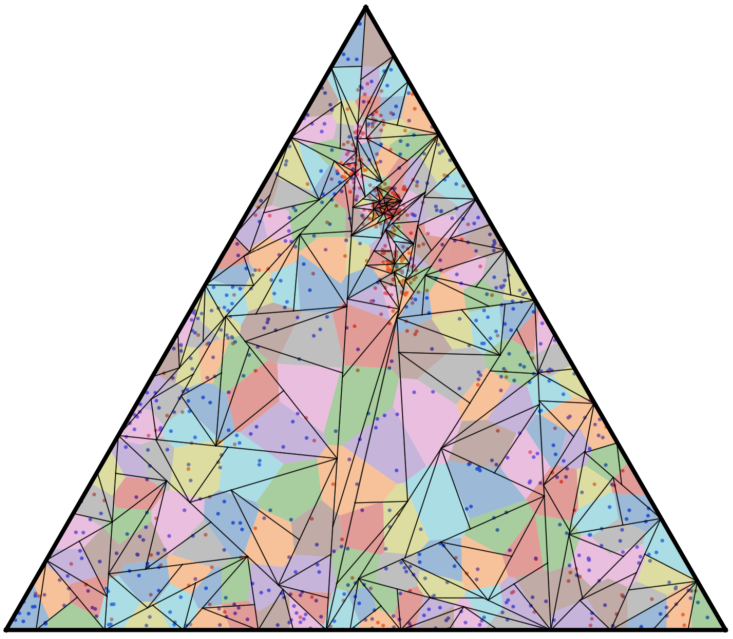
COMS W 4444

PROGRAMMING AND PROBLEM SOLVING



Project 2: Threeland

Submitted By :
Rebecca Calinsky, John
Daciuk, Kelley Valentine

October 21, 2019

Contents

1 Introduction

1.1 Problem Statement

The wise leaders of Threeland are in the midst of devising an election scheme and request the help of computational wizards. We are tasked with helping choose between two alternatives for electing 243 representatives:

1. 243 districts with a winner-takes-all election for 1 representative in each.
2. 81 districts with 3 representatives each, roughly divided between political parties with a significant number of votes in each district. [1]

The leaders are interested to find out which system is more robust to curb gerrymandering and want us to quantify the difference succinctly.

1.2 Goal

In order to help Threeland avert the vicious cycle of gerrymandered elections, we shall *ourselves* attempt to gerrymander the two alternative systems of representation. After all, actual implementation of gerrymandering is far more convincing to non-technical leaders than theory. Through the process, we shall employ various metrics to quantify the extent to which we can gerrymander the two systems, first and foremost being the election outcomes.

We want to make clear that with many challenges to overcome, we choose a special case of the problem in which we only ever have two parties and there are no campaigns.

1.3 Summary

It's worth noting the following intuition from the outset: more representatives per district should be generally less prone to gerrymandering. Consider the limit where each voter gets his or her own representative in the house; we have squashed the threat of gerrymandering entirely at the expense of introducing many other problems and impossibilities. In general, as the number of representatives per district increases, the number of districts decreases and quickly becomes intolerable to political philosophers considering systems of representation.

To colloquially consider 1 representative vs 3, consider the following: any successful machination to gerrymander must do so by "wasting" rival party's votes, but be-

[1] Technicalities of the 3 rep system can be found in the full problem statement: http://www.cs.columbia.edu/ kar/4444f19/node18.html

cause the 3-representative system is able to reflect voter preferences in each district at a more granular level, it's not possible to waste as many votes as it is in the winner-takes-all system. For example, for two parties in the winner-takes-all system, a party may have 49% of the vote in every district and still be left with zero representatives in the house. In the alternative, that party would at least be left with 1/3 representation.

All of this, however, is over simplified worst-case analysis and our real task is to quantifying the difference for actual population data, which is far more challenging and computationally intensive.

Let us begin our summary of solutions by first enumerating the many hurdles:

1. Ultimately, we care about the outcome of elections with 333,333 voters in either 81 or 243 polygonal regions we must draw. The space of possible district configurations is practically infinite and optimal configurations are a function of the large number of voters, each of whom is multidimensional in location and preferences. Considering case by case scenarios and manually drawing is out of the question if we want a final solution that can scale. Even with the use of a computer, many techniques are out of the question due to run time complexity.

2. The districts are constrained to have 9 sides or less, contain within 10% of the mean per district population and tessellate the map. The side restriction precludes the possibility of building districts out of very many pieces without some post process smoothing, while the mean restriction left many of our districting attempts invalid.

3. Generally speaking, population densities are not uniformly distributed, thereby complicating most of the issues mentioned above. We must cope with the extremes of cities and grasslands by finding a set of tools that is flexible yet still sophisticated enough to exhibit interesting behavior.

We eventually abandoned bottom-up algorithms and hand-drawn schemes for they either could not obey all constraints or were too painstaking to be effective. We soon converged on Voronoi polygons as the linchpin of our strategy. From Voronoi, we gain the leverage to build districts from centroid points, which greatly simplifies logistics while also yielding somewhat sophisticated designs. Furthermore, our core strategy for gerrymandering blossoms from the ease of Voronoi:

- Generate a large number of variant random Voronoi centroids and keep the set that delivers the best result for the party in our favor.

Our strategy critically depends on our ability to compute seed points with as much

variation as possible while still yielding valid districts. To build Voronoi districts within 10% of the mean population, we employ balanced k-means clustering–cutting short before getting equally balanced and less gerrymandered results. When faced with maps in which balanced k-means clusters are too arduous to compute, we can also resort to top down triangular district building as an alternative. We can first split the map into three regions of roughly equal population with three triangles, then recurse down to splitting those triangles and iterate until we have the desired number of districts. We lose the ability to generate variant solutions of which to choose the most gerrymandered; however, we nearly guarantee valid districts. Furthermore, the centroids of these triangles give us yet another initialization for Voronoi.

In the process of narrowing our focus to the above, we also explored many other ideas for computing Voronoi seeds which will be discussed in the report.

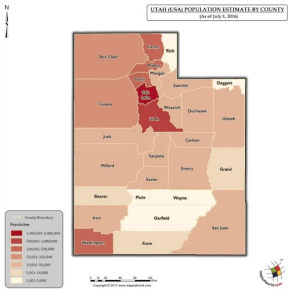
We reserve the result summary for the letter to Threeland.

