Repairing Redistricting:
Using an Integer Linear Programming Model to Optimize Fairness in Congressional Districts

By Benjamin Carman
Fall 2019

A Project for MATH 3970T
A Tutorial in Operations Research
Dr. Vardges Melkonian

https://github.com/benjamincarman/repairing-redistricting

Repairing Redistricting: Using an Integer Linear Programming Model to Optimize Fairness in Congressional Districts

***Abstract*—Historically, redistricting has been a process ridden with political manipulation in which politicians "gerrymander" districts to achieve a competitive advantage in future elections. However, the process of redistricting can be left purely up to a mathematical model that prioritizes key characteristics of a fair district. This paper details one such Integer Linear Programming model implemented in AMPL that ensures just that—a fair district. The model produces districts that reflect the political distribution of the state, with no party favored to win more districts than their share of the statewide vote. This model is tested and evaluated on data from the state of Ohio complete with reflections for future improvements and extensions.**

Keywords—redistricting, optimization, integer linear programming, gerrymandering, AMPL

I. Introduction

Redistricting is the process by which new lines are drawn to create districts for the U.S. House of Representatives and state legislatures. This legally required process happens in each state every ten years following the recently completed census [1]. As new data suggests significant population changes, new districts are required to ensure equal representation among states and among districts within the states. Historically, however, redistricting has rarely been a process by which "equal representation" is a priority. In fact, many states have politically biased legislators and state officials draw the maps, making the process a political battle where each party spends millions of dollars to find ways to reach a competitive advantage in the next decade of elections [1]. The method by which parties gain this advantage is known as "gerrymandering". Through various specialized tactics, politicians are able to achieve district maps in which their party is likely to win far greater districts than are proportionally representative of the state's demographic. This ultimately leads to a myriad of issues including disenfranchisement, voter apathy, less competitive elections, and reduced motivation for representatives to engage with constituents. Even more recognizably, this leads to districts that are in many cases strangely shaped or unnecessarily large. Naturally, there have to be much better, well-defined ways to draw congressional districts based on several criteria. This project aims to employ some of these criteria along with a specification for proportional party representation through an integer linear programming (ILP) optimization model. Furthermore, this project tests and evaluates its findings by running its model on data from the state of Ohio—a swing state with a history of relentless gerrymandering.

II. What Makes a "Good" District?

There are several criteria which researchers and mathematicians have defined for the purpose of creating optimal, fair districts. To a certain extent, these are also given as guidelines and/or requirements to states when drawing new lines. The National Conference of State Legislatures gives many of these [2]:

1) *Compactness*

Districts should be in regular geometric shapes and distances between parts of a constituency should be minimized.

2) *Contiguity*

All parts of the district should be connected to all other parts via a path within the district.

3) *Integrity*

Districts should not divide territorial boundaries. Subdivisions like cities, counties, etc. should largely remain intact throughout a single district.

4) *Population*

The population of each district should be roughly the same. Some say 10% deviations between districts are acceptable, however based on many court rulings, states like Colorado require a maximum deviation less than 5% [2].

5) *Competitiveness*

Districts should be relatively balanced to avoid "safety" for one political party. This encourages representatives to be as engaging and fair to their constituencies as possible. It also encourages greater voter engagement in elections.

This project prioritizes one additional quality of "good" districts to ensure fairness politically throughout a state:

6) *Proportionality*

The number of districts in a state that one political party is likely to win should be representative of the state demographic as a whole. This criterion of political fairness is one slowly being adopted nationally and is beginning to be made a greater priority in Ohio for the 2020 districts after the passage of Issue 1 regarding congressional redistricting reform [3]. Naturally, there could be several ways to define and achieve this requirement. The model described in this paper considers this requirement to be achieved when the proportion of voters statewide that favor one party is the same as the proportion of districts in which that party is favored to win.

III. An Integer Linear Programming Optimization Model

In order to achieve optimized districts for so many competing district qualities, it is best done mathematically. An Integer Linear Programming (ILP) optimization model can be implemented to formalize and optimize this process. This project implements such a model using the modeling language AMPL and begins by defining some data regarding a state.

A. *Data and Variable Formulation*

1) *Formulating Counties and Districts*

In maintaining the integrity of territorial boundaries, this model chooses districts solely from a set of a state's counties without additional territorial divisions beyond the input. First, we define the number and name of counties to be defined and the number of districts to be used in a particular state's model.

```
#DEFINITIONS
param n; #number of counties in the state

param m; #number of electoral districts

set counties; #the set of state counties

set districts := 1..m; #the set of districts
```

2) *Defining Restricted Possibilities*

As the model chooses counties to be in districts, it does not make sense for it to consider every county-district pairing. This would immediately use n*m decision variables, which would be an excessive waste of time and memory when it is impractical to consider counties on opposite sides of the state for the same district. To alleviate this unnecessary complexity, we define a parameter for whether a particular county is allowed to be included in a given district. This parameter is then related to a set for iteration purposes.

```
param p{i in counties, j in districts} binary;
#is 1 if county i can be included in district j

set possiblePairs := {i in counties, j in districts: p[i,j] == 1};
#set of possible county/district pairings
```

3) *Defining County Relationships*

Next, in order to implement constraints that guarantee connectivity and compactness within a district, we must define the distance between pairs of counties. The distance defined by this model is the length of the shortest path between two counties in terms of the number of adjacent counties that must be traveled through to reach the opposite county. If counties are adjacent, this distance is 1. More information on how these distances can be computed based on adjacency data for the counties in a state can be found in Appendix A. We also define a tuning parameter here that expresses the maximum allowable distance between any pair of counties in a chosen district.

```
param distances{i in counties, j in counties} integer;
#distance between counties i and j measured by how many adjacent counties
#away i is from j

param MAXDISTANCE; #maximum allowable distance between two counties in a
                   #district
```

