A REVERSIBLE RECOMBINATION CHAIN FOR GRAPH PARTITIONS

SARAH CANNON, MOON DUCHIN, DANA RANDALL, PARKER RULE

Contents

Abstract

In the last decade, computational approaches to graph partitioning have made major inroads in U.S. courts considering political redistricting. Mathematically, a districting plan can be viewed as a balanced partition of a graph into connected subsets. Examining a large sample of valid alternative districting plans can help us recognize **gerrymandering** against an appropriate neutral baseline.

One algorithm that is widely used to produce random samples of districting plans is a Markov chain called ReCom, which repeatedly fuses adjacent districts, forms a spanning tree of their union, and splits that spanning tree in a balanced cut to form new districts. One drawback is that this chain's stationary distribution has no known closed form when there are three or more districts because it does not satisfy the conditions for reversibility or detailed balance. In this paper, we modify ReCom slightly to make it reversible, resulting in a new Markov chain, RevReCom. This new chain converges to the simple, natural stationary distribution that ReCom was originally designed to approximate: a plan's stationary probability is proportional to the product of the number of spanning trees of each district. This is a measure of district "compactness" that is aligned with notions of **community structure**. After deriving the steady state formally, we present heuristic/empirical evidence that the convergence is efficient enough for the method to be realistically useful. In addition to primary use benchmarking a normal range for statistics over redistricting plans, this chain can also be used to validate other methods that target the spanning tree distribution.

1. Introduction

1.1. **Redistricting and ensembles.** Around the world, many countries divide their territory into regions that conduct elections, from the provinces of South Africa to the departments of France to the states of Brazil. The placement of those boundary lines has a significant impact on the representational outcomes. In the United States, the stakes are raised by the fact that many electoral districts are used to choose a single representative in a winner-take-all fashion; lines are regularly erased and redrawn; and in most cases, the elected officials themselves control this *redistricting* process. This applies most famously to the national House of Representatives.

When you administer the partition of a territory into single-member districts, you can often exercise a strong degree of control over the outcomes. A minority designated as group A with just over 40% of the vote can get anywhere from zero to 80% of the representation in an orchestrated scenario. At one extreme, A voters could have 40% share in each district, controlling none; by contrast, lines could be drawn to give just over half share for A in 80% of districts, with A voters totally absent from the rest. The practice of abusing line-drawing power to favor some interests over others is called *gerrymandering*, a term that dates to a newspaper broadside from 1812.

But what is the non-gerrymandered baseline? If A voters make up 40% of an electorate, there is no mathematical reason that the partition process should deliver an expectation of 40% of the seats. Instead, the expectation depends heavily on the geographic distribution of those voters. Starting in around 2013, experts working in litigation have attempted to use computational techniques to sample from the (very large) space of plausible districting plans, effectively arguing that a random sample chosen without partisan data provides the unadulterated normal baseline. That is, we can overlay possible partitions (i.e., district maps) on the recent election patterns to study the representational outcomes. For instance, if the A share of representation tends to be 25-35% in a sample constructed from the legitimate rules, then this gives us guidance to what outcomes are explained by the mathematics of partitioning, given the actual distribution of voters. Extreme outliers may be regarded as flags that there is some agenda present that is external to the coded rules—though of course this comparison method can not tell us whether that agenda is benign or self-serving. The underlying idea of comparing a proposed plan to a sample of alternative plans is called the *ensemble method* for redistricting analysis, and the random sample of plans is called an *ensemble*. For this entire line of reasoning to be compelling requires an understanding of the probability distribution on the space of plans from which the random sample is drawn.

Below, we describe redistricting as a graph partition problem and give an overview of the obstructions to effective sampling encountered by off-the-shelf MCMC methods in §1.2. We introduce an attractive choice of probability distribution on plans in §1.3 and a new *reversible recombination*

chain in §2 which is shown to target that distribution. Finally, we present empirical/heuristic evidence of effective sampling in §3 and discuss the applications of the work in §4.

1.2. **Graph models and Markov chains.** Most mathematical treatments of redistricting have a discrete setup in common: the electoral geography in a given state or region can be represented as a graph, where nodes correspond to small geographical regions or units (such as census blocks or voting precincts) and edges indicate adjacency of the units. Nodes are weighted according to the population of the corresponding unit. In this formulation, a *districting plan* is a partition of the vertices into k connected subgraphs with approximately balanced population. The problem of building an effective baseline for redistricting purposes can be reduced to the problem of randomly sampling such balanced graph partitions.

Markov chain Monte Carlo algorithms are ubiquitous across scientific disciplines as a computational means for randomized study of large, complicated sets (for surveys, see [24, 5, 28]). The idea is to design a random walk that moves among configurations—in this case, the random walk steps from partition to partition. Even though each plan might only be connected to a small set of nearest neighbors, an MCMC process is designed so that the random walk will eventually converge to a useful stationary distribution over the entire space of configurations. In this case, plans visited by the random walker will be added to the ensemble of alternative options.

Chains that are *reversible* (satisfying a property known as *detailed balance*) are particularly nice because they tend to have more easily described limiting distributions.[1] A key consideration with applied Markov chains generally is the mixing time; MCMC can be impractical if the convergence is too slow. Some of the challenges are elucidated by understanding the efficacy of sampling for problems from statistical physics, such as the ferromagnetic Ising model.[2]

For the redistricting application, local or quasi-local moves can generally be called *flip chains*, since they change the district assignment of one or a small number of units at a time, which can be visualized as flipping the color of the nodes. Flip chains can face impediments to efficiency when trying to draw a diverse sample while imposing global constraints on partitions. In addition to population balance, plausible districting plans must satisfy geometric constraints: districts must be *connected*, and partitions must have a controlled cutset so that districts meet in tolerably short boundaries—in redistricting lingo, connected plans are called *contiguous* and reasonably shaped plans are called *compact*. For a graph partition P, compactness is often managed by looking at the cut edge set ∂P, which consists of the edges in the graph whose endpoints lie in different districts of the partition. The difficulty of managing these constraints in a flip chain is discussed at length in the survey by DeFord–Duchin–Solomon [22], where it is shown that flip chains face extremely long convergence times (similar to a solid in statistical physics). Additional results in [30] show that flip chains can still face long convergence times even for simple subgraphs of the square gird.

To improve convergence, one line of research inspired by obstacles to efficient sampling for low temperature statistical physics models is to introduce temperature variation, such as employed by the research teams of Jonathan Mattingly and Kosuke Imai using simulated annealing and/or tempering [32, 29]. But even these modifications may not be adequate to handle forbidding complexity

[1] Other valuable techniques, like some significance testing results, are also available when the chain is reversible. See, for instance, [15, 14].

[2] The Ising model assigns "spin" to vertices in a grid, not unlike district assignments in the redistricting setting. Markov chains making small, localized updates in each step tend to perform well at high temperatures, when the models behave like a liquid or gas, but perform poorly at low temperatures, when the models behave like solids exhibiting long-range order that take a long time to converge (see, e.g., [43, 36]). For redistricting specifically, the important PNAS paper of Chikina–Frieze–Pegden avoids any need for convergence, instead giving methods for significance tests that work for reversible chains in great generality [15]. The CFP test can rigorously detect if a plan is unlikely to have been drawn from the stationary distribution of a given reversible Markov chain. If that stationary distribution is useful, then this mismatch can be damning. This approach was improved in the follow-up paper [14]. At the same time, because it works by local comparisons, this test is not able to report the global baseline or normal range for any summary statistics.

obstructions. In [39], Najt–DeFord–Solomon show that sampling from an approximately uniform distribution is computationally intractable on the class of 2-connected planar graphs, in the sense that the existence of any efficient algorithm implies $RP = NP$. As with all complexity results of this flavor, this provides limited practical insight, because real-world graphs may belong to a smaller class with no complexity obstruction. However, the papers [39, 22] also provide empirical evidence of performance obstructions for flip chains at scale, even with accelerations implemented.[3] Importantly, both the theoretical and the empirical obstructions also apply to approximate sampling from a distribution where the probability of a graph partition P is proportional to $\lambda^{|\partial P|}$ for any $0 < \lambda \leq 1$. This suggests daunting challenges for the most obvious choices of distributions that control ∂P directly.

Thus, there has been great interest in new ideas of good target distributions that sample from balanced, connected partitions with short boundaries for the study of redistricting. In this paper, we will emphasize the *spanning tree distribution* on partitions as an excellent default choice. We will describe a reversible Markov chain that targets that distribution, together with an efficient implementation.

1.3. **Spanning trees and community structure.** Graphs are defined by sets of nodes, with edges defining node adjacency. A *tree* is a minimally-connected graph—it is defined by the property that removing any edge would disconnect the graph. A vast number of applications in theoretical computer science and engineering rely on the use of *spanning trees* of a graph G, which are tree subgraphs of G using all vertices. The number of spanning trees $N_{\mathsf{ST}}(G)$ of a graph G is often referred to as its *complexity* and is used as a measure of well-connectedness [37]. For a simple example, four vertices arranged in a path are a tree, so $N_{\mathsf{ST}} = 1$ for that graph. Four vertices arranged in a square have $N_{\mathsf{ST}} = 4$ because removing any one edge leaves a tree; four vertices in a complete graph (with all six possible edges present) have $N_{\mathsf{ST}} = 16$, and the number grows quickly as the graph gets larger and more internally connected. Kirchhoff's celebrated Matrix-Tree theorem expresses this count elegantly: $N_{\mathsf{ST}}(G) = \frac{1}{n}\prod_{i=1}^{n-1}\lambda_i$, the product of the non-zero eigenvalues of the graph Laplacian.

