Technical report Spring 2022

Formulating Exact Contiguity Constraints in Integer Programming

AUTHOR: Cheng-Wei Lu

1 Introduction

Gerrymandering refers to a strategy where politicians try to maximize the votes they get by redistricting and manipulating district boundaries. The term gerrymandering got its first appearance in 1810s with Elbridge Gerry, the Governer of Massachusetts at that time, signing a bill that created a partisan distict in the Boston area that was compared to the shape of a mythological salamander. There are still on going discussions about how Gerrymandering is affecting elections such as the recent congressional map proposed in Wisconsin.

There are many people doing research on Gerrymandering and how to generate great district map. A great district map should pursue three goals. The first goal is to ensure that each district in the district map has a compact shape. In other words, instead of districts that have long and thin shape, we would prefer the ones that have round shape. The second goal is the population constraints, which makes sure the populations among districts do not differ too much. The last goal is the contiguity constraint, which guarantees every units inside a district form a connected graph where the edges in that graph represent the geographical adjacency among units.

There are two main research directions in Gerrymandering. The first one aims to design heuristics which randomly generates maps that satisfy the above mentioned goals [1] [2]. Generating multiple maps with heuristics allows us to understand the whole solution space better and analyze maps with different metrics such as party-wise voting ratio. The other one aims to formulate direct integer programming models in which contiguity is imposed as the constraints, which is the main focus of this project. In the integer programming model, the goal is to solve for a single optimal map. The definition of the optimal map will be introduced in later section. We hope that by using integer programming model and choosing appropriate objective, we can have a "baseline" of how a good map could look like and then we can incorporate it into our decision process.

This project will be presented in the following order. In section 2, I will introduce the main problem to solve with MIP model that contains contiguity constraints. There are multiple ways to formulate contiguity constraints, and different variations will be introduced in section 3. The methods I implemented [1] are the following:

1. SHIR [5] : For each district, we can formulate a set of Steiner Tree constraints to guarantee contiguity.
2. MCF : Instead of using $|T| - 1$ as big-M constraint in the Steiner Tree formulation of SHIR, we use multi-commodity formulation to obtain a tighter formulation.
3. Distance-based contiguity constraint : Using the distance of adjacent pair of units to formulate contiguity constraint.
4. CUT[4]: It uses the idea of a-b separator and add cutting planes to guarantee contiguity.

[1] Code and results are available at https://github.com/WalterLu3/ContiguityConstraints

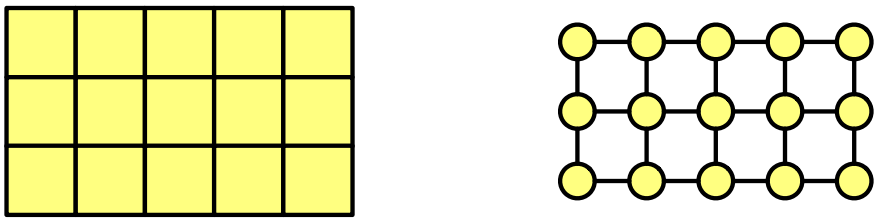

Figure 1: Each unit (grid) can be represented by a vertex and two adjacent vertices are connected by an edge .

In section 4, experiment results on synthetic data set will be presented. Finally, in section 5, I will apply CUT method to Wisconsin county-level data.

2 Problem Description

When we are trying to solve a redistricting problem, we can think of the original map as a graph $G = (V, E)$. Each vertex in that graph is a unit to be assigned (we will call it "unit" for the rest of the project), and each edge represents the geographical adjacency of a pair of units (See figure 1).

In the introduction section, we have described three goals for a good district map:

1. Great Compactness
2. Satisfying Population Constraint
3. Satisfying Contiguity Constraint

Then, the problem we seek to solve is something similar to a knapsack problem : we want to "pack" units into several districts with the objective of maximizing the compactness, and we also want to satisfy the capacity constraint of districts and the contiguity constraints.

2.1 The Hess model (model without contiguity constraints)

The Hess model [3] is widely recognized as the first optimization model for redistricting. The model seeks a plan which optimizes the compactness, where compactness is measured by summing up the distance between every unit in a district map and its assigned district center, subject only to constraints on population equality (no contiguity constraint is imposed). Therefore, only the compactness and population goals are considered in the model. The following is the formulation of the Hess model:

Decision Variables:

$$x_{ij} = \begin{cases} 1, & \text{if unit i is assigned to (the district centered at) unit j} \\ 0, & \text{otherwise} \end{cases} . \; i, j \in V$$

(Note that if a unit i is assigned to itself; i.e., $x_{ii} = 1$, then unit i is selected as a district center.)