4) *Expressing Demographics*

One of the primary features of this ILP model is its choices of districts that are proportional to the state's political leanings. To achieve this, we need number of pieces of demographic political data regarding the number of democrats, republicans, and total voters by county. We also define a value LARGE that is initialized to the total number of voters in the state. This value acts as a maximum, or relative infinity value, in the context of the model for later logical constraints. Next, the model defines two important target values: the number of ideal Democrat districts (Democrat is chosen arbitrarily as the model assumes a two-party system) and the ideal population of each district. The ideal number of Democratic districts is defined by the proportion of Democratic voters multiplied by the number of districts. The ideal population for a district is defined as the number of statewide voters divided by the number of districts. Finally, we define a tuning parameter for the allowable error of the actual number of Democrat districts from the ideal number of these districts in theory.

```
param d{i in counties}; #the number of Democrat voters in a county

param r{i in counties}; #the number of Republican voters in a county

param v{i in counties} := d[i] + r[i];
#the number of total Democrat and Republican voters in county

param LARGE := sum{i in counties}v[i]; #sum of all voters in state

param idealNumDemocratDistricts := round(((sum{i in counties}d[i]) /
                                    (sum{i in counties}v[i])) * m);
#proportion of democrat voters multiplied by total number of districts

param targetPopulation := round((sum{i in counties}v[i]) / m);

param politicalLeeway;
#a tuning parameter for the allowable error of actual Democrat districts
```

B. *Decision Variables*

In order to constrain and optimize a redistricting model with the previously defined data, we need 3 classes of decision variables. The most important is the decision variable 'c' for every county/district pair, which chooses whether a county is included in a certain district. The next variable is called 'leansDemocrat' and there is one for every district to tell whether a given district's configuration has more Democrat or Republican voters. The last decision variable is 'z' which will be updated to be the objective value our model is minimizing.

```
var leansDemocrat{j in districts} binary;
#is 1 if more Democrats in a district, 0 if not

var c{i in counties, j in districts: p[i,j] == 1} binary;
#is 1 if county i is chosen for district j

var z; #the largest difference from target population among all districts
```

C. *Objective and Objective Constraints*

Currently, the strictest requirement during redistricting is that districts remain near equal. This requirement is one strictly upheld by the courts under constitutional law in the 14th ammendment [2]. For this reason, the model given here will seek to produce a solution with the smallest possible maximum deviation from a district's target population. This is done by minimizing the decision variable 'z' and updating the variable to be the largest population deviation from the target population of all districts. This is done through two constraints, one to account for positive deviations and one for negative deviations.

```
#OBJECTIVE
minimize LargestDifference: z;


#CONSTRAINTS
subject to zIsLargestDifferencePos{j in districts}:
      z >= ((sum{(i,j) in possiblePairs}c[i,j]*v[i]) - targetPopulation);

subject to zIsLargestDifferenceNeg{j in districts}:
      z >= (-1 * ((sum{(i,j) in possiblePairs}c[i,j]*v[i]) –
            targetPopulation));
```

D. *Additional Constraints*

There are several additional logical constraints considered in this model which serve to produce the fairest, most optimal districts possible according to the criteria mentioned in Section II. We now consider each of these constraints, what they achieve, and how they are implemented.

1) *Constraining County-District Relationship*

First and foremost, we need a constraint to ensure that each county is assigned to one district and one district only. Note that we don't explicitly require the converse, that each district is assigned at least one county, because this constraint is captured by the objective which evens out the population distribution among all districts.

```
#CONSTRAINTS
subject to Exactly1DistrictPerCounty{i in counties}:
      sum{(i,j) in possiblePairs}c[i,j]=1;
```

2) *Constraining Political Distribution*

In order to write constraints regarding the number of districts with political leanings toward one party or the other we need to use the previously defined variables 'leansDemocrat'. For these variables to work, they must be kept up to date via two constraints that set this variable accordingly. These constraints are 'setOneIfLeansDemocrat' and 'setZeroIfLeansRepublican'. Note the 'setOneIfLeansDemocrat' constraint works because if there are more democratic voters, the value on the right of the inequality will be positive, requiring the 'leansDemocrat' variable to be 1. In the 'setZeroIfLeansRepublican' case, the right side of the inequality will also be 1, requiring (1 – 'leansDemocrat') to be positive and thus 'leansDemocrat' to be 0.

```
subject to setOneIfLeansDemocrat{j in districts}:
      LARGE * leansDemocrat[j] >=
      ((sum{(i,j) in possiblePairs}c[i,j]*d[i])
      - (sum{(i,j) in possiblePairs}c[i,j]*r[i]));

subject to setZeroIfLeansRepublican{j in districts}:
      LARGE * (1 - leansDemocrat[j]) >=
      (-1 * ((sum{(i,j) in possiblePairs}c[i,j]*d[i]) -
      (sum{(i,j) in possiblePairs}c[i,j]*r[i])));
```

Now with these variables constrained to be up to date, we can require the number of Democrat-leaning districts to be the same as the previously defined ideal plus or minus some predetermined leeway (usually 0 or 1). Thus, we can constrain this number of Democratic districts to be in the range of the ideal number ± leeway via two linear constraints.

```
subject to lessThanMaxDemocratDistricts:
    sum{j in districts}leansDemocrat[j] <= idealNumDemocratDistricts + 1;

subject to greaterThanMinDemocratDistricts:
    sum{j in districts}leansDemocrat[j] >= idealNumDemocratDistricts - 1;
```

3) *Constraining Compactness*

In order to help capture another criterion of good districts, we can constrain for compact districts by setting a maximum distance between any two counties in a particular district. We create this constraint by considering every pair of counties greater than the maximum distance and only allowing one of them to be in a particular district.

```
subject to maxDistance{i in counties, j in counties, k in districts:
                       p[i,k] == 1 and p[j,k] == 1 and
                       distances[i,j] > MAXDISTANCE}:
c[i,k] + c[j,k] <= 1;
```

