THESIS

SPECTRAL PARTITIONING OF GRAPHS INTO COMPACT, CONNECTED REGIONS

Submitted by

Maxine F. Kampbell

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2025

Master's Committee:

Advisor: Ewan Davies

Sanjay Rajopadhye
James Wilson

ABSTRACT

SPECTRAL PARTITIONING OF GRAPHS INTO COMPACT, CONNECTED REGIONS

Partitioning a graph into regions that are both compact and connected is an important problem with applications in many areas, for example, circuit design, social network analysis, and electoral redistricting. Our work builds on existing ideas of spectral bipartitioning and Markov chain Monte Carlo (MCMC) recombination methods to provide a new method for partitioning graphs: spectral recombination. Previous methods utilized these ideas independently, for example, partitioning a graph directly using its spectrum or using MCMC recombination methods that do not rely on its spectrum. Our work represents a novel approach that combines the two ideas. We provide empirical evidence that spectral recombination methods generate partitions with low cut edge counts, that is, more compact regions with shorter boundaries. Moreover, we demonstrate that our base spectral recombination algorithm can be modified to prioritize different metrics, such as balanced vertex weights among regions. We note that there appears to be a trade-off between achieving low cut edge counts and maintaining approximate weight balance, illuminating an avenue for future research. Our code and data can be found at https://github.com/MaxFlorescence/spectral_redistricting.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

This thesis explores the topic of computing compact and connected graph partitions. The key data structures in question, *graphs*, are collections of vertices and edges between those vertices. A graph is defined as an ordered pair $G = (V_G, E_G)$, where V_G is the set of the graph's vertices and $E_G \subset \binom{V_G}{2}$ is the set of the graph's edges. For example, figure 1.1 depicts a triangle graph (C_3) with corners labeled a, b, c:

$$C_3 = \Big(\{a, b, c\}, \{ab, bc, ca\}\Big)$$

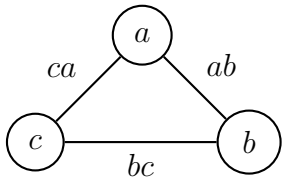


Figure 1.1: A triangle graph.

A graph T is called a *tree* if it contains no cycles (non-empty sequences of distinct edges that form loops) and is connected. If G is a graph and $T = (V_G, E_T)$ is a tree, then T is referred to as a *spanning tree* of G.

Graphs provide the setting for which *partitions* can be defined. A partition of a graph G is a set of subsets $\mathcal{D} \subseteq V_G$, called *regions*, that form a partition of V_G. For example, consider the four-cycle (C_4) with corners labeled a, b, c, d in figure 1.2.

$$C_4 = \Big(\{a, b, c, d\}, \{ab, bc, cd, da\}\Big)$$

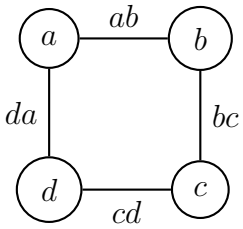


Figure 1.2: A square graph.

If the top (ab) and bottom (cd) edges of C_4 were to be removed, the result can be represented by a partition with two districts: the left and right halves of C_4. Figure 1.3 shows the subgraphs of C_4 induced by the partition $\mathcal{D}$.

$$\mathcal{D} = \Big\{ \{a, d\},\ \{b, c\} \Big\}$$

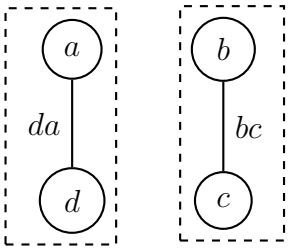


Figure 1.3: A size-2 partition of a square graph.

The *cut edges* of a partition is the set of edges that span two distinct regions of the partition. In the par of C_4 in figure 1.3 there are two cut edges, ab and cd.

We study the problem of graph partitioning specifically through the lens of electoral redistricting. We can apply graph partitioning to real-world maps by utilizing their *dual graphs*. Each subdivision of the map (country, state, county, etc.) is a vertex, and vertices are connected by edges if and only if they share a border. In this context, a graph partition is called a *districting plan*, and each region of the plan is called a *district*.

To make real life elections practical, maps can be divided into districts and votes are counted for each district. However, depending on exactly how this districting process is carried out, the results of the election may not be representative of the population's votes. This phenomenon is known as *gerrymandering*, and is described in more detail in section 2.1.

In general, it is difficult to intuitively say if a given districting plan is gerrymandered or not. One technique in the literature uses *ensembles* of automatically generated districting plans for comparison. In short, if a computer can generate tens of thousands of random districting plans without any bias, then we can compare the plan in question to this ensemble and determine if it is an outlier or not. If it is, then we can say that the plan was likely gerrymandered. In this way, we

can shift our focus away from the difficult question of "is this districting plan biased?" and to the easier question of "is this plan similar to ones that were generated to be unbiased?"

To generate these ensembles of districting plans, *Markov chain Monte Carlo* (MCMC) methods can be used. These methods start using an initial, provided districting plan, and then iteratively update it in small ways over and over again. After hundreds, thousands, or sometimes millions of small updates, a chain can end up at a districting plan that is completely different from the original plan it started on.

In particular, we study one type of MCMC method called *recombination*. For recombination chains, these iterative small steps involve recombining two adjacent districts in the current plan, and then splitting them back apart to form the next plan. Of course, there are many ambiguities in this statement. How does the chain decide which two districts to join? What considerations go into how the joined districts are split again? Different answers to these questions give rise to different types of recombination methods, including the two methods we propose in this thesis.

1.1 Thesis Structure

The remainder of this thesis is structured as follows. In chapter 2, we provide the necessary motivation for why the congressional redistricting problem is important. Next, in chapter 3, we explain the background needed to fully understand our contributions. Chapters 4 and 5 describe our contributions in detail, and chapter 6 contains our empirical evaluations of them. Finally, chapter 7 lists our conclusions, and chapters 8 and 9 discuss related and future works respectively.

Chapter 2

Motivation

Our motivation in the development of the spectral recombination algorithms is two-fold. First, our algorithms provide additional methods of generating districting plan ensembles which can be used in the detection of partisan gerrymandering. Second, other current methods use different metrics (such as the number of spanning trees a district admits) of a plan's districts as a surrogate measure for district compactness. Therefore, exploring the number of edges cut by a districting plan and how that can be used to inform a redistricting algorithm is an interesting area of research.

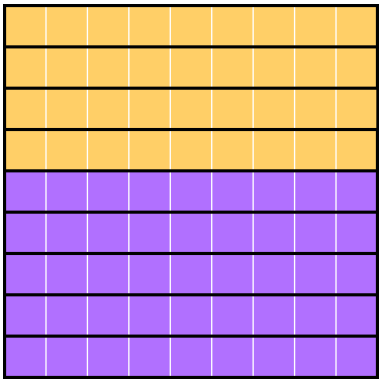
2.1 Gerrymandering

In the United States, most states elect local and federal representatives using some form of district maps. These divide the population of the state into multiple districts, within which votes are cast for a representative. While it might seem that any map which partitions the population into equally sized districts would also be equally fair, the map can be drawn to give certain groups an advantage or disadvantage. This process is called *gerrymandering*.

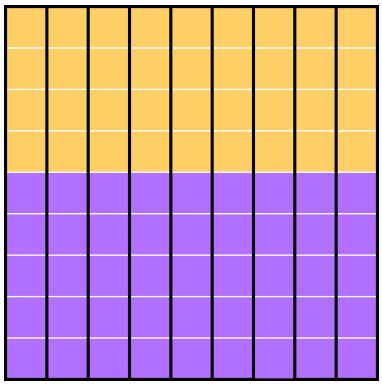
The word "gerrymander" is derived from the 19^{th} century politician Elbridge Gerry and the word salamander. Purportedly, as the governor of Massachusetts, Gerry had signed a bill calling for its redistricting. His party ended up creating a districting plan that favored themselves, with one of the oddly-shaped districts resembling a salamander. This resulted in a depiction of the district with a salamander imposed appearing in the *Boston Gazette* with the caption "The Gerry-mander", and the portmanteau caught on [1]. Originally, the word "gerrymander" was pronounced with a hard g (like "great"), the same way that "Gerry" was. But over time, it came to be mostly pronounced with a soft g (like "giraffe").

For example, suppose there are two populations of interest: 45 purple voters and 36 yellow voters distributed across a 9×9 square grid as shown in figure 2.1. The figure gives three example ways that plans can be drawn such that the total population is partitioned equally into nine districts,

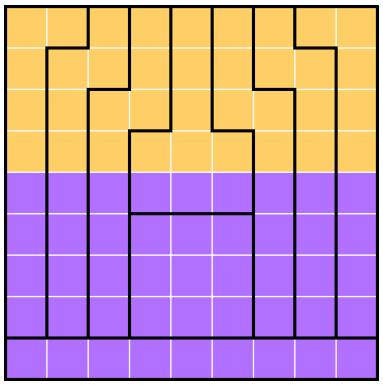
each with nine voters that elect one representative. Each square represents one voter, and voters are grouped into districts as indicated by the black boundaries.



(a) A districting plan that closely aligns with the voting population (5:4).



(b) A districting plan that favors purple voters (9:0).



(c) A districting plan that favors yellow voters (2:7).

Figure 2.1: Three valid districting plans demonstrating voter representation and misrepresentation.

Plan 2.1a might be an ideal solution, where the ratio of purple to yellow majority-districts is close to that of the ratio across the population as a whole. In such a plan, the ratio of elected officials would be representative of the entire population.

Plans 2.1b and 2.1c however, give an advantage to purple and yellow voters, respectively. If elections are winner-take-all, where in each district the candidate that receives the most votes is elected, then the ratio of elected purple representatives to elected yellow representatives would not correspond to the actual ratio of purple to yellow voters. One population is over-represented while the other is under-represented.

If the populations of interest correspond to different political parties, then this would be referred to as *partisan gerrymandering*. The district map was created to intentionally provide an advantage to one political party over the others.

Of course, real world maps are not as neat as the example map in figure 2.1, and so it is much harder to visually identify when gerrymandering was done intentionally. Counter-intuitively, in some cases it is impossible to draw a map that is representative of the voting population. For example, on pages 9-11 of Duchin and Walch's book *Political Geometry* [2], it can be seen that due to the way individual voters are distributed across the state of Massachusetts, all possible districts have the same political leaning.