As we have seen, the redistricting application requires that the cutsets of partitions be controlled in the sampling process. One way to prioritize efficient cuts is to select for districts with high interior connectivity. Therefore we will seek to weight a graph partition by the product of the spanning tree counts of its districts—in other words, if P is a partition of a graph into connected subgraphs $P_1, \ldots, P_k$ representing its districts, then we want to sample P with a probability proportional to $\prod_i N_{\mathsf{ST}}(P_i)$. We call this the *spanning tree distribution* on partitions.

From this expression, it is clear that the spanning tree distribution favors districts with large interior, because those typically have more spanning trees than more "spindly" districts. Recent work by Clelland et al. has investigated, in the case where there are two districts, the strong linear relationship between $|\partial P|$ and the modified spanning tree weight $\log\left(N_{\mathsf{ST}}(P_1)\, N_{\mathsf{ST}}(P_2)|\partial P|\right)$, in both grids and real-world examples [17][4]. Relatedly, a recent paper by Procaccia–Tucker-Foltz [41] proves, using effective resistance, that there is an asymptotically exponential relationship between the cut edge count $|\partial P|$ and the spanning tree distribution. The constants in this exponential relationship can vary depending on the structure of the graph.

Selecting subgraphs with high internal connectivity has an ample literature of its own, going by the name *community detection* in the network science literature. As one survey article puts it, "One mesoscopic structure, called a *community*, consists of a group of nodes that are relatively densely connected to each other but sparsely connected to other dense groups in the network" [40].

[3] A new preprint of Frieze and Pegden [30] further explores Glauber dynamics on partitions, confirming exponenial mixing times even on grid-graphs.

[4] This distribution was explored because it is the stationary distribution of the Markov chain ReCom, introduced in the next subsection

The explicit use of spanning tree counts for community detection is employed in numerous papers, including [34].

From this point of view, the spanning tree count over the districts reflects how effectively the plan picks out districts that correspond to adjacency clusters in a geographical network. Looking at sample outputs makes it clear that weighting P by $\prod N_{\mathsf{ST}}(P_i)$ controls the size of the cutset (which we can denote by $|\partial P|$) and keeps the district appearances looking reasonable. Whether this geographic clustering might correspond well to *social* understandings of community is an interesting question. When using geographical units that were designed with social structure in mind (like electoral precincts, or block groups in the U.S. Census), this is a promising prospect and is worth serious attention in future work.

1.4. **Recombination.** Often, including large steps instead of only allowing small perturbations can significantly speed up a Markov chain. This is illustrated by various classic card-shuffling examples. Using riffle-type moves that affect the whole deck produces significantly faster mixing than iterating single-card moves—the mixing time drops to the order of $\log n$ instead of $n \log n$ for decks of n cards [4, 25, 23, 8]. Motivated by these examples, DeFord–Duchin–Solomon introduced a new family of large-step Markov chains called *recombination* (or ReCom) for sampling from balanced, connected, compact partitions. The basic step in a recombination chain is to propose two districts to be fused, then select a random spanning tree of the fused double-district, then (if possible) choose an edge from that spanning tree whose deletion leaves two districts with population balance within the prescribed tolerance.[5] Recombination has been applied in a variety of scholarly redistricting studies and reports [20, 19, 11, 21, 12, 27, 9], has been used in 2021-2022 litigation in Pennsylvania, South Carolina, and Texas, and has led to an array of theoretical advances about its properties [2, 41, 13].

There are two central features of recombination moves that provided significant conceptual advances: first, the proposal fuses two adjacent districts and entirely re-partitions them at each step, potentially reassigning hundreds or thousands of units at a time instead of just a few; second, using spanning trees in the re-partition step controls the sizes of cutsets without needing to introduce additional parameters or weights. See [22] for a survey, presenting empirical/heuristic evidence of effective sampling at the scale of real-world redistricting problems, and confirming reasonable compactness of the resulting districts.

While ReCom approximately targets the spanning tree distribution, no closed-form representation of its precise steady state is known when there are more than two districts. When there are only two districts, the stationary distribution of ReCom is simple: for a plan with districts P_1 and P_2 with $|\partial P|$ edges between them, its stationary probability is proportional to $N_{\mathsf{ST}}(P_1) \cdot N_{\mathsf{ST}}(P_2) \cdot |\partial P|$. This is the spanning tree distribution along with the additional term $|\partial P|$. However, this formula does not generalize to more than two districts.

Here, we introduce a simple modification to ReCom that makes it reversible and precisely targets the spanning tree distribution on plans. (Additionally, because transition probabilities are now explicitly calculated at each step, it could be used with a Metropolis-Hastings variant to target any desired distribution.) The new RevReCom algorithm again fuses two districts and re-partitions them with random spanning trees, but adds more rejection conditions based on graph properties. These rejections have the effect of making the chain *lazy*, or more likely to self-loop, but in a controlled manner that ensures detailed balance. Higher rejection rates do in general require more proposals to generate a sample of a given size, but this can be compensated in practice by a "batching" (partial parallelization) method described below in §3.

While our approach is one way to make ReCom reversible, a different approach was taken by Mattingly's team in [6, 7]. There, the state space of the Markov chain is extended: a state is a districting plan together with a spanning tree for each district, i.e., a *spanning forest* for the graph.

[5] We will compare four variants in the recombination family below, in §3.4.

Their Markov chain operates on this extended state space and uses a Metropolis filter to ensure that a desired distribution is targeted. Keeping a record of the spanning tree in each district, and not just storing the plan as a partition, makes the probability calculations in the Metropolis filter computationally feasible. For certain settings of the parameters of their chain ($\gamma = 0$ and $\beta = 0$), the resulting process targets the same spanning tree distribution described here. However, their primary focus is on targeting other distributions. Later, Charikar et al. proposed an alternative method to target the spanning tree distribution in [13], incorporating some additional balance factors.

1.5. **Sampling.** MCMC is frequently used to collect a sample of configurations; as noted above, this ensemble will be used to study proposed plans comparatively. We briefly note that in many applications, researchers will employ parameters b and m to implement *burn-in* and *sub-sampling*: a Markov chain process will skip the first b states before adding a state to the ensemble; subsequently, every mth state visited by the chain will be added. In principle, when b is set to the mixing time of the Markov process and m is set as its relaxation time, this (approximately) produces uncorrelated samples from the stationary distribution. For real applications that lack rigorous bounds on mixing time, some authors have argued in favor of *continuous observation* that records every step encountered by the chain, which is known to give identical statistical estimates in the limit [3, 31]. Continuous observation is the method we will employ here. Further discussion of sampling practices and tests of convergence can be found in the appendix.

2. The reversible recombination chain

2.1. **Exact balance.** We begin by describing reversible recombination, RevReCom, for the simple case where nodes have equal weight and graph partitions require exact balance. We then extend this to the case of imposing a threshold for population deviation in the next section, §2.2.

Let $\mathcal{P}_k G$ be the set of exactly balanced connected k-partitions on a graph G whose nodes have weight one. We let $P_1, \dots, P_k$ denote the k subgraphs induced by the pieces of the vertex partition, which we will call *districts*. Define $\mathsf{sp}(H) = \ln(N_{\mathsf{ST}}(H))$ to be the natural log of the number of spanning trees of H. For $P \in \mathcal{P}_k G$, let

$$\mathsf{sp}(P) := \sum_{i=1}^{k} \mathsf{sp}(P_i) = \ln\left(\prod_{i=1}^{k} N_{\mathsf{ST}}(P_i)\right).$$

We call e a *balance edge* of a tree T if its two complementary components have population balance within a set tolerance—for this equal weight case, this means that the complementary components have the same number of nodes. A tree with a balance edge at tolerance ϵ is called an *ϵ-balanced tree*. For subgraphs A, B of G, let $E(A, B)$ be the set of edges in G with one endpoint in $V(A)$ and one in $V(B)$. Let us also write $N_{\mathsf{ST}}(A, B)$ for the number of spanning trees of the induced graph on $V(A) \cup V(B)$ that have exactly one edge in $E(A, B)$, noting that

$$N_{\mathsf{ST}}(A, B) = N_{\mathsf{ST}}(A) \cdot N_{\mathsf{ST}}(B) \cdot |E(A, B)|.$$

Let π be the spanning tree distribution on $\mathcal{P}_k G$, namely

$$\pi(P) := \frac{e^{\mathsf{sp}(P)}}{Z},$$

where $Z = \sum_{R \in \mathcal{P}_k G} e^{\mathsf{sp}(R)}$ is the *normalizing constant* ensuring π is a distribution. This is the distribution from which RevReCom will sample.

We define a Markov chain proposal as follows. From a plan P, first choose pairs of indices uniformly from $\{1, \dots, k\}^2$ until P_i and P_j are adjacent, then let $A = P_i$ and $B = P_j$. (Equivalently, consider the quotient graph of the plan P made by identifying nodes with the same district assignment, sometimes called the *district-level dual graph*: choose one of its edges uniformly at

random to choose a pair of districts to fuse.) Consider the graph $A \cup B$ induced by G on the vertex set $V(A) \cup V(B)$ and choose a spanning tree uniformly at random. If it is not a balanced tree, reject. If it is a balanced tree, delete the balance edge and let the new components be called A', B' and the corresponding new plan be called Q. Accept Q with probability $1/|E(A', B')|$; else, reject. If any of the rejection conditions was met, we resample a new pair of indices and repeat.