2 Problem Analysis

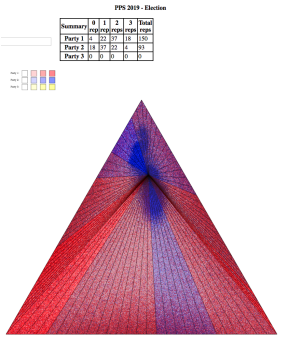
2.1 Motivation

One of the first maps explored for gaining insight into gerrymandering is a high-level representation of Utah, which was chosen for its notoriously red-favoring four districts that "crack" apart the high-density cluster of relatively liberal-leaning citizens in Salt Lake City and neighboring counties. "Cracking" is one of two of the most popular methods for gerrymandering (the other being "packing"), in which an opposing party's population is broken off and shared among the other party, diluting the opposing party's voting strength.

Within the simplified map, the high-density blue cluster towards the top of the triangle consists of five sub-clusters representing Salt Lake City, as well as high-density portions of Cache, Weber, Davis, and Utah County. The surrounding, less dense area consists of a mostly red, rural population. To further simplify the model, red and blue-leaning populations each share half of the total population. In the spirit of Utah's real district division, we have centered our district division at the heart of Salt Lake City, with 27 triangles jutting out to equally spaced vertices on each of the three sides. Below is the result for a two-party, three-representative election, resulting in red gaining roughly 62 percent of the 243 representatives:



(a) True Utah districts, mapsoftheworld.com



(b) our initial "hand-drawn" districts

It is worth noting how gerrymandered the above "results" map appears with its unnaturally long, needle-like districts. The word "gerrymander" itself originates from a case in 1812, where the governor of Massachusetts Elbridge Gerry was accused of flagrant abuses of redistricting power that resulted in a salamander-shaped district heavily favoring one party [2]. The propensity for gerrymandered districts to take on shapes that defy natural bounds begs the question: "is gerrymandering possible with districts roughly uniform in shape?"

Exploring a map like the one above opened our eyes to the challenges of satisfying the constraint of keeping each district's population within 10 percent of the mean district population. The difficulty in heeding this requirement becomes especially evident when managing a space such as this, where nearly half of the population is clustered in dense cities.

2.2 Identifying Gerrymandering: Metrics

Before we delve into our techniques for creating districts in detail, let's first think more about gerrymandering principals by discussing metrics. In practice, courts actually rely on a set of semi-objective methods to determine whether or not gerrymandering has taken place. Although we simplified our analysis to focusing on the "wasted vote" and used the proportional popular vote as a baseline, considering these metrics influenced the way we thought about the larger scope of the project.

2.2.1 Efficiency Gap

Perhaps the most widely-circulated method to detect gerrymandering is the efficiency gap.

[2] etymonline.com

The efficiency gap is an extension of "The Wasted Vote." Consider the usual "winner-takes-all" two-party scenario: any winning party vote over the 50% threshold is a wasted vote, and any losing party vote under the 50% threshold is a wasted vote. Any vote has utility *if and only if* it is part of the initial 50% for the winner, while all of the losing party's votes have no value.

The efficiency gap is calculated as follows: take the difference between each party's wasted votes and divide it by the total votes cast in the election. Political scientists have determined that once the efficiency gap breaches a 7% threshold, one party will have certain geographic advantage over the other.

For Threeland's three-representative plan, we must use a different method for calculating the wasted vote. Note that for this calculation we do not consider any advertising campaign boosts.

Each district offers at most 3 representatives, and each representative "costs" 1/4 of the district's total population. Thus, the wasted vote for one district can be calculated as follows:

```
p = district population
v = party's  votes
if v >= (3/4)*p:
     return v % (3/4)p
else:
     return v % (1/4)p
```

Because the 3-party wasted vote plan has a maximum loss rate of (1/4)*p per district, the overall efficiency gaps will be lower than the 1-party loss rate, which has a cap at 1/2*p per district.

2.2.2 Mean vs Median vote percentages

Compare mean and medium of a party's vote percentages by district; a significant difference can be used as evidence of gerrymandering. Consider a very close election between two parties in which each district is currently split 50% 50%. To gerrymander, a party should reduce 1 district to 0 and lift all others up slightly above 50%. Of course, now their mean is less than their medium.

2.2.3 Compactness: The Reock Score and the Popper-Polsby Test

Here, we define "compactness" as the area / perimeter ratio. A higher ratio would signify more compact districts.

There is a strong case to be made for desiring compact districts and being wary of those that are not. In the real world, communities tend to surround a locus, resulting in compact community shapes rather than elongated ones. A district that has a unique, long, non-compact shape yet purports to represent a single community is odd and may not be faithfully representative.

Political scientists have designed two methods to measure compactness: the Reock Score and the Popper-Polsby Test. To compute the Reock Score, compute the ratio of the district's area to the smallest circle that can be drawn to completely contain it. On the other hand, the Popper-Polsby Test requires you to compute the ratio of the district's area to the area of the circle whose perimeter is equal to the district's area. In both of these cases, the more "excess perimeter" a district has, the less compact it is.

Keeping these facts in mind, in our approach we chose to analyze gerrymandering in compact districts. We feel that a governing body would easily be able to identify an extreme non-compact gerrymander attempt, and that analysis would be more relevant to less blatant gerrymandering. Further, a professional gerrymanderer would know better than to present a very non-compact district layout to Threeland, so we consider what they might try.

2.2.4 Monte Carlo For Objective Results

To get a baseline more sophisticated than the popular vote proportion, one could randomly assign voters to districts, leaving alone geometry constraints. Running many such votes would give a good approximation to "fair" just given raw total preferences. In practice our compute time was limited and we didn't observe much difference, so we stuck with representation proportional to the popular vote as our baseline for "fair".

3 Plan Evolution

Based on the observations and discussion described in previous sections, we decided to focus on methods of district construction that would be conducive to producing more uniform polygons that satisfy the population restrictions as well as the nine-sided limit for districts. Such district shapes would also be easier to analyze with some of the methods discussed in Section 2.2. The evolution of our district builders is briefly described below. Further in-depth explanations of the algorithms can be found in Section 3.

3.1 Voronoi Tessellation

It's a delightful fact of nature that points can define a tessellation of polygons. The mapping is done by defining the sides of the resulting polygons to be equidistant from exactly two seed points. With two seed points we would divide the space with an infinite bisector; for three seeds, that bisector would fork off at the points' center and we'd have a vertex. In fact, all vertices are equidistant from more than 2 seeds.

