Choosing a fair map under adversarial attacks

Cheng-Wei Lu

Contents

There is an ongoing discussion about making the congressional map fairer in Wisconsin. Historical data has shown that the US House election result is in favor of Democrats before 2010 while it is in favor of Republicans after 2010 when the Wisconsin district maps were redrawn. Some even argue that the current map is "Gerrymandered". Gerrymandering refers to a strategy where politicians try to maximize the seats they get by redistricting and manipulating district boundaries. Therefore, we are interested in designing a framework that allows the user to choose the fairest map under the adversarial attack of election policies and the changes in raw votes. When legislators are proposing a map, it should follow three main guidelines - compactness, population equivalence, and contiguity. Many algorithms have been developed to generate maps that satisfy those guidelines. In this work, we modified one of the existing algorithms and relax the population equivalence constraint, which allows different population ratios to be assigned to different districts. We further look into the two most adopted vote seat methods - winner-take-all and proportional policies and investigate the impact of different population ratios and policies on the result of the election using the US House election data in Wisconsin. Finally, we propose a map selection framework based on a stochastic risk model that allows a map to be selected that preserves fairness among adversarial-defined voting policies, with the assumption that the election results can be "attacked" by adopting winner-take-all or proportional schemes.

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 the 1810s with Elbridge Gerry, the governor of Massachusetts at that time, signing a bill that created a partisan district in the Boston area that was compared to the shape of a mythological salamander. There are still ongoing discussions about how Gerrymandering is affecting elections such as the map in Wisconsin during the recent United State House election. For example, it is in favor of Democrats before 2010 while it is in favor of Republicans after 2010. One interesting fact is that there was a district redrawing between the election of 2008 and 2010, so it is possible that the current map is more beneficial to Republicans. Some data to support this claim is shown in table 6. In this work, we are interested in designing a framework that allows the user to choose the fairest map under an adversarial attack of election policies and changes in vote distribution.

A huge research focus is on how to generate good district maps. A decent map should at least follow three main guidelines - compactness, population equivalence, and contiguity. 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 shapes, we would prefer the ones that have round shapes. Numerous metrics are proposed in order to quantify the compactness of districts. In the method proposed by Hess [10], they use the sum of the distance of all units to their assigned district center as the metric of compactness. Some other metrics quantify the similarity between the district shape and a circle which is often regarded as the most compact shape. For example, the Polsby-Popper [14] score calculates the ratio of the area of a district to the circle that shares the same perimeter with it. Schwartzberg [16] score is similar to the Polsby-Popper score except that it calculates the ratios of the perimeter of a district and a circle that shares the same area. The second goal is population equivalence constraints, which make sure the populations among districts do not differ too much. Most generation algorithms require the districts to have about the same population. There are two ways to impose the population equivalence constraint in the solution process. One way to impose the population equivalence constraint is to add it as a constraint in the mixed-integer program(MIP) as in Hess [10] or use it as a stopping criterion in the map generation algorithm as in Chen's work[4]. In this case, we are able to specify a population bound when we get a map from the above-mentioned methods. The other way to consider the population equivalence constraint is to add it as a penalty term in the objectives as shown in Gutiérrez-Andrade [9], where the deviation from the optimal population distribution is encoded as a penalty term in the function of a simulated annealing process. The last goal is the contiguity constraint, which

guarantees that all units inside a district form a connected graph where the edges in that graph represent the geographical adjacency among units. It is possible to incorporate the contiguity constraint into the MIP model as network flow constraints as shown in Shirabe [17]. Contiguity is also maintained in heuristic approaches by rejecting a given step which may lead to dis-contiguity as in [4]. Most research in political districting follows the three guidelines in order to generate candidate maps either for recommendation or analysis.

There are two main research directions in Gerrymandering. The first direction aims to formulate direct mixed-integer programming (MIP) models in which contiguity is imposed in the form of constraints. In the integer programming model, the goal is to solve for a single optimal map. The objective is typically maximizing the compactness and at the same time satisfying population equivalence and contiguity constraints. For example, the Hess model [10] aims to minimize the sum of the distance of all units to their assigned district center and find a solution that is also feasible for only the population equivalence constraint. A lot of research then derives from the Hess model and explores ways to impose contiguity constraints on the MIP model. Most of them achieve that goal by adding extra constraints to the original IP of the Hess model. One example is Shirabe's model mentioned above. In Validi's work [18], the author proposes a numerically stronger formulation compared to Shirabe's model, where they derive a network-flow-based formulation that has tighter lower-bound for the original problem by avoiding the use of the big-M constraints. The result shows that the benefits of deriving a better formulation do not outweigh the computational complexity brought by the increase in the number of constraints and variables in this case. There are also formulations that utilize the cutting plane method to satisfy the contiguity constraint. In Oehrlein's work [13], the author uses the idea of a-b separator to formulate an exponential number of constraints. Then, they provide an effective way to solve the separation problem by finding the minimum vertex cut of a pair of units. The above-mentioned methods all aim to find a single optimal map by formulating an IP problem that considers the three main guidelines - compactness, population equivalence, and contiguity constraints. However, as the problem scales, the MIP problem becomes intractable. Therefore, it leads to the second direction of designing a map generation algorithm.

The second direction aims to design algorithms that randomly generate maps that satisfy the above-mentioned goals. Generating multiple maps with algorithms allows us to understand the whole solution space better and analyze maps with different metrics such as party-wise voting ratio. Having multiple maps also allows us to choose a potentially fair map given the future fluctuation of the vote distribution. People are exploring the heuristics to generate candidate maps that satisfy the three guidelines mentioned above. For instance, Kim [11] developed a greedy local search algorithm to generate maps on the county-level data. Chen [4] designs a local search method that starts from an initially compact map and gradually moves the boundary units in order to satisfy the population equivalence constraint without violating the contiguity constraint. They further made an observation of two kinds of Gerrymandering - intentional Gerrymandering and unintentional Gerrymandering. Unintentional Gerrymandering comes from the dense distribution of votes for a party whose population is mostly located in a few precincts. Evolutionary methods and simulated annealing methods also appear in the literature. In Liu's work[12] they develop a method based on an evolutionary process that utilizes crossover to add randomness into the map generation process. Gutiérrez-Andrade [9] develops a simulated annealing method that considers the co-optimization of compactness and population equivalence objectives. Recently, Markov chain Monte Carlo (MCMC) methods are also being applied in the area of map generation. The MCMC method works as follows: Given an initial feasible solution, the heuristic will perform random walks to generate new districting maps. The goal of the MCMC-type method is to sample from the global map distribution nicely. If we can sample from the distribution uniformly, we can analyze the quality of a given plan. In Fifield's work [7], the author propose a random walk method for doing MCMC sampling, and then discuss the variation of the algorithm after adding more constraints to it. The author randomly chooses edges in the graph and identifies connected components in that graph. Then the random walk is conducted by using those connected components. In Cho's work [5], the author conducts experiments on a Monte Carlo (MC) method and the MCMC method, and the result implies that the MCMC method is better after the aggregation of different starting solutions on Florida

data. In DeFord's work[6], they propose the ReCom method that combines two adjacent districts and then utilizes the idea of a spanning tree to perform bi-partitioning among the units of the two districts. The result shows that ReCom is better than the simplest Flip method mentioned in their work. The development of map generation methods is an active area, where researchers try to find computationally less expensive ways to generate high-quality maps.

In this paper, we are firstly interested in the effects of the election policies and the number of districts on the result. For example, Wisconsin currently adopts a winner-take-all policy for the US House seat. Therefore, the total congressional seats for a party are calculated based on the number of districts where that party has the majority of votes. However, there exist different policies for congressional seat assignment. The proportional policy is the one we are particularly interested in. We would like to see how each system performs on real-world voting data, and analyze their advantages and disadvantages.