Note the number of spanning trees for $A \cup B$ that could have produced districts A' and B' is exactly $N_{\mathsf{ST}}(A', B')$, as such a tree must have exactly one edge in $E(A', B')$. The process above prescribes the following transition probability $X_P(Q)$ for transitioning from state P to state Q in a single step:

$$X_P(Q) = \frac{2}{k^2} \cdot \frac{1}{N_{\mathsf{ST}}(A \cup B)} \cdot \frac{1}{E(A', B')} \cdot N_{\mathsf{ST}}(A', B') = \frac{2}{k^2} \cdot \frac{N_{\mathsf{ST}}(A')\, N_{\mathsf{ST}}(B')}{N_{\mathsf{ST}}(A \cup B)}.$$

One can verify reversibility with a simple calculation.

$$\begin{aligned} \pi(P) \cdot X_P(Q) &= \frac{\prod_l N_{\mathsf{ST}}(P_l)}{Z} \cdot \frac{2}{k^2} \cdot \frac{N_{\mathsf{ST}}(A')\, N_{\mathsf{ST}}(B')}{N_{\mathsf{ST}}(A \cup B)} \\ &= \frac{2}{k^2 Z} \frac{\prod_{l \neq i,j} N_{\mathsf{ST}}(P_l)}{N_{\mathsf{ST}}(A \cup B)}\, N_{\mathsf{ST}}(A)\, N_{\mathsf{ST}}(B)\, N_{\mathsf{ST}}(A')\, N_{\mathsf{ST}}(B'). \end{aligned}$$

Note that $V(A) \cup V(B) = V(A') \cup V(B')$ and also that $P_l = Q_l$ for $l \neq i, j$. We conclude that $\pi(P) \cdot X_P(Q) = \pi(Q) \cdot X_Q(P)$, which is precisely the detailed balance condition that constitutes the definition of reversibility. This same balance condition also ensures that π is a stationary distribution for the chain; this is because if we start with probabilities distributed by π, the probability of being at any state P after one application of X is given by $\sum_Q \pi(Q) \cdot X_Q(P) = \sum_Q \pi(P) \cdot X_P(Q) = \pi(P)$.

2.2. **Approximate balance.** We generalize the results in the previous subsection to the case where we only aim to approximately balance populations of districts. That is, given weights on the vertices $w : V \to \mathbb{R}$ and a small population deviation tolerance $\epsilon > 0$, a graph partition P is *approximately balanced* if for all districts P_i,

$$(1 - \epsilon)\frac{w(G)}{k} \leq w(P_i) \leq (1 + \epsilon)\frac{w(G)}{k},$$

where $w(G) = \sum_{v \in V(G)} w(v)$.

In this setting, a tree may have multiple edges whose removal separates the vertices into two approximately balanced parts. We modify the algorithm as follows. Let m be any upper bound on the number of edges in a tree that satisfy population balance to within tolerance ϵ, which we again call *balance edges*. Note this upper bound must apply to all possible spanning trees that could be generated in a proposal. Each possible balance edge is chosen with probability $1/m$. If there are b balance edges in a particular spanning tree T, this means there is a probability $1 - \frac{b}{m}$ that no edge will be chosen, and the proposal will be rejected. Recalculating transition probability:

$$X_P(Q) = \frac{2}{k^2} \left(\frac{N_{\mathsf{ST}}(A', B')}{N_{\mathsf{ST}}(A \cup B)} \right) \cdot \frac{1}{m \cdot E(A', B')} = \frac{2}{mk^2} \cdot \frac{N_{\mathsf{ST}}(A')\, N_{\mathsf{ST}}(B')}{N_{\mathsf{ST}}(A \cup B)},$$

and detailed balance follows as before. Because of the rejection probability of $1 - \frac{b}{m}$, it is advantageous to find an upper bound m that is as good as possible. (Certainly one can use $m = n$, but in practice smaller values of m appear to be safe.) However, we caution that unless m is a valid upper bound, this process may not converge to π. Even if the number of balance edges is never observed to exceed m, the resulting limiting distribution is conditioned on never encountering a spanning tree that has more than m balance edges. This could differ from π in subtle ways.

2.3. **Ergodicity.** From the detailed balance expressions above, we know that the spanning tree distribution π is a steady state of RevReCom, which we note has the same state space as ReCom but with different transition probabilities. However, the fundamental theorem that guarantees that this is the unique stationary distribution, and that all other distributions converge to π under iterations of the Markov process, requires the hypothesis that the system is *ergodic*, i.e., aperiodic (the greatest common divisor of all self-looping path lengths is 1) and irreducible (it is possible to transition from any state to any other state in finite forward time).

In all Markov chains used to sample graph partitions on real-world data, including the one proposed here, proofs of irreducibility have remained elusive. This is largely due to population balance constraints: the tighter the population balance must remain, the more likely it is that the valid moves no longer connect the state space. The best known result comes from Akitaya et al., who have shown that the chain is ergodic when $\epsilon = 1$, that is, when districts can shrink arbitrarily small and grow up to double their ideal size [2]. This is because allowed moves can create one large district and several single-vertex districts, and moving through such configurations allows greater flexibility. A forthcoming result of Cannon proves ergodicity under much tighter balance conditions (± 1 from ideal size), but is limited to the special case of a triangular grid-graph.

Even on smaller, more structured examples, ergodicity proofs remain challenging. For example, to divide a 6×6 grid-graph into three equal-sized districts, every known proof of ergodicity for ReCom involves extensive case analysis.[6] For the $7 \times 7 \to 7$ problem, all known proofs of irreducibility use brute-force computation.

We know several examples of tilings of the infinite grid by m-ominoes, for $m = 3, 4, 5$, that are *recombination rigid*, meaning that they are not connected by any perfect-balance recombination transition to any other configuration. Recently Jamie Tucker-Foltz discovered a rigid 3-omino tiling of the 6×6 grid, meaning that the state space for dividing a 6×6 into 12 equal parts has at least two isolated points coming from this example and its reflection. However, despite these carefully-constructed rigid counterexamples, recombination is widely believed to be irreducible on most real-world examples under reasonable values of the population deviation tolerance ϵ.

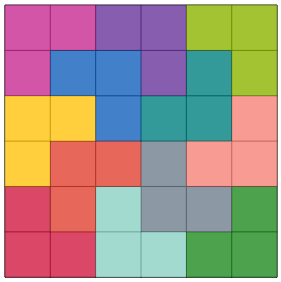

FIGURE 1. A recombination-rigid partition for the $6 \times 6 \to 12$ problem: no two adjacent districts admit a different balanced splitting. This makes this plan and its reflection *isolated points* in the state space of 80,092 exactly balanced plans. However, if districts are allowed to vary ± 1 from ideal size, the number of valid plans rises to over 4.5 billion, and these configurations are no longer isolated.

Without a proof of irreducibility, we only know that a RevReCom chain initialized at a particular state converges to the spanning tree distribution on its connected component of the state space. This provides another reason that verifying similar properties for samples seeded at multiple different starting points—also known as the multistart heuristic, as in Figure 4—can be valuable to raise confidence that our ensemble is representative of a large and diverse set of plans.

[6]Note that recombination moves include flips as a special case, so the state spaces for flip chains are strictly more likely to be reducible.

3. Experimental results

3.1. **Implementation and setup.** The main implementation of RevReCom used in this paper is written in Rust, a programming language increasingly popular in the scientific computing community for its performance and compile-time memory safety guarantees.[7]

With the Rust codebase, we compare RevReCom to original ReCom in three settings: the seven-by-seven square grid divided into 7 districts; the precincts of Virginia divided into 11 districts; and the precincts of Pennsylvania divided into 18 districts. The first is included because it is small enough that all possible districting plans can be enumerated, and the second two come from recent redistricting case studies.[8]

We apply RevReCom as described above, and unless otherwise indicated we impose exact balance on the $7\times 7 \to 7$ problem and $\leq 1\%$ population deviation in Pennsylvania and Virginia. The choice of spanning tree is made with Wilson's algorithm, which is provably uniformly random [44]. For an upper bound m on the number of balance edges, we use $m = 30$ in Virginia and Pennsylvania. These values are both larger than the largest number of balance edges we ever observed during an execution of RevReCom.[9] For the PA and VA runs, the batching was executed with 1024 proposals per batch, split across 8 cores.

[7]Architecturally, RevReCom resembles previous implementations of ReCom, including an experimental implementation of RevReCom in Julia [38, 42]. Novel elements include extensive use of reusable buffers, partly to avoid allocating excess memory during long runs. Due to these enhancements, our Rust implementation outperforms the principal Python library GerryChain by a factor of over 200 for a non-reversible variant of ReCom, processing 1 million proposals in a benchmarking run on Virginia precincts in 1-2 minutes. In the Rust code, RevReCom processes 1 million proposals in a similar amount of time, ultimately accepting about 50 plans per second. But in fact, the memory safety guarantees in Rust make it well suited to multithreading. We take advantage of this feature to mitigate the high rejection rate of RevReCom through a partially parallelized *batching* strategy, which improved the acceptance rate in the benchmarking run to over 300 plans per second. See Appendix D for more details.

We sketch the batching strategy here. The low acceptance rate of RevReCom on real-world graphs (see Table 1) enables us to secure performance gains by using multiple cores. Markov chains are fundamentally hard to parallelize because, by definition, the next step is probabilistically determined by the current state, so a single random walker must consider a proposal at each step. However, if 99% of proposals are being rejected in a given chain, then we can get an efficiency boost by using worker threads at the proposal stage that collectively consider a few hundred proposals simultaneously. They can then be ordered randomly and the accepted proposal with lowest index can be passed back to the main thread, at a potentially significant time savings. This is an instance of what is sometimes called "speculative execution"—work is performed that may not be used, in order to minimize lag time. We have found work by Brockwell from 2006 proposing a similar batching strategy, where he uses the term "pre-fetching" for the partial parallelization [10]. The batch size should be large enough for the advantages to surpass the cost of synchronization overheads, but without leading to many wasted samples. Empirically, we find that a batching strategy is effective for reducing wall-clock compute time on full-scale redistricting problems.