2.2 Detecting Partisan Gerrymandering

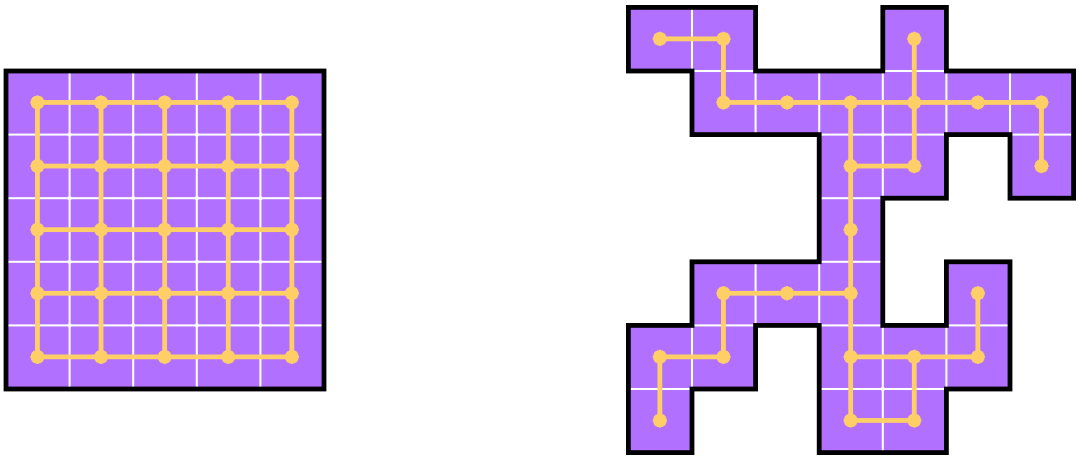
One technique proposed in the literature to detect the presence of a partisan gerrymander in a districting plan is ensemble comparison [3–8]. Suppose a particular districting plan comes under review to determine if it had been gerrymandered. If there exists a large ensemble of non-gerrymandered plans to compare it against, then we have a way to obtain evidence: if the plan in question is an outlier compared to the ensemble (with respect to some set of metrics), then we can be fairly confident that the plan was gerrymandered to some degree. Furthermore, ensemble comparison can also be robust to edge cases like the one discussed at the end of the previous section. The ensemble data would likely report the same result, implying that it is at least extremely unlikely that a non-gerrymandered plan would have proportional representation.

This idea is a step further than Vickery's argument that the "elimination of gerrymandering would seem to require the establishment of an automatic and impersonal procedure for carrying out a redistricting" [9], and is referred to by Guest et al. [10] as a "division of labour in which humans debate and formulate districting criteria whereas machines optimise the criteria to draw the district boundaries". Moreover, this idea has been used in court cases before, as discussed by Herschlag et al. [3]. Building on the analysis used in these cases, they were able to confirm that past districting plans of North Carolina were gerrymandered by creating ensemble using a Markov

chain Monte Carlo (MCMC) method and simulated annealing. Since then, MCMC methods for generating ensembles have become popular in the literature. In general, these involve designing a Markov chain that targets a particular distribution over the space of desirable plans. And recently, the so-called spanning tree distribution has been the target of several chains [4].

2.3 Spanning Trees and Cut Edges

Clelland et al. [4] explored the relationship between cut edge count and number of spanning trees. The Recombination and Reversible Recombination methods from the literature [4, 7] are both based on the idea that the number of spanning trees that a district admits correlates with how compact it is. That is, a district with more possible spanning trees can be considered more compact, an intuition demonstrated in figure 2.2. In district (a) the nodes of the dual graph (in yellow) are "more connected" than those of district (b). One concrete way this holds is that district (a) admits more spanning trees.



(a) A district with over 5 billion spanning trees.

(b) A district with just 16 spanning trees.

Figure 2.2: The number of spanning trees as an indicator of compactness.

Clelland et al. showed that the number of spanning trees across any pair of districts ($P_{\mathcal{D}}$) appears to decay exponentially with the number of cut edges between those districts ($E_{\mathcal{D}}$):

$$P_{\mathcal{D}} \approx Ce^{-kE_{\mathcal{D}}},$$

where C and k are constants that depend on the graph in question. And so, the intuition can be re-framed in terms of cut edge counts: a districting plan with fewer cut edges is more compact than one with more cut edges. This is good because, as Clelland et al. describe, the number of edges cut by a districting plan is easier to calculate, visualize, and understand than how many spanning trees it admits.

Furthermore, Procaccia and Tucker-Foltz [11] agree that there is a relationship between cut edges and spanning trees. They state that, for example, if one districting plan for a large grid graph has at least 7.23 times the number of cut edges as another plan for the same graph, then the plan with fewer cut edges is more likely to be sampled under the spanning tree distribution. Specifically, they found that there is an inverse exponential relationship between a plan's number of cut edges and the probability that the plan is sampled under this distribution. If $\mathcal{D}$ and $\mathcal{D}'$ are two districting plans for a graph G such that G and its dual have second-largest degrees bounded by a constant, then:

$$\frac{\Pr_{\mu^*}[\mathcal{D}]}{\Pr_{\mu^*}[\mathcal{D}']} \geq 2^{\Theta\left(|\partial_G \mathcal{D}'|/|\partial_G \mathcal{D}|\right)},$$

where μ^* is the spanning tree distribution and $\partial_G \mathcal{D}$ denotes the set of edges of G that are cut by $\mathcal{D}$. The condition of G and its dual having bounded second-largest degrees corresponds to the types of graphs typically of interest to the congressional districting problem; all census block have few neighbors, and no large group of blocks intersect at the same boundary point.

Our idea therefore utilizes the cut edge count of a districting plan as the fundamental metric to consider, rather than number of spanning trees. One use case we envision for our methods is to provide a standard initial state for other redistricting algorithms. To our knowledge, there is no standard way to begin a redistricting MCMC run, and without rigorous bounds on the chain's mixing time, it is possible that the chain's initial state has some influence on its final state. However, our two algorithms might be able to provide unbiased initial states. As can be seen in section 6.2, they both seem to successfully generate districting plans with low cut edge counts in just a few hundred steps, and our second algorithm also maintains low population deviation from the ideal.

This alternative method also speaks to an open question from DeFord et al. [5], "Propose other balanced bipartitioning methods to replace spanning trees, supported by fast algorithms". Additionally, an invitation written by DeFord and Duchin [2] reads, "New methods should strive for easy implementation, low rejection rate, adaptability to varied districting criteria, and of course theoretical properties like provable ergodicity or reversibility". While we do not completely address the points raised in these texts, our work makes good progress on these problems.

Chapter 3

Background

This chapter provides the information necessary to understand our contributions fully. We start by precisely defining the congressional redistricting problem and describing the different metrics we will use. Next, we briefly describe the framework under which our algorithms operate and specify the precursors to our algorithms from the literature. We then finish with an overview of relevant ideas from spectral graph theory.

3.1 Problem Statement

We formulate the congressional redistricting problem (or simply, the redistricting problem) as follows.

Given an n-vertex graph $G = (V_G, E_G)$, a positive integer k, and a set of constraints C, create a districting plan $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k\}$ that satisfies all constraints in C.

That is, the districting plan $\mathcal{D}$ must identify each vertex of G with exactly one of k districts. This can be visualized by associating each district with a unique color, and coloring each vertex of G with its district's color. When applied to states in the USA, we consider a state's dual graph constructed by associating each of the state's precincts with a vertex, and connecting two vertices if and only if the two corresponding precincts share a border. Figure 3.1 demonstrates this.

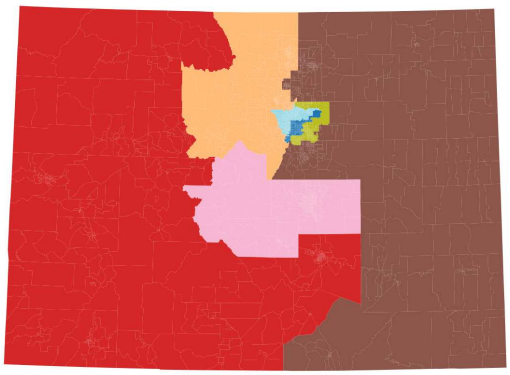
3.1.1 Notation

For simplicity, we make the following definitions.

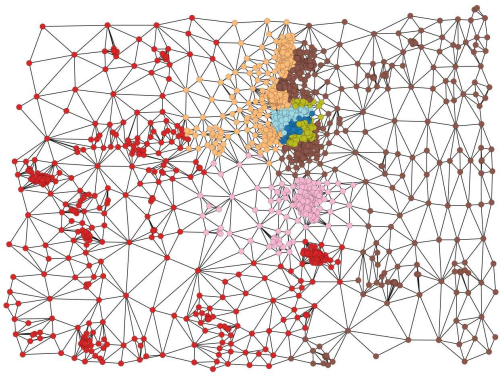
$\mathcal{D}(v)$ is defined to be the unique district of which v is a member.

$E_G(U, V)$ is defined as the edges of G between the vertex sets U and V:

$$E_G(U, V) := \big\{uv \in E_G : u \in U \text{ and } v \in V\big\}.$$



(a) Colorado's 2018 congressional districts.



(b) The corresponding dual graph representation.

Figure 3.1: How U.S. states are represented in the redistricting problem.

$\partial_G \mathcal{D}$ is defined to be the edges of G that are cut by the districting plan $\mathcal{D}$:

$$\partial_G \mathcal{D} := \bigcup_{\substack{\mathcal{D}_i, \mathcal{D}_j \in \mathcal{D} \\ i \neq j}} E_G(\mathcal{D}_i, \mathcal{D}_j).$$

$p(v)$ is defined to be the population of vertex v, $p(G)$ is the population the entire graph, and $p(V)$ is the population of a vertex subset $V \subseteq V_G$:

$$p(G) := \sum_{v \in V_G} p(v), \quad p(V) := \sum_{v \in V} p(v).$$

$G[V]$ is defined as the subgraph of G induced by the vertex set V. That is, the subgraph of G that contains all of the vertices in V, and all edges that have both end points in V:

$$G[V] := \big(V, E_G(V, V)\big).$$

$\mathrm{ST}(G)$ is defined as the set of all spanning trees of G:

$$\mathrm{ST}(G) := \{T : T \text{ is a spanning tree of } G\}.$$

$T \setminus e$, where T is a tree, is defined as the pair of connected graphs obtained from T by removing the edge $e \in E_T$.