In order to allow the analysis of different election policies, it is necessary that we generate maps that have different population ratios for the proportional policy to work. That is, we would like to generate maps that allow multiple representatives. For example, Wisconsin has 8 US house seats and it currently has 8 US house districts, each of which is assigned only 1 seat. In the case where multiple representatives are allowed, we can have 7 districts where one of the districts will be assigned 2 seats and the rest will be each assigned 1 seat. Intuitively, the district with 2 seats should have about 2 times the population of the districts that are only assigned 1 seat. Most previous methods aim to generate population-balanced districts where every district has about the same population, including the ones that have been mentioned above. Our map generation is a variation of Chen's [4] work where the population equivalence constraint is relaxed so that districts can have different population weights (population balanced). There are some benefits of developing a map generation algorithm that allows different populations for different districts, such as parallel computations. We will describe its importance in the conclusion and future work section. With the modified version of the map generation algorithm, we are able to conduct the analysis of election systems.

In addition to analyzing different election systems, we are able to use the generated maps to make recommendations under different scenarios. For example, here we can make an assumption that we are unsure about both the vote distribution of every unit and also the election policy in the coming elections. Therefore, we can formulate a two-stage stochastic model to recommend the fairest map that does not have any bias towards any party. Different risk measures can also be incorporated into the recommendation process, such as average risk and conditional value at risk (CVaR).

To sum up, here are some contributions we made in this work:

1. Propose a map generation algorithm that allows population unbalanced districts
2. Analyze the performance of different election systems under differently populated districts
3. Propose a framework to make recommendations on maps using two-stage stochastic models based on map compactness and fairness risk.

The contributions will be presented in the following sections. In section 2, we will introduce the basic notation and the MIP problem related to the design of our map generation algorithm. In section 3, we will provide a detailed description of our algorithms, and discuss how we identify maps with better qualities in our map pools. In section 4, we will introduce different existing election policies and how seats will be assigned under each policy. In section 5, we will introduce a two-stage stochastic map evaluation framework that recommends a fair map with respect to different district numbers by using various risk measures. In section 6, we report the details of our experiments, including the basic setting of our map generation process, a comparison between the winner-take-all policies, and a demonstration of our map choice framework. Finally, in section 7 and 8 we will talk about some conclusions and possible future work.

2 Preliminaries

In this section, we will first introduce graph notations related to the districting problem. Then, we introduce a general MIP problem formulation of optimally compact map generation in order to connect with the goal of our map generation algorithm.

2.1 Graph representation of districting problems

A districting problem is often formulated as a graph partitioning problem that is subject to both contiguity and population constraints. Suppose we would like to have N districts in our redistricting map. Let $G = (V, E)$ be a graph that represents a state where we are going to perform redistricting. Each $v \in V$ is the smallest **unit** for the partition problem and is also associated with a population value p_v and a centroid coordinate (x_v, y_v). The number of units, $|V|$, varies depending on different use cases. Take Wisconsin for example. Two possibilities include performing redistricting on the county level (72 counties) or the census-tract level (1409 census tracts). The edge set E is used to represent the geographical adjacency of the units in V. Let $D = \{1, \ldots, N\}$ be the set of districts we would like to form

Definition 1 ***Adjacent***. *We say two units are* ***adjacent*** *if they share at least a border whose length is strictly larger than 0. In other words, we do not consider two units that only overlap at a single point to be adjacent. Let $u, v \in V$, we say u and v are adjacent if and only if $(u, v) \in E$.*

Definition 2 ***Assignment***. *An assignment $f : V \to D$ is a many-to-one and onto mapping from a unit to its assigned district, and each element in $dom(f)$ is defined. For example, $f(v) = d$ means the unit $v \in V$ is assigned to district $d \in D$.*

We can then define $H(\cdot, \cdot)$ where $H(d, f) = \{v | f(v) = d, v \in V\}, d \in D$ and f is a mapping function mentioned above. $H(d, f)$ can be understood as the set of units assigned to district d under the assignment function f. The goal is then to find a valid mapping f such that given any $d \in D$, the total population of district d represented by $\sum_{v \in H(d,f)} p_v$ satisfies the population bound, and the set of units in $H(d, f)$ formulates a connected graph.

Definition 3 ***Connected district***. *We say that the district d is connected under the assignment f if $H(d, f)$ formulates a connected graph. That is, for any $u, v \in H(d, f)$ and $u \neq v$, there exists a path $(u, n_1, n_2, \ldots, n_{r-1}, n_r, v)$ such that $(u, n_1) \in E$, $(n_r, v) \in E$ and $(n_i, n_{i+1}) \in E$ where $i = 1, \ldots, r-1$, and $n_i \in H(d, f)$ where $i = 1, \ldots, r$. The parameter r here is a non-negative integer.*

With the notation defined here, we are introduce the problem formulation for the map generation algorithm in the next section.

2.2 Problem formulation for map generation algorithm

In this section, we will use the notations defined in the section 2.1 and introduce a widely used MIP model to generate an "optimal map", the one that maximizes compactness. We assume that we would like to perform redistricting on a state represented by $G = (V, E)$ here.

The first important thing in our work is to generate maps with good quality. As mentioned in the introduction, there are two directions in redistricting research. One is solving a MIP model for an exact optimal map,

and the other is designing a map generation algorithm that produces feasible maps efficiently. In this paper, we are following the second direction where we aim to generate a pool of feasible maps. However, since both directions are closely connected, it is helpful to understand the goal of a map generation algorithm from a MIP formulation point of view. Therefore, we will start by introducing MIP formulations of the optimal map-searching model.

In section 1, we have introduced the three basic guidelines of a decent map.

1. compactness
2. population equivalence
3. contiguity

Many works ([13, 18, 17]) focus on formulating a model that considers all three guidelines, and they all share a similar structure, as shown in the section 2.2.1.

2.2.1 Direct model structure for finding an optimally compact map

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:

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

(The distance between two units is 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:

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

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

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

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

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

contiguity constraints (1g)

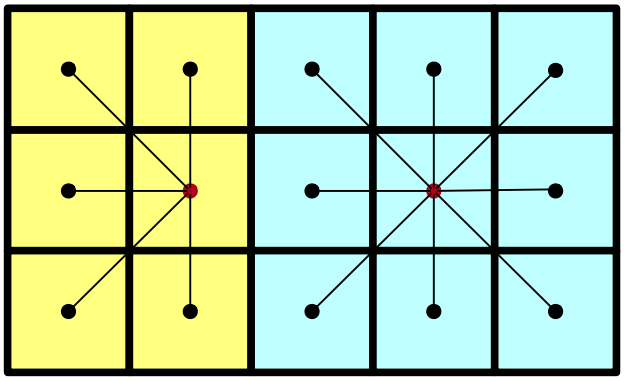

Figure 1: Objective function for the optimal MIP model. Suppose there are two districts. One in blue and the other in yellow, and the district centers are units that contain the red centroid point. The objective value is the sum of the distance between each unit and its assigned district center's centroid; i.e., the sum of thin black lines length in the figure.

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 satisfies the population bound. Constraint (1e) ensures that unit i can be assigned to unit j only if unit j is chosen as a district center. Constraint (1g) makes sure every district generated is contiguous. The objective (1a) of the model here is chosen to be minimizing the sum of the distance between every unit in a district map and its assigned district center. The visualization of how the objective value is acquired is in figure 1. The constraints (1b)-(1f) along with the objective (1a) come from the Hess Model[10, 18], which aims to find a map that maximizes compactness and satisfies population constraints. The Hess model does not guarantee contiguity, so we need an extra set of constraints (1g) to ensure contiguity [13, 18, 17]. The contiguity constraints will utilize the information of adjacency encoded in the edge set E. Details of the explicit form of the contiguity constraints are non-trivial and will be introduced in the appendix A. We also have done experiments of imposing contiguity constraints on a set of smaller-sized synthetic problems [2].