Parameters:

$$w_{ij} : \text{the distance from unit } i \text{ to } j. \; i, j \in V$$

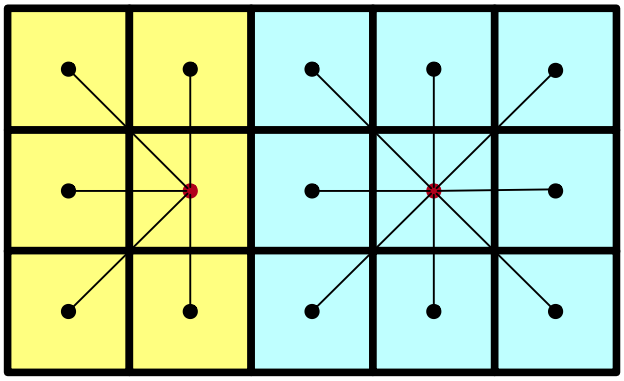

Figure 2: Objective function for the Hess model. Suppose there are two districts. One in blue and the other in yellow, and the district centers are units that contain the red centroid point. The objective value is the sum of distance between each unit and its assigned district center's centroid; i.e., the sum of thin black lines length in the figure.

(The distance between two units are usually calculated as the Euclidean distance between their centroids.)

$$p_i : \text{the population of unit } i.\ i \in V$$

Objective:

$$\min \quad \sum_{i \in V} \sum_{j \in V} w_{ij} x_{ij} \tag{1a}$$

Constraints:

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \tag{1b}$$

$$\sum_{j \in V} x_{jj} = k \tag{1c}$$

$$L x_{jj} \leq \sum_{i \in V} p_i x_{ij} \leq U x_{jj} \qquad \forall j \in V \tag{1d}$$

$$x_{ij} \leq x_{jj} \qquad \forall i, j \in V \tag{1e}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{1f}$$

Constraint (1b) ensures every unit can only be assigned to 1 unit. Constraint (1c) ensures that there are exactly k units selected as district centers; i.e., there are exactly k districts (k is user input). Constraint (1d) ensures population of each district satisfy population bound. Constraint (1e) ensures that unit i can be assigned to unit j only if unit j is chosen as a district center. The objective (1a) of the Hess model here is chosen to be minimizing the sum of distance between every unit in a district map and its assigned district center. The visualization of the how the objective value is acquired is in figure 2.

Since the objective is to find an optimally compact map, it can generate map where the districts are nearly contiguous, though there is no contiguity constraint. In this project, I use the Hess model as the base of my implementation. I will use a variety of methods to add additional contiguity constraints to the Hess model. For the ease of notation, let's define the feasible region of the Hess problem as :

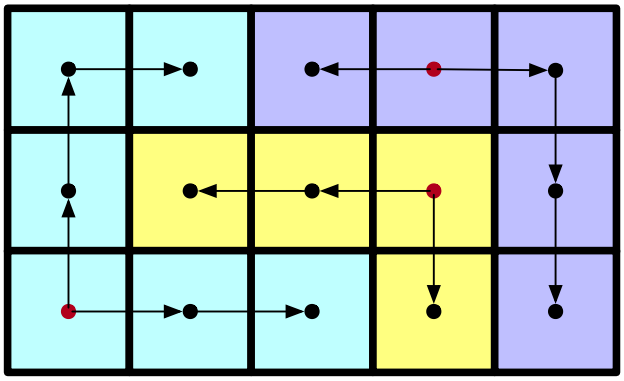

Figure 3: Visualization of the SHIR model. We can regard the contiguity constraint as a combination of multiple Steiner tree problems. In the figure, we have a solution that satisfies contiguity, and there are 3 districts in three different color. Let red dots be the choice of district centers, then each district center is able to send out flow to the units that are assigned to it.

$$P_{\text{Hess}} = \{x : x \text{ satisfies } (1b), (1c), (1d), (1e) \text{ and } (1f)\}.$$

3 Methods of Formulating Contiguity Constraints

In the last section, we learn that the Hess model only pursues the goal of compactness and population balance. Contiguity constraint can easily be added to the Hess model in the form of additional constraints. We will discuss four ways of formulating contiguity constraints here.