3.2 Metrics

There are different metrics we can use in the evaluation of a given districting plan. For this thesis, we focus on metrics involving the plan's cut edges, district sizes and populations, and contiguity.

3.2.1 Cut Edges

A districting plan's *cut edge count* quantifies the total length of the plan's borders in a graph-theoretic way. It is the sum of the number of edges cut between the vertices of each distinct pair of districts:

$$|\partial_G \mathcal{D}| = \sum_{\substack{\mathcal{D}_i, \mathcal{D}_j \in \mathcal{D} \\ i \neq j}} |E_G(\mathcal{D}_i, \mathcal{D}_j)|.$$

A plan with a low cut edge count will have districts that can be considered more compact, since more precincts will be on the interiors of districts rather than on the borders. Our algorithms are mainly concerned with this metric since, as discussed in section 2.3, it seems to correspond to the compactness of districts without being too influenced by specific district boundary shapes.

3.2.2 Deviation and Tolerance

A plan's *deviation* is, in concept, the furthest any individual district is from the "ideal" value of some metric. In terms of size, we can consider a plan's *size deviation*:

$$\text{sizeDev}(G, \mathcal{D}) := \max_{\mathcal{D}_i \in \mathcal{D}} \left| k \frac{|\mathcal{D}_i|}{n} - 1 \right|,$$

and in terms of population, we consider *population deviation*:

$$\text{popDev}(G, \mathcal{D}) := \max_{\mathcal{D}_i \in \mathcal{D}} \left| k \frac{p(\mathcal{D}_i)}{p(G)} - 1 \right|.$$

Deviation ranges in value from 0 (all districts are equal in size/population), to $\approx k-1$ (all districts contain a single vertex/population unit, except for one). We also note that size deviation can be thought of as a special case of population deviation, where each vertex has population 1.

A more natural way to phrase approximate population equality among districts might be through *tolerance*. That is, for all $\mathcal{D}_i \in \mathcal{D}$, the population of $\mathcal{D}_i$ is within some tolerance ε of the ideal population, $p(G)/k$:

$$(1-\varepsilon)\frac{p(G)}{k} \leq p(\mathcal{D}_i) \leq (1+\varepsilon)\frac{p(G)}{k},$$

with a similar pair of inequalities holding for size tolerance.

From this we can see that, given a population/size tolerance of ε, an equivalent way of stating the above inequalities is to say that the plan's population/size deviation should be no more than ε:

$$\begin{aligned}
& (1-\varepsilon)\frac{p(G)}{k} \leq p(\mathcal{D}_i) \leq (1+\varepsilon)\frac{p(G)}{k}, \ \forall \mathcal{D}_i \in \mathcal{D} \\
\iff \quad & -\varepsilon \leq k\frac{p(\mathcal{D}_i)}{p(G)} - 1 \leq \varepsilon \quad , \ \forall \mathcal{D}_i \in \mathcal{D} \\
\iff \quad & \max_{\mathcal{D}_i \in \mathcal{D}} \left| k\frac{p(\mathcal{D}_i)}{p(G)} - 1 \right| \leq \varepsilon
\end{aligned}$$

3.2.3 Contiguity

A districting plan is *contiguous* if each of its districts induces a connected subgraph of G:

$$\text{contiguous}(G, \mathcal{D}) := \begin{cases} \texttt{True} & \text{if } G[\mathcal{D}_i] \text{ is connected, for all } \mathcal{D}_i \in \mathcal{D}, \\ \texttt{False} & \text{otherwise.} \end{cases}$$

Contiguity is often included in the constraint set C.

3.3 Markov Chain Monte Carlo Methods

A Markov chain is a type of memory-less random process which explores a state space; in our case, the space of valid districting plans. We can efficiently sample a plan from the state space

by running a corresponding chain, and our choice of chain influences which types of plans are sampled. The state space for our MCMC methods of interest is parameterized by k and G; the set of all valid partitions of V_G into k districts:

$$\Omega_{G,k} = \{\mathcal{D} : \mathcal{D} \text{ is a } C\text{-feasible districting plan of } G \text{ with } k \text{ districts}\}$$

Using MCMC methods, we can efficiently generate many districting plans. A new random plan is sampled by starting from an initial plan $\sigma_0 \in \Omega_{G,k}$ and running some random walk over $\Omega_{G,k}$ for some number of steps N. The specific computations performed during each step are determined by which *proposal function* p is being used, that is, which chain is being run. The proposal function defines the transition matrix of the chain, and in turn which distribution to sample plans from.

The library "GerryChain" [12], created in 2018 by the Data and Democracy Lab (formerly the MGGG Redistricting Lab), provides a framework for implementing such MCMC methods in Python. A simplified version of GerryChain's generic MCMC method is shown in algorithm 1.

Algorithm 1 Markov chain Monte Carlo Redistricting

```
procedure REDISTRICTING(N, C, σ_0, p)    # input: steps, constraints, initial state, proposal
    for 1 ≤ i ≤ N do
        repeat
            σ_i ← p(σ_{i−1})
        until σ_i is feasible under C    # don't step out of the state space
    end for
    returnσ_N                            # output: final state
end procedure
```

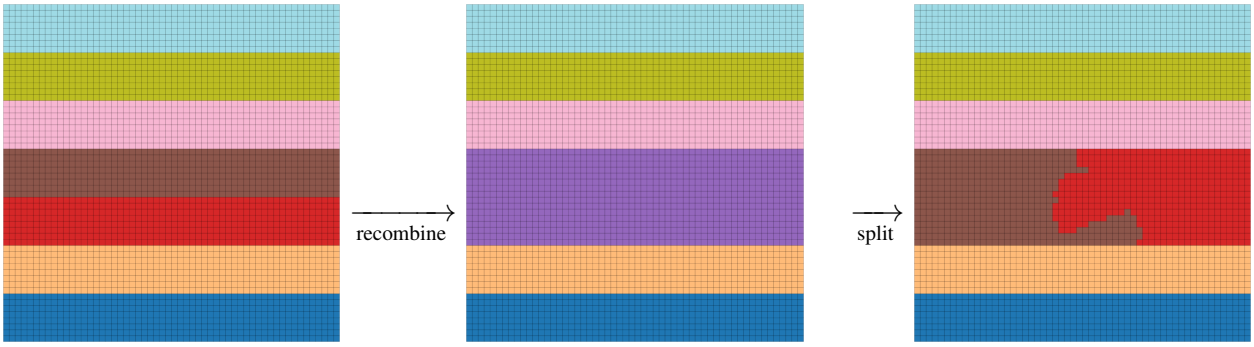
3.4 Recombination Algorithms

One group of MCMC methods studied in the literature are "recombination" proposals. These types of algorithms follow the same basic structure (depicted in figure 3.2):

1. Select two adjacent districts in the current districting plan

2. Recombine the two districts into one connected graph

3. Split the combined graph back into two districts



(a) A 56×56 grid graph with seven districts.

(b) The graph after the brown and red districts have been recombined.

(c) The graph after the combined districts are split again.

Figure 3.2: An example step of a recombination algorithm on a 56×56 grid graph.

We refer to two recombination algorithms from previous literature: Recombination, and Reversible Recombination.

3.4.1 Spanning Tree Recombination

The first recombination chain (ReCom) [5, 13] is given by algorithm 2. In general, it functions by sampling a uniform random spanning tree of the dual graph of the combined districts. Then it can split the districts back into two by sampling an edge from the tree to cut.

In order to maintain roughly balanced district populations, the algorithm cannot select edges of the spanning tree arbitrarily. Instead, after an edge is sampled uniformly, it ensures that the difference in population between the districts is limited by a population tolerance parameter, ε (line 10 in algorithm 2). But for any given spanning tree, it is not guaranteed that there exists an edge that satisfies the population tolerance when cut (see figure 3.3 for an example). So in this case, the algorithm rejects the tree and samples a new one.

Because ReCom samples spanning trees in determining how to split the recombined districts, the overall Markov chain targets the "spanning tree" distribution. For example, consider the two

Algorithm 2 Recombination Proposal [4]

procedure RECOM($\mathcal{D}; G, \varepsilon$) *# input: current state; graph, tolerance*
  select$\{u, v\} \in \partial_G \mathcal{D}$ uniformly at random (UAR) *# sample a cut edge*
  $H \leftarrow G[\mathcal{D}(u) \cup \mathcal{D}(v)]$ *# recombine the districts spanned by the edge*
  edges $\leftarrow \emptyset$
  cuttable $\leftarrow$ FALSE
  while $\neg$ *cuttable* **do**
    select$T \in \mathrm{ST}(H)$ UAR *# sample a spanning tree*
    for $e \in T$ **do** *# find all ε-balanced cuts of T*
      $T_1, T_2 \leftarrow T \setminus e$
      if $\big||T_1| - |T_2|\big| < \varepsilon |T|$ **then**
        edges $\leftarrow$ *edges* $\cup \{e\}$
        cuttable $\leftarrow$ TRUE
      end if
    end for
  end while
  select$e \in$ *edges* UAR *# pick one of the ε-balanced cuts, e*
  $T_1, T_2 \leftarrow T \setminus e$ *# cut T at e to form new districts*
  $\mathcal{D}' \leftarrow (\mathcal{D} \setminus \{\mathcal{D}(u), \mathcal{D}(v)\}) \cup \{V_{T_1}, V_{T_2}\}$
  return $\mathcal{D}'$ *# output: next state*
end procedure

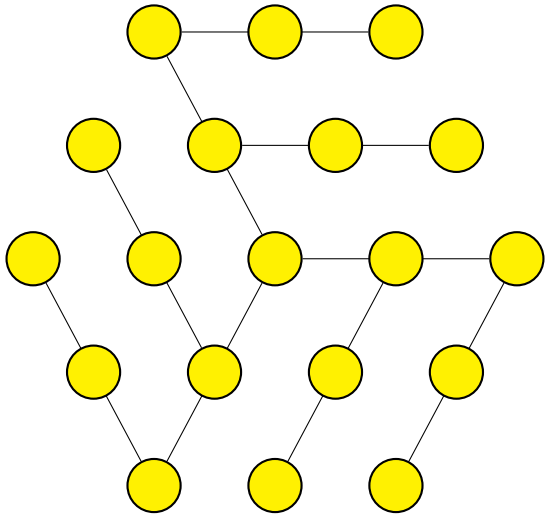


Figure 3.3: A spanning tree with no ε-balance edges for $\varepsilon < 7/19$.

districts in figures 2.2a and 2.2b. Since 2.2b admits fewer spanning trees than 2.2a, it is less likely to be chosen over 2.2a when ReCom samples a random spanning tree. (Of course, the districts shown in figure 2.2 would each have complementary districts such that the union of 2.2a and its complement equals the union of 2.2b and its complement.)