4) *Constraining Contiguity*

Finally, an important component to this model of choosing counties to be placed in districts is in ensuring that all the counties within a district are connected (i.e. there always exists a path between two counties in a district such that all counties along that path are also in the district). In order to make sure this is always the case, we define a set of ‘bridgeCounties’ over every distance between 2 and ‘MAXDISTANCE’ and all county pairs which are that distance away from each other. These bridge counties are the set of counties that are adjacent to 1 county in the pair and a distance of 1 closer to the other county in the pair. Then, for every pair of counties ‘dist’ distance apart in a district (‘dist’ between 2 and ‘MAXDISTANCE’), we require that at least one bridge county be included in the district as well. By induction, it can be proven that this will guarantee connectedness amongst the counties of a district. It will also aid in our compactness constraint. A detailed proof of the logic behind this constraint as well as stronger intuitive details can be found in Appendix B.

```
set bridgeCounties{i in counties, j in counties, dist in 2..MAXDISTANCE:
             distances[i,j] == dist} :=
{k in counties: (distances[i,k] == 1 and distances[j,k] == dist - 1)
                or (distances[i,k] == dist - 1 and distances[j,k] == 1)};

subject to countyBridge{i in counties, j in counties, l in districts,
                  dist in 2..MAXDISTANCE: distances[i,j] == dist and
                  p[i,l] == 1 and p[j,l] == 1}:
sum{k in bridgeCounties[i,j,dist]: p[k,l] == 1}c[k,l] >=
                                                    c[i,l] + c[j,l] - 1;
```

E. *Data*

This model was tested on Ohio, a swing state with a history of gerrymandering, using relevant data from the 2016 election. This data includes all 88 counties with 3 counties split into smaller "sub-counties". These are Cuyahoga, Franklin, and Hamilton. These counties include Ohio's largest cities Cleveland, Columbus, and Cincinnati, respectively, and are therefore the three largest counties in Ohio. For this reason, the counties were split into smaller sections to allow for greater flexibility in the model's choices for districts. The full data used for the state of Ohio can be found in the project's repository at https://github.com/benjamincarman/repairing-redistricting and included the numbers of Democrat/Republican voters for each county based on how those counties voted in 2016. The parameter for whether a county was able to be included in a given district was defined methodically such that a given county could be included in its current district (from the 2012-2022 map) or any district adjacent to its current district. This still allowed for a great amount of flexibility in the model's options while also ensuring many unnecessary decision variables were not even considered. Finally, data was included giving distances for each county-county pair generated from 2010 census adjacency data (See Appendix A).

F. *Results*

The model was then run for maximum distance values of 4, 5, and 6. Each resulted in an objective of 32,741 meaning the model was always able to find a redistricting solution where the maximum deviation of a district's population from the ideal population of 321,909 was 32,741. This is a deviation of about 10.17% which could likely stand to be improved in future models to meet constitutional and federal law. This could be done by splitting counties up further into smaller functional units thus giving the model more districting options. This could also be achieved by relaxing other constraints, but this may not be ideal.

The test of this model was very successful in that it met the constraint most central to this project—it produced districts with as many favored for each party as should be expected by statewide voter data. Specifically, each model produced 6 Democrat-leaning districts which was within the ±1 range of an expected 7 Democrat-leaning districts. The specific choices for county-district pairs can be seen in the colored Congressional maps in Figure 1 and in the model's output found on the project's Github repository. These maps are in stark contrast to the current Congressional map of Ohio (Figure 2) which has districts that are visibly less compact and politically disproportional.

Not only did the model produce maps meeting the proportionality criteria but it also produced fully connected, compact districts. The proof of why this is guaranteed to be the case can be found in Appendix B.

As far as complexity of the model in terms of runtime, the following was observed for each of the three models' solve times:

	Solve Time (summed over all threads)
Model w/ MAXDISTANCE 4	292.393
Model w/ MAXDISTANCE 5	268.303
Model w/ MAXDISTANCE 6	271.808

In each of these cases, this solve time across all threads translated to only a few seconds when run on the NEOS servers. It was also observed however, that in the case of this model and Ohio dataset, setting a leeway parameter of 0 from the ideal proportionality of Democrat to Republican districts is either infeasible or much more expensive computationally. This is something that could potentially be alleviated when a greater number of districting ensembles are allowed through the use of smaller functional units (e.g. precincts instead of counties).

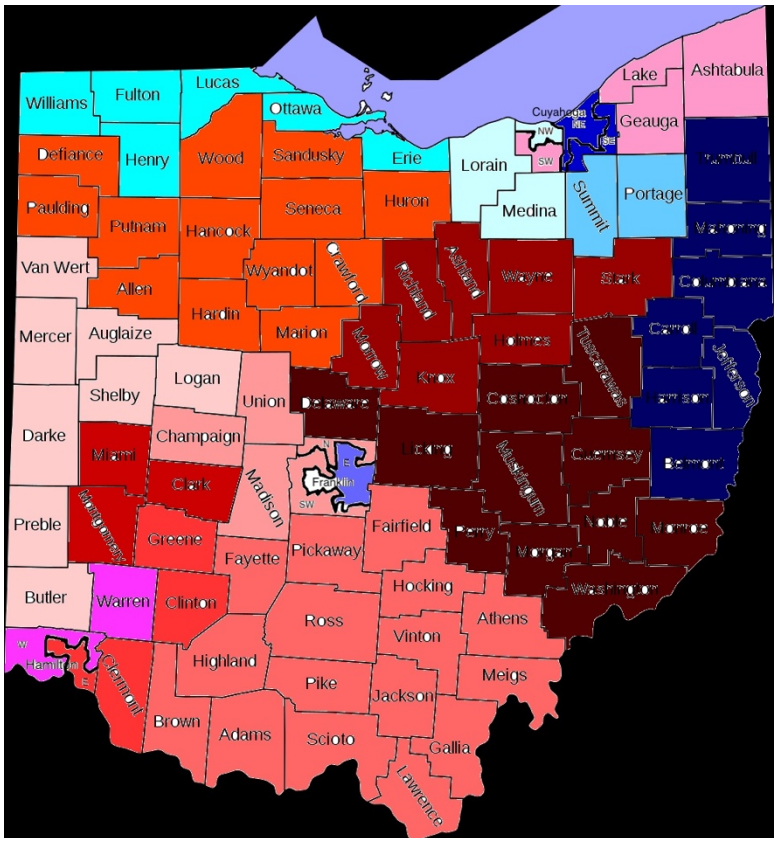


Figure 1a (above). Resulting Congressional map of Ohio of model run with maximum distance of 4 between any two counties in the state