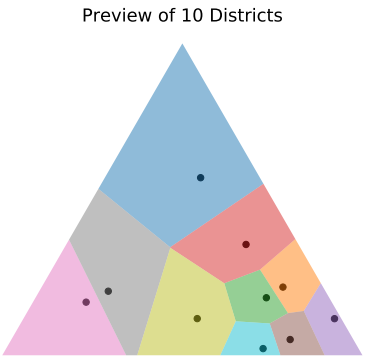


Figure 2: Simple voronoi tessellation on Threeland

Voronoi diagrams were probably first conceived in the 17th century but did not gain traction for engineering applications until the 20th. Applications range from tracking the sources of disease to providing robots with safe paths. Voronoi is a partially applicable to district building because it creates compact polygons that resemble the way populations of voters actually cluster.

We conclude this subsection by enumerating the reasons we grew excited enough about Voronoi to make it the focus of our implementation:

- Creating a tessellation is trivial, so we would be able to create many district configurations and gerrymander by keeping those which deliver the most extreme results to our party of favor. Moreover, by taking the mean result we would have a baseline for "fair" districting to contrast with.
- The resulting polygons tend to not have many sides, which was a restraint burdening the attempting tessellations of many groups. The resulting polygons are compact, which allows Voronoi districts to realistically map to actual population clusters.
- We could make a strong argument to the leaders of Threeland that they should

enforce the use of Voronoi districts to prevent some of the more blatant gerrymander strategies. Thus, our analysis on the extent to which Voronoi districts can still be gerrymandered in the two systems would be of great interest to them, while analysis of other schemes may be rendered irrelevant.

Because a Voronoi tessellation is completely determined by the seeds, we expand the discussion by turning to them.

3.2 Voronoi Centroids

The variability in the districts generated using Voronoi, central to our strategy, stems from the method of generating the seed centroids. For starters, a popular algorithm for creating centroids is k-Means, which works in the following way:

1. k centroids [3] are randomly initialized
2. points are assigned to their closest centroid using a squared-distance measurement
3. each centroid location is recalculated based on the new mean of their cluster
4. the cluster populations and centroid locations are updated back and forth until some measure of convergence is achieved

An interesting exploration in creating wasted votes involves creating cluster centers influenced only by the density of a subset of the population preferring the opposing party. However, since restrictions are placed on how high or low a single district's population is allowed to be [4], Voronoi districts with a valid population become even harder to achieve.

3.3 Complications

One drawback to k-Means even in its most basic "vanilla" form is that less uniform populations will certainly fail to provide districts close to uniform in population count. This problem is likely to occur because k-Means alone does not have a limit on how many or how few points are allowed in each cluster.

3.4 Solutions

One strategy is to break 3-land into nine smaller equilateral triangles, measure the percentage of the total population enclosed in each triangle, and then run k-Means for each triangle separately with k set proportional to the population of each piece.

[3] in our case, 81 or 243

[4] 235 - 1508 for 243 districts, and 3704 - 4526 for 81

Further building on the above idea, a top-down approach to breaking up 3-land into triangles works very well to balance the districts. The method works by recursively breaking a triangle in three separate triangles of approximately equal population until the appropriate number of districts is achieved. Then, the centroid at the center of each triangle generated can be used to create a Voronoi polygon.

It is worth noting that even though top-down implementation without Voronoi does build districts that satisfy population and polygon side constraints, Voronoi districts from the centroids of those triangles, however, does not. To see why, consider the following image:

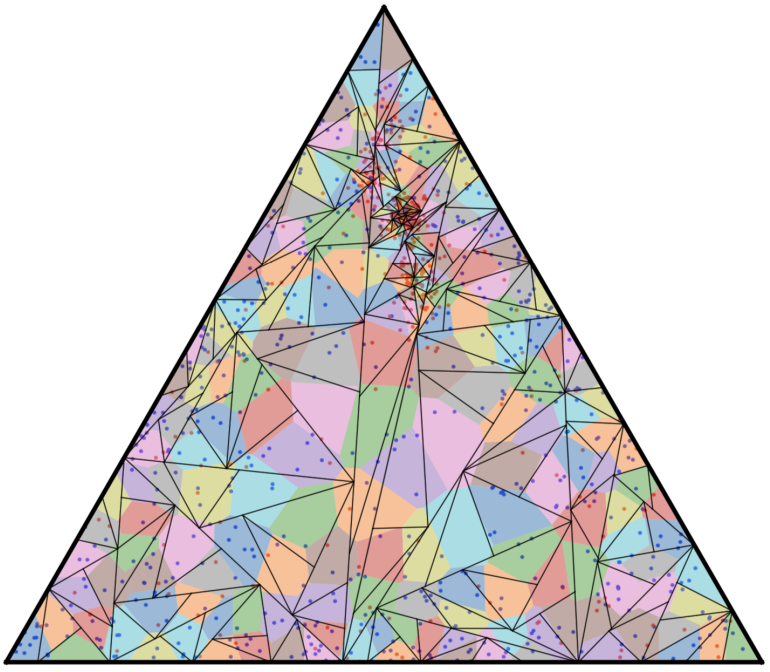


Figure 3: Voronoi districts do not conform to triangles!

This is all to beg the question: what is the solution to districts that break the mean population constraints? A k-Means centroid generator that is capable of supplying centroids for equally-sized clusters *would* actually solve this problem, although in exponential time. [5]. This was the solution we ultimately stuck with and base much of our advice to Threeland on.

One other idea we had was a centroid "shifter" algorithm, capable of moving centroids with too-low populations towards those with populations that are too high.

[5]https://www.brown.edu/news/2017-11-07/redistricting

This actually worked, but only to an extent.

4 Implementation

The following algorithms are used in the construction of various districts for analyzing one and three-party gerrymandering susceptibility.