Though, ReCom is only able to approximate the spanning tree distribution. In order to create a chain that provably has the spanning tree stationary distribution, ReCom was modified to be *reversible*.

3.4.2 Reversible Recombination

The Reversible Recombination (RevRecom) algorithm, created by Cannon et al. [7], modifies ReCom to give it a property called *detailed balance*. This, in turn, allowed for the proof that RevRecom's stationary distribution is the spanning tree distribution.

A description of RevRecom is given by algorithm 3. The notation "RAND(0,1)" indicates choosing a uniform random real between 0 and 1, for use in probabilistic tests. Similar to ReCom, RevRecom functions by sampling a random spanning tree of the recombined districts' graph, and then cutting an edge that maintains some population tolerance. This tolerance is characterized by "ε-balance edges", edges of the spanning tree that, when cut, maintain a population tolerance across the whole graph G of at most ε.

However, the two algorithms differ in key areas. Most importantly, RevRecom provides many chances for the proposed split to be rejected. The parameter m indicates the absolute maximum number of ε-balance edges that can be found for any spanning tree of any two districts of G. This value is impractical to calculate, and so it is often determined empirically. As such, choosing a value of m that is too large causes many more rejections than necessary, since m determines how likely RevRecom is to reject the sampled spanning tree based just on the number of ε-balance edges it contains (line 11 of algorithm 3).

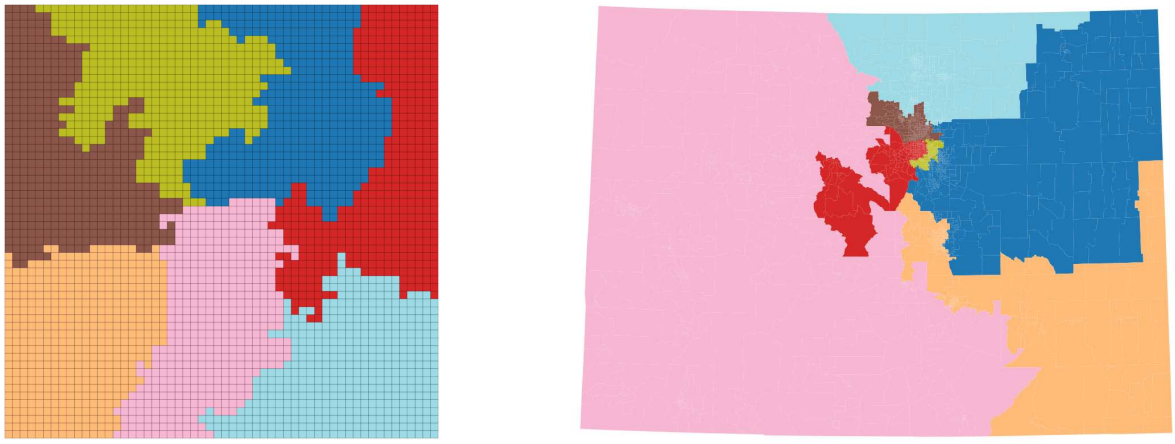
Moreover, RevRecom can still reject the split based on how many edges it would cut in G (line 16 of algorithm 3). The overall increased likelihood of rejection means that RevRecom runs more

Algorithm 3 Reversible Recombination Proposal [7]

procedure REVRECOM($\mathcal{D}; G, \varepsilon, m$) *# input: current state; graph, tolerance, maximum balance edge count*
  accept $\leftarrow$ `False`
  repeat
    select$\mathcal{D}_i, \mathcal{D}_j \in \mathcal{D}$ such that $E_G(\mathcal{D}_i, \mathcal{D}_j) \neq \emptyset$ UAR *# sample adjacent districts*
    $H \leftarrow G[\mathcal{D}_i \cup \mathcal{D}_j]$ *# recombine*
    select$T \in \mathrm{ST}(H)$ UAR
    $b \leftarrow$ `Null`
    for $e \in E_T$ **do** *# repeatedly find ε-balance edges*
      $T_1, T_2 \leftarrow T \setminus e$
      $\mathcal{D}' \leftarrow (\mathcal{D} \setminus \{\mathcal{D}_i, \mathcal{D}_j\}) \cup \{V_{T_1}, V_{T_2}\}$
      if $(\mathrm{popDev}(G, \mathcal{D}') < \varepsilon) \wedge (\textsc{rand}(0,1) < 1/m)$ **then**
        $b \leftarrow e$ *# choose this ε-balance edge b with probability $\frac{1}{m}$*
        break
      end if
    end for
    if $(b \neq$ `Null`$) \wedge (\textsc{rand}(0,1) > 1/|E_G(T_1, T_2)|)$ **then**
      accept $\leftarrow$ `True` *# accept split made by b with probability $\frac{1}{|E_G(T_1,T_2)|}$*
    end if
  until *accept*
  return $\mathcal{D}'$ *# output: next state*
end procedure

slowly than ReCom, as it wastes time rejecting splits. This problem was mitigated in implementation by allowing many different spanning trees to be considered in parallel, and moving on once one of them is accepted [14].

Figure 3.4 provides example districting plans produced by RevRecom. While they do maintain approximate population balance, the district borders are visually more jagged. This is quantified by the cut edge counts of the plans shown in section 6.2.



(a) 56×56 grid graph.

(b) Colorado graph.

Figure 3.4: Example plans after 100,000,000 steps of RevRecom.

3.5 Algebraic Connectivity

Our recombination algorithm (described in chapter 4) uses the idea of algebraic connectivity, also named the Fiedler value after Miroslav Fiedler [15], to determine how to split the recombined districts.

For an undirected, finite, simple graph G with n vertices, we consider three $n \times n$ matrices that are indexed by pairs of vertices of G. The adjacency matrix A_G of G is the symmetric matrix that

has the weight of each edge of G at its corresponding entries:

$$A_G[u,v] = \begin{cases} \text{weight}(uv) & \text{if } uv \in E_G, \\ 0 & \text{otherwise.} \end{cases}$$

We note that, in the case that G is an unweighted graph, the weight of each edge is defined to be 1.

The degree matrix D_G of G is the diagonal matrix that has the degree of each vertex of G along its diagonal:

$$D_G[u,v] = \begin{cases} \text{degree}(v) & \text{if } u = v, \\ 0 & \text{otherwise.} \end{cases}$$

And finally, the *Laplacian* L_G of G is defined to be the difference between the degree matrix and adjacency matrix of G: $L_G = D_G - A_G$. In some contexts, the *normalized Laplacian* matrix ($\mathcal{L}_G = D_G^{-1/2} L_G D_G^{-1/2}$) is used. In this thesis though, the term Laplacian will be used to refer the un-normalized Laplacian matrix L_G.

The spectrum of the graph G is the set of eigenvalues of L_G, exactly one of which is zero for connected graphs [16]. Listing the spectrum in order, we have:

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n,$$

where λ_2 is referred to as the algebraic connectivity of G. An eigenvector with eigenvalue λ_2 is known as a Fiedler vector of G, $\boldsymbol{f}_G$. While it is possible that the geometric multiplicity of λ_2 is greater than 1, corresponding to meaningfully different choices of Fiedler vector, we do not explore this possibility in this work. We discuss this further in section 9.1.

Urshel and Zikatanov [17] show that for any connected, undirected, simple graph G, there exists a Fiedler vector $\boldsymbol{f}_G$ such that the bipartition given by

$$\{\boldsymbol{f}_G^+, \boldsymbol{f}_G^-\} = \big\{\{v \in V_G : \boldsymbol{f}_G[v] \geq 0\},\ \{v \in V_G : \boldsymbol{f}_G[v] < 0\}\big\}$$

results in two connected subgraphs of G: $G[\boldsymbol{f}_G^+]$ and $G[\boldsymbol{f}_G^-]$.

The fact that spectral graph partitioning is a technique found in the literature [15, 16, 18], combined with Urshel and Zikatanov's results, provides motivation for considering the approach in congressional redistricting. The bipartition described above makes use of the positive (including zero) and negative entries in the Fiedler vector. Notably, cut-vertices of graphs (vertices that, when removed, split the graph into two components) have corresponding entries of zero, and so other vertices can be thought of as being on the "positive side" or "negative side" of this cut. This intuition is demonstrated in figure 3.5, in which each vertex is labeled and colored to correspond with its Fiedler vector entry, rounded to the nearest thousandth (the graph contains no cut-vertices, but positive and negative sides still emerge).