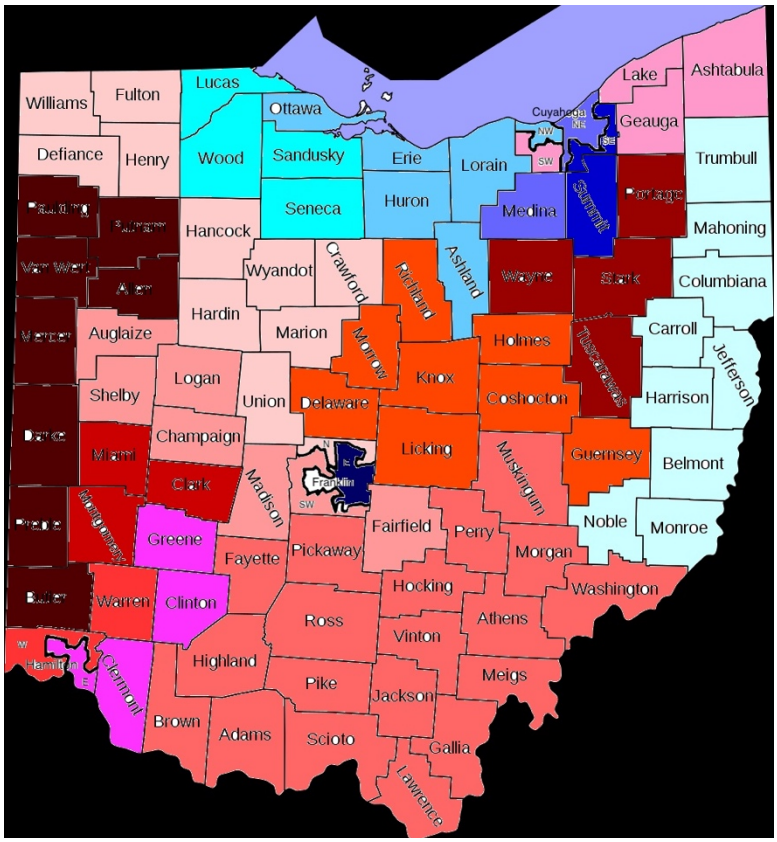


Figure 1b (above). Resulting Congressional map of Ohio of model run with maximum distance of 5 between any two counties in the state

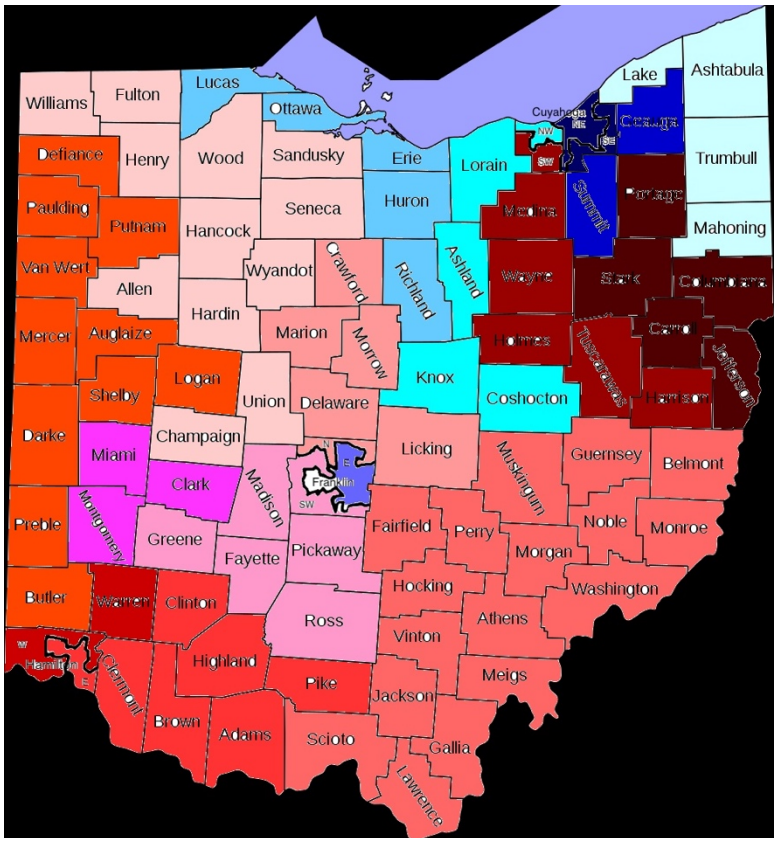


Figure 1c (above). Resulting Congressional map of Ohio of model run with maximum distance of 6 between any two counties in the state