2.2.2 Goal of generation algorithm

In section 2.2.1, the stated model structure aims to maximize the compactness and at the same time satisfy the contiguity and population constraint. Therefore, our map generation should also follow the same direction. We would like to have a method that generates a map that is feasible for the problem specified in section 2.2.1. Also, we want to make sure that the algorithm maintains compactness as much as possible at every step. In the next section, we will introduce the main generation algorithm.

3 Generation algorithm

We've developed an algorithm that allows different population ratios among districts. The algorithm is inspired by the methods proposed in Chen's work [4], where the author proposed local search methods to effectively generate feasible maps while maintaining the compactness objective.

Our algorithm has two parts. The first part generates a random initial graph that preserves compactness by

combining graph units in random order and using the information of unit coordinates. After the first part, the initial graph may violate the population bound. Therefore, the second part uses the initial graph and modifies it in order to satisfy the population bound. We provide the details of the algorithms below.

3.1 Algorithm Description

Algorithm 1: $initMap(G, N)$: Generate initial districts

Input: A graph $G = (V, E)$ for a given state, and N as the number of districts to create. Each $v \in V$ also has the coordinate (x_v, y_v) of its geographical center (Let $A(v) = \{n|((n, v) \in E) \vee ((v, n) \in E)\}$ denote the set of neighboring vertices of v) ;

Output: An initial map with N districts that satisfies the contiguity constraint

while $|V| > N$ **do**

Randomly choose $v_1 \in V$, and let $v_2 \in A(v_1)$ be the closest point to v_1 among its adjacent neighbors;

Create a dummy vertex $v_{1\&2}$ that represents the merged node of v_1 and v_2;

Calculate and keep track of the centroid coordinate $(x_{v_{1\&2}}, y_{v_{1\&2}})$ of $v_{1\&2}$;

$E' = \{(v, v_{1\&2})|v \in A(v_1) \vee v \in A(v_2), v \notin \{v_1, v_2\}\} \cup \{(v_{1\&2}, v)|v \in A(v_1) \vee v \in A(v_2), v \notin \{v_1, v_2\}\}$;

$D_{v_1} = \{(v, v_1)|v \in A(v_1)\} \cup \{(v_1, v)|v \in A(v_1)\}$;

$D_{v_2} = \{(v, v_2)|v \in A(v_2)\} \cup \{(v_2, v)|v \in A(v_2)\}$;

$E \longleftarrow (E \cup E') \setminus (D_{v_1} \cup D_{v_2})$;

$V \longleftarrow (V \cup \{v_{1\&2}\}) \setminus \{v_1, v_2\}$;

Now the node $v_{1\&2}$ representes the merged node of v_1 and v_2;

end

return G;

The first part of the generation algorithm is described in the algorithm 1. In algorithm 1, the following steps were done iteratively:

1. Randomly choose a unit v_1 in the current graph G.
2. Identify the set $A(v_1) = \{n|((n, v_1) \in E)\}$, which is the set of units that are adjacent to unit v_1.
3. Identify a unit v_2 in $A(v_1)$ that has the shortest distance from v_1. The distance can be calculated using the unit coordinates.
4. Merge v_1 and v_2 and call the new unit $v_{1\&2}$. Add the new unit into the graph and also link it to the units that are adjacent to v_1 or v_2.
5. Remove the units and edges related to v_1 and v_2.

Each iteration will decrease the value of $|V|$ by 1 because exactly 2 nodes are merged. The iteration stops when the number of final units is exactly N. Then, we can view each final unit as a district, and thus get the initial assignment graph. Empirically, the initial graph looks compact because of step 3. It is also contiguous by construction. However, since the information on population value is never used here, the population constraint may be violated. Therefore, we need the second part to satisfy population feasibility.

Algorithm 2: $genMap(G, N, T, \epsilon)$: Generate feasible maps

Input: A Graph $G = (V, E)$ for a given state, the population p_v for each unit, N as the number of districts to create, and $T_i, i = 1 \dots N$, are the population weights determined by the user. ϵ is the population bound that should be satisfied. Each $v \in V$ also has the coordinate (x_v, y_v) of its geographical center (Let $A(v) = \{n|((n, v) \in E) \vee ((v, n) \in E)\}$ denote the set of neighboring vertices of v) ;

Output: A final map with N districts that satisfies the contiguity constraint and population constraint

---

$initG \longleftarrow initMap(G, N)$;
Match the districts in $initG$ with population weights $T_i, i = 1 \dots N$;
while *True* **do**
  Identify a set $P = \{(d_i, d_j)|\exists u, v \in V, v$ is assigned to d_i and u is assigned to $d_j\}$;
  Among $(d_i, d_j) \in P$, choose the one that has the highest $popDist(d_i, d_j)$ value. Suppose the district pair we found is (d_k, d_l) ;
  if $popDist(d_k, d_l) \leq \epsilon \times \frac{total\ population}{total\ congress\ seats}$ **then**
    break ;
  end
  if $\frac{pop(d_k)}{T_{d_k}} > \frac{pop(d_l)}{T_{d_l}}$ **then**
    $d_{\text{large}} = d_k$;
    $d_{\text{small}} = d_l$;
  else
    $d_{\text{large}} = d_l$;
    $d_{\text{small}} = d_k$;
  end
  Identify a set of vertices $Q = \{v|v$ is assigned to d_{large} and $\exists u \in A(v), u$ is assigned to $d_{\text{small}}\}$;
  Among all $v \in Q$, find the one that has the smallest $compactChange(v, d_{\text{large}}, d_{\text{small}})$ value. Suppose the vertex we found is w;
  Reassign w to district d_{small} ;
  if *district* d_{large} *remains connected* **then**
    pass ;
  else
    Reassign all the connected components except the largest one to district d_{small} ;
  end
end
return G;

The complete map generation process is specified in algorithm 2. The following is the detailed description of the algorithm 2.

1. Run *initMap* (algorithm 1) to generate an initial graph *initG* that splits the whole map into N partitions.
2. Match the population ratios specified by the user with the N partitions generated in step 1.

Steps 1 and 2 are executed just once in the whole process. In step 2, we are assigning a population ratio to each of the partitions in *initG*. For example, suppose $N = 3$ and there are three partitions in *initG*, which are A, B, and C. Let A, B, and C have populations of 20, 40, and 10 respectively. Suppose that the user-specified population ratios are $T_1 = 3$, $T_2 = 2$, and $T_3 = 1$. We can first sort the partitions by population and then sort the population ratios from small to large. After that, we have (C,A,B) and (T_3,T_2,T_1). Then,

we can let all the units in C, A, and B be assigned to districts 3, 2, and 1 respectively. Now we have assigned every unit to a specific district that has a specified population ratio to satisfy.

The remaining steps in algorithm 2 are executed iteratively.

3. Find the pair of adjacent districts (d_k, d_l) that has the largest value of weighted population disparity (WPD), $popDist(d_k, d_l) = |\frac{pop(d_k)}{T_{d_k}} - \frac{pop(d_l)}{T_{d_l}}|$.
4. If $popDist(d_k, d_l) < \epsilon \times \frac{\text{total population}}{\text{total congress seats}}$, terminate the process. If not, go to the next step.
5. Identify the over-populated district between d_k and d_l by finding the larger value between $\frac{pop(d_k)}{T_{d_k}}$ and $\frac{pop(d_l)}{T_{d_l}}$. Let the over-populated district be d_{large} and the under-populated be d_{small}.
6. Identify all units from d_{large} that borders d_{small}. Among all the identified units, assign the one that has the best contribution to compactness to d_{small}. The compactness contribution of unit v is calculated as $compactChange(v, d_{\text{large}}, d_{\text{small}})$ = (Euclidean distance of v and d_{small}'s centroid) − (Euclidean distance of v and d_{large}'s centroid).
7. If d_{large} remains contiguous, go to step 3. If not, identify the connected components in d_{large} and assign all the connected components except the largest one to district d_{small} (more details in section 3.3). Here we defined the size of a connected component to be the number of vertices contained in it. Go to step 3.

