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

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 the rest.

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 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 placed 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 passed 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 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 is there is one for every district and is updated 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 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 ± 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 adjacent to 1 county in the pair and a distance of 1 closet to the other county in the pair. Then, for every pair of counties 'dist' distance apart included in a district, we require that at least one bridge county is 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 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] >=
```

E. *Data*

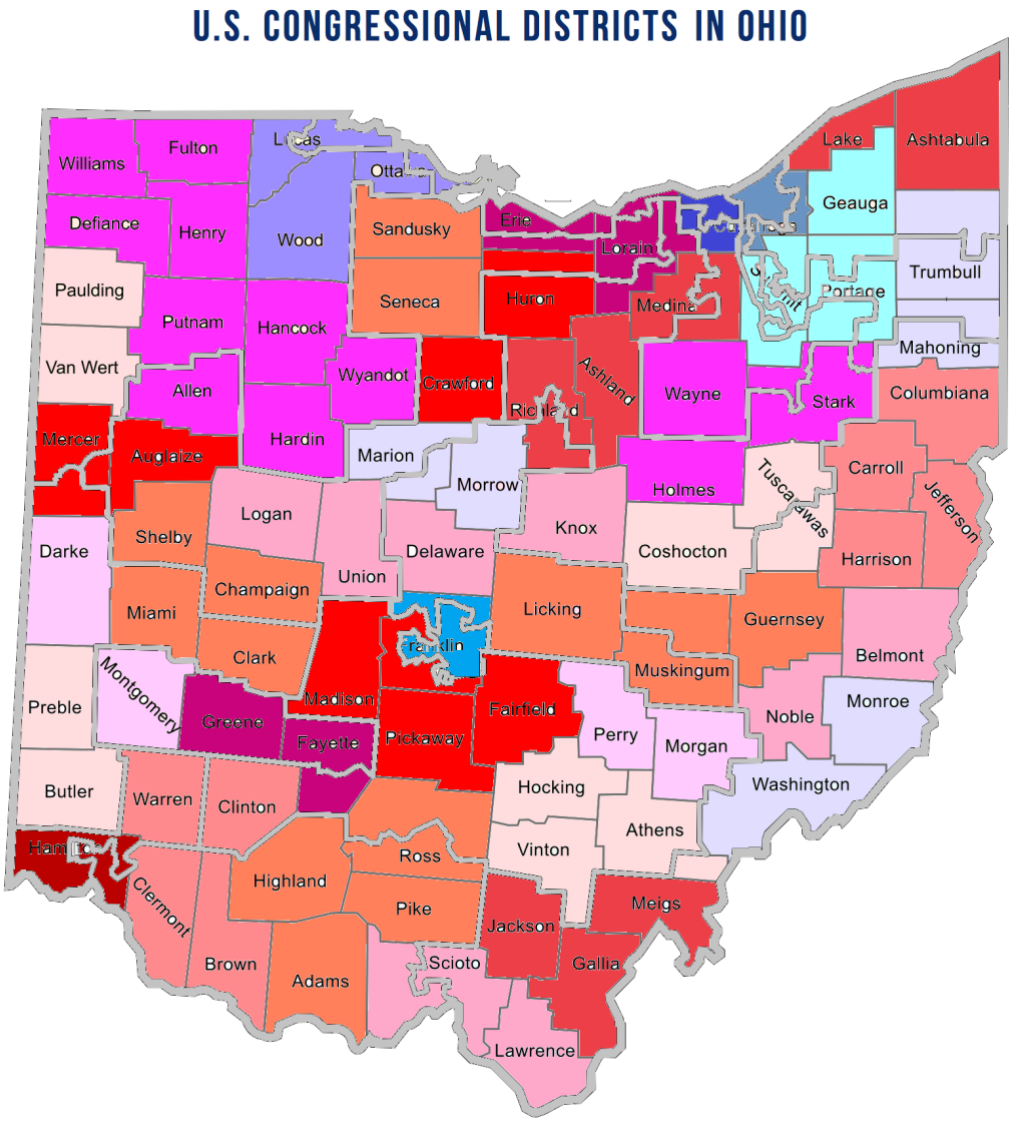
This model was tested on Ohio, a swing state with a history of gerrymandering, using relevant data from the 2016 election. This data includes all 88 counties with 3 counties split into smaller "sub-counties". These are Cuyahoga, Franklin, and Hamilton. These counties include Ohio's largest cities Cleveland, 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. This data can be seen in Appendix C and includes 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*

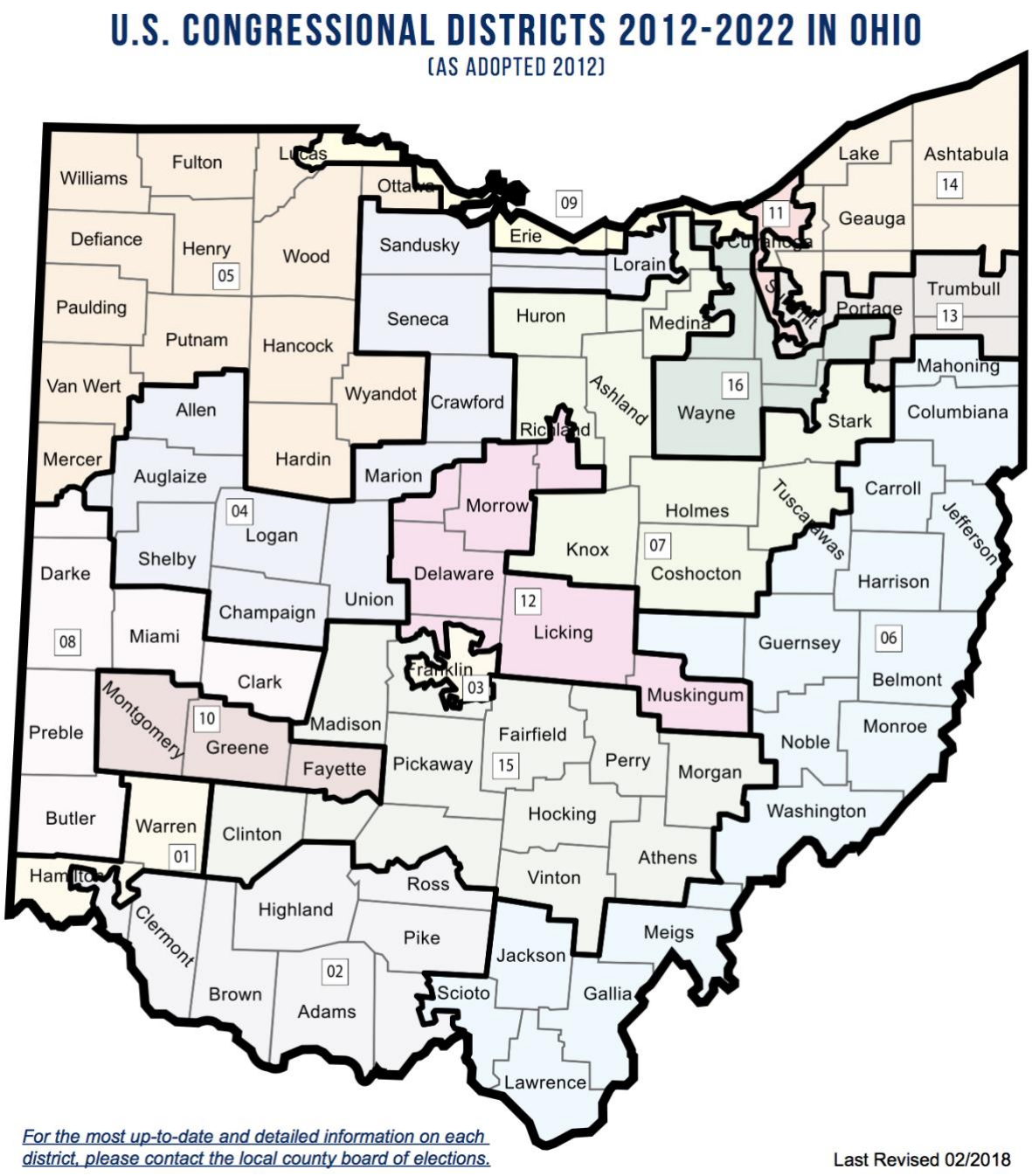
Upon running the model, a feasible solution was found with an objective of 87,171. This means the model was able to find a solution where the maximum deviation in population of a district from the ideal population of 321,909 was 87,171. There was a deviation in population of about 27% from optimal which should be improved to meet constitutional and federal law. This could be done by splitting counties up further to give the model more options or by relaxing other constraints.

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 districts for each party as should be expected by statewide voter data. Specifically, it produced 6 Democrat-leaning districts which was within the ±1 range of an expected 7 districts. The specific choices for county-district pairs can be seen in Appendix B.