In some of the contiguity formulations below, flow-based methods are used. Therefore, we need to redefine the network for a map as a directed map $G_d = (V, A)$, where V is the set of all units in a map and A is the set of arcs which contains both directions of every edges in E we mentioned at the start of section 2.

3.1 SHIR model

The SHIR model [5] formulates the contiguity constraint as a combination of multiple Steiner trees in the redistricting network (See figure 3).

Decision Variables:

$$f_{ij}^{v} = \text{the amount of flow sent across arc } (i, j), \text{ originating at district center v. } v \in V, (i, j) \in A.$$

Constraints:

$$x \in P_{\text{Hess}} \tag{2a}$$

$$\sum_{j \in \delta^+(i)} f_{ij}^v - \sum_{j \in \delta^-(i)} f_{ji}^v = -x_{iv} \qquad \forall i \in V \setminus \{v\}, \forall v \in V \tag{2b}$$

$$\sum_{j \in \delta^-(i)} f_{ji}^v \leq (|V| - 1) x_{iv} \qquad \forall i \in V \setminus \{v\}, \forall v \in V \tag{2c}$$

$$\sum_{j \in \delta^-(v)} f_{jv}^v = 0 \qquad \forall v \in V \tag{2d}$$

$$f_{ij}^v \geq 0 \qquad \forall (i,j) \in A, \forall v \in V \tag{2e}$$

$$\delta^+(i) = \{j : (i,j) \in A\}$$
$$\delta^-(i) = \{j : (j,i) \in A\}.$$

Constraint (2b) makes sure if a unit i is assigned to district center v, then i has a flow demand of -1 for flow type v; i.e., i has a flow demand -1 in the Steiner tree whose root or supply vertex is v. Constraint (2c) ensures that i can have inflow of type v only if i is assigned to v. (2b) and (2c) together make sure that a unit i does not have any flow exchange of type v if it is not assigned to v. Constraint (2d) makes sure the supply node v of a Steiner tree does not have any inflow of type v. This constraint also guarantees that the root node v will send out flow that meets the demand of its represented Steiner tree.

Using the constraints (2a)-(2e) and letting the objective be (1a), we can obtain a model that maximizes compactness and satisfies both population and contiguity constraints.

3.2 MCF model

The MCF[6] model is an modification of the SHIR model. In the SHIR formulation, it has a big-M constraint (2c), which leads to weak linear relaxation. We can avoid the use of big-M constraint by using a stronger formulation for Steiner tree problem. In this formulation, the flow variables have type that is dependent on original-destination pair in the network. The formulation of the MCF model is as follow:

Decision Variables:

$$f_{ij}^{ab} = \begin{cases} 1, & \text{if there is flow through arc } (i,j) \in A \text{ that is originated at } b \text{ and has destination } a \\ 0, & \text{otherwise} \end{cases}.$$

Constraints:

$$x \in P_{\text{Hess}} \tag{3a}$$

$$\sum_{j \in \delta^+(a)} f_{aj}^{ab} - \sum_{j \in \delta^-(a)} f_{ja}^{ab} = -x_{ab} \qquad \forall a \in V \setminus \{b\}, \forall b \in V \tag{3b}$$

$$\sum_{j \in \delta^+(i)} f_{ij}^{ab} - \sum_{j \in \delta^-(i)} f_{ji}^{ab} = 0 \qquad \forall i \in V \setminus \{a, b\}, \forall a \in V \setminus \{b\}, \forall b \in V \tag{3c}$$

$$\sum_{j \in \delta^-(b)} f_{jb}^{ab} = 0 \qquad \forall a \in V \setminus \{b\}, \forall b \in V \tag{3d}$$

$$\sum_{j \in \delta^-(i)} f_{ji}^{ab} \leq x_{ib} \qquad \forall i \in V \setminus \{b\}, \forall a \in V \setminus \{b\}, \forall b \in V \tag{3e}$$

$$f_{ij}^{ab} \geq 0 \qquad \forall (i,j) \in A, \forall a \in V \setminus \{b\}, \forall b \in V \tag{3f}$$

Constraint (3b) makes sure if a is assigned to b, then it will have 1 demand of flow type ab (the origin-destination pair formed by origin b and destination a). Constraint (3c) keeps the sum of all flow exchange (type ab flow) for all the other nodes to be 0. Constraint (3d) is similar to the constraint (2d) in the SHIR model, where the inflow for root node b is limited to be 0 for all flow type of ab. Constraint (3e) ensures that unit i can only receive flow of type ab if i is assigned to b.