4.1 Voronoi

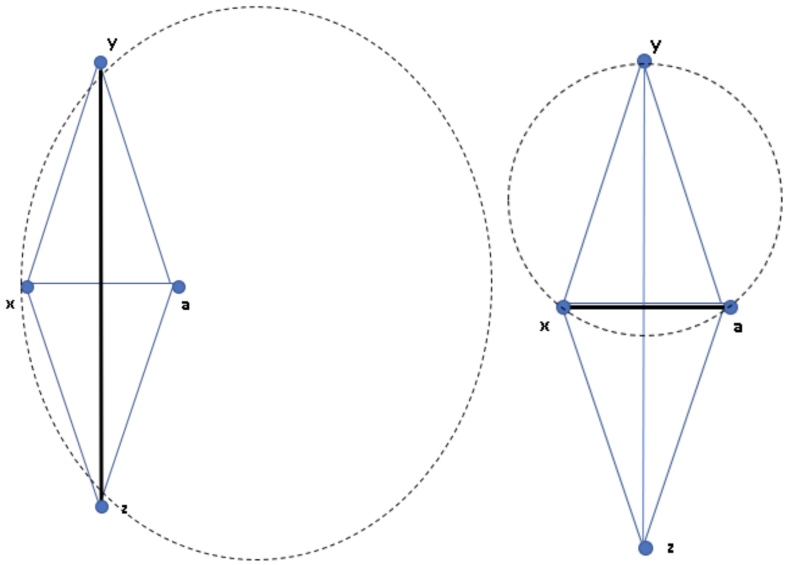
To create the Voronoi, first recognize that each seed point will have a vertex set associated with it; these are the vertices that will create the polygon for that seed. Now, every vertex in that set must be equidistant from more than two seed points; recall that the points equidistant from exactly two seeds make up the sides of the polygons. So a brute force approach would first draw all the bisector lines between seeds, and then consider all intersections of these bisectors as a possible vertex. If an intersection is equidistant from more than two seeds, add it to the vertex set of each of those seeds. After collecting the vertex set for each seed, drawing the polygons is trivial.

To extend this intuition and make the algorithm more efficient, note that the Voronoi Diagram is the dual of the Delaunay Triangulation. Thus, an interesting (and Threeland friendly) way to calculate the Voronoi is to first find the Delaunay Triangulation of the seeds.

First, recall the Delaunay Triangulation: a triangulation is a Delaunay Triangulation if no vertex lies within any triangle's circumcircle.

A simple Delaunay Triangulation algorithm hinges upon the idea of a "locally Delaunay edge".

Consider two adjacent triangles xyz and ayz. Let yz be the edge in question. Construct a circle that passes through x, y, and z; if a lies within that circle, the edge yz is not a locally Delaunay edge.



"Flipping" a nonDelaunay edge for a Delaunay edge

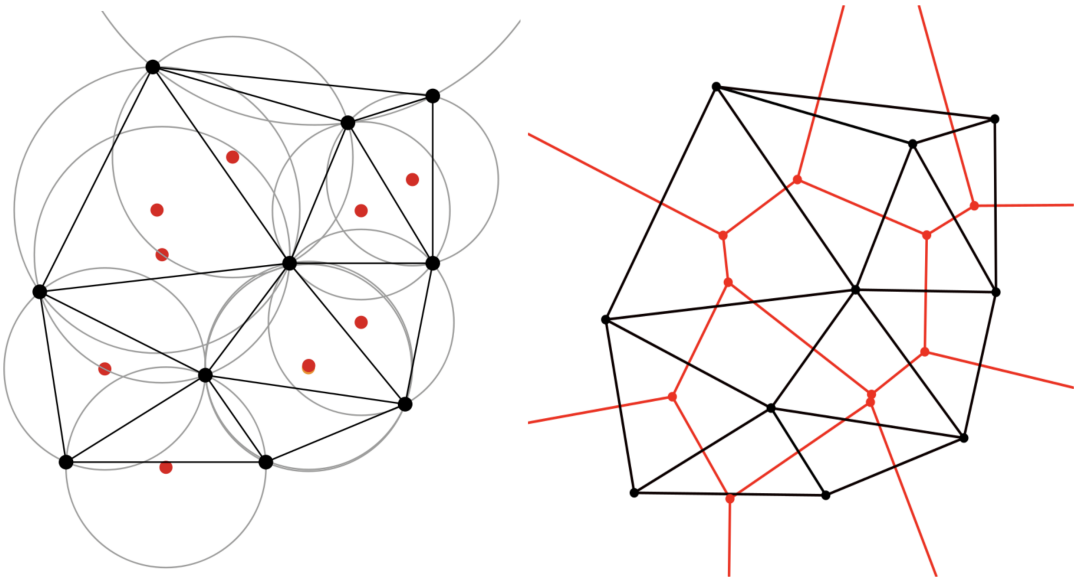
Given the seed points, a Delaunay Triangulation algorithm is as follows, with the points references from above:

```
Create any triangulation using the centroids as vertices of these triangles;
Push every edge onto a stack and mark each edge;
While len(stack) > 0:
      yz = stack.pop()
      unmark yz
      if yz is not a local Delaunay edge (because it leaves the circle:
            replace yz with ax
            for ab in {xy, ya, az, zx}:
                  if ab is unmarked:
                        mark ab
                        stack.push(ab)
```

This technique is known as "flipping", and it's runtime is $O(N^2)$.

Once you have the Delaunay Triangulation, the conversion to the Voronoi Diagram is simple.

For each triangle, compute the circumcenter. Create edges connecting the circumcenter of all triangles that share an edge. The circumcenters will be vertices of the Voronoi polygons and the new edges will be the sides of the Voronoi polygons.



A Delaunay Triangulation and its respective Voronoi Diagram

4.2 Vanilla K-Means

The simplest implementation of k-Means used works in the following way:

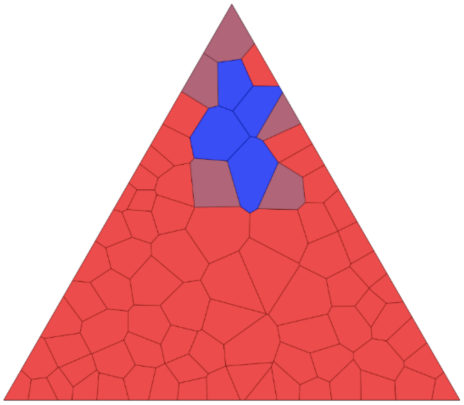
1. Initialize k random centroid coordinates
2. Measure the squared distance between every point and every centroid, assigning points to their nearest centroid
3. Update each centroid's location to be the mean of their respective cluster
4. Repeat steps 2 and 3 until lowest variance within each cluster achieved

Due to the randomness in the initialization, different centroids can result. However, with Vanilla k-Means alone, resulting district maps vary roughly 30 percent from the mean population on average. More advanced "equal" or "balanced" k-Means implementations are able to result in valid maps.

4.3 Party Preference k-Means

In order to create more gerrymandered results under the constraints of Voronoi, centroid coordinates can be generated focusing on sub-populations preferring a specific party, heightening the chance that the resulting districts will experience wasted votes and cracking.