[8]Code for these experiments is publicly available at `github.com/mggg/reversible-recom` and `github.com/pjrule/frcw.rs`.

[9]Recall that the resulting distribution in both these cases could differ slightly from π because it is conditioned on the process never having observed a tree with more than m balance edges. However, observing a large number of balance edges in a single tree is an extremely rare event: even in 2 billion proposals in Pennsylvania, there were at most 23 balance edges (observed 1 time). In Virginia, there were at most 18 balance edges (observed 4 times) in 2 billion proposals. Because the bound m was set well higher than this, the effect on any conclusions reached because of this subtle conditioning should be negligible.

3.2. **Convergence diagnostics using full enumeration.** While π was shown above to be an asymptotic limit for RevReCom, this in itself provides no guarantee of getting a good sample in a reasonable time. A user must seek to raise confidence that enough steps have been run so that the ensemble of districting plans is distributed approximately by π. Since no mixing time upper bounds bounds are known for recombination, a heuristic check of convergence is typically made by computing various statistics over the ensemble and confirming that further enlargement of the sample, or re-start from very different starting points, has a minimal effect on aggregate statistics.

We first look at the seven-by-seven grid, where we have the ability to compare collected statistics to the ground truth by enumerating all districting plans and directly calculating their probability of occurring in stationary distribution π. Figure 2 compares the distribution of cut edges in the ensemble created by 10,000, then 1 million, and then 100 million steps of RevReCom to the distribution of cut edges under π. (The 100 million step sample was collected in under TODO: 5 hours, and the time growth is linear.) By the time we have taken 1 million steps, the histograms are nearly identical, and the 100-million-step sample is visually indistinguishable from the ground truth histogram. Besides giving a sense that 1 million steps was likely enough for most practical purposes, this also supports the use of continuous observation, as we are able to nearly approximate π without any burn-in or subsampling.[10]

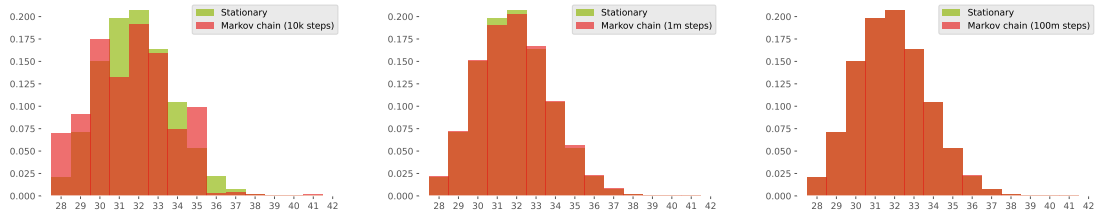

FIGURE 2. For the seven-by-seven grid divided into seven equal-sized districts, the distribution of cut edges in the ensemble of districting plans created after 10,000, 1 million, and 100 million steps (red) compared to the distribution of cut edges in all districting plans, weighted by π (green)

Next, we can assess convergence quantitatively, and not just by visual comparison. The notion of similarity we use is the Wasserstein distance (also known as the earth-mover distance) between distributions, which will be familiar to some readers from the optimal transport setting. In Figure 3, we see that from two different starting points (i.e, seeds), the ensembles produced by RevReCom are similar to the full enumeration after around 500,000 steps, with fluctuating but moderately improving similarity after that. Notably, the time it takes the two ensembles generated by RevReCom to be similar to each other tracks with the time it takes for each to be similar to the full enumeration. This offers some validation for the value of the multi-start heuristic when there is no full enumeration available.

[10]To see how many distinct plans are encountered by the random walk over time, see Supplementary Figure 7.

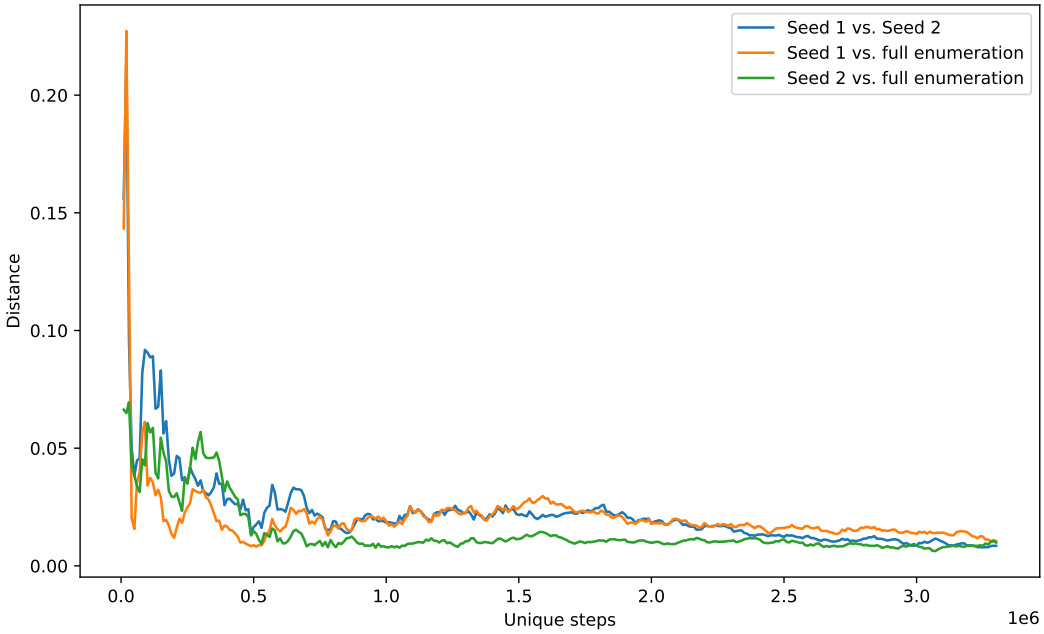

FIGURE 3. Pairwise Wasserstein distance between cut edge distributions. For the $7 \times 7 \to 7$ redistricting problem, the sampling distribution quickly approaches the stationary distribution of RevReCom from extreme seeds (horizontal and vertical stripes). Importantly, the multi-start distance (blue) is confirmed to track close to each chain's distance from stationarity (orange, green).

3.3. **Convergence diagnostics without full enumeration.** In Virginia and Pennsylvania, the number of possible districting plans is so large that there is no hope of enumerating them all to get a ground-truth distribution. However, we can still compare runs across different seeds. The first statistic we look at is of particular relevance for redistricting: the vote share for one party by district. For each districting plan in our ensemble, we order the districts by the Democratic vote share in that district according to election data from a recent election, in this case the 2016 Presidential election.[11] The left-most column shows the range of shares in the least Democratic district in each plan; the next column in the second-least Democratic; and so on.[12] Figure 4 shows the resulting boxplot for Virginia for three different ensembles, each created by 1 million steps of RevReCom from a different starting condition. (Two different maps—the enacted Congressional plans from 2012 and 2016—are used as seeds. We run one chain from the former and two chains, with different random sequences of steps, from the latter.) The similarity in this statistic across the three runs lends support to the idea that 1 million steps is sufficient to produce high-quality partisan statistics by district.

We can also promote the Wasserstein distance between distributions to a vector-valued generalization that is well suited to our setting: for any district-level statistic (such as partisan vote share in this case), we can order the values of a plan from lowest to highest. Then we measure distance coordinatewise and sum over the k coordinates.

[11]Using data from other recent elections gave similar results.

[12]Here and in all boxplots in this paper, the box shows the 25th to 75th percentile, the whiskers range from 1st to 99th percentile, and the median is marked.

