of the Popular Vote." *Business Insider*. Business Insider, 18 May 2016. Web. 08 Dec. 2016.

Levitt, Justin. "Where Are the Lines Drawn?" *All About Redistricting -- Where the Lines Are Drawn*. N.p., n.d. Web. 04 Dec. 2016.

"Simulated Annealing." *Simulated Annealing - MATLAB & Simulink*. N.p., n.d. Web. 04 Dec. 2016.