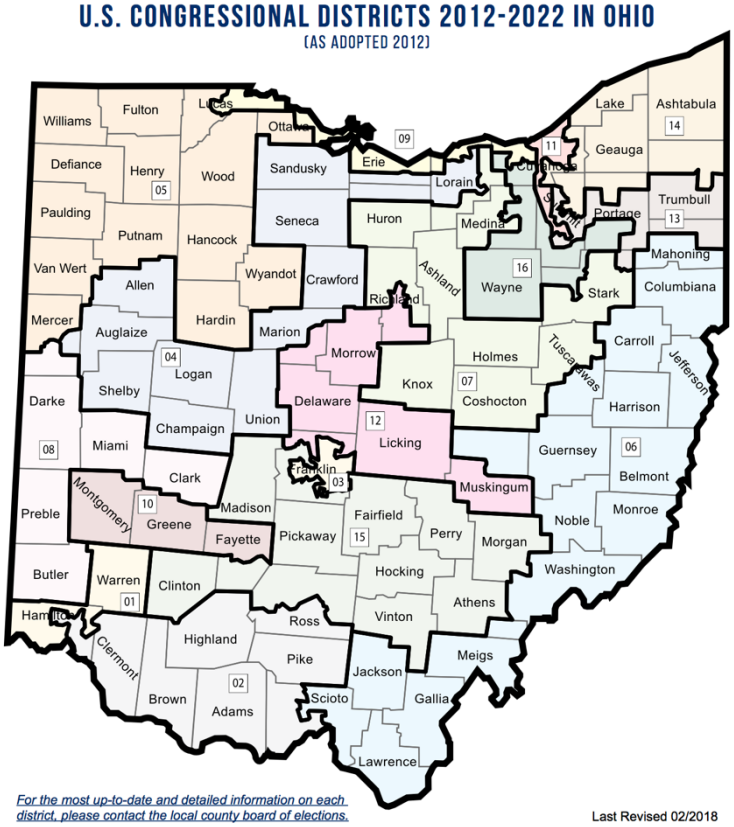


Figure 2 (above). Current gerrymandered map of Ohio Congressional districts

IV. Model Evaluation and Future Extensions

This ILP model is a successful start to a considerably complex problem of redistricting. Most notable in this model is its ability create districts of connected counties with political leanings proportional to the statewide ratio of Democrats and Republicans. However, this model stands to improve in a number of areas according to how we defined a "good" district. In order to see where this model can be improved, we reconsider each of the criteria initially defined on what makes a district "good":

A. *Reevaluating and Extending Upon 'Good' District Criteria*

1) *Compactness*

This model achieves compactness considerably well by simply restricting the distance between pairs of counties in a district. It is safe to say that the model given here achieves compact districts, however, more could be done. Even with small maximum distances like 4, districts can still string around in interesting ways.

Future Extension: Thus, we can expand upon this compactness criteria by engineering new constraints that achieve compactness. For instance, some might suggest that a compact district should maximize the Polsby-Popper measure. This is the ratio of a district's area to the area of a circle whose circumference is the same as the district's perimeter [4]. Similarly, an extension could be made to constrain districts to be absent of enclaves (i.e. a district shouldn't border a county of another district on more than 2-3 'sides' or a certain percentage of its perimeter for example). One might even consider simply defining distance in other ways to see what achieves the greatest compactness. Would districts be more compact if we simply defined distance as the number of miles directly between the centers of two counties?

2) *Contiguity*

As proven in Appendix B, this model achieves districts that are guaranteed to be connected. This criterion has clearly been achieved by the proposed model.

3) *Integrity*

To a certain extent, this model achieves good integrity by using counties as functional units in a district. Keeping counties generally intact within a district is one way to maintain a geographical region's integrity. However, in the future smaller functional units like precincts could be used as counties are currently used in this model. Thus, it is probable that new constraints will be necessary to maintain a certain level of integrity.

Future Extension: Thus, in the future when smaller functional units might be encoded as the "counties" are in this model, new constraints can be engineered to require that only a certain number of counties can be divided up. One can also consider constraints that further maintain the integrity of regions as defined by more focused attributes such as neighborhoods within cities or regions with a distinct, cohesive culture or identity.

4) *Population*

Overall, as the population criterion is designed to be the objective in this model, population distribution is what this model achieves best. However, with deviations just over 10% from the optimal population of a district, some courts would rule the map unconstitutional (like Colorado which requires a maximum deviation less than 5%) [2]. Thus, in order to reduce this objective value, in the future the model needs to be given greater possibilities for ensembles of districts.

Future Extension: Thus, in order to avoid sacrificing other important constraints, the best way to give the model more options for creating districts of uniform population is to create ensembles of districts using smaller functional units, like precincts. A future extension would be to gather more data for a state like Ohio to test the proposed model using smaller functional units. One could see whether the greatly increased number of decision variables required by having more functional units has a strong impact on the model's complexity and whether doing so gives greater performance in criteria like population.

5) *Competitiveness*

While this model does a great deal in promoting proportionality of districts leaning toward a particular party to the number of that party's voters statewide, it doesn't necessarily have any components directly addressing the issue of competitive districts. For now, it simply considers the political *leanings* of a district for one party or another. Some districts might still win a candidate by a landslide while others might lean towards a certain party, but not by much.

Future Extension: Thus, more research needs to be done into how to best encode competitiveness for a district to be considered "good". Perhaps this means optimizing the model using a second objective such that all districts are also as competitive as possible while maintaining the proportional political leaning constraint. Perhaps this means allowing some districts to likely give a certain party a landslide victory but while making sure each party has a proportional number of landslide victories to its voters statewide. In either case, more needs to be done to ensure competitive districts exist in order to continually encourage the connection between an elected representative and their constituents.

6) *Proportionality*

This model's main focus is to achieve this new criterion of proportionality. It does this quite successfully with the only possible improvement coming from two previously mentioned extensions. The first improvement is in being able to achieve the exact optimal proportion without a leeway. In order to do this however, it seems that a greater number of possible ensembles of districts might be required. Thus, this could be fixed by using smaller functional units like precincts. The other improvement is in considering how proportionality should be related to competitiveness and help to achieve it as described by the future extensions to competitiveness. Overall, however, this model achieves proportionality quite well, especially in comparison to the current gerrymandered maps that exist in many states throughout the nation.

As discussed, this model creates decently 'good' districts in a short amount of time and provides a strong basis for mathematically constructing districts that have a fairness component politically. With a few extensions as previously described, the model could truly capture each criterion defined for a 'good' district. There are also a few additional extensions that could be done in the future:

B. *Additional Future Extensions*

1) *Literature Review*

The report accompanying this model should contain a thorough literature review into all the other research being performed on algorithmic Congressional redistricting. Are other researchers using Integer Linear Programming to construct optimal districts? What are other possible criteria for defining a compact district? How can political fairness and district competitiveness both be achieved?