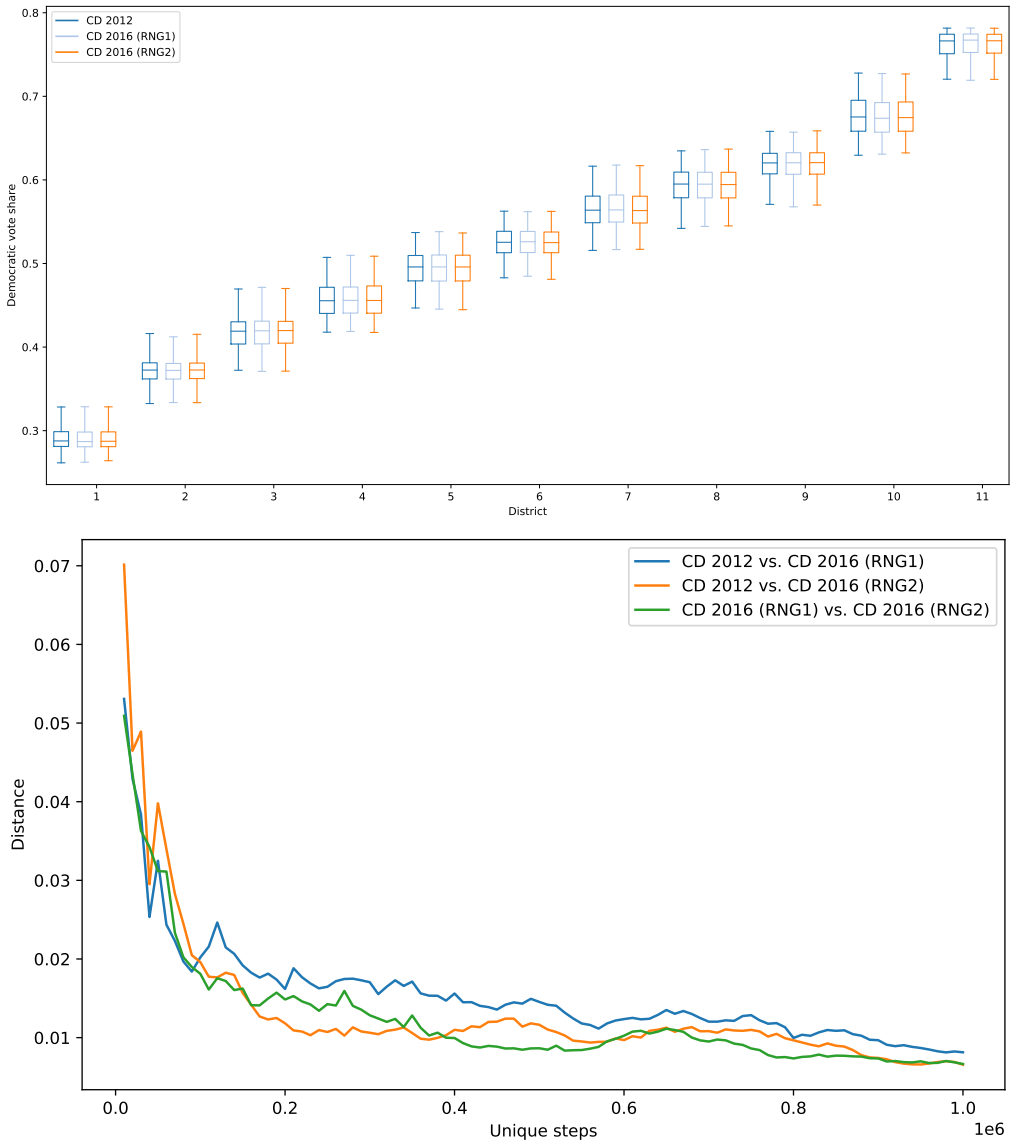

FIGURE 4. Top: the box-and-whiskers plot compares the Democratic vote share by district across three different ensembles of 1 million steps on Virginia Congressional plans. Bottom: The corresponding Wasserstein trace plot.

In this case, the similarity between ensembles is still continuing to improve after a million accepted steps, but the statistics might be regarded as qualitatively stable at this point. Note also that the distance quickly drops to below .02, despite being summed over 11 districts, while Figure 3 shows a single Wasserstein distance on a smaller state space taking longer to drop below that level. This is partly explained by the observation that the cut edge count is a much more sensitive measure of convergence to the spanning tree distribution than a typical partisan statistic would be, because of the close relationship between the two compactness scores. As many authors have noted, convergence in the push-forward distribution to a summary statistic may occur well before representative sampling in the state space.

3.4. **Distributional comparisons.** The use of this RevReCom chain is unlikely to overtake the faster, simpler ReCom chains in practice, because the asymptotic guarantees come at a significant performance cost. However, because we know the stationary distribution of this reversible chain exactly, it has an additional use: very long runs of RevReCom can be used as a tool to help us benchmark and better understand the sampling distributions found by other methods.

3.4.1. *Choice of district pair.* The first step of both ReCom and RevReCom is to pick a random pair of districts to recombine, but there are many possible methods to do so. In standard ReCom, a uniformly random cut edge of the geography dual graph (i.e., a random pair of blocks or precincts that are assigned to different districts) is selected, and the two districts spanned by that cut edge are fused. Instead, we can select districts as in the reversible chain—by choosing a random edge of the district-level dual graph. We can call these *cut edge* and *district pair* selection, respectively. Under cut edge selection, districts sharing a long boundary are more likely to be recombined, which will clearly have the effect of making the sampled plans more compact on average. A priori, the size of this effect is unclear.

3.4.2. *Choice of spanning tree.* Spanning tree methods are clearly very powerful, but they require a choice of spanning tree at every step to repartition the fused districts, and that choice may in turn impact the outputs. We can examine variants of simple recombination using, for instance, the uniform distribution on spanning trees (UST), say via Wilson's algorithm [44], or that introduce random weights and draw a minimum spanning tree (MST) via Prim's or Kruskal's algorithms (see standard reference [18] for a description of these algorithms). All these algorithms run in polynomial time, but the MST algorithms are faster. It may at first seem that if the weights are drawn in an i.i.d. manner, then the MST will also draw uniformly, but elementary calculations show this not to be the case. See, for instance, classic work that calculates expected diameter for a spanning tree of the complete graph K_n. Under UST, the diameter is on the order of $n^{1/2}$, while the MST expectation is $n^{1/3}$ [16, 1]. This suggests that uniform trees may be more path-like, while minimum-weight trees are relatively "bushier," and therefore likely to produce more compact districts.[13] On the other hand, path-like trees are more likely than star-like trees to have a balanced cut, which means that the efficiency gains from MST might be compensated by higher balance-edge rejection.

3.4.3. *Recombination variants.* For the purposes of this paper, these two choices give us four variants of the general recombination chain: ReCom-A (cut edge, MST), ReCom-B (district pair, MST), ReCom-C (cut edge, UST), and ReCom-D (district pair, UST). The standard software implementation uses ReCom-A, and the benchmarking provided here confirms that, for our test cases, its sampling distribution is the most like the spanning tree distribution. This might seem surprising because in its district selection and spanning tree generation, it makes the opposite choices from the reversible chain. As a partial explanation for this, note that the seam-length rejection step in RevReCom means that districts with longer boundaries are less likely to be accepted. ReCom-A imitates this by using cut edge selection (which makes districts with longer boundaries more likely to be fused) and by using minimum spanning trees (which have lower diameter, so also promote compactness).

Many other variants are possible within the recombination family of Markov chains: if we fail to find a balance edge for the first spanning tree drawn for $A \cup B$, we can draw more trees for that pair a variable number of times before selecting a new pair of districts. (See Appendix B.2.) Or we can allow steps that fuse three rather than two districts at a time, among many other means to potentially boost performance by avoiding bottlenecks in the state space, each likely having subtle but non-trivial impacts on the stationary distribution.

[13]Randomly assigning a weight to every edge of $A \cup B$ is, for the purposes of the MST algorithms, the same as choosing random ordering of the edges. Even determining the number of different edge orderings which produce a given spanning tree is a NP-hard problem, equivalent to counting the number of linear extensions of a poset.

3.5. **Rejection and runtime comparison.** We should expect that RevReCom will take longer to converge than ReCom: in order to make the chain reversible, we have introduced more rejection steps (for example, each balance edge is only selected with probability $1/m$ and a new plan is only accepted with probability $1/E(A', B')$). A comparison of rejection rates is shown in Table 1.[14] The table compares the reversible chain to two variants in the ReCom family that employ different ways of choosing a pair of districts to recombine: one picks a random cut edge of the dual graph whose endpoints are in different districts and recombines those districts, while the other picks a random pair of district indices and recombines the districts if they are adjacent. The cut edges method is more widely used (and is the software default in GerryChain), but the district pair method more closely matches the steps in RevReCom.

			how many rejected at each stage			
state	type	attempts	adjacency	balance	seam length	accept
VA	ReCom-A	19,999,999	–	15,490,379 *77.5% rej*	–	4,509,620 22.5%
VA	ReCom-B	59,999,960	41,688,137 *69.5% rej*	12,926,450 *70.6% rej*	–	5,385,373 9.0%
VA	RevReCom	5,000,000,204	3,485,950,207 *69.7% rej*	1,055,448,861 *69.7% rej*	455,177,592 *99.3% rej*	3,423,544 0.07%
PA	ReCom-A	19,999,997	–	14,971,153 *74.9% rej*	–	5,028,844 25.1%
PA	ReCom-B	60,000,005	46,416,385 *77.4% rej*	9,263,366 *68.2% rej*	–	4,320,254 7.2%
PA	RevReCom	4,364,648,325	3,380,978,397 *77.5% rej*	663,754,599 *67.5% rej*	317,847,592 *99.4% rej*	2,067,737 0.05%
7x7	RevReCom	100,000,001	53,408,509 *53.4% rej*	17,304,987 *37.1% rej*	12,765,182 *43.6% rej*	1,652,323 1.7%

TABLE 1. Rejection statistics for ReCom runs on Virginia (precincts, $k = 11$, $\epsilon = .01$), Pennsylvania (VTDs, $k = 18$, $\epsilon = .01$), and a 7×7 grid ($k = 7, \epsilon = 0$).

[14]The irregular numbers of total steps show in Table 1 are a result of a batching technique used to speed up our long runs of these three algorithms.

3.6. **Exploration of state space.** It is common to compare different Markov chains (and other sampling methods) for their runtime or convergence properties, but we can also use visualizations to explore how they transit the state space differently.[15]

We will present a comparison using a toy example: partitions of a graph that is built by suturing together square grids on different scales. The graph is shown in Figure 5: a 2×2, 4×4, 8×8 and 16×16 graph are combined to create a multi-scale grid with 340 vertices. Each quadrant has population 256, so every node in the finest graph has unit population, while the nodes of the 2×2 have population 64. This graph is chosen to exaggerate the disparities in population across nodes that would be viewed in a realistic districting problem.