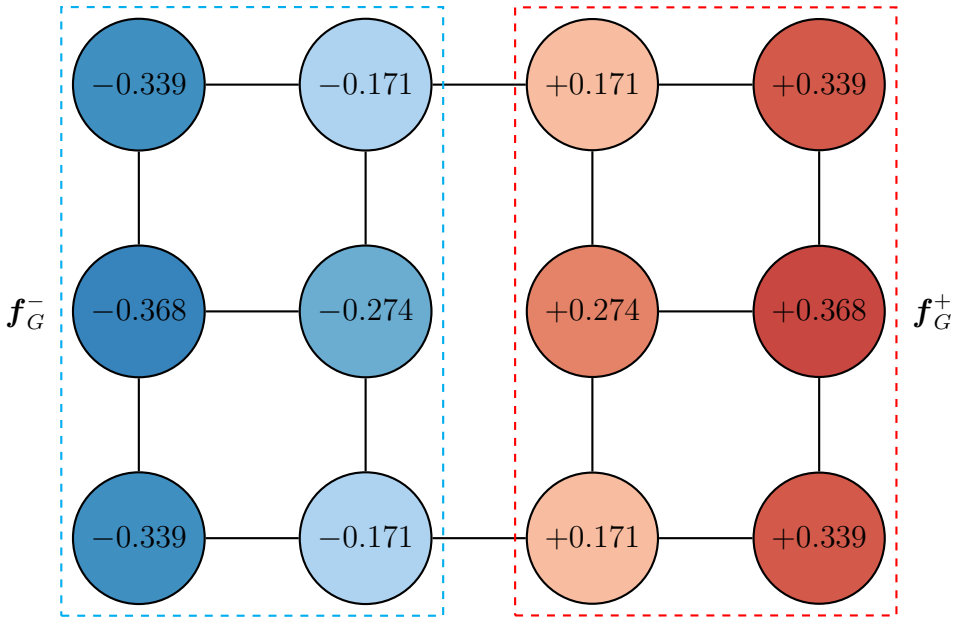


Figure 3.5: A spectral bipartitioning example. Each vertex is labeled with its Fiedler vector entry.

Chapter 4

The Spectral Recombination Proposal

In this chapter we describe our two algorithms that build on the ideas present in ReCom and RevRecom from the literature. Our first algorithm, Spectral Recombination, utilizes the spectral bipartitioning technique as discussed in section 3.5. The second algorithm, Balanced Spectral Recombination, illustrates how Spectral Recombination can be modified to account for the population deviation metric, ensuring that district populations stay as balanced as possible.

4.1 Spectral Recombination

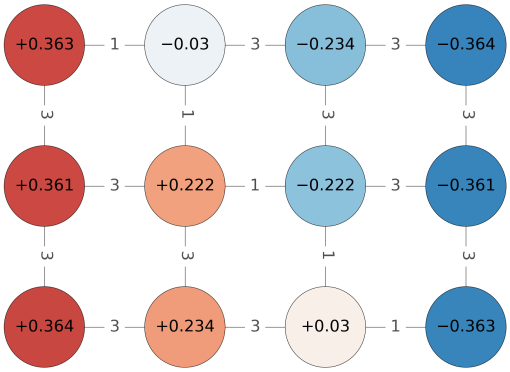
Our first algorithm, Spectral Recombination (SpecRecom), uses the graph's spectrum as described in section 3.5. First, like ReCom, SpecRecom chooses two districts to recombine based on the number of edges between them. District pairs that cut more edges of G are more likely to be chosen. After the recombination step, the induced subgraph's Laplacian and Fiedler vector are computed. From here, the split step proceeds by associating each of the subgraph's vertices with one of the entries in the Fiedler vector, and assigning those vertices with negative entry to one district and those with non-negative entry to the other.

Performing the split based on each vertex's Fiedler vector entry is a common theme for both this basic version of SpecRecom, as well as our "balanced" version in section 4.2. Algorithm 4 provides a precise description of SpecRecom.

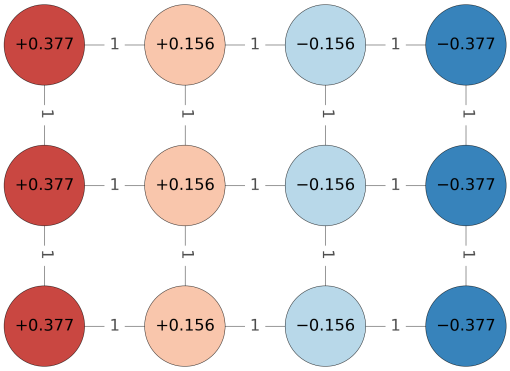
Lines 4-6 of algorithm 4 show that, to introduce non-determinism into SpecRecom, we first randomize the edge weights of G to be between 1 and 2. For the purposes of bipartitioning via the Fiedler vector, edge weights determine the strength of each edge—weaker edges are easier to cut. Without random edge weights, all edges would have equal weight and the proposal would be deterministic. Various edge weight configurations, along with the Fiedler vector entries they induce, are shown in figure 4.1. Our choice to use weights between 1 and 2 was arbitrary, but it appears to provide sufficient variety in generated plans.

Algorithm 4 Spectral Recombination Proposal

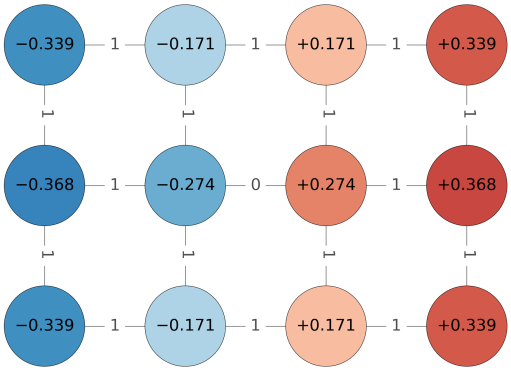
procedure SPECRECOM($\mathcal{D}; G$) *# input: current state; graph*
  select$\{u, v\} \in \partial_G \mathcal{D}$ UAR *# sample a cut edge*
  $H \leftarrow G[\mathcal{D}(u) \cup \mathcal{D}(v)]$ *# recombine*
  for $e \in E_H$ **do**
    $\text{weight}(e) \leftarrow \text{RAND}(1,2)$ *# randomize edge weights*
  end for
  $H_1 \leftarrow H[\boldsymbol{f}_H^+]$ *# apply spectral bipartitioning*
  $H_2 \leftarrow H[\boldsymbol{f}_H^-]$
  $\mathcal{D}' \leftarrow (\mathcal{D} \setminus \{\mathcal{D}(u), \mathcal{D}(v)\}) \cup \{V_{H_1}, V_{H_2}\}$
  return $\mathcal{D}'$ *# output: next state*
end procedure



(a) Edge weights determine how "easy" they are to cut.



(b) A graph with all edges having equal weight.

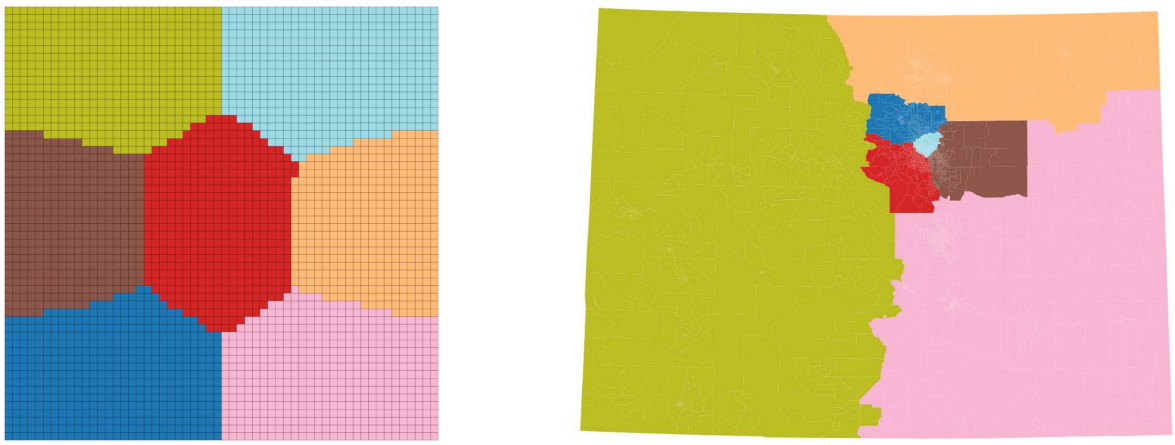


(c) A zero-weight edge recreates figure 3.5.

Figure 4.1: How edge weights impact Fiedler vector entries.

Figure 4.2 depicts two example districting plans produced by SpecRecom with 7 districts after 400 steps of the chain. Visually, both plans are more "compact" than RevRecom's, in the sense that the district shapes featured have lower cut edge counts.

We note that SpecRecom does not take into account vertex population. Therefore, while the plans may appear to be sufficient, their district populations are not well-balanced. For example, the grid graph and Colorado graph in figure 4.2 have population deviations of approximately 0.15 and 0.33 (respectively), both of which are worse than the majority of plans seen to be generated by RevRecom (see section 6.2).



(a) 56×56 grid graph.

(b) Colorado graph.

Figure 4.2: Example plans after 400 steps of SpecRecom.

Preliminary work by Davies et al. [19] attempted to incorporate vertex population in SpecRecom by using a Laplacian with vertex weights, as described by Chung and Langlands [20]. Unfortunately, they found that this version of SpecRecom struggled to enforce contiguity and decrease cut edge counts in generated plans. We hypothesize that including vertex weights in this way over-emphasized the importance of population balance. So to address these concerns, we added a separate step to SpecRecom in which population balance is considered *after* the Fiedler vector has been calculated using the population-oblivious Laplacian.

4.2 Balanced Spectral Recombination

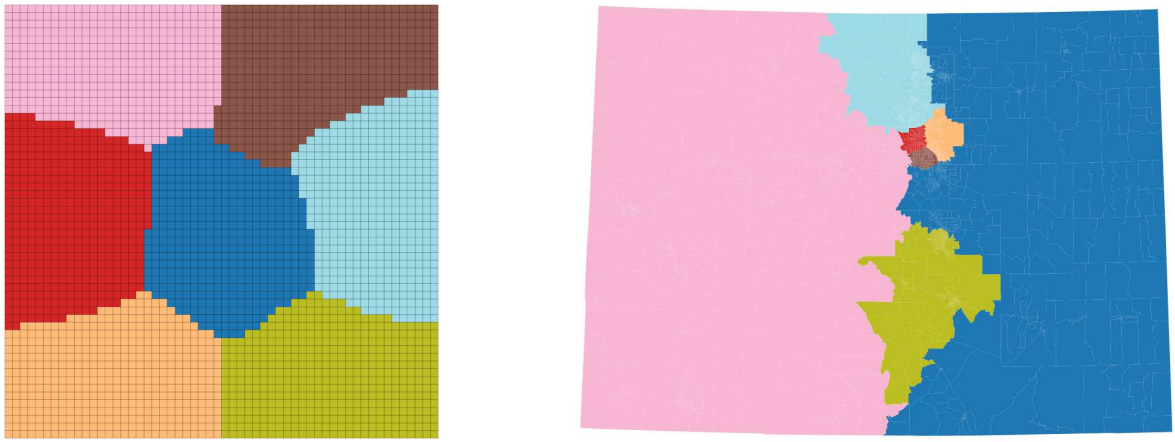
Balanced Spectral Recombination (BalSpecRecom) attempts to resolve the population balance issues of SpecRecom. In SpecRecom, the Fiedler cut is always made using a threshold of zero. That is, vertices whose entries are below this threshold are assigned to one district, and the rest to the other. In BalSpecRecom, we include a brute-force search for the threshold that optimizes for population balance, while attempting to maintain district contiguity.