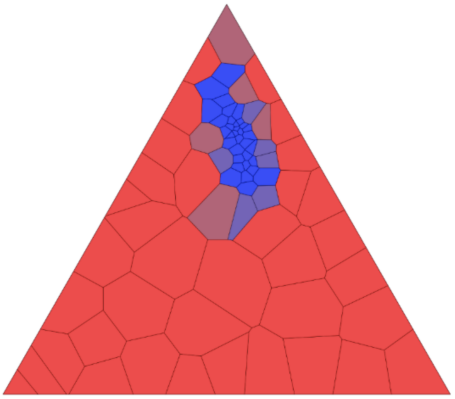
Below are results for the 2-party Utah map's 3-representative election using k-Means centroids generated by looking solely at the Party 1 (red) population, and k-Means centroids generated by looking solely at the Party 2 (blue) population:



Summary	0 rep	1 rep	2 reps	3 reps	Total reps
Party 1	4	0	5	72	226
Party 2	72	5	0	4	17
Party 3	0	0	0	0	0

NORMALIZED

219.7
23.3



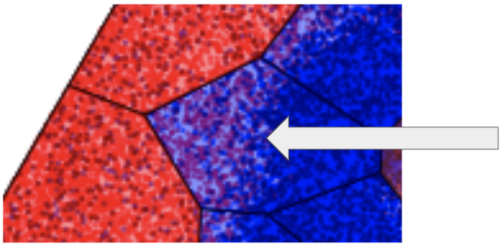
Summary	0 rep	1 rep	2 reps	3 reps	Total reps
Party 1	34	6	5	36	124
Party 2	36	5	6	34	119
Party 3	0	0	0	0	0

NORMALIZED

103.4
139.6

In the above map on the left, where centroids are set by looking solely at Party 1's (Red's) population, big swaths of Party 2's population are being rounded up into single districts, wasting their votes. Similarly, when looking solely at Party 2's (Blue's) population, the tables are turned as more of Party 1's voters become rounded up and wasted into single districts.

Below is a close-up of the second image (with districts influenced solely by the Party 2 population), focusing in on the second dark blue district down. In this region, blue is "cracking" into a portion of red's votes, but swallowing just under the threshold that red would need to gain a representative:



Note that the above districts neglect the population limit constraint in order to show the effects of this method more drastically, as they are unaffected by equalizing methods that would help combat these effects in this form.

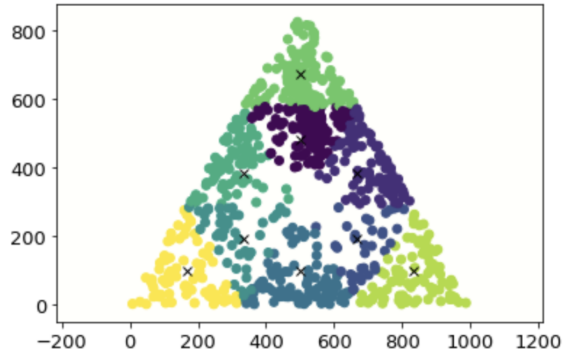
4.4 k-Means inside Nine Equilateral subset Triangles

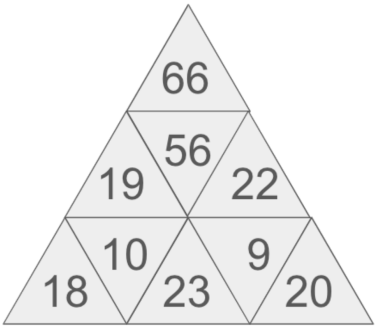
In order to create centroids that are more likely to create clusters more equal in size, 3-land can first be broken into smaller sub-lands, and then the number of k allocated to that particular sub-land is determined by the percentage of the total population residing in that area.

Our implementation of this method involves first splitting up 3-land into nine sub-populations within nine equilateral sub-lands.

The image below to the left shows the population of the Utah map (thinned to 1,000 to make the image easier to digest). The new sub-populations are represented by different colors. To the right are the number of centroids that will be allocated to each sub-population for 243 districts:

```
Pops_Nine = get_nine_pops(thinned_Population_coords)
```





An efficient way to allocate sub-groups to regions is by finding the centers (represented as x's in the above left image) of each equilateral triangle and then measure the distances of 3-land's whole population to each center. Then, assign voters to the sub-region whose center is closest.

Once sub-region populations are determined, the percentage of k district centroids for that region can be calculated by using only a fraction of total k equal to the fraction of the total population present in that area. Then, k-Means on each sub-population with its respective fraction of k can be used to find more suitable centroid locations.

4.5 Balanced K-Means

We now turn to the strategy that ended up working the best for building more equal districts while still gerrymandering: the more sophisticated "Equal" k-Means algorithm.

The same approach is used by researchers at Brown, and is described in the following article: https://www.brown.edu/news/2017-11-07/redistricting

Equal k-Means differs from vanilla versions by assigning a desired number of voters (total number of voters divided by the number of districts) to each center, as opposed to all the closest voters to each of their respective centroids.

The resulting clusters are roughly equal in size, as illustrated in the following image:

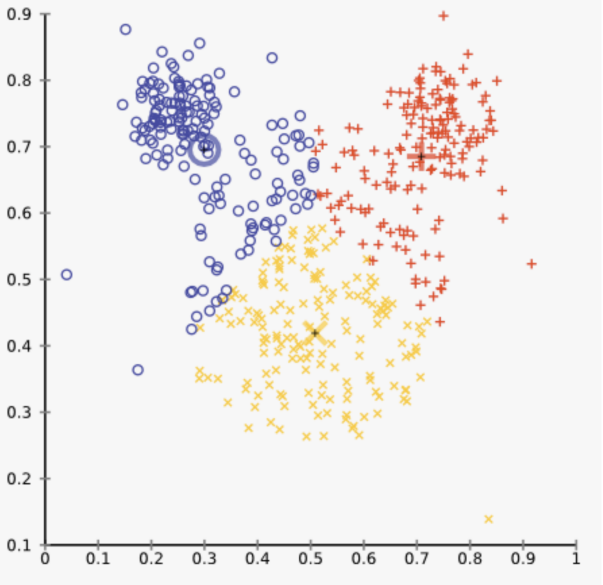


Figure 4: credit to: https://elki-project.github.io

Note how in the above image blue and red's Voronoi boundaries creep beyond their respective clusters into what would otherwise be yellow with a vanilla k-Means implementation. This picture illustrates why balanced k-means works so harmoniously with Voronoi to meet our district constraints. Although the combination of these two algorithms winded up being a complete solution for us, we also developed a way of shifting centroids when in a pinch to validate district population constraints.