2) *Additional Testing*

The model should be tested on other states as well. How do the results compare in states other than Ohio with totally different shapes and political makeups? Will models using data from these states still perform as well? Similarly, more can be done to compare models using different values for the maximum distance, the leeway, etc. How does changing these hyperparameters affect the quality of the resulting map? How do we pick the best values for these hyperparameters? How can we validate the optimality of the resulting map?

3) *Multi-Objective Consideration*

One should consider how techniques like multi-objective integer linear programming can be applied to this model. One problem is that it seems there are many possible solutions that reach the same optimal objective in terms of the uniformity of population distribution. In order to ensure that the resulting map is the best possible, it might be worth translating some of the constraints regarding compactness and political fairness into objectives instead, so as to ensure the resulting map does not only meet some predetermined constraint, but rather is the best possible map under those criteria.

4) *Broader Model Extension*

Another future extension is in considering how some of the constraints defined regarding compactness and others might apply to other graphical models. The issue of compactness in a graphical structure is an issue with its own theoretical interest. How might the techniques and findings used here apply in other examples or to broader theoretical questions we might have about graphs?

V. Conclusion

Clearly, this research addresses an issue of vital, timely concern in the United States. States all over the country have districting plans that clearly aim to disenfranchise voters and allow politicians to choose their constituents rather than other way around. As more and more states and courts begin to address this problem, and more courts rule maps unconstitutional, it is necessary to have a constructive, algorithmic solution to the problem of redistricting. Furthermore, it is important that the solution consider criteria beyond compactness, population, and contiguity. It needs to address political fairness both in terms of proportionality and competitiveness. The model proposed here is a big step forward in achieving just that. As every state approaches a new redistricting process following the upcoming 2020 census, models like this one will have to be considered and enhanced in order to save the country from another decade of broken democracy.

VI. Works Cited

[1] Blake, Aaron. "Redistricting, Explained." The Washington Post, WP Company, 1 June 2011, www.washingtonpost.com/politics/redistricting-explained/2011/05/27/AGWsFNGH_story.html?utm_term=.9a20b654409b.

[2] "Redistricting Criteria." NCSL.org, National Conference of State Legislatures, 26 Nov. 2018.

[3] "Issue 1." FairDistrictsOhio.org, League of Women Voters of Ohio, www.fairdistrictsohio.org/issue-1.html.

[4] Fisher, Zachary. "Measuring Compactness". https://fisherzachary.github.io/public/r-output.html

APPENDIX A. USING DIJKSTRA'S ALGORITHM TO COMPUTE COUNTY DISTANCES

A key component to the constraints of the model considered in this project is a 'distances' parameter. By defining a distance between any two counties in a state, we can constrain our model to select districts that never have two counties more than some maximum distance value apart. This distance parameter could be defined as the direct distance in miles between two counties for example. In the case of our testing on the state of Ohio, we employ a distance defined in the following way due to its benefit for logical constraints regarding contiguity:

$$Distance(i,j)\ \ s.t.\ i,j\ \in counties$$
$$= minimum\ number\ of\ edges\ that\ must\ be\ traversed\ to\ reach\ j\ from\ i$$

Thus, in order to generate this data, we can first obtain adjacency information for all the counties in a state (i.e. for every pair of counties recording whether they adjacent to one another). Then we can create an instance of a graph from the counties in the state where each node is defined to be a county and there exists an edge between two nodes if the corresponding counties are adjacent.

Using this graph structure modeling the state as a whole, we simply apply Dijkstra's shortest path algorithm to find the shortest path between every pair of nodes in the state. The length of this shortest path is then defined to be the distance between those two nodes.

For more details on the implementation of this algorithm, see the project repository and its corresponding subdirectory on 'shortestDistances':
https://github.com/benjamincarman/repairing-redistricting/tree/master/shortestDistances

APPENDIX B. GUARANTEED DISTRICT CONGUTITY PROOF

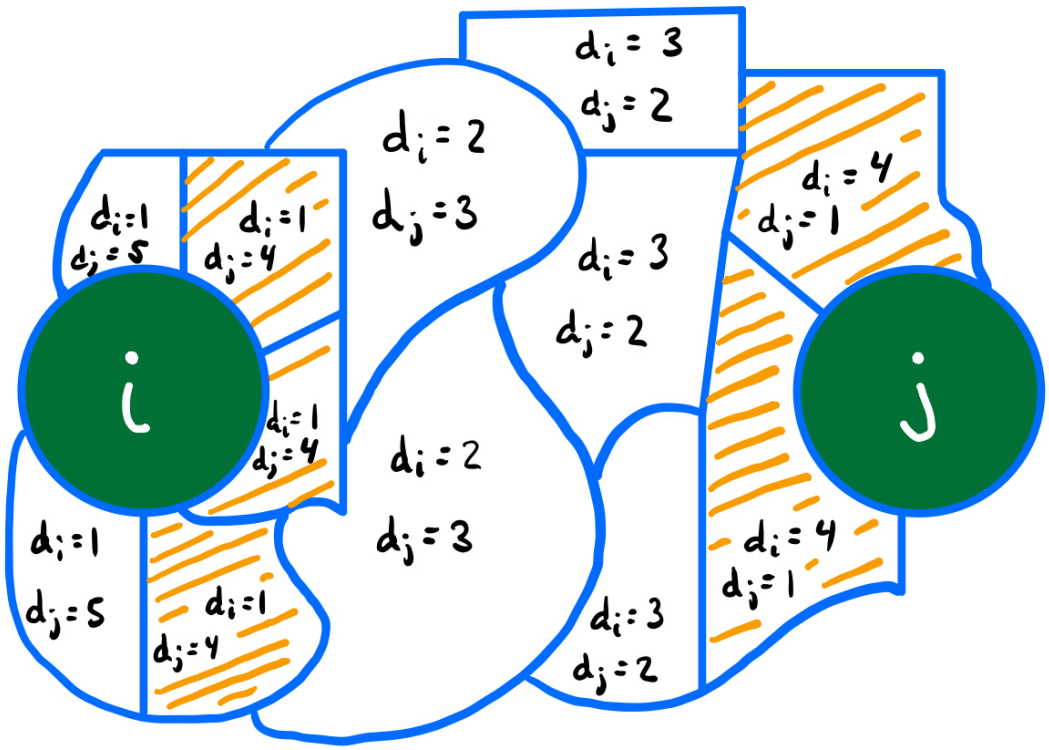
In order to generate "good" Congressional districts, a basic requirement is being able to produce districts where the set of counties contained within that district are all connected to one another. Recall that in order to achieve this, our model put in place the following connectivity constraint:

```
set bridgeCounties{i in counties, j in counties, dist in 2..MAXDISTANCE:
            distances[i,j] == dist} :=
{k in counties: (distances[i,k] == 1 and distances[j,k] == dist - 1)
                or (distances[i,k] == dist - 1 and distances[j,k] == 1)};

subject to countyBridge{i in counties, j in counties, l in districts,
                 dist in 2..MAXDISTANCE: distances[i,j] == dist and
                 p[i,l] == 1 and p[j,l] == 1}:
sum{k in bridgeCounties[i,j,dist]: p[k,l] == 1}c[k,l] >=
                                                    c[i,l] + c[j,l] - 1;
```