Naively, the sheer number of available thresholds would render this brute-force search infeasible. However, recall that the purpose of a threshold is to partition the vertex set of G using the Fiedler vector. Therefore it suffices to consider just the entries of the Fiedler vector themselves as the threshold candidates. Algorithm 5 describes this process.

Algorithm 5 Balanced Spectral Recombination Proposal

procedure BALSPECRECOM($\mathcal{D}; G$) *# input: current state; graph*
  select $\{u, v\} \in \partial_G \mathcal{D}$ UAR *# sample a cut edge*
  $H \leftarrow G[\mathcal{D}(u) \cup \mathcal{D}(v)]$ *# recombine*
  for $e \in E_H$ **do**
    $\text{weight}(e) \leftarrow$ RAND(1,2) *# randomize edge weights*
  end for
  popDiff $\leftarrow \infty$
  splits $\leftarrow \{\{\mathcal{D}(u), \mathcal{D}(v)\}\}$
  for $t \in \boldsymbol{f}_H$ **do** *# brute force search for optimal threshold(s)*
    $\boldsymbol{f}_H^{\geq t} \leftarrow \{v \in V_H : \boldsymbol{f}_H[v] \geq t\}$
    $\boldsymbol{f}_H^{<t} \leftarrow \{v \in V_H : \boldsymbol{f}_H[v] < t\}$
    if $\text{contiguous}(H, \{\boldsymbol{f}_H^{\geq t}, \boldsymbol{f}_H^{<t}\})$ **then**
      $h_1, h_2 \leftarrow H[\boldsymbol{f}_H^{\geq t}], H[\boldsymbol{f}_H^{<t}]$
      if $|p(h_1) - p(h_2)| =$ *popDiff* **then**
        splits $\leftarrow$ *splits* $\cup\{\{h_1, h_2\}\}$
      else if $|p(h_1) - p(h_2)| <$ *popDiff* **then**
        splits $\leftarrow \{\{h_1, h_2\}\}$
        popDiff $\leftarrow |p(h_1) - p(h_2)|$
      end if
    end if
  end for
  $H_1, H_2 \leftarrow \underset{S \in \textit{splits}}{\text{argmin}} |\partial_H S|$ *# break ties with cut edge count*
  $\mathcal{D}' \leftarrow (\mathcal{D} \setminus \{\mathcal{D}(u), \mathcal{D}(v)\}) \cup \{V_{H_1}, V_{H_2}\}$ *# split*
  return $\mathcal{D}'$ *# output: next state*
end procedure

In figure 4.3, example plans generated by BalSpecRecom are shown. While the districts appear less "neat" than those generated by SpecRecom, the population deviations are greatly improved. The grid graph plan in figure 4.3 has a perfect population deviation of 0.0, and the Colorado graph plan has a population deviation of approximately 0.002. This improvement is quantified in section 6.2.



(a) 56×56 grid graph.

(b) Colorado graph.

Figure 4.3: Example plans after 400 steps of BalSpecRecom.

Chapter 5

Implementation

Our implementations of SpecRecom and BalSpecRecom are both written in Python, and utilize the GerryChain library for political redistricting [12]. We found that experimentation time could be reduced by consolidating our code, so we wrote a "Redistricting" Python package to contain experiment instances. Each instance holds all of the relevant data for a particular redistricting algorithm, ready to run with a set of tunable parameters, that automatically formats and outputs the necessary data at the end of the instance's run.

Moreover, to provide a baseline with which to compare cut edge counts, we implemented a "spectral k-means clustering" (SpecKMeans) algorithm. This algorithm simply combines the idea of embedding vertices in a vector space, as done by Lee et al. [18], with k-means clustering to partition these embedded vertices into districts. This algorithm is deterministic and blind to contiguity, but serves as a good baseline by achieving low cut edge counts with few lines of code.

All source code for the Redistricting package, as well as the data collected by it, is available at https://github.com/MaxFlorescence/spectral_redistricting.

```
r = Redistricting(
    graph = 'CO_graph/co_precincts.shp',
    k = 7,
    assignment = 'CD116FP',
    proposal = 'specrecom',
    steps = 400,
    single_updaters = ['cut edges', 'population disparity'],
    population_key = 'TOTPOP',
    graph_name = 'Colorado graph'
)

r.run(
    interactive_level = 'progress',
    plot_interval = 100,
    output_level = 'less',
    output_parent = 'Colorado_graph-output',
    checkpoint_interval = 100,
    checkpoint_dest = 'Colorado_graph-test_run-checkpoint.json'
)
```

Figure 5.1: Code snippet demonstrating how to use the Redistricting package.

Chapter 6

Evaluation

6.1 Methodology

To evaluate if SpecRecom and BalSpecRecom sample districting plans with low cut edge counts, we experimentally compared the two to RevRecom's performance. We wrote several Slurm scripts to run many Redistricting instances on the CSU Computer Science department's Falcon cluster. Our independent variables for each experiment were the algorithm that was run and the graph that it was run on. Our dependent variables were cut edge count and population deviation of the resulting plan. The details are summarized in table 6.1. This included running 10,000 SpecRecom and BalSpecRecom chains for 400 steps each, across 20 cores.

Table 6.1: Summary of experimental configurations.

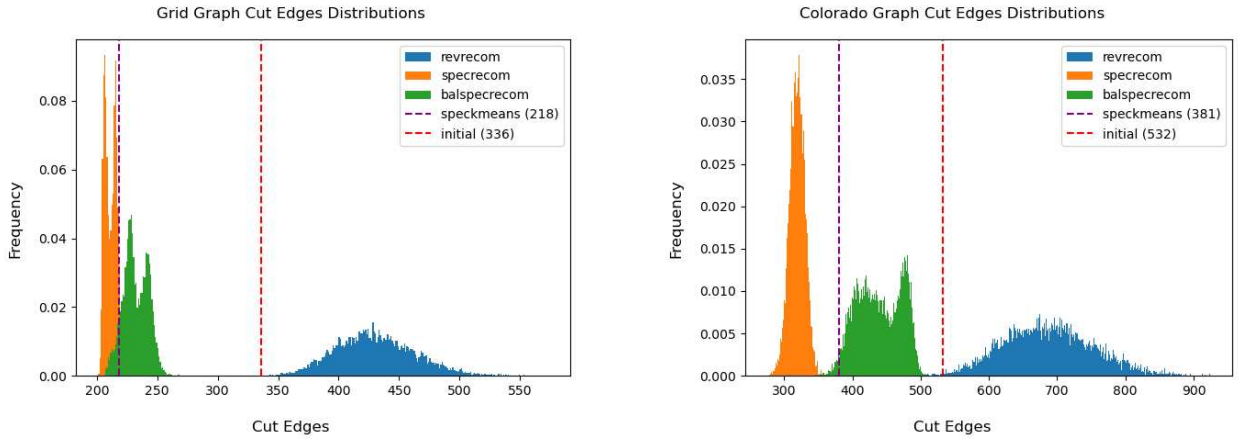
Algorithm	**Graph**	**Steps** (N)	**Districts** (k)	**Ensemble Size** (m)
RevRecom	56×56 grid graph 2018 Colorado districts	10,000	7	10,000
SpecRecom	56×56 grid graph 2018 Colorado districts	400	7	10,000
BalSpecRecom	56×56 grid graph 2018 Colorado districts	400	7	10,000

While GerryChain provides an implementation of RevRecom, it does not utilize parallelism to consider multiple spanning trees, as discussed in section 3.4.2. Therefore, we used a Rust implementation written by Rule et al. [14] to first generate many different plans. Then to facilitate data collection, we used a technique known as sub-sampling, as described in *Political Geometry* [2]. Instead of running m chains independently for N steps each, we run a single chain for $m \cdot N$ steps, sampling one state every N steps to produce m plans. In our case, $m = N = 10{,}000$.

To provide a baseline with which to compare our results from the Colorado graph, we chose to use a square grid graph with 56 vertices per side. This value was chosen as it was the integer whose square ($56^2 = 3136$) was the closest to the number of vertices in the Colorado graph (3135).

6.2 Results

Figures 6.1 and 6.2 show the distribution of cut edge counts and population deviations respectively, across the 6 different combinations of algorithm and graph. Each histogram is normalized independently. In orange is the data for RevRecom, in blue is SpecRecom, and in green in BalSpecRecom. A vertical, dashed red line indicates the initial values of the graph for each metric. A vertical, dashed purple line indicates the baseline SpecKMeans value. For the 56×56 grid graph, the initial districting plan was seven equally-sized horizontal stripes (as seen in figure 3.2a), and for the Colorado graph the initial plan was the 2018 congressional district map (as seen in figure 3.1a).