The formulation is similar to the SHIR model. The main difference between the SHIR model and the MCF model is that in the MCF model, the total demand for each flow type is 1, and thus when we are limiting the flow in constraint (3e), we do not need to introduce big-M constraint as the Hess model does in constraint (2c).

This kind of stronger reformulation can sometimes help solve the problem faster, but for the problem that is more complicated, it can have negative effect since the problem size grows a lot faster.

3.3 Distance-Based (DB) model

This method utilizes the distances among units to construct contiguity constraint. The idea is that a unit a can be assigned to b only if among all the units that are adjacent to a and closer to b, at least one of them is assigned to b.

The DB model formulation is the following :

Constraints:

$$x \in P_{\text{Hess}} \tag{4a}$$

$$x_{ij} \leq \sum_{k \in \Delta(i,j)} x_{kj} \qquad \forall i \in V \setminus \{j\}, \forall j \in V \tag{4b}$$

where

$$Dist(i,j) : \text{distance between the centroid of unit i and unit j}$$
$$\Delta(i,j) = \{k : (i,k) \in A, Dist(k,j) \leq Dist(i,j)\}.$$

To visualize how this method works, we can look at figure 4.

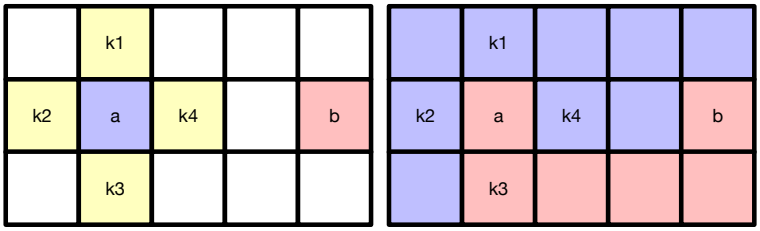

Figure 4: The visualization of the DB model and the difference between its feasible region and other methods'. First we look at the graph on the left. In the DB model, a can be assigned to b if at least one of the unit in $\Delta(a, b)$ is also assigned to b. Here, $\Delta(a, b)$ contains all units that are adjacent to a and are closer to b than a is. Therefore, $\Delta(a, b) = \{k4\}$. For the graph on the right, the units are assigned to two districts in two different colors. Suppose the district center of the red district is b. The solution in this graph is contiguous, but it is not feasible in the DB model constraint, since $k4$ is not assigned to b.

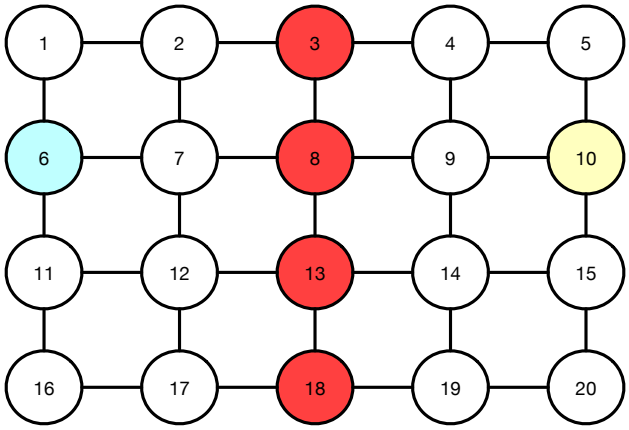

Figure 5: Example of an a-b separator C . Let $a = 6$ and $b = 10$. $C = \{3, 8, 13, 18\}$.

The feasible solution of the DB model will then be contiguous, but the whole feasible region is not the same as those of the SHIR model and the MCF model. The example in figure 4 shows this fact. This also implies that the DB formulation rejects the cases where the connecting path between a pair units is not curvy, which to some extent reflects the goal of maximizing compactness. To my knowledge, this formulation has not appeared in any literature yet.

3.4 CUT model

The CUT model [4] is a cutting plan method based on the idea of a-b separator. An example is in figure 5.

Definition 1 (a-b separator) *Given a graph G = (V,E), the an a-b separator is a set of vertices $C \subseteq V$, such that there is no path from a to b in G-C.*

We can use the idea of a-b separator to define our contiguity constraint. Suppose a unit a is assigned to b, and the solution satisfies the contiguity constraint; i.e. there is a path from a to b, then since an a-b separator C is able to disconnect a and b, we know that at least one of the vertices in C is also assigned to b.