The heatmaps of Figure 5 show the frequency with which each node is reassigned from one district to another. Comparing heatmaps gives us an indication of why RevReCom requires more steps to converge. The southwest quadrant, which contains many unit-weight nodes, will require many changes before the sample of partitions collected has encountered enough variety to approach stationarity. Compared to all other alternatives—and especially ReCom-A—our new reversible chain is least likely to reassign those nodes.

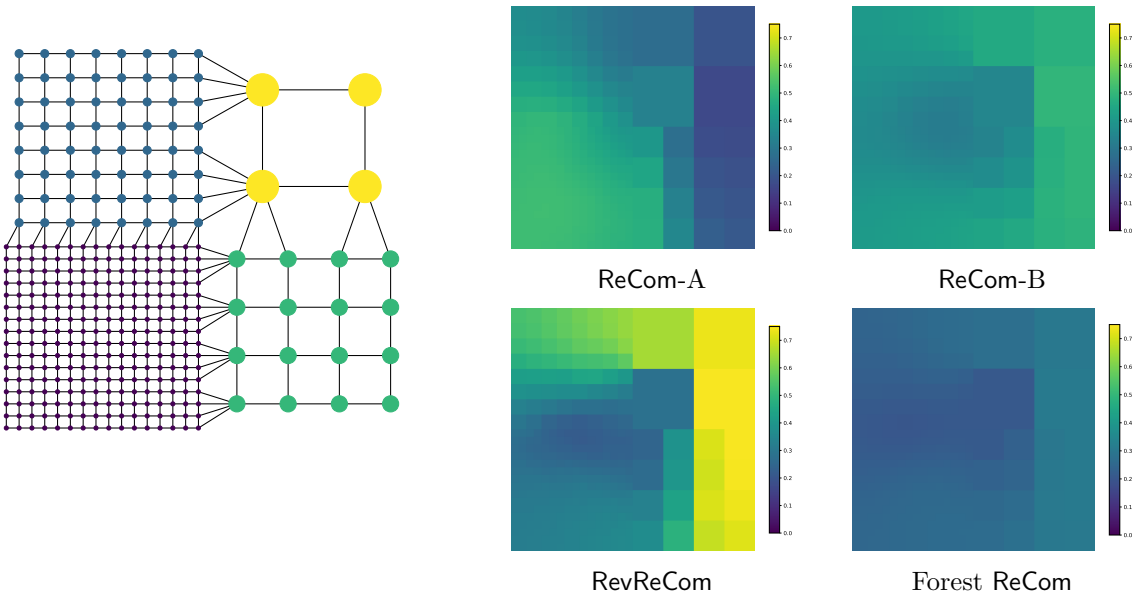

FIGURE 5. These heatmaps provide a visualization of how each chain transits the state space of 4-district partitions of this 340-node graph.

[15]It is easy to see that two Markov chains may have the same stationary distribution, but may transit the state space very differently. For example, consider a state space with n states, and recall that the stationary distribution of the simple random walk on a graph is proportional to vertex degree. One Markov chain that targets the uniform distribution is the simple random walk on the path 1–2–…–n that has self-loops at the endpoints. Connecting the nodes in any k-regular fashion will also yield a simple random walk that converges to the uniform distribution, but with different transitions.

4. Impacts

In practice, ensembles are used to benchmark the neutral consequences of a given redistricting framework. For instance, it is widely believed that early 21st century political geography in the United States creates significant structural partisan advantages—one party, the Republican party, will tend to outperform its rival party, all other things being equal. Reasonable observers can disagree about how to define fair redistricting: is a partisan-blind plan necessarily fair? should policymakers prefer plans that tend to favor representation that is proportional to public preference? and so on.

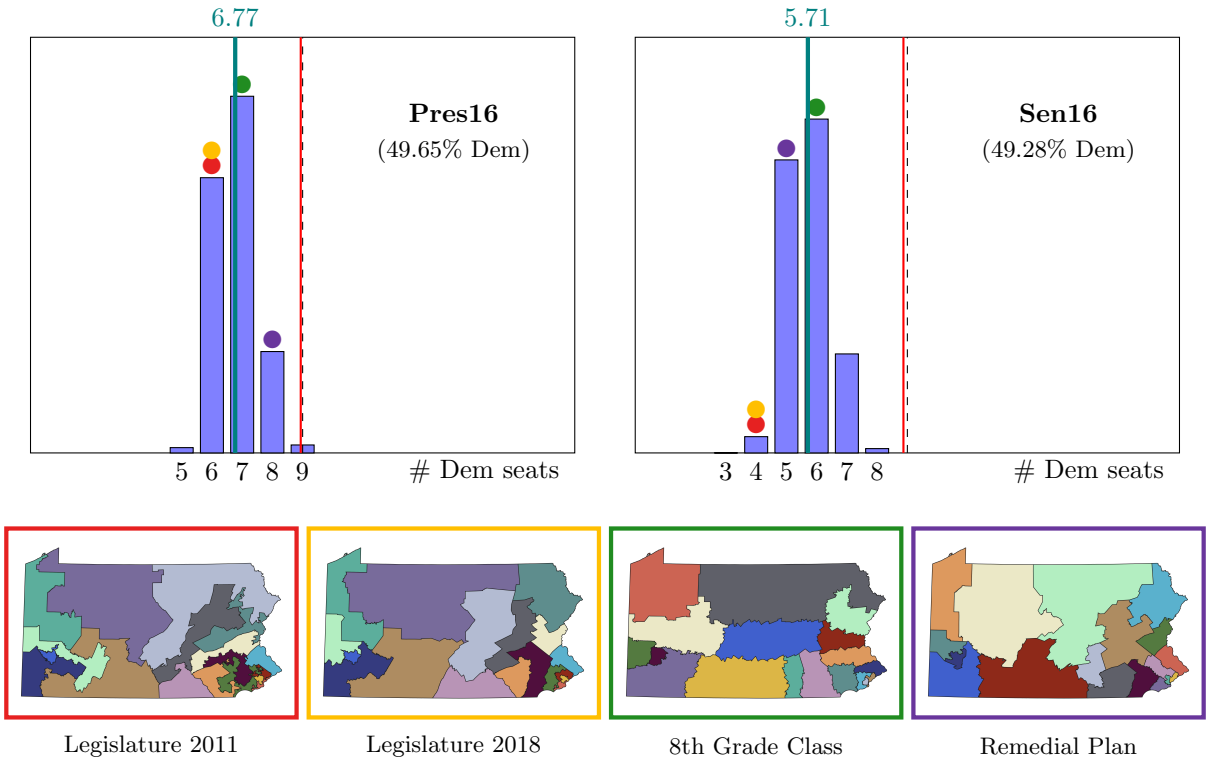

Figure 6. Four proposed 18-district plans for Pennsylvania are compared against the vote distributions in the contests for President and U.S. Senator from 2016. The legislature's enacted plan from 2011, and its much more compact proposal from the lawsuit in 2018, look different but perform similarly against both vote patterns. Both elections are very nearly 50-50, so that a proportional outcome would award nine seats to each party. But the political geography of Pennsylvania—the locations of the votes—creates a situation in which partisan-neutral plans significantly disfavor Democrats. The legislature's plans are even more favorable to Republicans than a blind plan.

The method of ensembles allows us to measure the consequences of a chosen framework, but, importantly, it does not commit us to a view that *neutral is fair*. Rather, it gives us a much-needed means to study the central tendencies of single-member districts. Moreover, because Markov chain methods are easily amenable to heuristic optimization methods (hill-climbing, annealing, short bursts, and so on), we can also produce runs that seek to drive various scores and summary statistics up or down. Local search methods give a view of the "elasticity" of districting outcomes by probing

how effective it is to guide the exploration process. In this way we can start to understand, to the extent that we might have districting goals beyond the simple neutral consequences of the rules, how attainable those goals might be.

Taken together, ensemble methods give us essential tools for the 21st century as we engage in continued debate about our requirements, and our aspirations, for representative democracy.

5. Acknowledgements

The authors are grateful to the developer team behind GerryChain, with special thanks to Daryl DeFord, Max Hully, JN Matthews, Bhushan Suwal, Anthony Pizzimenti, and Max Fan, as well as other members and collaborators of the MGGG Redistricting Lab. The authors are deeply grateful for funding support from multiple sources. SC acknowledges NSF grants DMS-1803325 and CCF-2104795; MD acknowledges NSF DMS-2005512; DR acknowledges NSF grants CCF-1733812 and CCF-2106687.

References

[1] Louigi Addario-Berry, Nicolas Broutin, and Bruce Reed. Critical random graphs and the structure of a minimum spanning tree. *Random Structures and Algorithms*, 35:323–347, 2009.

[2] Hugo A. Akitaya, Matias Korman, Oliver Korten, Diane L. Souvaine, and Csaba D. Tóth. Reconfiguration of connected graph partitions via recombination. *Theoretical Computer Science*, 923:13–26, 2022.

[3] David Aldous. On the Markov chain simulation method for uniform combinatorial distributions and simulated annealing. *Probability in the Engineering and Informational Sciences*, 1:33–46, 1987.

[4] David Aldous and Persi Diaconis. Shuffling cards and stopping times. *The American Mathematical Monthly*, 93(5):333–348, 1986.

[5] David Aldous and James Allen Fill. Reversible Markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/$\sim$aldous/RWG/book.html`.

[6] Eric Autry, Daniel Carter, Gregory Herschlag, Zach Hunter, and Jonathan C. Mattingly. Metropolized forest recombination for Monte Carlo sampling of graph partitions. Submitted. Available at `https://arxiv.org/abs/1911.01503v2`.