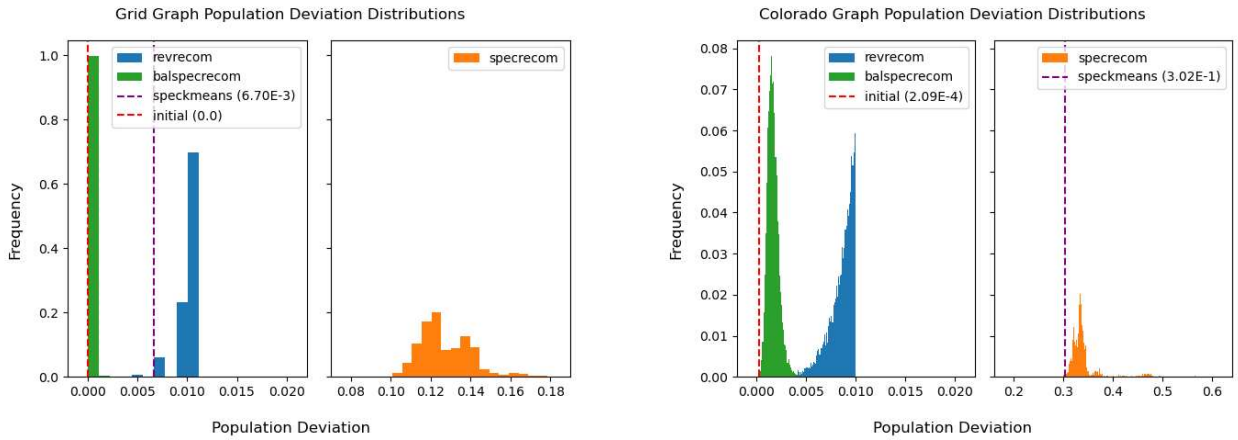
(a) 56×56 grid graph.

(b) Colorado graph.

Figure 6.1: Cut edge distributions across each algorithm.

6.3 Discussion

In terms of cut edges, we see a clear difference between our SpecRecom and BalSpecRecom algorithms and RevRecom. Figure 6.1 shows that for both the 56×56 grid graph and the Colorado



(a) 56×56 grid graph.

(b) Colorado graph.

Figure 6.2: Population deviation distributions across each algorithm.

graph, SpecRecom achieves the lowest numbers of cut edges in its final generated districting plans, followed closely by BalSpecRecom plans, and then RevRecom plans with around twice as many cut edges on average. RevRecom's distribution also lies mostly above the initial cut edges value for each graph, while SpecRecom and BalSpecRecom lie below it. Relative to the SpecKMeans baseline, only SpecRecom manages to mostly generate plans with fewer cut edges.

In terms of population deviation, the large difference between SpecRecom and BalSpecRecom or RevRecom necessitated that the plot be split. Figure 6.2 shows that BalSpecRecom and RevRecom both manage to get population deviation values below 0.01 (a 1% tolerance), while SpecRecom achieves values for the Colorado graph that are an order of magnitude higher. For the 56×56 grid graph, BalSpecRecom most frequently manages to maintain the perfect population balance of the initial plan, and for the Colorado graph it consistently achieves deviations less than 0.005. In both cases, SpecRecom achieved values mostly above the SpecKMeans baseline, and BalSpecRecom achieved values below it.

In the case of RevRecom, the sharp cutoff at 0.01 population deviation can be attributed to the algorithm's implementation. Using ε-balance edges, RevRecom never accepts plans that exceed a population deviation of ε. For the implementation of RevRecom that we used, $\varepsilon = 0.01$.

Chapter 7

Conclusions

Our results provide empirical evidence in favor of the hypothesis that Fiedler vectors can be used to create districting plans with much shorter boundaries than existing algorithms from the literature. Figure 6.1 demonstrates that SpecRecom and BalSpecRecom both tend to generate districting plans with fewer cut edges than RevRecom, and in much fewer steps (400 as opposed to 10,000). Both the 56×56 grid graph and the Colorado graph exhibit this behavior, indicating that spectral recombination methods can work across different types of graphs. Moreover, SpecRecom produces plans with fewer cut edges and a higher affinity for contiguity than a simple clustering approach like SpecKMeans. That is, SpecRecom is more likely to produce contiguous plans, whereas SpecKMeans would need to be modified to guarantee contiguous plans.

Figure 6.2 demonstrates that the SpecRecom algorithm can be used as a base to create new algorithms that effectively take different redistricting metrics into consideration, specifically population deviation. RevRecom achieves population deviation of at most 0.01, since this is the value of ε used in our experimentation. However, due to SpecRecom making no attempt to maintain population balance within the same ε tolerance, it achieves significantly higher population deviation values for both graphs. In contrast, BalSpecRecom mostly achieves population deviation values lower than those achieved by RevRecom, owing to its brute-force search for the optimal spectral bipartitioning threshold (with respect to population deviation) during each step. The values given by the SpecKMeans also reveal that a smarter algorithm is necessary; while it does achieve a population deviation below 0.01 for the 56×56 grid graph, it is on-par with SpecRecom for the real-world Colorado graph.

When taken together, these results imply that there exists a trade-off between generating districting plans with few cut edges and with low population deviation. Figure 6.2 clearly shows that BalSpecRecom generates plans with population deviations lower than SpecRecom, but 6.1 indicates that this comes at the cost of increasing the number of cut edges relative to SpecRecom.

Chapter 8

Related Work

8.1 Early Redistricting Methods

Chen and Rodden [21] first published the idea of districting plan comparison in 2015. With the intent to detect partisan gerrymanders in districting plans, they focused on generating a benchmark plan to compare currently implemented plans against. In essence, their algorithm works by starting with each precinct as its own district, and whittling down the number of districts by combining two at a time. Then, the algorithm repeatedly flips units to improve population balance. They applied their result to the 2012 Florida congressional district map and were able to successfully detect that it had been gerrymandered.

In 2017, Kawahara et al. [22] described an algorithm for exhaustively enumerating graph partitions for districting plans. They focus on using a zero-suppressed decision diagram data structure to explore the space of all districting plans given a graph and district count. They also detail how to modify the original algorithm to consider population disparity, making sure that only plans with bounded disparity are enumerated.

Cohen-Addad et al. [23] provided another algorithm for creating districting plans utilizing power diagrams in 2018. Their algorithm uses centroidal power diagrams to determine the districts to which population units should be assigned, by solving the balanced k-means clustering problem. These districts end up being polygons whose centroids define the center points of each power region in the diagram. Notably, the average number of sides for all polygons is less than 6 and their populations differ by at most 1, and so the districts end up being balanced and visually compact.

Levin and Friedler [24] in 2019 used a similar idea with Voronoi diagrams. Like Cohen-Addad et al, their algorithm iterates using two components, a Voronoi component to bipartition regions and a swapping component to ensure population balance and contiguity. But in contrast, Levin and Friedler's is a divide-and-conquer algorithm, bipartitioning the whole state first then recursing

downward. Ultimately, their algorithm generates plans with strong compactness scores by their measures.

8.2 MCMC Methods in the Literature

Fifield et al. [25] proposed the first MCMC redistricting algorithm in 2020. Their motivation for investigating an MCMC method is that there were not many simulation methods in the literature, and that the state-of-the-art algorithm was not ideal. A step of their chain functions by randomly selecting clusters of nodes along the current district boundaries, and then proposing swaps where clusters can switch membership to adjacent districts. This new plan is then accepted or rejected before continuing to the next step.

The idea to apply an ensemble of generated districting plans for comparison with a proposed real-world plan was enacted by Herschlag et al. [3], also in 2020. Building on Fifield et al.'s work, they use their MCMC method with an added simulated annealing procedure to generate this ensemble of plans. Using this ensemble of 66.5 thousand plans, they were able to determine that the enacted North Carolina districting plan was biased, whereas previous plans drawn by bipartisan judges were not.

In 2021, Clelland et al. published on the ReCom MCMC method developed by the MGGG redistricting lab in 2018, and DeFord et al. characterized it as one method among a family of recombination methods [4, 5, 13]. These groups introduce the idea that the spanning tree distribution is important for compactness in districting plans, and they show how to target this distribution using MCMC methods.

Then, in 2022, Cannon et al. [7] modified the ReCom method to be reversible, making RevRecom. This improved on previous work by ensuring that the spanning tree distribution was the exact distribution that the MCMC method converges to. They also describe a parallel implementation of RevRecom that makes up for the increased mixing time resulting from this modification.

In the same year, Cannon et al. [8] also published work improving biased random walk MCMCs for redistricting by adding "short bursts". Their motivation for this work was to maximize the

number of simultaneous majority-minority districts in generated plans. Their idea was to run the chain in periods of unbiased random walks called short bursts. At the end of each burst, the chain would be re-started from the most extreme plan seen within the most recent burst window.

8.3 Spectral Graph Theory and Redistricting

So far, we have seen many different types of solutions to the redistricting problem. However, to our knowledge no redistricting algorithms have been published that utilize the idea of spectral partitioning. Most relevantly, spectral bipartitioning was refined in Urschel and Zikatanov's work [17], and applied to k-partitioning by Lee et al. [18]. Our work can be seen as an attempt to unite these two areas of research to produce contiguous spectral k-partitions.

In 2014, Urschel and Zikatanov refined the work of Miroslav Fiedler, for whom the Fiedler vector is named. Their work proved many theorems around the robustness of using the Fiedler vector to partition a graph into two connected subgraphs. Notably, with their generalized bisection theorem, they prove that for any connected, undirected graph with no self-loops there exists a Fiedler vector with which the graph can be bipartitioned into connected subgraphs.

In the same year, Lee et al. [18] built on the idea of Cheeger's inequality. It states that a graph has a sparse cut if and only if there are two eigenvalues in its spectrum that are close to zero–and generalize it to show that a sparse k-cut exists if and only if there are k eigenvalues close to zero. Their general-purpose algorithm works by embedding each vertex of the graph in a real vector space, then outputting subsets of vertices that form least-expanding sets.

While this technique can produce a k-partition with small cut edge count and population deviation, it is not necessarily suited for the problem of congressional redistricting. It is unclear if the approach would be able to guarantee contiguity in districting plans, but we hypothesize that it does not. Moreover, the approach may need to be modified to ensure population balance.

Chapter 9

Future Work

Here, we list several areas in which our work can be expanded. The first two relate to improving our algorithms' robustness, the next two to fully addressing the open questions mentioned in section 2.3, and the last to what work needs to be done before applying our algorithms to redistricting in the real world.

9.1 Alternate Bipartitioning Vectors

As Urschel and Zikatanov note, there may be more than one Fiedler vector for a particular graph, depending on the geometric multiplicity of the second eigenvalue of the Laplacian. Moreover, Colin de Verdière's invariant [26] reveals that the corank (and thus the geometric multiplicity of the algebraic connectivity) of a planar graph's Laplacian can be at most three. Therefore, even though our spectral recombination algorithms seem to be well-suited for producing ensembles, we do not thoroughly explore the full eigenspaces. Future work might consider an optimization over the entire eigenspace of Fiedler vectors.