Therefore, the necessary condition of a solution which satisfies the contiguity constraint is that for any a-b separator, at least one of the vertices in it is also assigned to b. We can easily prove the opposite direction (sufficient condition for contiguity) by proving the contra-positive statement.

With the above statement, we can formulate the CUT model as the following:

$$x \in P_{\text{Hess}} \tag{5a}$$

$$x_{ab} \leq \sum_{c \in C} x_{cb} \qquad \forall a\text{-}b \text{ separator } C, \forall a \in V \setminus \{b\}, \forall b \in V \tag{5b}$$

Constraint (5b) makes sure if for every a-b separator C at least one of the vertices in C is also assigned to b, then a can be assigned to b . This is the sufficient condition for contiguity. One may notice that there can be a lot of constraints in (5b); therefore, it would be appropriate to implement a cutting plane method, and add some violated constraints on the fly. We will discuss the main separation problem next.

3.4.1 Fractional Separation

Here we discuss how the separation works when we are dealing with relaxed solutions where x variables can be fractional in the branch and bound process. Note that in the original problem formulation, x variables are binary and represent the choices of assigning a unit to another. In Oehrlein's paper [4], they define a separation problem that aims to identify the most-violated inequalities for constraint (5b). The method to find the most-violated inequality is the same as that of finding the minimum vertex cut. The following is the reasoning. Suppose we are now trying to find the most violated inequality for a pair of nodes (a, b) that corresponds to the (a, b) in constraint (5b), and let the weight for each vertex v be value x_{vb} (current relaxed solution value) when finding the minimum vertex cut. Let K be the minimum vertex cut for node pair (a, b). By definition, we know K is also an a-b separator. If there is another a-b separator C that violates constraint (5b), then we know that

$$x_{ab} \geq \sum_{c \in C} x_{cb} \geq \sum_{k \in K} x_{kb}$$

The first inequality shows the constraint violation for (5b) of the a-b separator C, and the second inequality comes from the fact that K is a minimum vertex cut with sum of vertex weights equals $\sum_{k \in K} x_{kb}$. Therefore, we can identify that for the choice of nodes (a, b), if there is any a-b separator that violates constraint (5b), then the most violated inequality would be

$$x_{ab} \leq \sum_{k \in K} x_{kb}$$

To sum up, we can find the most violated inequality for every choice of (a, b) pairs by solving a minimum vertex cut problem.

However, as the size of problem grows, the separation problem will also grow quickly. In addition, maybe we do not need to add the most violated inequalities for all (a, b) pairs. We can just choose the most probable ones. We would like to choose (a, b) if x_{ab} and x_{bb} are large enough ; i.e., the probabilities of a being assigned to b and b is chosen to be a district center are high. The implementation details will be included in the appendix A.

3.4.2 Integer Separation

When an integer solution is found, we can also try to find the most violated inequalities. It turns out that the separation problem for fractional solution can be applied directly to the integer separation. We can find the most violated inequalities in integer solution by solving minimum vertex cut problem, and we can observe that if there is any violated inequality for an (a, b) pair, then the minimum vertex cut K for it will have sum of vertex weights equals $\sum_{k \in K} x_{kb} = 0$.

4 Experiments

In this section, experiments are conducted to compare the performance among all the contiguity methods mentioned in the previous section. All of the experiments were performed on a machine with an Intel Xeon E5-4640 v2 2.20GHz CPU with 40 cores and 256GB RAM.

4.1 Dataset

For the experiments, I design an adjustable synthetic data set. The synthetic data is created as an $m \times n$ rectangle, where each grid in the rectangle is a unit, and the adjacency among units is derived directly from the adjacency of grids in the rectangle. Specifically, $(m, n) = (5, 8), (6, 9)$ and $(7, 10)$ in our dataset. For each unit, it has a randomly generated population from the interval $[10, 300]$. For the population bound in constraint (1d), I tried two set of bounds for the dataset. Let the average population for a given synthetic dataset be p. One bound is $(L, U) = ((1 - 0.02)p, (1 + 0.02)p)$, and the other is $(L, U) = ((1 - 0.03)p, (1 + 0.03)p)$. Lastly, I let the models assign units into 4, 6 and 8 districts.