Steps 3-7 are done iteratively until the WPD is within a given bound, $\epsilon \times \frac{\text{total population}}{\text{total congress seats}}$. The process aims to reassign a unit at each iteration, which potentially moves the current map closer to population feasibility and at the same time make sure the reassignment does not hurt the compactness badly in step 6. Some of the innovations of our algorithm are further introduced in the following sections.

3.2 Varying population ratios

Different from most implementation in the previous study, our algorithm allows users to generate districts with different population ratios. For example, if we have a state that has 4 congressional votes, and we would like to have 3 districts, we can make one districts have a population ratio of 2 and the other two have a population ratio of 1. This is achieved by imposing a bound on the maximum weighted population disparity (MWPD), the largest WPD among all pairs of adjacent districts. The intuition is that if we are able to find a map when $\epsilon = 0$, that means the MWPD will be zero, and we will have $\frac{pop(d_1)}{T_{d_1}} = \frac{pop(d_2)}{T_{d_2}} = \cdots = \frac{pop(d_N)}{T_{d_N}}$. Since each iteration will aim to decrease the MWPD, it is equivalent to forcing the populations to follow the population ratios.

3.3 Reassigning connected components

In step 7 of algorithm 2, we need to check if d_{large}, the district that gives away one of its units, is still contiguous. This step is different from the original implementation of Chen's work [4], where they only choose the unit that won't violate the contiguity in step 6. Empirically, it may lead to a district shape that contains long and thin branches as shown in Figure 2. A similar observation is also made in Gutiérrez-Andrade's work [9], where the author argues that the optimal solution will never be found if rejection based on disconnection is incorporated into the algorithm process. Therefore, we need to fix the disconnected solution if any appears. One possible way is to first identify the connected components in the disconnected graph of d_{large}. Then, we can keep the largest connected component in d_{large} and assign the other connected components to d_{small}. An example is shown in Figure 3.

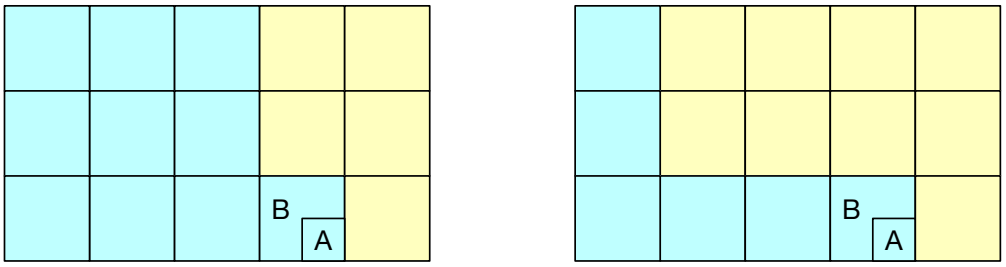

Figure 2: A long branch example that can happen in real dataset. The left graph is the original assignment. Suppose the blue district is over-populated and needs to assign border points to the yellow district. Even if unit B has the best compactness contribution, B can never be switched to the yellow district because the blue district will be disconnected. After many iterations, we will get the result like the right graph.

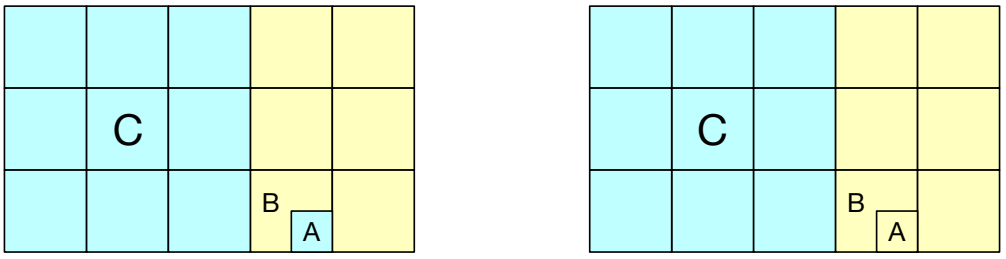

Figure 3: An example of assigning connected components. Here we can follow the example from Figure 2. Since we allow the violation of contiguity in a district, we can first assign B to the yellow districts (left graph). Then, we can identify connected component A and C. We then keep the one that has large number of units (connected component C) in the blue district, and assign A to yellow district as shown in the right graph. Note that the contiguity is still maintained.

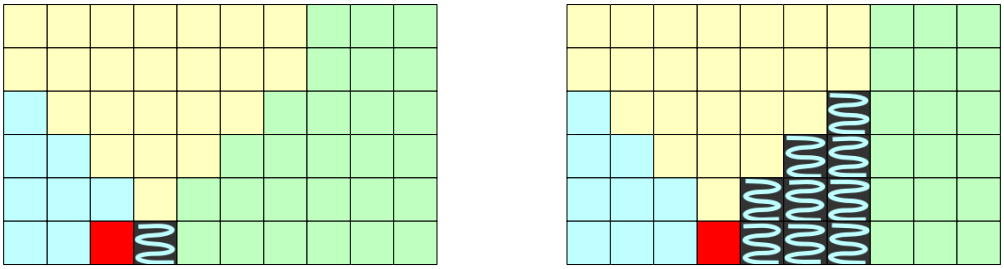

Figure 4: Demonstration of the calculation for DS. It follows from the example in figure 5. DS of the left graph is the largest branch by removing the red unit, which will have a length of 1 (the scribbled unit). The DS score on the right graph is found after removing the red unit and has the value of 9, which is the number of the scribbled units.

3.4 Choosing high-quality solutions

In this work, we also identify a metric to quantify the compactness of graphs generated by a local search method like Chen's method [4], where a border point is swapped between two adjacent districts at each iteration. An algorithm of this kind can generate districts whose shape resembles an hourglass, which was not discussed in the papers that implement similar local search methods to the best of our knowledge. An example is shown in Figure 5. This kind of phenomenon can decrease the compactness of the generated maps; therefore, we propose a metric, disconnection score (DS), to quantify this phenomenon. Intuitively, the DS of a map gives information on the "largest branch" in a districting map, which is good at identifying quality solutions in our case.

Given a map that contains N districts, we can calculate its DS as follows. For each district d, we can find its longest branch by removing a unit at a time and then sum up the size of the second largest to the smallest connected components if the graph of d becomes disconnected. The longest branch of district d will be the largest sum among all the unit removal. Then, DS will be chosen to be the size of the longest branch among those N districts. We provide an example for DS calculation in figure 4.

4 Election policy

District	Congress Seats	REP Raw Votes	DEM Raw Votes	Winner-take-all		proportional	
				REP	DEM	REP	DEM
D1	2	130	70	2	0	1	1
D2	2	40	160	0	2	0	2
D3	1	45	55	0	1	0	1

Table 1: An example for winner-take-all and proportional policies. There are three districts D1, D2, and D3. The congress seats column contains the congressional votes assigned to each district. The winner-take-all and proportional columns contain the final assignment of the congressional seats to the Republican or Democrat party. In the winner-take-all policy, the winner in a district can take away all the congressional seats. In the proportional policy, the seats are assigned according to the voting ratio. For instance, Republican gets $2 \times round(130/(130+200)) = 1$ seat in D1 and $2 \times round(40/(40+160)) = 0$ seats in D2.

In this section, we are going to introduce the election policies discussed in this work. In particular, we are interested in the two policies - the winner-take-all and the proportional policies.