The largest room for improvement in this current model is its ability to produce compact, contiguous districts. Currently, there are only constraints that help this to be the case but are not enough to guarantee a district is completely connected when there are still many possible candidate county-district pairs. This output can be seen in the map below where there is a unique color for each county. Colors of a red hue indelicate the county leans Republican and colors of a blue hue indicate the county leans Democrat.



This map is in stark contrast the existing gerrymandered districts in the following map:



IV. Future Extensions

This ILP model is a successful start to solving a complex problem. Most notable in this model is its ability create districts according to the statewide ratio of Democrats and Republicans. However, the model could be greatly improved through further research in optimizing the contiguity and compactness of the districts. If constraints could be formulated to ensure these criteria are satisfied, the model could still check many possible county-district candidates and produce the very best results. In its current form, the model relies on a stringently defined p parameter to ensure candidates across the state are included in the district. One such way this extension could be defined might be through a parameter that defines the distance between any two counties and constraining that distance to a certain "close" value or even optimizing based on the distance instead of the population.

In performing this type of extension, one must also consider the need to ensure that population deviation is as minimal as possible for the sake of complying with federal law. There are many such ways to ensure a small deviation is possible, such as creating more candidate counties relaxing other constraints. This necessity must be considered as new constraints are added. Overall, this current model demonstrates a great basis for a certain set of constraints and criteria. With a few extensions, this model could produce quite viable, legitimate districts for the next round of redistricting.

V. 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.

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.

Appendix B. Output of AMPL Model with Ohio Test Data