4.6 Centroid Shift

To reduce the amount of districts that fall below and exceed the population thresholds, we iteratively take the optimal low-density to high-density centroid neighbors, move the low towards the high and recalculate Voronoi. The low-density centroid absorbs some of its neighbors former population, thereby slightly evening the district population assignments.

4.6.1 Finding the optimal shift

Two districts neighbor each other if and only if their boundaries share at least two vertices. We iterate through each underpopulated district and determine its best shift by comparing it to its neighbors– whichever neighbor has the highest density is the best direction for this district's centroid shift. The direction is calculated by taking a difference vector between the two centroids. Ultimately, we take the shifts moving across the entire map's highest density gradients.

4.6.2 Complexity

Determining each low-density district's neighbors takes on average O(D*N*V), where D is the low-density districts, N is the total amount of districts, and V is the amount of Vertices in D. This is achievable with Python's set in operation, which has amortized O(1) complexity.

Since each district's population is precomputed, and since each district has a maximum of 9 neighbors, determining the neighbor with the highest density is constant-time.

Thus, the overall runtime is O(D*N*V). Of course, if we want to iterate and do more than one shift, which we do, then we must again recompute district populations.

4.7 Top Down Triangle Splitting

The top-down method is simple but effective and helps us achieve districts with near equal populations on maps that thwart our other ideas because of extreme variation in densities. Given a triangle, split it into three triangles such that each triangle has as equal the amount of voters as possible. Repeat this method until you have a total of 81 or 243 districts.

This method can be separated into two processes: root vertex calculation and splitting.

Root Vertex Calculation

Recall we want to design compact districts. Thus, when splitting a triangle into smaller triangles, we want to split it such that the resulting triangles have the largest area to perimeter ratio as possible. To do this, we split the original triangle's longest edge. The new border line extends from the longest edge and into the "root vertex". This root vertex is essential in the splitting process, because it is one of the two constant vertices throughout the calculation.

Splitting:

Every triangle goes through a splitting process twice. The first split cuts it into a 1/3 population triangle and a 2/3 population triangle, and the second split cuts the 2/3 population triangle into two 1/3 population triangles.

The main driver in the splitting process is the binary search. We modified the binary search such that in each round it calculates the population of the leftmost triangle it would spawn.

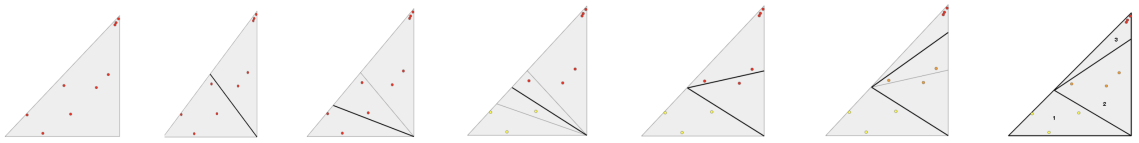
The binary search code:

```
binary_search_tri_split (vertex A, vertex B, vertex root, integer
target_population):

      lower, upper = A, B
      mid = None
      floor = target_population
      while lower <= upper:
            mid = (lower.x + upper.x)/2, (lower.y + upper.y)/2
            if population(A, mid, root) == target_population:
                  return mid
            elif population(A, mid, root) < floor:
                  lower = 1.05*mid.x, 1.05*mid.y
            else:
                  upper = .95*mid.x, .95*mid.y

      return mid
```

The binary search method works because the population is sorted; as the "splitting border" goes up from the bottom edge to the top edge, it continuously encapsulates more voters, and vice versa. Naturally, a perfect cut cannot happen every time; in those cases, we simply have the binary search return the closest cut. However, because the district constraint allows us to be as far as mean+-10%, having slightly

less than 100% precision is not an issue. Indeed, when we use this method, 100% of our districts' populations lie within the thresholds.

See the following figures for a visualization of the process.



4.7.1 Complexity Analysis

Let N be the length of the longest edge in the triangle. Each run of binary search has a runtime of O(logN). Each step yields three subproblems, each of size N/3.

This gives us a recurrence equation of $T(N) = 3T(N/3) + \log N$, which, by Master's Theorem, yields a runtime of O(N).

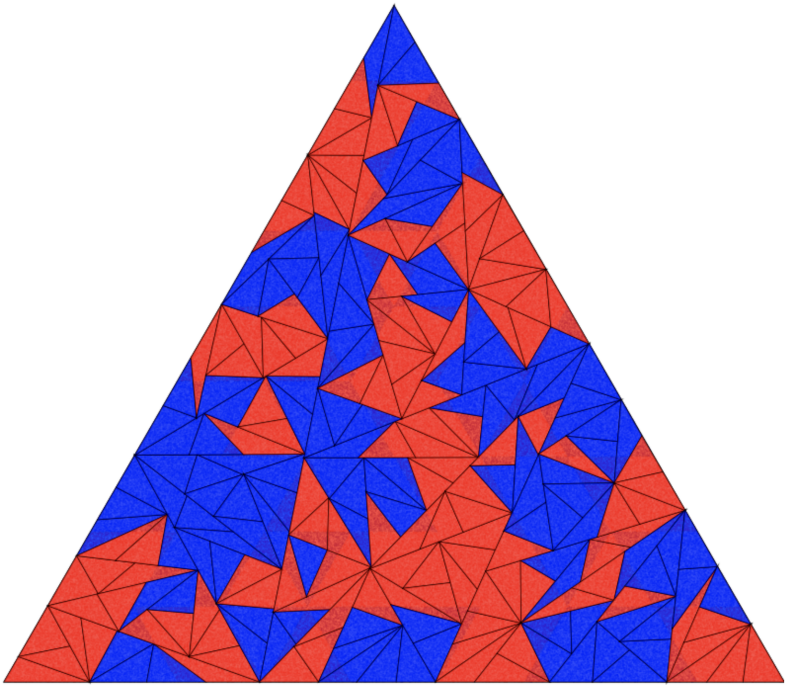


Figure 5: Top down approach with 243 districts

5 Data and Results Analysis

Balanced K-Means: Tournament (Popular Vote: 123 vs 120)