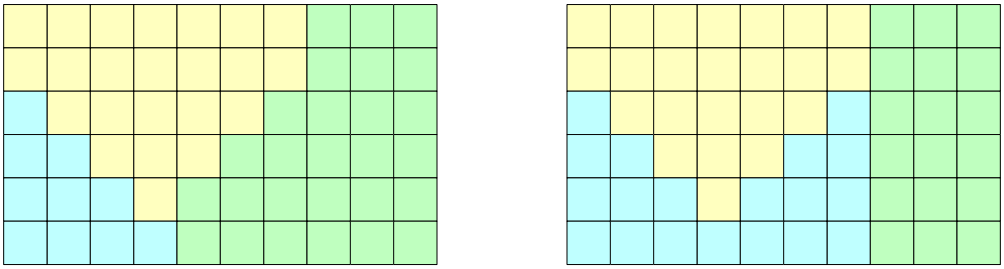

Figure 5: Example of the hourglass-shaped district in the local search method. This phenomenon happens when the identified pair of districts for swapping units are connected through only a small number of units. Suppose the two districts with the maximum weighted population disparity are the blue and the green district and the green district is over-populated. The green district will assign its units in the next few iterations to the blue districts. The ending graph may contain districts with a shape like an hourglass (blue district in the right graph).

4.1 Winner-take-all policy

The winner-take-all policy is the current policy of the US House election. Under the current policy, if a state is assigned N congressional seats, there can be less than or equal to N congressional districts. Each congressional district is assigned an integer value of seats, and the sum of seats from all the districts will equal to N. Under the winner-take-all policy, all of the congressional votes of a district will be assigned to the party with the majority within that district. The district boundaries within a map play an important role in this case. Most Gerrymandering research is based on the assumption of a winner-take-all policy. Note that the current US policy is a special case of our above-defined policy, where there exist exactly N congressional districts and each district has exactly 1 congressional seat.

4.2 Proportional policy

The proportional policy is partly adopted in Portugal, Spain, and Germany, where the congressional representation is assigned according to the total vote ratio within the election unit. In this work, the proportional policy is adapted to the US House election in the following way. If a state is assigned N congressional seats, there can be less than N congressional districts, where a district has at least one seat and the sum of seats from all districts will equal N. Then, within each district, the congressional seats for parties will be calculated based on the voting ratio within that district. The formula for calculating the assigned seats for a party is as follows:

$$\text{Party A's congressional seats in district } d = (\text{congressional seats in } d) \times round\left(\frac{\text{raw votes obtained by A in } d}{\text{total raw votes in } d}\right),$$

where *round* is a function that rounds that value to the nearest integer. An example of the calculation for both policies is included in Table 1.

5 Two-stage stochastic risk model

After the generation of maps by using the algorithm described in a previous section, a natural question to ask is which map produces the fairest result. Here we describe a general stochastic programming framework

in order to pick the fairest map A general risk model has the following form.

$$\min \quad c(x) + R(x) \tag{2a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i = 1 \tag{2b}$$

$$x \in \{0,1\}^n \tag{2c}$$

Suppose there are n maps in the database. The constraint (2b) states that only one of the maps is chosen. The objective (2a) is represented by the addition of two terms. The first term is associated with the direct cost (i.e. fixed cost) of making a decision. In our example, making a decision is the same as choosing a map in the database. The direct cost of choosing a map includes the compactness cost or the difference between the chosen map and the currently adopted map (i.e. some would want the new map to be as close as the current map). The risk term, $R(x)$, is related to the costs that can differ under various scenarios (i.e. unfixed cost). For instance, the scenarios can be multiple predicted voting patterns paired with different election policies. In this case, the corresponding unfixed costs may include the deviation between the congressional seats and the ideal seats or the quantification of the population imbalance. In the following sections, we will introduce some common candidates for the risk term, and finally introduce our general risk model. In the following discussion, we assume that constraint (2b) and (2c) are both satisfied.

5.1 Average risk

$R_{AVG}(x)$ calculates the average risk given a map decision, and is formulated as follows:

$$\begin{aligned} R_{AVG}(x) := \min \quad & t \\ \text{s.t.} \quad & t \geq x_i \sum_{s \in S} P(s) Deviation(i,s) \quad \forall i = 1, \ldots, n \\ & t \geq 0 \end{aligned}$$

The set S represents the set of all scenarios. The $Deviation(i,s)$ is a function that measures the unfixed cost of the map i under the scenario s. $P(s)$ is the probability for the scenario s to take place. The first inequality says t is greater than the expected deviation of the map i if it is chosen.

5.2 Conditional value at risk

Here, we introduce the formulation of the conditional value at risk (CVaR), which calculates the average cost in the tail among all the scenarios. It is formulated as follows:

$$\begin{aligned} R_{CVaR}(x,\alpha) := \min \quad & \sum_{i=1}^{n} y_i + \frac{1}{1-\alpha} \sum_{s \in S} P(s) u_{is} \\ \text{s.t.} \quad & u_{is} \geq Deviation(i,s) x_i - y_i \quad \forall s \in S, i = 1, \ldots, n \end{aligned}$$

α is a user-input value that determines the "tail" among all the scenarios, and $\alpha \in (0,1]$. For example, if we let $\alpha = 0.95$, then $R_{CVaR}(x,\alpha)$ will give the value of the average among worst 5% (i.e. $(1-\alpha)$) scenarios.

y_i is a free variable that will reach the value of VaR_α, which is the cost value that is larger than α of all scenarios' costs for map i. u_{is} will have a positive value if the cost of the map i in scenario s is larger than VaR_α of the map i. In this case, u_{is} is exactly the difference between Deviation(i,s) and its VaR_α. This formulation follows from the formulation in Rockafellar's work [15].

5.3 General risk model

Here, we combine the risk measures in the above sections and write our map selection model as the following.

$$
\begin{aligned}
\min \quad & c(x) + \lambda t + (1-\lambda)\{\sum_{i=1}^{n} y_i + \frac{1}{1-\alpha}\sum_{s\in S} P(s)u_{is}\} \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\
& t \geq x_i \sum_{s\in S} P(s)Deviation(i,s) \quad \forall i = 1,\dots,n \\
& u_{is} \geq Deviation(i,s)x_i - y_i \quad \forall s \in S, i = 1,\dots,n \\
& x \in \{0,1\}^n
\end{aligned}
$$

The notations follow from the previous sections. $\lambda \in [0,1]$ is a parameter that models the trade-off between the average risk and the conditional value at risk. Therefore, the objective is a combination of the c(x) that relates to the properties of the solutions such as compactness, and the random cost represented by risk measures whose values may change according to the scenarios. With this general risk model, we are able to choose a fair map by evaluating each map on different scenarios based on different risk measures.

6 Experiments and results

In this section, we introduce the details of our experiments. All of the experiments were conducted on a machine with an Intel Xeon E5-4640 v2 2.20GHz CPU with 40 cores and 256GB RAM. First, we will talk about the data sources, and the relevant data preprocessing on the raw shape data. Next, we will talk about the generation settings that specify the parameter in map generation algorithms. Finally, we will describe the results of our experiments in two sections. These two sections correspond to the two goals we mentioned in the introduction :

1. Analyze the performance of different election systems under differently populated districts
2. Propose a framework to make recommendations on maps using two-stage stochastic models based on map compactness and fairness risk.

The results we are showing below are related to the US House election in Wisconsin, where there are 8 congressional seats to be assigned. The 8 congressional seats correspond to 8 districts the legislators need to form in the original setting.