```
Presolve eliminates 105 constraints.
Adjusted problem:
614 variables:
        613 binary variables
        1 linear variable
6641 constraints, all linear; 23869 nonzeros
        94 equality constraints
        6547 inequality constraints
1 linear objective; 1 nonzero.

Gurobi 7.5.1: threads=4
Gurobi 7.5.1: optimal solution; objective 87171
6019 simplex iterations
1 branch-and-cut nodes
plus 2 simplex iterations for intbasis

z = 87171

targetPopulation = 321909

idealNumDemocratDistricts = 7

leansDemocrat [*] :=
 1  0
 2  0
 3  0
 4  0
 5  1
 6  1
 7  0
 8  0
 9  0
10  0
11  1
12  1
13  0
14  1
15  0
16  1
;

c [*,*]
:            1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
:=
Adams        0   0   .   .   .   0   .   .   .   .   .   .   .   .   1   .
Allen        .   .   0   0   0   .   0   0   1   .   .   0   .   .   0   .
Ashland      .   .   .   0   .   0   0   .   0   .   .   0   1   .   .   0
Ashtabula    .   .   .   .   .   .   .   .   .   .   0   .   1   0   .   .
Athens       0   0   0   1   .   0   0   .   .   0   .   0   0   .   0   .
Auglaize     .   .   1   0   0   .   0   0   0   .   .   0   .   .   0   .
Belmont      .   0   .   .   .   0   1   .   .   .   .   0   0   .   0   .
Brown        0   1   .   .   .   0   .   .   .   .   .   .   .   .   0   .
Butler       0   .   .   1   0   .   .   0   .   0   .   .   .   .   0   .
Carroll      .   1   .   .   .   0   0   .   .   .   .   0   0   .   0   .
Champaign    .   .   0   0   0   .   0   0   0   .   .   0   .   .   1   .
Clark        0   .   .   0   0   .   .   0   .   0   .   .   .   .   1   .
Clermont     0   1   .   .   .   0   .   .   .   .   .   .   .   .   0   .
Clinton      0   1   0   0   .   0   .   .   .   0   .   0   .   .   0   .
```

Columbiana	.	1	.	.	.	0	0	.	.	.	.	0	0	.	0	.
Coshocton	.	.	.	1	.	0	0	.	0	.	.	0	0	.	.	0
Crawford	.	.	1	0	0	.	0	0	0	.	.	0	.	.	0	.
Cuyahoga-NE	.	.	.	.	.	.	.	.	0	.	0	.	0	1	.	0
Cuyahoga-NW	.	.	.	0	0	.	0	.	0	.	0	.	.	.	.	1
Cuyahoga-SE	.	.	.	.	.	.	.	.	.	.	0	.	0	1	.	.
Cuyahoga-SW	.	.	.	.	.	.	0	.	0	.	0	.	0	.	.	1
Darke	0	.	.	0	0	.	.	0	.	1	.	.	.	.	0	.