Representatives	*favorite*	*P1 vote*	*P2 Vote*	*P1 Wasted Vote*	*P2 Wasted Vote*
1	Party 1	125	118	82153	84513
1	Party 2	115	128	88307	78359
3	Party 1	123	120	42003	41330
3	Party 2	121	122	44378	38955

Let's begin by analyzing our balanced k-means results for the tournament map, where the popular vote is about even. Recall that we ran the algorithm 20 times and kept the best resulting districts for party 1 and party 2 in order to attempt a gerrymander. We can see evidence that the 3 rep system is less prone to gerrymandering as in those rows we were not able to bring party 1 or 2 significantly above their popular vote proportional representation. In contrast, looking at 1 rep system we see about twice the number of wasted votes, moreover we were able to give a clear win to either party 1 and 2, depending on our favor. When we favor party 1, the wasted votes of party 2 increases and vice versa. These results may be even more stark when we consider that the majority party may gain advantage in the house disproportional to their majority. That is to say, bringing party 2 up from 120 votes to 128 could totally transform the power dynamic in Threeland.

Balanced K-Means: Utah (Popular Vote: 142 vs 101)

Representatives	*favorite*	*P1 vote*	*P2 Vote*	*P1 Wasted Vote*	*P2 Wasted Vote*
1	Party 1	149	94	79334	87333
1	Party 2	148	95	27358	55974
3	Party 1	151	92	26245	57088
3	Party 2	150	93	27359	55975

Now considering the balanced k-means strategy in Utah, the result is not consistent. In this case the 1 rep system was closer to the popular vote; however, the difference between 1 and 3 reps is also less drastic. Also, we note that, still, the 1 rep system led to far more wasted votes on average. Although in this case they were wasted for both sides, we can imagine a gerrymandering scheme that wastes votes less even-handedly. Ultimately we get less randomness in the Utah result to exploit because Utah has lower entropy than Tournament. It's a more ordered map with 1 dense

city population (party 2), and we get approximately half the centroids in that city every time. By contrast, the tournament map has mild concentrations of party 1 and party 2 adjacent across Threeland, with lots of opportunity to break the parties across different fractions of districts.

Top Down: Tournament (Popular Vote: 123 vs 120)

Representatives	*P1 vote*	*P2 Vote*
1	122	121
3	123	120

For the top down triangle building, we did not implement any significant variance in the process and so were not able to use the strategy of running the algorithm many times to favor a party; therefor even with 1 rep we still were unable to break the symmetry of the map.

Top Down Utah: (Popular Vote: 142 vs 101)

Representatives	*P1 vote*	*P2 Vote*
1	156	87
3	153	90

Now we run top down on a map that's less symmetric. As with balanced k-means, we observe that 3 reps is closer to the popular vote than 1 rep.

Top Down Seeded Voronoi: Tournament (Popular Vote: 123 vs 120)

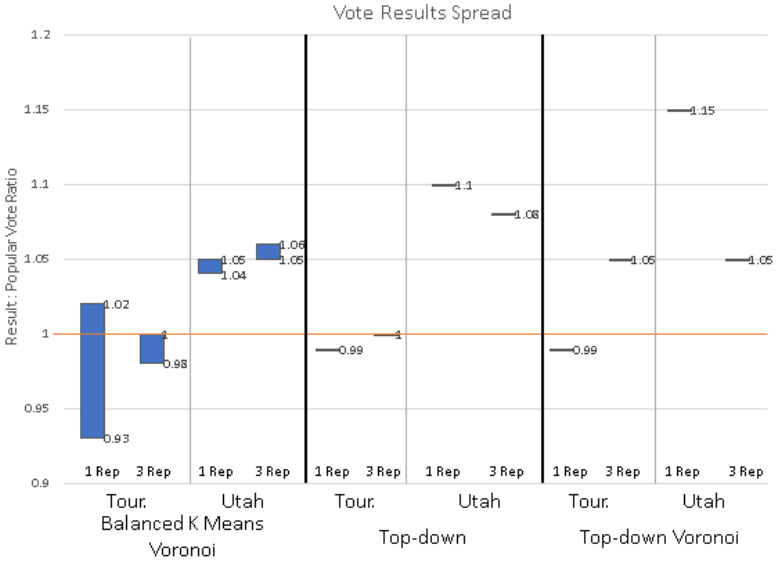
Representatives	*P1 vote*	*P2 Vote*
1	122	121
3	129	114

Top Down Seeded Voronoi: Utah (Popular Vote: 123 vs 120)

Representatives	*P1 vote*	*P2 Vote*
1	163	80
3	149	94

For Voronoi seeded with the centroids of the top down districts, the results of the two maps once again contrast with resepect to 3 reps being more true to the popular preferences. It's likely that some of these districts were more than 10 % away from the mean population as discussed earlier, so it may not be wise to not further analyze the results of this process.

We move onto a summary visual of the charts discussed so far:



Each box represents the range of representatives Party 1 won in proportion to the popular vote throughout our simulations. The y=1 line marks an outcome that would match popular representation. For example, if a popular vote would have awarded Party 1 123/243 representatives, but in the simulation Party 1 actually won 115, then its score would be 115/123, or .93. So, out of our presented cases, only the Top-down method when applied to the Tournament map yielded an outcome totally inline with the popular proportion.

Again, there is a much wider spread in the Balanced K Means Voronoi method compared to the Top-down and Top-down Voronoi approaches. This is due to the randomness that occurs when seeding the clustering algorithm as compared to the deterministic Top-down and Top-down Voronoi methods.

This chart also illustrates another relevant point: the 1 Representative method is more susceptible to straying from the fair result than the 3 Representative, even when no intentional Gerrymandering is in force. Consider the distances from "fair" both

scenarios result in: the worst-case 1 Representative is on average 6% off, whereas the worst-case 3 Representative is on average 3% off. Although this might not sound like much, it makes a difference when applied: 6% of 243 representatives is roughly 14 representatives; 3% is roughly 7.

However, averages of distances alone do not fully express the advantages of the 3 Representative format. Consider the spread of the non-deterministic methods. The 1 Representative formats have an average spread of .05 compared to the 3 Representative .015 spread average. This means that the 3 Representative format is nearly 4 times as predictable than its 1 Representative counterpart. This tighter window of certainly makes it harder for a gerrymanderer to make any significant strides while hiding under the guise of fairness.