[7] Eric A. Autry, Daniel Carter, Gregory Herschlag, Zach Hunter, and Jonathan C. Mattingly. Multi-scale merge-split Markov chain Monte Carlo for redistricting. Submitted. Available at `https://arxiv.org/abs/2008.08054`.

[8] Dave Bayer and Persi Diaconis. Trailing the dovetail shuffle to its lair. *The Annals of Applied Probability*, 2(2):294–313, 1992.

[9] Amariah Becker, Moon Duchin, Dara Gold, and Sam Hirsch. Computational redistricting and the Voting Rights Act. *Election Law Journal: Rules, Politics, and Policy*, 20(4):407–441, 2021.

[10] A. E Brockwell. Parallel Markov chain Monte Carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics*, 15(1):246–261, 2006.

[11] Sophia Caldera, Daryl DeFord, Moon Duchin, Samuel C Gutekunst, and Cara Nix. Mathematics of nested districts: The case of Alaska. *Statistics and Public Policy*, pages 1–22, 2020.

[12] Sarah Cannon, Ari Goldbloom-Helzner, Varun Gupta, JN Matthews, and Bhushan Suwal. Voting rights, Markov chains, and optimization by short bursts. Submitted. Available at `https://arxiv.org/abs/2011.02288`.

[13] Moses Charikar, Paul Liu, Tianyu Liu, and Thuy-Duong Vuong. On the complexity of sampling redistricting plans. Available at `https://arxiv.org/abs/2206.04883`.

[14] Maria Chikina, Alan Frieze, Jonathan Mattingly, and Wesley Pegden. Separating effect from significance in Markov chain tests. *Statistics and Public Policy*, 7, 2020.

[15] 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.

[16] Fan Chung, Paul Horn, and L. Lu. Diameter of random spanning trees in a given graph. *Journal of Graph Theory*, 69(3):223–240, 2012.

[17] Jeanne N. Clelland, Nicholas Bossenbroek, Thomas Heckmaster, Adam Nelson, Peter Rock, and Jade VanAusdall. Compactness statistics for spanning tree recombination. Available at `https://arxiv.org/abs/2103.02699`, 2021.
[18] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, fourth edition, 2022.
[19] Daryl DeFord and Moon Duchin. Redistricting reform in Virginia: Districting criteria in context. *Virginia Policy Review*, XII:120–146, Spring 2019.
[20] Daryl DeFord, Moon Duchin, and Justin Solomon. Comparison of districting plans for the Virginia House of Delegates. White Paper. Available at https://mggg.org/VA-report.pdf, 2018.
[21] Daryl DeFord, Moon Duchin, and Justin Solomon. A computational approach to measuring vote elasticity and competitiveness. *Statistics and Public Policy*, 7(1):69–86, 2020.
[22] Daryl DeFord, Moon Duchin, and Justin Solomon. Recombination: A Family of Markov Chains for Redistricting. *Harvard Data Science Review*, 3(1), 2021.
[23] Persi Diaconis. *Group Representations in Probability and Statistics*, volume 11 of *Institute of Mathematical Statistics Lecture Notes - Monograph Series*. 1988.
[24] Persi Diaconis. The Markov chain Monte Carlo revolution. *Bulletin of the American Mathematical Society*, 46:179–205, 2009.
[25] Persi Diaconis and Mehrdad Shahshahani. Generating a random permutation with random transpositions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57(2):159–179, Jun 1981.
[26] Charles R. Doss, James M. Flegal, Galin L. Jones, and Ronald C. Neath. Markov chain Monte Carlo estimation of quantiles. *Electronic Journal of Statistics*, 8(2):2448–2478, 2014.
[27] Moon Duchin and Douglas M. Spencer. Models, Race, and the Law. *The Yale Law Journal*, 130:744–797, 2021.
[28] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968.
[29] 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.
[30] Alan Frieze and Wesley Pegden. Subexponential mixing for partition chains on grid-like graphs. Submitted. Available at `https://arxiv.org/abs/2206.00579`.
[31] Charles J. Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.
[32] 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):452–479, 2020.
[33] Richard M Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3):429–448, 1989.
[34] J. Kim and T. Wilhelm. What is a complex graph? *Physics A*, 387:2637–2652, 2008.
[35] Leslie Kish. *Survey Sampling*. John Wiley & Sons, 1965.
[36] Eyal Lubetzky and Allan Sly. Cutoff for the Ising model on the lattice. *Inventiones Mathematicae*, 191:719–755, 2013.
[37] Russell Lyons. Asymptotic enumeration of spanning trees. *Combinatorics, Probability, and Computing*, 14(4):491–522, 2005.
[38] Metric Geometry and Gerrymandering Group. Gerrychain. Python Library. `https://github.com/mggg/GerryChain`.
[39] Elle Najt, Daryl DeFord, and Justin Solomon. Complexity and geometry of sampling connected graph partitions. Available at `https://arxiv.org/abs/1908.08881`.
[40] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. Communities in networks. *Notices of the American Mathematical Society*, 56(9):1082–1097, 1164–1166, 2009.
[41] Ariel D. Procaccia and Jamie Tucker-Foltz. Compact redistricting plans have many spanning trees. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2022.
[42] Parker Rule, Matthew Sun, and Bhushan Suwal. mggg/gerrychainjulia: Minor fixes + save as hdf5. March 2021.
[43] Lawrence E. Thomas. Bound on the mass gap for finite volume stochastic Ising models at low temperature. *Communications in Mathematical Physics*, 121:1–11, 1989.
[44] David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC, page 296–303, 1996.

Appendix A. Burn-in, subsampling, and sample size

As noted in Section 1.5, many MCMC sampling methods use parameters b and m to implement *burn-in* and *subsampling*: a Markov chain process will skip the first b states, and then add only every m^{th} state to the ensemble. Instead, we conduct *continuous sampling*: every state visted by the Markov chain is added to the ensemble. Additionally, as is necessary to ensure the desired stationary distribution, we build our ensemble with multiplicity: if we are at state P when a proposal fails, then we count this as a step of the Markov chain and increase the multiplicity of P in our ensemble and in the weight of the statistics that we gather.

For real applications that lack rigorous mixing or relaxation time bounds, some authors have argued in favor of such continuous observation. For example, Aldous proves in [3] that for estimating means, continuous observation for $2kt_{rel}$ steps after a burn-in phase is at least as good as taking k samples, each at time $m = 2t_{rel}$ apart. (Here t_{rel} is the *relaxation time*, which is the inverse of the spectral gap of the transition matrix; it is closely related to the mixing time.) Similarly, Geyer argues in [31] that in practice, if runs are long enough, one long run of the Markov chain with continuous observation suffices for estimating many quantities of interest.

Separately from issues of whether to continuously observe a chain or subsample, there is a large body of literature focusing on how many samples are needed to reliably estimate statistics [28, 33, 26]. When samples are independent, if the underlying distribution has mean μ and variance σ^2, the average of n samples has variance σ^2/n. For statistics that take on values in a bounded range, upper bounds on σ^2 are easy to find; for however small one would like the variance of the sample average to be, it is then straightforward to determine how many samples are necessary. However, when samples are correlated, the average of n random samples may have much higher variance; the *effective sample size* n_{eff} is the value such that the variance of the average of n samples is σ^2/n_{eff}. The value n_{eff} is often a function of n, and depends on the correlation between all the random samples collected. Approximations or bounds for n_{eff} in terms of n can be used to determine how many samples are needed until the variance of the sample average is small enough, where "small enough" may vary depending on the application. When the samples are weighted (for example, when districting plans appear in the ensemble with multiplicity) and computing the effective sample size is too complicated or too computationally expensive, an approximation formula by Kish [35] is widely used:

$$n_{\text{eff}} \approx \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{\sum_{i=1}^{n} w_i^2}$$

However, this approximation can under-estimate the effective sample size when the weights are correlated with the statistic you're trying to estimate, and it can over-estimate the size when there is clustering in the data. For estimating a statistic other than a mean, one can simply define an auxiliary random variable whose mean is the statistic of interest and apply the same analysis to that random variable.

Appendix B. Mitigating high rejection probabilities

RevReCom is slowed down by the high probability of rejection when proposing a new transition. Here we outline some approaches toward improving the rejection rate while ensuring that RevReCom still converges to the correct distribution.

B.1. **Combining rejection steps.** One factor contributing to the slow runtime of RevReCom is that each balance edge in a spanning tree is only picked with probability $1/m$; in most cases there are far fewer than m possible balance edges, and this means there is a high rejection probability. However, by combining this rejection step with the seam length rejection (with probability $1/E(A', B')$), we can use a smaller value of m and thus have a greater probability of choosing a particular balance edge.

Once two districts A and B (along with a spanning tree T of their union) are chosen, proceed as follows. If T has b balance edges, pick one uniformly with probability $1/b$. Then, accept the proposed new plan created by cutting this edge, resulting in districts A' and B', with probability $b/(mE(A', B'))$. In order for this step to be valid, we just need that $b/(mE(A', B')) \leq 1$ is always true. Previously, we needed to guarantee that $b \leq m$, that is, that $b/m \leq 1$.

This does not remove the concern that using a value of $m < n$ conditions the limiting distribution on having never observed $b > m$ at any step of X. Here that condition is just changed: the limiting distribution, which is likely nearly π, is conditioned on having never observed $b/(mE(A', B')) > 1$ at any step of X.