A. *Intuition*

What this constraint says in layman's terms is that for every pair of counties in a district of a distance *d* greater than or equal to 2 apart, we require that one of that pair's 'bridge counties' be included in the district. A bridge county is defined as being adjacent to 1 node from the pair and a distance *d – 1* from the other. Consider the following graphic:



Let every enclosed space be a county in the state. Consider *i* and *j* to have already been chosen for the same district. Let d_i be the distance of that county from *i* and d_j be the distance of that county from *j*. Notice that *i* and *j* are a distance of 5 from one another. Let those counties highlighted orange be the set of bridge counties (as they are a distance of 1 from either *i* or *j* and a distance of 5 – 1 = 4 from the other). Intuitively, via the constraint above, we require that one of these orange bridge counties also be included in the district. Since these bridge counties bring the connectedness of the disjoint subgraphs containing *i* and *j* closer together and the included bridge county will then require its own bridge county to be included in the district, from a recursive perspective, bridge counties will continue to be added until these subgraphs are connected via some path.

B. *Proof of Contiguity*

Connectedness Definition.

A graph *G* is *connected* if and only if for any two nodes $x, y \in G$, $\exists$ a sequence of nodes b_1, b_2 , … , b_n where $n < MAXDISTANCE$ s.t. $distances[x, b_1] = 1$, $distances[b_n, y] = 1$, and $\forall\, i \in [2, n], distances[b_{i-1}, b_i] = 1$

Theorem.

All resulting districts from the redistricting model described in this paper will only contain a set of counties C which can be expressed as a connected graph where nodes are the counties and edges exist between adjacent counties.

Proof.

We must show that for any pair of counties in C corresponding to its district *l*, that they are guaranteed to be connected by some subset (or component) of nodes from the district graph. We will proceed *by induction* over possible distances d in the range [1, MAXDISTANCE] for a pair of counties in the district.

Base Cases:

d = 1 (Trivial Case): Consider any two nodes *x, y* in a district graph to be distance 1 apart. If this is the case *x* and *y* are connected by default (by the definition of connectedness).

d = 2: Consider any two nodes *x, y* in a district graph to be distance 2 apart. By the countyBridge constraint, some node *z* in the set of *bridgeCounties[x, y, 2]* must be included in the district graph. By the definition of *bridgeCounties, z* must be a distance of 1 from both *x* and *y.* Thus, by the definition of connectedness, *x* and *y* are connected via *z*.

Inductive Case:

Thus, by proving the above base cases we can assume the following:

Inductive Hypothesis: Any two nodes (counties) in the district graph distance *d* from one another are guaranteed to be connected for any *d* s.t. $0 < d < d + 1 \leq MAXDISTANCE$.

We must now show that any two nodes *x, y* in the district graph, s.t *x, y* are distance *m = d + 1* apart, are connected.

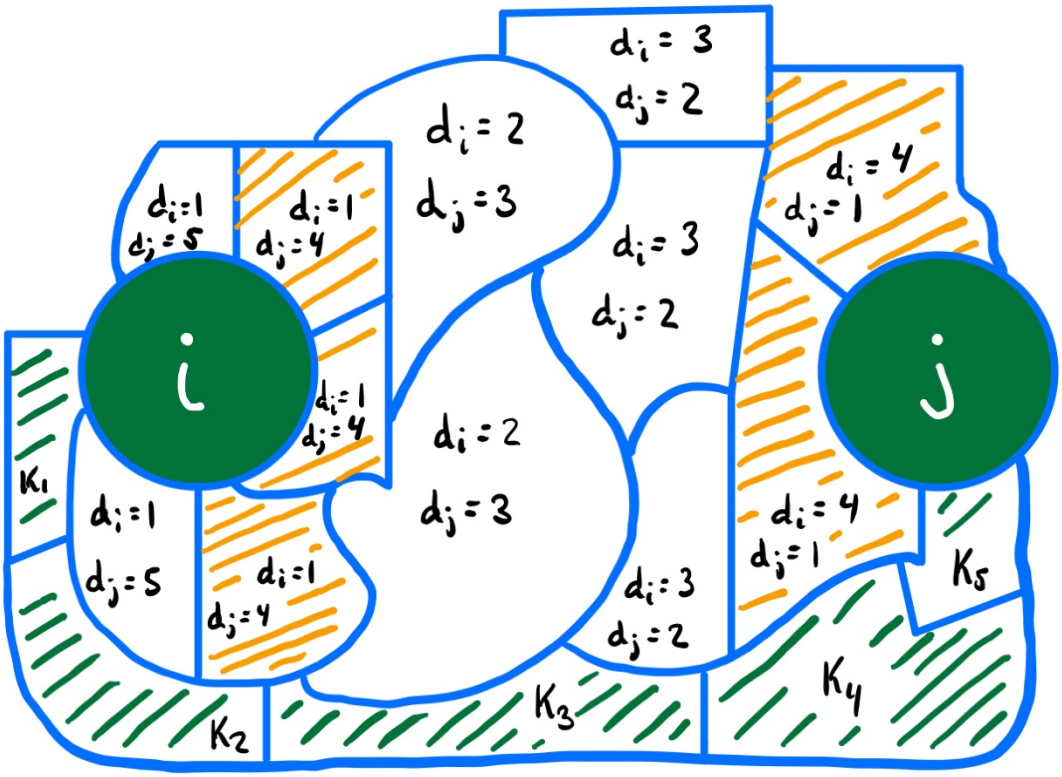
By the constraint *countyBridge(x, y, l, m),* some node *z s.t. z* is adjacent to *x* and distance *m – 1* from *y* or s.t. *z* is adjacent to *y* and distance *m – 1* from *x* must be included in the district graph.