5.1 Targeting clusters to sub-populations

In order to create more gerrymandered results under the constraints of Voronoi, centroid coordinates can be generated focusing on sub-populations preferring a specific party, heightening the chance that the resulting districts will experience wasted votes and cracking. In order to explore the effects of this method, we set up elections for five different settings of weighted party populations. Note that the districts built for this particular test neglect the population limit constraint in order to show the effects of this method more drastically (equalizing methods would help combat these effects). Also note that we focus on the tournament map and Utah map, as they show the different effects on a population that is more uniform versus one that is densely packed for one party but more sparse and rural for another:

PARTY 1: 0, PARTY 2: 100%

	TOURNAMENT MAP RESULTS			UTAH MAP RESULT		
Popular vote:	**red to blue ratio: 123, 120**			**red to blue ratio: 142, 101**		
	UNNORMALIZED		**NORMALIZED**	**UNNORMALIZED**		**NORMALIZED**
reps = 1	PARTY 1:	88	86.9	PARTY 1:	85	67.8
	PARTY 2:	155	**156.9**	PARTY 2:	156	**175.2**
reps = 3	PARTY 1:	103	101.7	PARTY 1:	124	103.4
	PARTY 2:	140	**141.8**	PARTY 2:	119	**139.6**

PARTY 1: 25%, PARTY 2: 75%

	TOURNAMENT MAP RESULTS			UTAH MAP RESULT		
Popular vote:	**red to blue ratio: 123, 120**			**red to blue ratio: 142, 101**		
	UNNORMALIZED		**NORMALIZED**	**UNNORMALIZED**		**NORMALIZED**
reps = 1	PARTY 1:	72	71.1	PARTY 1:	126	105.4
	PARTY 2:	171	**173.1**	PARTY 2:	117	**137.6**
reps = 3	PARTY 1:	95	93.8	PARTY 1:	154	**134.1**
	PARTY 2:	148	**149.9**	PARTY 2:	89	108.9

PARTY 1: 50%, PARTY 2: 50%

Popular vote:	TOURNAMENT MAP RESULTS red to blue ratio: 123, 120			UTAH MAP RESULT red to blue ratio: 142, 101		
	UNNORMALIZED		NORMALIZED	UNNORMALIZED		NORMALIZED
reps = 1	PARTY 1:	122	120.5	PARTY 1:	160	**140.5**
	PARTY 2:	121	**122.5**	PARTY 2:	83	102.5
reps = 3	PARTY 1:	126	**124.5**	PARTY 1:	181	**164.0**
	PARTY 2:	117	118.5	PARTY 2:	62	79.0

PARTY 1: 75%, PARTY 2: 25%

Popular vote:	TOURNAMENT MAP RESULTS red to blue ratio: 123, 120			UTAH MAP RESULT red to blue ratio: 142, 101		
	UNNORMALIZED		NORMALIZED	UNNORMALIZED		NORMALIZED
reps = 1	PARTY 1:	163	**161.5**	PARTY 1:	200	**186.6**
	PARTY 2:	80	81.5	PARTY 2:	43	56.4
reps = 3	PARTY 1:	146	**144.5**	PARTY 1:	209	**187.8**
	PARTY 2:	97	98.5	PARTY 2:	34	45.2

PARTY 1: 100%, PARTY 2: 0%

Popular vote:	TOURNAMENT MAP RESULTS red to blue ratio: 123, 120			UTAH MAP RESULT red to blue ratio: 142, 101		
	UNNORMALIZED		NORMALIZED	UNNORMALIZED		NORMALIZED
reps = 1	PARTY 1:	160	**158**	PARTY 1:	225	**218.4**
	PARTY 2:	83	85	PARTY 2:	18	24.6
reps = 3	PARTY 1:	103	101.4	PARTY 1:	226	**219.7**
	PARTY 2:	140	**141.6**	PARTY 2:	17	23.3

In the above results, 1-representative cases gerrymander far worse than 3-representative cases overall. Also note that the rural Party 1 in the Utah map is given more opportunities to gerrymander than Party 2.

6 Conclusion

We gerrymandered by creating many randomized district configurations and cherry picking the ones that favored our party of choice. We were able to get semi-balanced districts through a variation of k-means and tessellate the space with Voronoi diagrams. Based on the results of our data analysis for 2 parties, we conclude that 3-representative systems are generally less prone to gerrymandering than 1-representative systems. It's harder to waste as many votes when even a 26% minority would be enough to win a party some representation. Furthermore, we conclude that district boundaries restricted to more uniform - and thus less gerrymandered-looking - shapes are still susceptible to gerrymandering, even if less so, as described in our analysis.

7 Ackowledgements

We enjoyed how this project involves a real practice in U.S. politics that deserves attention, and we appreciate Professor Ken Ross' care in coming up the 3-land concept - it helped us understand gerrymandering on a much deeper level. We would also like to acknowledge our teaching assistant Chengyu Lin for his commitment to the students in our class. Thank you to Group 3 and 6 for bringing up the idea of top-down triangle splitting. We would also like to thank Group 6 for providing base functions to run the vote in python.

We would like to acknowledge the following resources from which we repurposed code and ideas for implementing some of our strategies:

We initially got the idea for Voronoi districts from: https://www.brown.edu/news/2017-11-07/redistricting

For building Voronoi from seeds, we were greatly aided by code found at:

https://stackoverflow.com/questions/20515554/ colorize-voronoi-diagram/2067864720678647

https://en.wikipedia.org/wiki/Delaunay_triangulation

https://www2.cs.duke.edu/courses/fall08/cps230/Lectures/L-21.pdf
Equal k-Means: https://github.com/ndanielsen/Same-Size-K-Means
http://gerrymander.princeton.edu

8 Team member contributions

Rebecca Calinsky: Implemented all k-Means strategies; contributed to map design and relevant distribution code, data analysis, and brainstorming for strategies; Took on the responsibility of ensuring that deliverable deadlines were met.

John Daciuk: Proposed Voronoi, Top-Down and centroid shifting strategies and contributed to the code to get them all running; integrated team code into a master file and generally worked to ensure our code pieces worked as parts of larger functions; set up data-collection pipeline and ran result analysis; contributed to map design and relevant distribution code, data analysis, and researching various gerrymandering techniques and metrics.

Kelley Valentine: Implemented the core processes of the Centroid Shifter and the Top-Down strategies; proposed the Utah map; contributed to data analysis; brain-

stormed for strategies. Took on the responsibility of researching various methods for measuring gerrymandering.

All team members contributed extensively to the report and presentation.