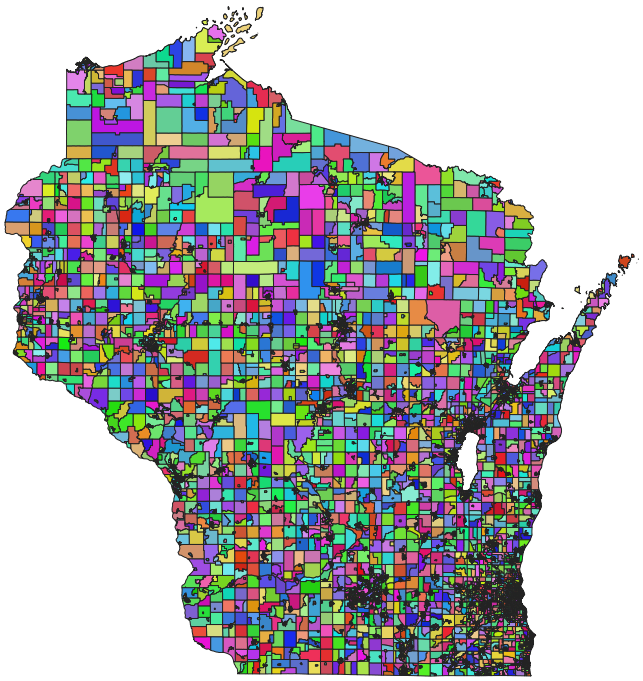

Figure 6: Wisconsin is composed of 7078 ward units in the dataset we used.

6.1 Data description and preprocessing

In our work, we conduct experiments on the Wisconsin ward-level data set, where there are 7078 ward units that constitute the whole state of Wisconsin (figure 6). The data we use comes from Wisconsin State Legislature's GIS Open Data Portal [1]. The dataset contains a shape file that describes 7078 multi-polygons that are linked to the 7078 ward units. Each multi-polygon has the raw election votes of the US House and presidential elections from 2002 to 2020 within their feature sets. One important piece of information is the adjacent relationship among all the units. We used the $nb2mat()$ method in the *spdep* package in R to identify adjacency. Specifically, $nb2mat()$ will turn on the corresponding elements in the adjacency matrix if they have an overlapping border that has a length longer than 0. For each unit, we also calculate its centroid coordinate by averaging the coordinates of vertices in the corresponding multi-polygon. In many cases, the map might contain wards that might be isolated, which makes the graph of the original map disconnected. For example, in figure 6, there are some isolated islands at the northernmost position of Wisconsin. In this case, $nb2mat()$ cannot detect adjacency for those islands. In order to solve this problem, we add some extra adjacency by connecting every isolated unit to its closest unit. By adding these extra edges, the graph of the original map will be connected. With the information on adjacency and population, we can then conduct the following experiments.

6.2 Map generation

In this section, we introduce the parameters we use in order to generate the maps. The population we used in the generation algorithm is the raw votes of the 2020 presidential election in Wisconsin. To be more specific, the population of a unit is the sum of votes for Democrats and Republicans. We generate maps for different

District Number	Population Weights	ϵ	Total # of Samples	Filtered # of Samples
5	2:2:2:1:1	0.05	1000	300
6	2:2:1:1:1:1	0.05	1000	300
7	2:1:1:1:1:1:1	0.05	1000	300
8	1:1:1:1:1:1:1:1	0.05	1000	300

Table 2: Map Generation Setting

numbers of districts according to table 3. As shown in the table, in the case where the district number is less than 8, the districts can have different population ratios. Take the 7-district case as an example. The map will contain 1 district with a population ratio of 2 and the other 6 with a population ratio of 1. The population ratio of a district also represents the number of congressional seats assigned to that district. We also set $\epsilon = 0.05$ which is related to the bound of the weighted population. For each district number, we generated 1000 maps. After the generation of the 1000 maps for a district number, we only take the 300 maps that have the smallest DS, which is specified in the section 3.4 as a metric to identify high-quality solutions. We also report the running time of map generation in table 3.

District Number	average	standard deviation	minimum	maximum
5	185	126	8	828
6	132	77	17	674
7	139	70	33	579
8	199	88	45	643

Table 3: Time for generating a map in seconds.

We conclude that the running time might be related to the population distribution in the initial graph that is described in step 1 of algorithm 2. When the population distribution of that is close to the population weights stated in table 3, fewer iterations might be needed to reach the given population weights. We also provide access to our generation code [3].

6.3 Comparison Between Winner-take-all and Proportional Policies

number of districts	1 seat	2 seats	3 seats	**4 seats**	5 seats
5	4	7	73	197	19
6	3	15	121	159	2
7	0	37	160	103	0
8	0	116	155	29	0

Table 4: Seats won by Democrats under the winner-take-all policy. (2020 US House election data)

number of districts	1 seat	2 seats	3 seats	**4 seats**	5 seats
5	0	0	179	96	25
6	0	69	125	105	1
7	0	130	134	30	6
8	0	116	155	29	0

Table 5: Seats won by Democrats under the proportional policy. (2020 US House election data)

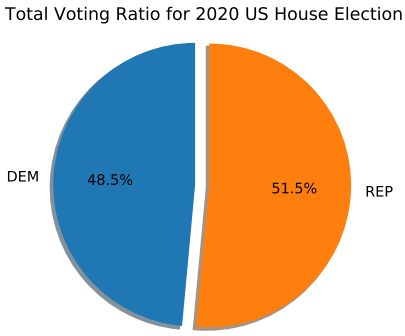

Figure 7: Voting ratio of 2020 US House election

In this section, we compare the distributions of the US House election results of winner-take-all and proportional policies based on the maps generated in the section 6.2. We use the 2020 US House raw votes as the data for calculating the seat shares in our experiments. The process of calculating seat shares for both policies was introduced in the section 4. There are 8 seats in Wisconsin's House election of 2020. Here we define a fair election result as the seat assignment that is closest to the raw vote ratios. According to figure 7, the fairest seat assignment for the Democrats should be $round(8 \times 48.5\%) = 4$.

In table 4, we first look at the cases where the number of districts is 8, which is the current setting in Wisconsin to formulate US House districts. According to the table, we can observe that the distribution is shifted in favor of the Republicans. As the number of districts decreases, the distribution slowly shifts to the position where the mean is close to the fairest seat assignment (4 votes). However, making the number of districts smaller leads to extreme cases where there is only 1 seat assigned to Democrats. This result infers that the existence of bigger-sized districts allows the possibility of seat assignment that is largely beneficial to a specific party. The results for the proportional policy are shown in table 5. First, note that the results for the 8-district case of winner-take-all and proportional policies coincide because of how they are calculated. As the number of districts decreases, the distribution does not shift to the fairest case as much as it is under the winner-take-all policy. However, the extreme cases where Democrats only get 1 or 2 seats completely vanish when the number of districts is decreased to 5. The results show us the completely different characteristics of the two policies. With randomly generated maps, the winner-take-all will potentially provide election results closer to the fairest case on average while allowing more extreme cases to happen when the number of districts is smaller. In contrast, the proportional policy would avoid extreme cases from happening.

The results align with the intuition that the proportional policy performs better when the number of districts is smaller. In the extreme case where there is only 1 district, it is optimal for the proportional policy while it is the worst for the winner-take-all policy because all the seats will be assigned to the one party that has the majority of the raw votes. This motivates having both of these policies considered "adversaries" to our map selection process.

6.4 Result of the two-stage stochastic framework

In this section, we use our proposed stochastic framework to choose a map that is robust to the changes in both election policies and voting. The policies we consider are winner-take-all and proportional policies (2 policies). For the voting data, we use the US House election data from 2002 to 2020 (a total of 10 elections) to

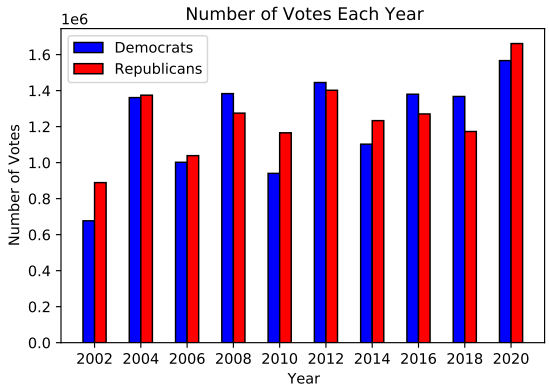

Figure 8: US House election data from 2002 to 2020

conduct our experiments. Therefore, when we are evaluating the risk in our model, the number of scenarios, $|S|$, would be 2(policies) $\times$ 10 (elections) = 20. We visualize the voting data in figure 8. With these data, we can calculate the fairest seat assignment for each election just as we did for the 2020 US House election in section 6.3. The result is shown in table 6.