4.2 Methods

The methods implemented are the follows:

1. HESS : Since the Hess model does not consider contiguity constraint, we can compare the solution time of it to those that have contiguity constraints.
2. SHIR : This serves as the **baseline** method for our experiment. It is also the earliest developed model among those we introduced.
3. MCF : It provides stronger formulation compared to SHIR. Therefore, it would be interesting to see if adding more variables and constraints in MCF makes the performance better.
4. DB : Method inspired by Professor Thomas Rutherford's lecture notes. We can still look at the performance of it, though the solution space is different from the other methods.
5. CUT : The CUT model with separation problem solved only at integer solutions; i.e., the violated inequalities are added only as lazy constraints.
6. CUTALL : The CUT model with separation problem solved at both integer and fractional solutions. The violated inequalities are added as both lazy constraints and cuts.
7. CUTPRE : I run several loops to add cuts by solving the relaxed problem (here I set the number of loops to be at most ten). Then, after all the cuts are "pre-added", I then solved the original un-relaxed problem, and added the violated inequalities as lazy constraints.

(m,n)	k	w/o contiguity		w/ contiguity						
		Obj.	HESS	Obj.	SHIR	MCF	DB	CUT	CUTALL	CUTPRE
loose population bound $(L,U) = (0.97p, 1.03p)$										
$(5,8)$	4	51.72	0.50	51.72	1.13	40.03	0.63	1.76	7.19	1.67
$(5,8)$	6	41.03	0.70	4.12	2.53	45.40	1.13	1.53	11.52	1.43
$(5,8)$	8	37.54	6.44	37.72	10.93	109.05	4.24	9.34	472.63	6.25
$(6,9)$	4	77.46	0.25	77.46	0.50	45.90	0.43	2.03	2.28	1.89
$(6,9)$	6	61.22	0.55	61.22	1.73	113.40	1.02	2.59	9.55	2.50
$(6,9)$	8	56.04	4.59	56.04	10.86	266.17	7.28	7.12	386.64	6.24
$(7,10)$	4	115.43	0.6	115.43	2.51	171.18	0.85	2.91	6.25	2.85
$(7,10)$	6	92.53	1.49	92.53	4.94	256.96	3.21	5.90	12.60	5.46
$(7,10)$	8	81.35	6.72	81.35	12.48	511.99	5.92	7.65	256.59	8.11
tight population bound $(L,U) = (0.98p, 1.02p)$										
$(5,8)$	4	51.72	0.99	51.72	1.95	34.11	0.66	1.59	11.68	1.58
$(5,8)$	6	41.03	0.70	42.37	4.58	101.81	2.33	3.83	159.01	3.63
$(5,8)$	8	38.79	11.07	-	TL	TL	TL	TL	TL	TL
$(6,9)$	4	78.04	0.31	78.04	1.00	56.18	0.52	1.01	1.55	1.07
$(6,9)$	6	62.82	1.04	62.87	2.99	113.20	2.46	2.92	21.95	2.83
$(6,9)$	8	57.04	19.31	57.10	34.26	TL	16.55	16.97	TL	27
$(7,10)$	4	115.43	0.47	115.43	2.01	149.26	0.78	2.81	6.23	2.80
$(7,10)$	6	92.53	1.88	92.53	6.00	213.94	2.73	4.16	14.88	4.27
$(7,10)$	8	82.17	23.37	82.17	24.71	TL	15.86	17.95	TL	17.13

Table 1: Time and objective values of experiments with different population bounds. The p in the population bound is the average population for each unit in the synthetic data set. (m,n) represents the size of rectangle. There will be $m \times n$ units to be assigned into k districts. The running time for each model is in seconds. TL denotes that 10 minutes time limit was reached and the methods do not reach the optimal solution.

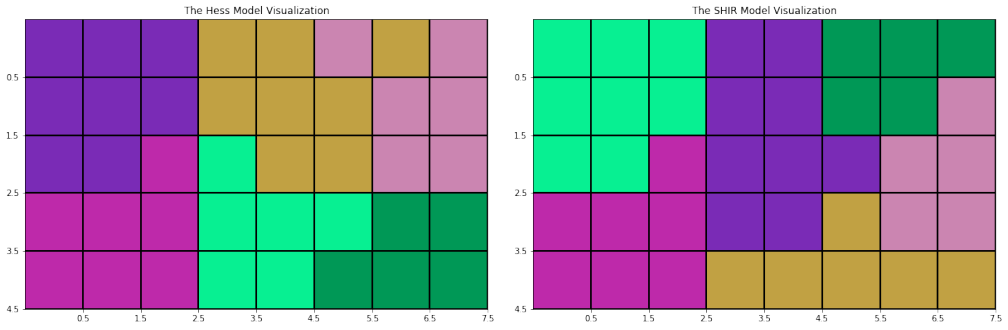

Figure 6: Example of SHIR model successfully cutting off infeasible solution from the Hess model's solution. The example is the result when we apply tight population bound , $(m,n) = (5,8)$ and $k = 6$. Note that in table 1, the objective value of Hess is smaller than that of SHIR in this case.