Defiance	.	.	.	0	0	.	.	0	1	.	.	.	.	.	.	.
Delaware	.	.	0	0	.	0	1	.	.	.	.	0	.	.	0	.
Erie	.	.	0	0	0	.	0	1	0	.	0	0	.	.	0	0
Fairfield	0	0	1	0	.	0	.	.	.	0	.	0	.	.	0	.
Fayette	0	0	0	0	.	0	.	1	.	0	.	0	.	.	0	.
Franklin-E	.	.	0	.	.	.	.	.	.	.	.	1	.	.	0	.
Franklin-N	.	.	0	0	.	0	0	.	.	.	.	1	.	.	0	.
Franklin-SW	0	0	1	0	.	0	.	.	.	0	.	0	.	.	0	.
Fulton	.	.	.	0	0	.	.	0	1	.	.	.	.	.	.	.
Gallia	.	0	.	.	.	0	0	.	.	.	.	0	1	.	0	.
Geauga	.	.	.	.	.	.	.	.	.	.	1	.	0	0	.	.
Greene	0	.	.	.	.	.	.	1	.	0	.	.	.	.	0	.
Guernsey	.	0	.	.	.	0	0	.	.	.	.	0	0	.	1	.
Hamilton-E	1	0	.	.	.	0	.	.	.	.	.	.	.	.	0	.
Hamilton-W	1	0	.	.	.	.	.	0	.	0	.	.	.	.	0	.
Hancock	.	.	.	0	0	.	.	0	1	.	.	.	.	.	.	.
Hardin	.	.	.	0	0	.	.	0	1	.	.	.	.	.	.	.
Harrison	.	1	.	.	.	0	0	.	.	.	.	0	0	.	0	.
Henry	.	.	.	0	0	.	.	0	1	.	.	.	.	.	.	.
Highland	0	0	.	.	.	0	.	.	.	.	.	.	.	.	1	.
Hocking	0	0	0	1	.	0	.	.	.	0	.	0	.	.	0	.
Holmes	.	.	.	0	.	0	0	.	1	.	.	0	0	.	.	0
Huron	.	.	1	0	0	0	0	0	0	.	.	0	0	.	0	0
Jackson	.	0	.	.	.	0	0	.	.	.	.	0	1	.	0	.
Jefferson	.	1	.	.	.	0	0	.	.	.	.	0	0	.	0	.
Knox	.	.	.	0	.	0	1	.	0	.	.	0	0	.	.	0
Lake	.	.	.	.	.	.	.	.	.	.	0	.	1	0	.	.
Lawrence	.	0	.	.	.	0	1	.	.	.	.	0	0	.	0	.
Licking	.	.	0	0	.	0	0	.	.	.	.	0	.	.	1	.
Logan	.	.	0	0	0	.	1	0	0	.	.	0	.	.	0	.
Lorain	.	.	0	0	0	0	0	1	0	.	0	0	0	.	0	0
Lucas	.	.	.	0	1	.	0	0	0	.	0	.	.	.	.	0
Madison	0	0	1	0	.	0	.	.	.	0	.	0	.	.	0	.
Mahoning	.	0	.	.	.	1	0	.	.	.	0	0	0	0	0	0
Marion	.	.	0	0	0	1	0	0	0	.	.	0	.	.	0	.
Medina	.	.	.	0	.	0	0	.	0	.	0	0	1	.	.	0
Meigs	.	0	.	.	.	0	0	.	.	.	.	0	1	.	0	.
Mercer	0	.	1	0	0	.	0	0	0	0	.	0	.	.	0	.
Miami	0	.	.	0	0	.	.	0	.	0	.	.	.	.	1	.
Monroe	.	0	.	.	.	1	0	.	.	.	.	0	0	.	0	.
Montgomery	0	.	.	.	.	.	.	0	.	1	.	.	.	.	0	.
Morgan	0	0	0	0	.	0	.	.	.	1	.	0	.	.	0	.
Morrow	.	.	0	0	.	1	0	.	.	.	.	0	.	.	0	.
Muskingum	.	0	0	0	.	0	0	.	.	.	.	0	0	.	1	.
Noble	.	0	.	.	.	0	1	.	.	.	.	0	0	.	0	.
Ottawa	.	.	.	0	1	.	0	0	0	.	0	.	.	.	.	0
Paulding	.	.	.	1	0	.	.	0	0	.	.	.	.	.	.	.
Perry	0	0	0	0	.	0	.	.	.	1	.	0	.	.	0	.
Pickaway	0	0	1	0	.	0	.	.	.	0	.	0	.	.	0	.
Pike	0	0	.	.	.	0	.	.	.	.	.	.	.	.	1	.
Portage	.	.	.	.	.	0	0	.	0	.	1	.	0	0	.	0
Preble	0	.	.	1	0	.	.	0	.	0	.	.	.	.	0	.
Putnam	.	.	.	0	0	.	.	0	1	.	.	.	.	.	.	.