	2002	2004	2006	2008	2010	2012	2014	2016	2018	2020
Fairest seats	3	4	4	4	4	4	4	4	4	4
Real Seats	4	4	5	5	3	3	3	3	3	3

Table 6: Fairest and the real seat assignment for Democrats. (2002-2020 US House election data)

The row of "real seats" in the table represents the actual result of the election. We can observe that the election results are in favor of Democrats before 2010 and are in favor of Republicans after 2010 when there is a congressional map change in Wisconsin. However, we do not claim that the map change in 2010 is made unfair intentionally since more data is needed to verify such statements.

6.4.1 Find a map that has low average deviation

We now use our stochastic framework (in section 5.3) to choose the best map that produces fair results within the 1200 maps we generated (300 maps for 5, 6, 7, and 8 districts). The related parameters and the function definition we used are included in table **??**. $c(x)$ is chosen to a multiple of the compactness measure(DS) of a map. $\lambda = 0.999$ means the model will prioritize average risk and use $CVaR_\alpha$ to break ties. $CVaR_{0.9}$ is the average of the worst deviation among the 10% worst scenarios. The deviation is defined as the absolute difference between the fair seats and the calculated seats.

	2002	2004	2006	2008	2010	2012	2014	2016	2018	2020
Fairest seats	3	4	4	4	4	4	4	4	4	4
winner-take-all	**4**	4	4	4	4	4	4	4	4	**2**
Proportional	**4**	4	4	4	4	4	4	4	4	4
Average	**4**	4	4	4	4	4	4	4	4	**3**

Table 7: The results of the fairest map in each scenario

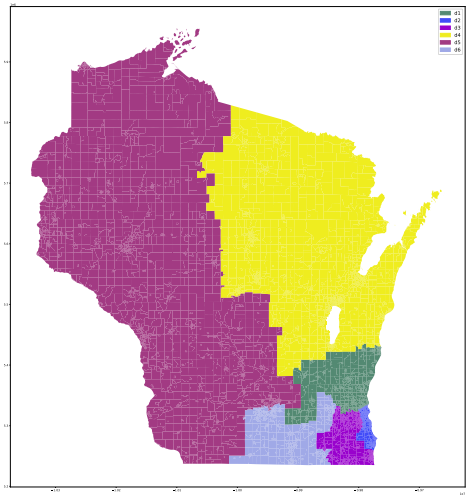

Figure 9: Map with low average risk

The fairest map given by the stochastic model is shown in figure 9, and its deviations are reported in table 7. The map in this case is a 6-district map. The table reports that the fairest map chosen is able to achieve the fairest seat allocation in the elections from 2004 to 2018. It only has deviation in 2002 with both policies, and in 2020 with winner-take-all policies. However, the map has a deviation of 2 under the winner-take-all policy in 2020. Compared with the real seats in table 6, the map chosen by our model is able to improve the fairness that is measured by the average deviation.

6.4.2 Find a map that has the low worst deviation

$c(x)$	0
λ	0.001
α	0.9
$deviation(.)$	absolute difference between the fair seats and the calculated seats

Table 8: Parameter for the experiment with low tail deviation

The next thing we would like to do is to find a map with a small tail deviation, the parameters we use here are in table 8. The parameters λ are set to have a value of 0.001. Now, the model will prioritize the search for a map whose worst 10% deviation is small, and break the tie using the average risk. The chosen map is visualized in figure 10 and the result is reported in table 9.

The map picked by the model, in this case, is an 8-district map where every district has the same population weight. It also shows a large improvement from the real seats shown in table 6. Further, the improvement is actually strict, meaning the deviation is less than or equal to the real case under any scenario. The chosen map will only have deviations of 1 seat in 2010, 2014, and 2016 under both policies, but it does not have a large deviation (larger than 1 seat) under any scenario compared to the map chosen in section 6.4.1.

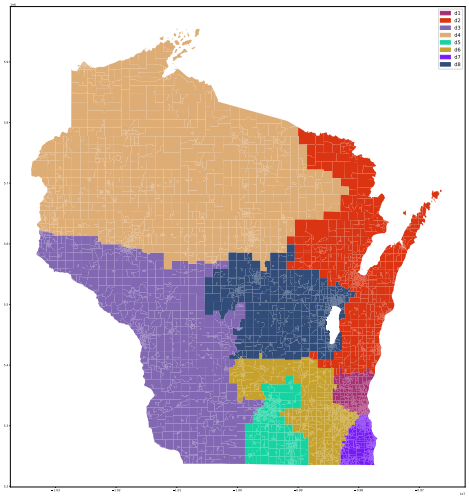

Figure 10: Map with low worst risk

	2002	2004	2006	2008	2010	2012	2014	2016	2018	2020
Fairest seats	3	4	4	4	4	4	4	4	4	4
winner-take-all	3	4	4	4	**3**	4	**3**	**3**	4	4
proportional	3	4	4	4	**3**	4	**3**	**3**	4	4
Average	3	4	4	4	**3**	4	**3**	**3**	4	4

Table 9: The results of the robust map in each scenario

To sum up, we have proposed a general stochastic framework that is able to choose a fair map in terms of average deviation or worst deviation. The results also show that we can find a fairer map by picking from a set of random generations if we are able to predict different voting data.

7 Future direction

In this section, we will discuss some potential future work. First, we can explore different population weights in the multi-representative district other than the ones used in our current generation process. Using our stochastic framework, we can study the Pareto front of the set of maps we are having by adjusting the weights among average risk, CVaR, and fixed costs. Some researchers are exploring possibilities of using parallel computing to generate maps [8]. We believe that by allowing different population weights in our map generation algorithm, we are able to use it to generate maps effectively for a larger problem set (more units and districts) by using hierarchical parallel computing as shown in figure 11. Comparing the effectiveness of direct generation and parallel generation will be an interesting topic to explore. Finally, we are currently using historical election data in our scenarios. To choose a map that produces a fair result in future elections, we can use our historical data to generate several future votes predictions and then use those predictions in the scenarios of our stochastic model.

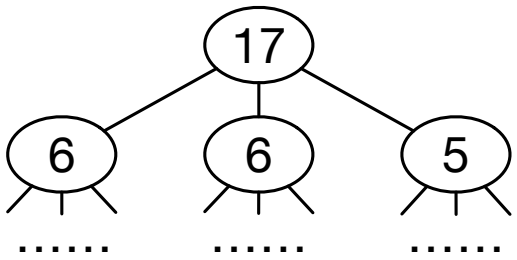

Figure 11: Example of hierarchical parallel computing. Suppose we would like to have 17 districts with equal populations. Then, we can first have 3 districts with population weights 6, 6, and 5 (as shown in the 3 nodes at the bottom). Then, we can parallelize the computing for those 3 nodes until all the leaf nodes have an exact population of 1.

8 Conclusion

We first introduced a new version of the algorithm [4] that allows different population weights among districts and incorporates a novel step where the connected components are reassigned when the contiguity is violated (section 3.3). We further introduce a new compactness metric, map disconnection score (DS), to quantify the compactness in order to filter out the maps that have hourglass-shaped districts. Then, using the maps generated by the algorithm, we analyzed the distributions of election results for winner-take-all and proportional policies as the number of districts decreased. Finally, we proposed a stochastic framework that is able to choose a map within our dataset based on scenarios generated by pairing voting data and election policies.