4.3 Results

4.3.1 Model without Contiguity vs. Models with Contiguity (Correctness of implementation)

In section 2, we mentioned that the Hess model can produce contiguous or nearly contiguous solution due to the nature of its objective function. By looking at the objective value, we can see that in many scenarios, the objective value of the Hess model indeed is the same as those of the other models with contiguity constraints.

In some cases, the objective value of the Hess model is lower. This means that the dis-contiguous solution of the HESS model is successfully cut off by imposing the contiguity constraint . The example in figure 6 demonstrates this, and it also shows that our implementation of contiguity is correct.

4.3.2 SHIR (baseline) vs. MCF

MCF provides a stronger formulation for the Steiner tree formulation by introducing more variables and constraints. Sometimes stronger formulation can yield better results, but according to table 1, we can see that the running time of MCF is a lot worse than that of SHIR, since the problem size of redistricting is larger than a simple Steiner tree problem. Therefore, the benefits of a stronger formulation do not out-weight the cost brought by the increase of problem complexity in this case.

4.3.3 SHIR (baseline) vs. CUT

In the result with tighter population bounds in table 1, CUT's performance is generally better than that of the SHIR, and under loose population bounds, they have comparable performance. From the result, we can say that CUT generally performs better under the problem size of assigning less than 70 units and strict population restriction. Though it is harder to implement CUT, but it is worth it in this case.

4.3.4 SHIR (baseline) vs. DB

In table 1, we can see the DB has the best performance among all methods, but I think that might be because the synthetic data is perfectly shaped as a rectangle with square grids. Unexpected error may happen when one unit is completely contained in another. In addition, as explained in section 3, the feasible region of DB is different from the other model; therefore, more studies are needed to discuss the stability of the DB method.

4.3.5 CUT, CUTALL and CUTPRE

CUTALL and CUTPRE can be regarded as potential improvments of CUT.

CUTALL is CUT with extra cut adding at the fractional solutions. The result shows solving the separation problem for fractional solutions worsens the performance. We can also see that in table 1, the running time of CUTALL increases rapidly with the increase of district number k. CUTPRE aims to solve relaxed problem beforehand and add cuts that might be useful for the solving of CUT model. In the result, the performance does not improve in CUTPRE.

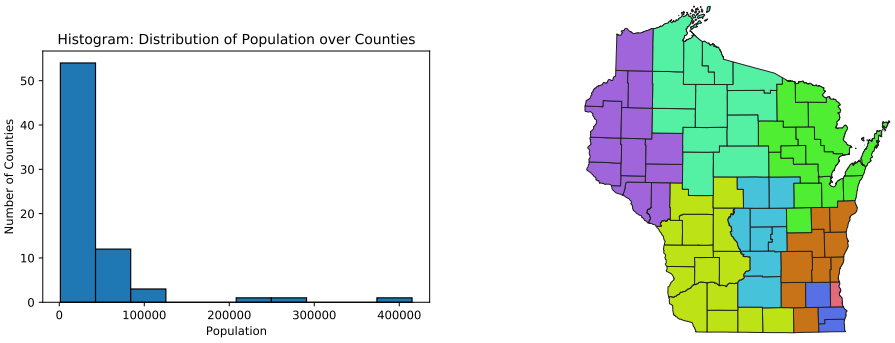

Figure 7: Population distribution and the final map.

5 Case Study

In this section, CUT method will be applied to the Wisconsin county-level dataset. Also, some difference between the synthetic data and real world data will also be discussed. The data I used contains three part:

1. Population data[2] (Voting data for 2016 US election in Wisconsin is used as population data.)
2. County adjacency data [3]
3. Wisconsin county-level shape data[4]

The Wisconsin county-level data contains 72 counties, and we want to assign them into 8 district, when the number 8 comes from the number of congressional seats in Wisconsin. In figure 7, we can see that the populations of the 72 counties varies a lot. Therefore, when we are choosing population bound for our optimization problem, it is usually loose. For example, the experiment here uses the bound $(L, U) = (0.8p, 1.2p)$, which allows a 40% variation with respect to the average population p. The graph on the right in figure 7 demonstrates the result of the final assignments, and again shows that out optimization model is able to create contiguous map. We can see the in real world dataset, the shape of each basic unit (county) are not perfect.