```
Richland      .   .   0   0   .   0   0   .   0   .   .   0   1   .   0   0
Ross          0   0   0   0   .   0   .   .   .   0   .   0   .   .   1   .
Sandusky      .   .   0   0   0   .   0   0   0   .   .   0   .   .   1   .
Scioto        0   0   .   .   .   0   1   .   .   .   .   0   0   .   0   .
Seneca        .   .   0   0   0   .   0   0   0   .   .   0   .   .   1   .
Shelby        .   .   0   0   0   .   0   0   0   .   .   0   .   .   1   .
Stark         .   .   .   0   .   0   0   .   1   .   0   0   0   0   .   0
Summit        .   .   .   .   .   0   0   .   0   .   1   .   0   0   .   0
Trumbull      .   .   .   .   .   1   .   .   .   .   0   .   0   0   .   0
Tuscarawas    .   0   .   1   .   0   0   .   0   .   .   0   0   .   0   0
Union         .   .   0   0   0   .   1   0   0   .   .   0   .   .   0   .
VanWert       .   .   .   1   0   .   .   0   0   .   .   .   .   .   .   .
Vinton        0   0   0   1   .   0   .   .   .   0   .   0   .   .   0   .
Warren        0   1   .   .   .   .   .   0   .   0   .   .   .   .   0   .
Washington    .   0   .   .   .   1   0   .   .   .   .   0   0   .   0   .
Wayne         .   .   .   .   .   .   0   .   1   .   0   .   0   .   .   0
Williams      .   .   .   0   0   .   .   0   1   .   .   .   .   .   .   .
Wood          .   .   .   0   1   .   .   0   0   .   .   .   .   .   .   .
Wyandot       .   .   .   0   0   .   .   0   1   .   .   .   .   .   .   .
;
```