By the inductive hypothesis, we have that any two nodes in the district graph of distance less than *m* are connected. Thus, in the first case for *z* it must be the case that *z* is adjacent to *x* and connected to *y* by the inductive hypothesis. In the second case for *z* it must be that *z* is adjacent to *y* and connected to *x* by the inductive hypothesis. In either case, by the definition of connectivity, *x* is connected to *y* via the included node *z.*

Qed.

C. *Additional Restrictiveness*

It is important to note that while this constraint guarantees connected districts as shown in B, having this constraint prohibits a few possible ensembles of districts. Consider the following modification to the previously considered graphic:



Let all K_m be counties included in the same district as *i* and *j*. Notice that the inclusion of these counties would make the district fully contiguous. However, it is clear by inspection that none of these counties fit the criteria to be considered a bridge county. Thus, their inclusion will not be sufficient for our model to consider *i* and *j* to be connected and it will require additional counties to be included as a result. Thus, a resulting district including only those counties highlighted green as above would not be possible under the constraints given in this model. This creates an additional, unintended restrictive property by including the connectedness constraint. It is important to realize this side effect and cost of having it, however and it can largely be seen as a bonus benefit to the connectivity constraint as districts like the one considered here are visibly not compact and tend to produce enclaves. These are not desirable qualities anyway so omitting these possibilities does not hurt the effectiveness of the model.

APPENDIX C. COMPLETE AMPL MODEL

```
#DEFINITIONS
param n; #number of counties in the state

param m; #number of electoral districts

set counties; #the set of state counties

set districts := 1..m; #the set of districts

param p{i in counties, j in districts} binary;
#is 1 if county i can be included in district j

set possiblePairs := {i in counties, j in districts: p[i,j] == 1};
#set of possible county/district pairings

param distances{i in counties, j in counties} integer;
#distance between counties i and j measured by how many adjacent counties
#away i is from j

param MAXDISTANCE;
#maximum allowable distance between two counties in a district

param d{i in counties}; #the number of Democrat voters in a county

param r{i in counties}; #the number of Republican voters in a county

param v{i in counties} := d[i] + r[i];
#the number of total Democrat and Republican voters in county

param LARGE := sum{i in counties}v[i]; #sum of all voters in state

param idealNumDemocratDistricts := round(((sum{i in counties}d[i]) / (sum{i
in counties}v[i])) * m);
#proportion of democrat voters multiplied by total number of districts

param targetPopulation := round((sum{i in counties}v[i]) / m);

param politicalLeeway; #a tuning parameter for the allowable error of
                       #actual Democrat districts from the ideal

var leansDemocrat{j in districts} binary;
#is 1 if more Democrats in a district, 0 if not

var c{i in counties, j in districts: p[i,j] == 1} binary;
#is 1 if county i is chosen for district j

var z; #the largest difference from target population among all districts


#OBJECTIVE
minimize LargestDifference: z;


#CONSTRAINTS
subject to zIsLargestDifferencePos{j in districts}:
      z >= ((sum{(i,j) in possiblePairs}c[i,j]*v[i]) - targetPopulation);

subject to zIsLargestDifferenceNeg{j in districts}:
      z >= (-1 * ((sum{(i,j) in possiblePairs}c[i,j]*v[i]) –
```

```
            targetPopulation));

subject to Exactly1DistrictPerCounty{i in counties}:
      sum{(i,j) in possiblePairs}c[i,j]=1;

subject to setIfLeansDemocrat{j in districts}:
      LARGE * leansDemocrat[j] >= ((sum{(i,j) in possiblePairs}c[i,j]*d[i])
                                 - (sum{(i,j) in possiblePairs}c[i,j]*r[i]));

subject to setIfLeansRepublican{j in districts}:
      LARGE * (1 - leansDemocrat[j]) >=
      (-1 * ((sum{(i,j) in possiblePairs}c[i,j]*d[i]) -
      (sum{(i,j) in possiblePairs}c[i,j]*r[i])));

subject to lessThanMaxDemocratDistricts:
      sum{j in districts}leansDemocrat[j] <= idealNumDemocratDistricts + 1;

subject to greaterThanMinDemocratDistricts:
      sum{j in districts}leansDemocrat[j] >= idealNumDemocratDistricts - 1;

subject to maxDistance{i in counties, j in counties, k in districts:
                       p[i,k] == 1 and p[j,k] == 1 and
                       distances[i,j] > MAXDISTANCE}:
      c[i,k] + c[j,k] <= 1;

set bridgeCounties{i in counties, j in counties, dist in 2..MAXDISTANCE:
                   distances[i,j] == dist} :=
      {k in counties: (distances[i,k] == 1 and distances[j,k] == dist - 1)
                  or (distances[i,k] == dist - 1 and distances[j,k] == 1)};

subject to countyBridge{i in counties, j in counties, l in districts,
                        dist in 2..MAXDISTANCE: distances[i,j] == dist
                        and p[i,l] == 1 and p[j,l] == 1}:
      sum{k in bridgeCounties[i,j,dist]: p[k,l] == 1}c[k,l] >=
                                                  c[i,l] + c[j,l] - 1;
```

Note: The data parameters are left undefined in this report. Please see https://github.com/benjamincarman/repairing-redistricting for the complete set of code, data, and results.