References

[1] 2002-2020 Election Data with 2020 Wards. `https://data-ltsb.opendata.arcgis.com/`, 2021. [Online; accessed July-2021].

[2] Formulating Exact Contiguity Constraints in Integer Programming. `https://uwmadison.box.com/s/exzn5y0nhn8rhyznv3xz20ubmjmnsexn`, 2022.

[3] Generation code. `https://github.com/WalterLu3/unbalancedGerryMandering`, 2022.

[4] 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.

[5] W. K. T. Cho and Y. Y. Liu. Sampling from complicated and unknown distributions: Monte carlo and markov chain monte carlo methods for redistricting. *Physica A: Statistical Mechanics and its Applications*, 506:170–178, 2018.

[6] D. DeFord, M. Duchin, and J. Solomon. Recombination: A family of markov chains for redistricting. *arXiv preprint arXiv:1911.05725*, 2019.

[7] 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.

[8] W. Gurnee and D. B. Shmoys. Fairmandering: A column generation heuristic for fairness-optimized political districting. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*, pages 88–99. SIAM, 2021.

[9] M. Á. Gutiérrez-Andrade, E. A. Rincón-García, S. G. de-los Cobos-Silva, P. Lara-Velázquez, R. A. Mora-Gutiérrez, and A. Ponsich. Simulated annealing and artificial bee colony for the redistricting process in mexico. *INFORMS Journal on Applied Analytics*, 49(3):189–200, 2019.

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

[11] M. J. Kim. Give-and-take heuristic model to political redistricting problems. *Spatial Information Research*, 27(5):539–552, 2019.

[12] Y. Y. Liu, W. K. T. Cho, and S. Wang. Pear: a massively parallel evolutionary computation approach for political redistricting optimization and analysis. *Swarm and Evolutionary Computation*, 30:78–92, 2016.

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

[14] D. D. Polsby and R. D. Popper. The third criterion: Compactness as a procedural safeguard against partisan gerrymandering. *Yale L. & Pol'y Rev.*, 9:301, 1991.

[15] R. T. Rockafellar, S. Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[16] J. E. Schwartzberg. Reapportionment, gerrymanders, and the notion of compactness. *Minn. L. Rev.*, 50:443, 1965.

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

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

Appendices

A Formulations for contiguity constraints

Here we introduce 2 options to formulate contiguity constraints that have appeared in previous work. One is based on network flow formulation, and the other is based on a-b separator. We also have done experiments of imposing contiguity constraints on a set of smaller-sized synthetic problems [2].

A.1 Shirabe's model

Shirabe's model [17] formulates the contiguity constraint as a combination of multiple Steiner trees in the redistricting network (See figure 12).

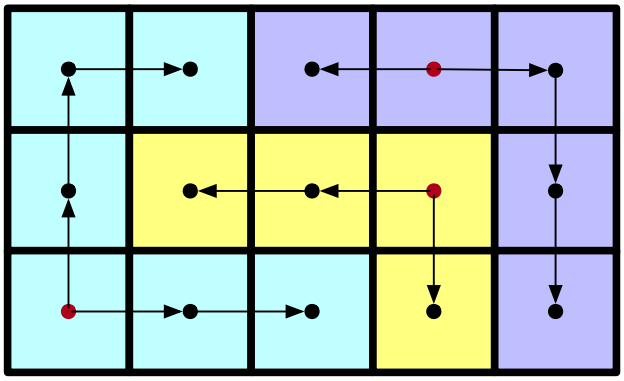

Figure 12: Visualization of Shirabe's model. We can regard the contiguity constraint as a combination of multiple Steiner tree problems. In the figure, we have a solution that satisfies contiguity, and there are 3 districts in three different color. Let red dots be the choice of district centers, then each district center is able to send out flow to the units that are assigned to it.

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:

$$x \text{ satisfies } (1b) - (1f) \tag{A1}$$

$$\sum_{j \in \delta^+(i)} f_{ij}^v - \sum_{j \in \delta^-(i)} f_{ji}^v = -x_{iv} \qquad \forall i \in V \setminus \{v\}, \forall v \in V \tag{A2}$$

$$\sum_{j \in \delta^-(i)} f_{ji}^v \leq (|V|-1) x_{iv} \qquad \forall i \in V \setminus \{v\}, \forall v \in V \tag{A3}$$

$$\sum_{j \in \delta^-(v)} f_{jv}^v = 0 \qquad \forall v \in V \tag{A4}$$

$$f_{ij}^v \geq 0 \qquad \forall (i,j) \in A, \forall v \in V \tag{A5}$$

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

Constraint (A2) 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 (A3) ensures that i can have inflow of type v only if i is assigned to v. (A2) and (A3) together make sure that a unit i does not have any flow exchange of type v if it is not assigned to v. Constraint (A4) 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 (A1)-(A5) and letting the objective be (1a), we can obtain a model that maximizes compactness and satisfies both population and contiguity constraints.

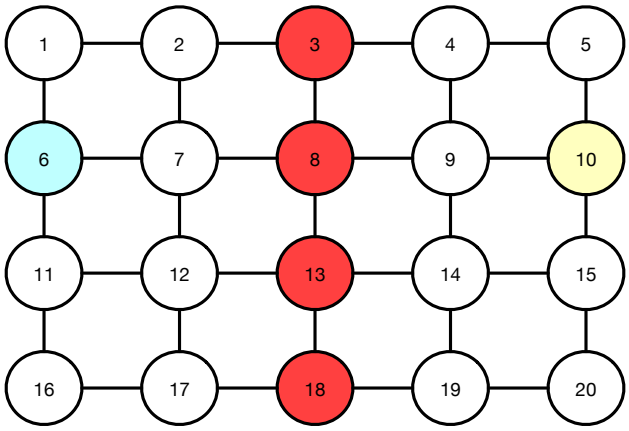

Figure 13: Example of an a-b separator C . Let $a = 6$ and $b = 10$. $C = \{3, 8, 13, 18\}$.

A.2 Oehrlein's model

Oehrlein's model [13] is a cutting plane method based on the idea of a-b separator.

Definition 4 (a-b separator) *Given a graph G = (V,E), the a-b separator is a set of vertices $C \subseteq V$, such that there is no path from a to b in G-C, which is the graph G after removing all vertices in C. An example of a-b separator is presented in figure 13. Note that a-b separator does not exist if $a = b$ or $(a, b) \in E$.*

We can use the idea of a-b separator to define our contiguity constraint. Suppose a unit a is assigned to b (a and b are not adjacent), 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 that satisfies the contiguity constraint is that for any a-b separator, at least one of the vertices in it is also assigned to b. It can also be shown that the above statement is sufficient for contiguity by direct construction of the violated a-b separator. Specifically, suppose a district is represented by $R \subset V$ and is not contiguous. Then there exist at least two connected components within R. We can construct the a-b separator as $S = V \setminus R$, and we know that there must exist at least a pair of vertices a and b such that a is assigned to b, and that S is an a-b separator where no vertex is contained in R.

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

$$x \text{ satisfies } (1b) - (1f) \tag{B1}$$

$$x_{ab} \leq \sum_{c \in C} x_{cb} \qquad \forall a\text{-}b \text{ separator } C, \forall a \in V \setminus \{b\}, \forall b \in V, (a, b) \notin E \tag{B2}$$

Constraint (B2) 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 an exponential number of constraints ($O(2^{|V|})$) in (B2) ; therefore, it would be appropriate to implement a cutting plane method, and add some violated constraints on the fly. We will not describe the separation problem to solve for constraint B2. Instead, we refer the reader to the appendix section of Oehrlein's paper [13] for more details.

Using the constraints (B1)-(B2) and letting the objective be (1a), we can also obtain a model that maximizes compactness and satisfies both population and contiguity constraints just as what we have in the Shirabe's model.