6 Conclusion and Lessons Learned

In this project, I implemented 6 models that are able to generate contiguous solution for a re-districting problem. From the flow based-method, SHIR and MCF, I learned to use Steiner tree formulation to ensure contiguity, and discover the trade-offs between using stronger formulation and the increase of problem size. I have implemented DB which has the best performance among all the methods, but the stability of DB still needs more explorations. In addition to directly modeling contiguity constraint, I implemented cutting plane method, CUT, and explore its derivation, CUTALL and CUTPRE, using Gurobi in python. The performance of CUT method tells us that sometimes spending a bit more work to implement separation algorithm might be rewarding. Lastly, I implemented CUT method on Wisconsin county-level dataset.

[2]Origin : https://www.opendatasoft.com/ (The data is removed recently.)
[3]Origin : https://www.census.gov/geographies/reference-files/2010/geo/county-adjacency.html
[4]Origin : https://data-wi-dnr.opendata.arcgis.com/datasets/county-boundaries-24k/explore

References

[1] J. Chen, J. Rodden, et al. Unintentional gerrymandering: Political geography and electoral bias in legislatures. *Quarterly Journal of Political Science*, 8(3):239–269, 2013.

[2] B. Fifield, M. Higgins, K. Imai, and A. Tarr. Automated redistricting simulation using markov chain monte carlo. *Journal of Computational and Graphical Statistics*, 29(4):715–728, 2020.

[3] S. W. Hess, J. Weaver, H. Siegfeldt, J. Whelan, and P. Zitlau. Nonpartisan political redistricting by computer. *Operations Research*, 13(6):998–1006, 1965.

[4] J. Oehrlein and J.-H. Haunert. A cutting-plane method for contiguity-constrained spatial aggregation. *Journal of Spatial Information Science*, (15):89–120, 2017.

[5] T. Shirabe. Districting modeling with exact contiguity constraints. *Environment and Planning B: Planning and Design*, 36(6):1053–1066, 2009.

[6] H. Validi, A. Buchanan, and E. Lykhovyd. Imposing contiguity constraints in political districting models. *Operations Research*, 2021.

A Algorithm

Algorithm 1: Separation Algorithm

Data: Adjacency graph $G = (V, A)$ for the underlying districting problem. V is the set of all units sorted by x_{vv} , $v \in V$ of the current relaxed solution in descending order

Result: Cuts of most violated inequalities added

$c \longleftarrow V.first()$;

$hasAdditionalCuts \longleftarrow$ false;

while $x_{cc} \geq \frac{1}{|V|}$ *and* $\neg hasAdditionalCuts$ **do**

  Let the graph $G_m = (V, A)$ be the same as G except that each vertex $v \in V$ has a weight of x_{vc};

  $U \longleftarrow \{v \in V :$ v is not c, $(v, c) \notin A$ and $x_{vc} \geq \frac{1}{|V|}\}$;

  for $u \in U$ **do**

    $S \longleftarrow$ find the minimum vertex cut for (u, c) in graph G_m

    if $\sum_{v \in S} x_{vc} \leq x_{uc}$ **then**

      addCut($\sum_{v \in S} x_{vc} \geq x_{uc}$);

      $hasAdditionalCuts \longleftarrow$ true;

    end

  end

  $c \longleftarrow V.next()$;

end

The main idea is to loop over all (a, b) pairs where $a, b \in V$, and find their minimum vertex cuts. The twist is what we mentioned in section 3 when we are trying to do the fractional separation. We do not want to add cuts for all (a, b) pairs, but only add those whose x_{ab} and x_{bb} values are big enough (Implemented as $x_{cc} \geq \frac{1}{|V|}$ and $x_{vc} \geq \frac{1}{|V|}$). I choose to not talk about the implementation details of finding the minimum vertex cut here because it may distract readers from the main idea of the separation process itself. The problem of finding minimum vertex cut can be transformed to finding minimum cut by constructing an auxiliary graph of the graph G_m. More details for finding minimum vertex cut can be found in the appendix section of Oehrlein's paper [4].