9.2 Improved Balancing Sweep

Since our BalSpecRecom algorithm considers applying a variable threshold to the Fiedler vector's entries, it strictly enforces the order of the graph's vertices according to those entries. This may not be ideal, as the optimal split in terms of population deviation might be inaccessible since it could require a slightly different ordering. To this end, future work might consider some kind of "noisy sweep", in which for each threshold the vertices near the threshold can be permuted. Considering all permutations would lead to an exponential increase in running time, and so perhaps a single, randomly chosen permutation would be considered instead.

Alternatively, we note that Recom and RevRecom take a population-balancing parameter, ε, while BalSpecRecom only considers the threshold that maximizes population balance at each step.

Instead, BalSpecRecom could consider "ε-balance thresholds"—any threshold that achieves population balance within some ε tolerance. This modification may allow plans with fewer cut edges *and* acceptable population deviation to be generated, addressing the trade-off discussed in section 6.3.

9.3 Algorithm Optimization

Recall that DeFord et al.'s open question asks researchers to "Propose other balanced bipartitioning methods to replace spanning trees, *supported by fast algorithms*" [5]. Regarding this, we made no attempt to optimize SpecRecom or BalSpecRecom beyond basic implementations in Python.

Rule et al.'s Rust implementation of RevRecom [14] was optimized to perform steps of the chain in parallel, decreasing the amount of time the chain spends rejecting possible moves. In our experimentation, we found that with 8 cores the Rust implementation of RevRecom completes approximately 55.3 thousand steps per second on the Colorado graph. While our implementations of SpecRecom and BalSpecRecom are not comparable to this implementation of RevRecom, we have observed practically that they take much more time to generate the same number of plans.

Given that the majority of SpecRecom's computational load is spent performing matrix operations, we hypothesize that optimizing it would be relatively easy. However, BalSpecRecom includes all of SpecRecom's code as well as a brute force loop through the Fiedler vector, adding an $O(n)$ factor to the time complexity. An optimized version might consider parallelizing this loop or replacing it with a smarter method of searching for an acceptable threshold, such as stopping early once an ε-balance threshold is found.

9.4 Statistical and Theoretical Analysis

Recall that DeFord and Duchin's invitation asks that "New methods should strive for easy implementation, low rejection rate, adaptability to varied districting criteria, and of course *theoretical properties like provable ergodicity and reversibility*" [2]. While our work demonstrates

that spectral recombination algorithms are viable, we do not know if either SpecRecom or BalSpecRecom is aperiodic, irreducible, or reversible. Future work could give insight in this area, providing a rigorous understanding of the chains by determining their stationary distributions and (in particular) mixing times.

In the abscence of theoretical guarantees, Chikina et al. describe several statistical tests. The "$\sqrt{\varepsilon}$" test [27] would allow it to be determined with significance $p = \sqrt{2\varepsilon}$ whether or not a given run of a Markov chain is mixed at its current state. Furthermore, an extension of this test allows for the effect size ε to be separated from the significance p [28].

9.5 Considering Additional Plan Constraints

Clelland et al. [6] list districting plan requirements for Colorado, two of which are that plans must "preserve communities of interest and political subdivisions" and "minimize the number of divisions when a city, county, or town is divided".

While our algorithms do not consider these constraints during the chains' runs, we note that GerryChain does provide methods for determining how many units are split by a given plan. Moreover, these methods are parameterized by which type of unit (counties, precincts, etc.) is being considered. Future work could analyze how SpecRecom and BalSpecRecom perform under these metrics, and propose modifications to comply with them, if necessary. Another constraint listed by Clelland et al. is that plans must "maximize the number of politically competitive districts". Again, GerryChain provides metrics for determining how competitive a given districting plan is, and so the same can be said about future work with regards to this constraint.

We have demonstrated that our SpecRecom can easily be modified to account for population deviation, yielding BalSpecRecom. Future work might be able to modify SpecRecom in similar ways to account for the metrics and constraints discussed above.

Bibliography

[1] gerrymander. In *American Heritage Dictionary of the English Language*. HarperCollins, 2022. https://www.ahdictionary.com/word/search.html?q=gerrymander.

[2] Moon Duchin and Olivia Walch. *Political geometry: Rethinking redistricting in the US with math, law, and everything in between*. Birkhäuser, 2022. Online pre-print available at https://mggg.org/gerrybook.html.

[3] Gregory Herschlag, Han Sung Kang, Justin Luo, Christy Vaughn Graves, Sachet Bangia, Robert Ravier, and Jonathan C. Mattingly. Quantifying Gerrymandering in North Carolina. *Statistics and Public Policy*, 7(1):30–38, 2020. https://doi.org/10.1080/2330443X.2020.1796400.

[4] Jeanne N. Clelland, Nicholas Bossenbroek, Thomas Heckmaster, Adam Nelson, Peter Rock, and Jade VanAusdall. Compactness statistics for spanning tree recombination. 2021. https://api.semanticscholar.org/CorpusID:234771721.

[5] Daryl DeFord, Moon Duchin, and Justin Solomon. Recombination: A Family of Markov Chains for Redistricting. *Harvard Data Science Review*, 3(1), mar 31 2021. https://doi.org/10.1162/99608f92.eb30390f.

[6] Jeanne Clelland, Haley Colgate, Daryl DeFord, Beth Malmskog, and Flavia Sancier-Barbosa. Colorado in context: Congressional redistricting and competing fairness criteria in colorado. *Journal of Computational Social Science*, 5(1):189–226, 2022. https://doi.org/10.1007/s42001-021-00119-7.

[7] Sarah Cannon, Moon Duchin, Dana Randall, and Parker Rule. Spanning Tree Methods for Sampling Graph Partitions. *arXiv*, 2210.01401, 2022.

[8] Sarah Cannon, Ari Goldbloom-Helzner, Varun Gupta, J. N. Matthews, and Bhushan Suwal. Voting rights, markov chains, and optimization by short bursts. *Methodology and Computing in Applied Probability*, 25(1):36, 2023. https://doi.org/10.1007/s11009-023-09994-1.

[9] William Vickrey. On the prevention of gerrymandering. *Political Science Quarterly*, 76(1):105–110, 1961. https://doi.org/10.2307/2145973.

[10] Olivia Guest, Frank J Kanayet, and Bradley C Love. Gerrymandering and computational redistricting. *J Comput Soc Sci*, 2(2):119–131, 2019. https://doi.org/10.1007/s42001-019-00053-9.

[11] Ariel D. Procaccia and Jamie Tucker-Foltz. Compact redistricting plans have many spanning trees. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3754–3771. https://doi.org/10.1137/1.9781611977073.148.

[12] Data and Democracy Lab. Gerrychain. Python Library, 2018. https://github.com/mggg/GerryChain.

[13] Metric Geometry and Gerrymandering Group. Comparison of districting plans for the virginia house of delegates. Technical report, Cornell University, Brooks School of Public Policy, 2018. https://mggg.org/VA-report.pdf.

[14] Parker J. Rule, Mike Sarahan, and Max Fan. Fastest recom chain in the west. GitHub, 2022. https://github.com/pjrule/frcw.rs.

[15] Miroslav Fiedler. Algebraic connectivity of graphs (english). *Czechoslovak Mathematical Journal*, 23, 1973. https://doi.org/10.21136/CMJ.1973.101168.

[16] Fan R. K. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[17] John C. Urschel and Ludmil T. Zikatanov. Spectral bisection of graphs and connectedness. *Linear Algebra and its Applications*, 449:1–16, 2014. https://doi.org/10.1016/j.laa.2014.02.007.

[18] James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *J. ACM*, 61(6), December 2014. https://doi.org/10.1145/2665063.

[19] Ewan Davies, Ryan Job, Hannah Kim, and Hyojin Seo. A spectral approach to legislative districting in colorado. Research poster, Summer 2023.

[20] F.R.K. Chung and Robert P. Langlands. A combinatorial laplacian with vertex weights. *Journal of Combinatorial Theory, Series A*, 75(2):316–327, 1996. https://doi.org/10.1006/jcta.1996.0080.

[21] Jowei Chen and Jonathan Rodden. Cutting through the thicket: Redistricting simulations and the detection of partisan gerrymanders. *Election Law Journal*, 14:331–345, 2015. https://api.semanticscholar.org/CorpusID:146890736.

[22] Jun Kawahara, Takashi Horiyama, Keisuke Hotta, and Shin-ichi Minato. Generating all patterns of graph partitions within a disparity bound. In Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen, editors, *WALCOM: Algorithms and Computation*, pages 119–131, Cham, 2017. Springer International Publishing. https://doi.org/10.1007/978-3-319-53925-6_10.

[23] Vincent Cohen-Addad, Philip N. Klein, and N. Young. Balanced power diagrams for redistricting. *ArXiv*, abs/1710.03358, 2017. https://api.semanticscholar.org/CorpusID:12961735.

[24] Harry A. Levin and Sorelle A. Friedler. Automated congressional redistricting. *ACM J. Exp. Algorithmics*, 24, April 2019. https://doi.org/10.1145/3316513.

[25] Benjamin Fifield, Michael Higgins, Kosuke Imai, and Alexander Tarr. Automated redistricting simulation using markov chain monte carlo. *Journal of Computational and Graphical Statistics*, 29(4):715–728, 2020. https://doi.org/10.1080/10618600.2020.1739532.

[26] Yves Colin de Verdière. On a new graph invariant and a criterion for planarity. In *Graph Structure Theory*, 1991. https://api.semanticscholar.org/CorpusID:30121343.

[27] Maria Chikina, Alan Frieze, and Wesley Pegden. Assessing significance in a markov chain without mixing. *Proceedings of the National Academy of Sciences*, 114(11):2860–2864, 2017. https://doi.org/10.1073/pnas.1617540114.

[28] Maria Chikina, Alan Frieze, Jonathan C. Mattingly, and Wesley Pegden. Separating effect from significance in markov chain tests. *Statistics and Public Policy*, 7(1):101–114, 2020. https://doi.org/10.1080/2330443X.2020.1806763.