B.2. **Resampling spanning trees.** When drawing a random spanning tree of the union of two districts, there is substantial probability that the tree chosen does not have any balance edges and therefore cannot be split into two new districts. In RevReCom as described above, at this point the attempt would be abandoned, the step would be considered a self-loop, the current plan would be added to the ensemble an additional time, and a new move would be attempted by choosing two new districts to recombine. However, in ReCom, if a spanning tree drawn for a union of two districts has no balance edge, a new spanning tree is drawn for the same two districts, and the attempt to recombine those two districts is only abandoned after some fixed number s spanning trees have been drawn. While s could be set to 1, more frequently s is set to be larger; the default behavior for ReCom is to use $s = 2$.

Note that implementing this modification into the chain—when an spanning tree without any balance edges is chosen, a new spanning tree is drawn on the same double-district region up to a maximum of s attempts, without this counting as a rejection—would not affect the stationary distribution of RevReCom. For a given pair of districts A and B, let $u_{A\cup B}$ be the fraction of spanning trees of $A \cup B$ that do not have a balance edge. Allowing multiple attempts to find a tree with balance edges on the same region, the transition probability of RevReCom becomes

$$\begin{aligned} X_P(Q) &= \frac{2}{k^2}\left(\frac{N_{\mathsf{ST}}(A', B')}{N_{\mathsf{ST}}(A \cup B)E(A', B')} + u_{A\cup B}\frac{N_{\mathsf{ST}}(A', B')}{N_{\mathsf{ST}}(A \cup B)E(A', B')}\right. \\ &\qquad \left. + u_{A\cup B}^2\frac{N_{\mathsf{ST}}(A', B')}{N_{\mathsf{ST}}(A \cup B)E(A', B')} + \ldots + u_{A\cup B}^{s-1}\frac{N_{\mathsf{ST}}(A', B')}{N_{\mathsf{ST}}(A \cup B)E(A', B')}\right) \\ &= \frac{2}{k^2}\cdot\frac{N_{\mathsf{ST}}(A')\,N_{\mathsf{ST}}(B')}{N_{\mathsf{ST}}(A \cup B)}\left(1 + u_{A\cup B} + u_{A\cup B}^2 + \ldots + u_{A\cup B}^{s-1}\right) \end{aligned}$$

Because the extra term added by these tree redrawing steps only depends on $A \cup B$ and not A, A', B, or B', it will cancel in the detailed balance condition and RevReCom will still have the same stationary distribution. While we did not implement this modification for the current study, doing so may result in faster convergence, though it also might change the way RevReCom explores the state space (which would show up in a different reassignment heatmap than the one shown in Figure 5).

B.3. **Picking a random district pair.** A large number of rejections in RevReCom are the result of choosing two random districts to fuse that are not adjacent. Although such rejections are not computationally costly (since checking adjacent is extremely fast), there are simple alternatives to this kind of step counting.

As one option, Euler's formula implies that an n-vertex planar graph has at most $3n - 6$ edges, so when there are k districts there can be at most $3k - 6$ pairs of adjacent districts (that is, at most $3k-6$ edges in the district-level dual graph). Rather than choosing each possible district pair with probability $2/k^2$, one could choose each adjacent pair with probability $1/(3k - 6)$, and reject with the remaining probability. That is, if there are a pairs of adjacent districts, one would pick each with probability $1/(3k-6)$ and reject with probability $1-a/(3k-6)$. This could equivalently be accomplished by picking a pair of adjacent districts uniformly at random and accepting that

pair with probability $a/(3k-6)$. The calculations for detailed balance will remain the same, with the $2/k^2$ term replaced by $1/(3k-6)$, resulting in the same stationary distribution with a lower proposal count.

Instead, one could observe that if the current state has a pairs of adjacent districts and each is being selected with probability $1/(3k-6)$, the number of attempts until an adjacent pair is selected is a geometric random variable with success probability $p = a/(3k-6)$. Thus, one could draw a random variable $r \sim \texttt{Geom}(a/(3k-6))$, add the current plan to the ensemble with multiplicity r, and then proceed to draw a uniformly random pair of adjacent districts.[16] Note that overall this has the same effect on the ensemble as performing each rejection step individually.

These modifications would have minimal impact on the runtime of RevReCom when the number of districts is small, as it is in the examples we consider, but could become useful when k is large.

Appendix C. Additional figures and demonstrations

C.1. **Coverage.** The power of Markov chain methods is that they are capable of producing excellent estimates while visiting only a tiny portion of a state space. However, it is also interesting to consider how many distinct plans are sampled.

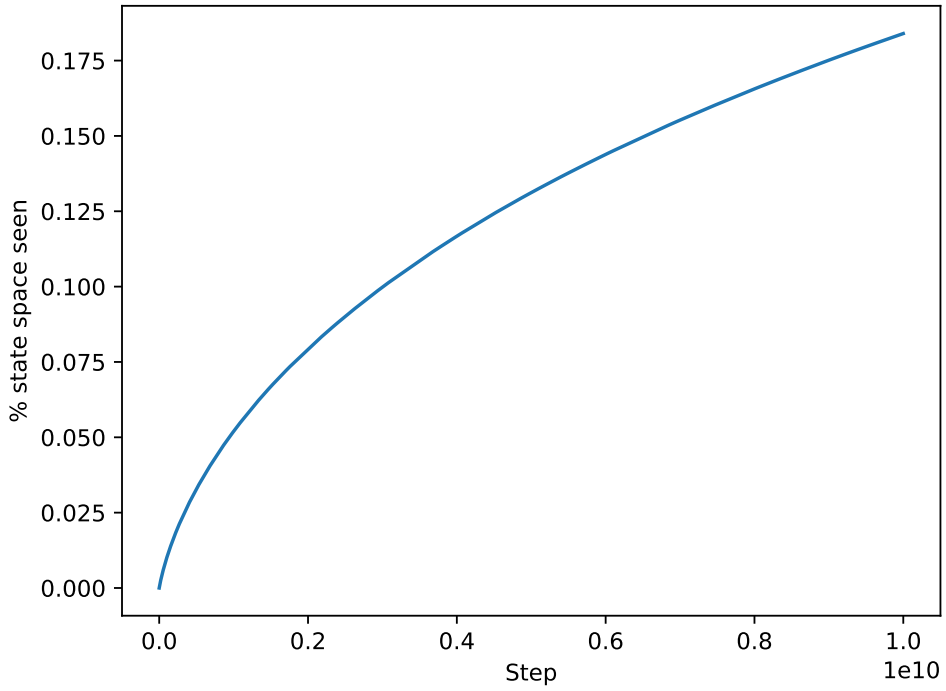

Figure 7. In ten billion proposed steps on the $7 \times 7 \to 7$ state space, this run of RevReCom has visited roughly 18% of states, or about 27 million out of the 150 million distinct plans. Comparing to Figure 2, where we achieved excellent convergence after 100 times fewer steps, reminds us that the power of MCMC is to approximate a sampling distribution long before cover time.

[16] A similar approach to self-looping is taken in [15].

C.2. **Multiscale experiments, continued.** Here we report a second multiscale grid experiment (following Figure 5), which demonstrates even more starkly that the chains move around the state space in remarkably different ways.

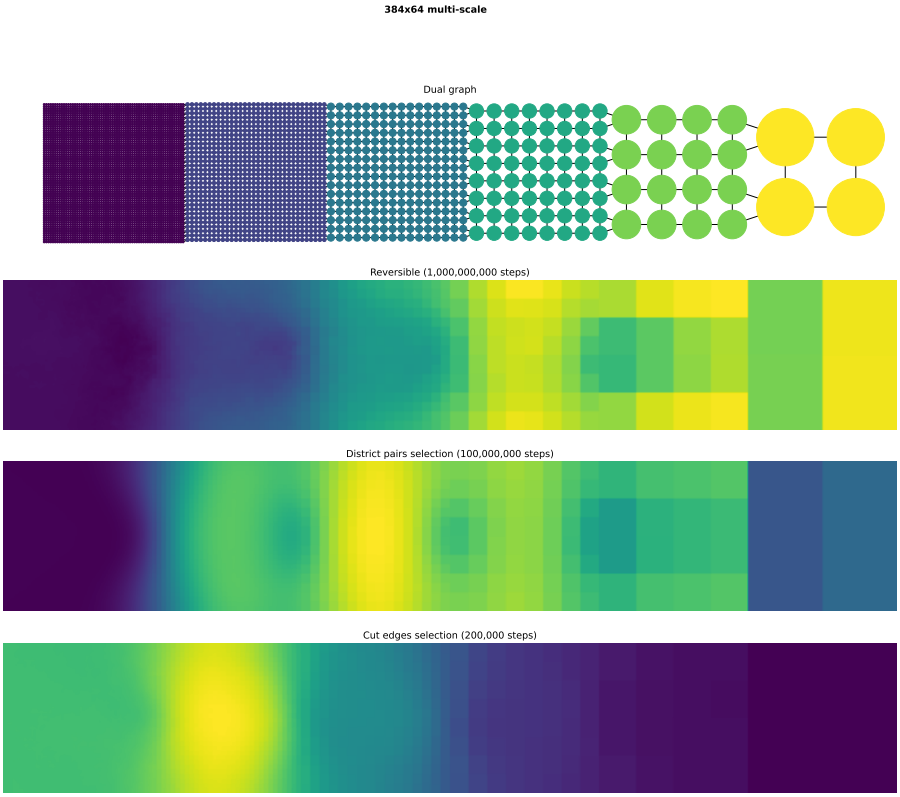

FIGURE 8. Relative frequency of flips across chain variants for 6-district chain runs on a 5460-node graph formed from suturing multiple square grids. Each constituent grid has the same total population, so that the nodes range from unit population at the far left to population 1024 at the far right.

C.3. **Recombination variants, continued.** Next, we present further evidence regarding the differences between variants of the recombination chain.

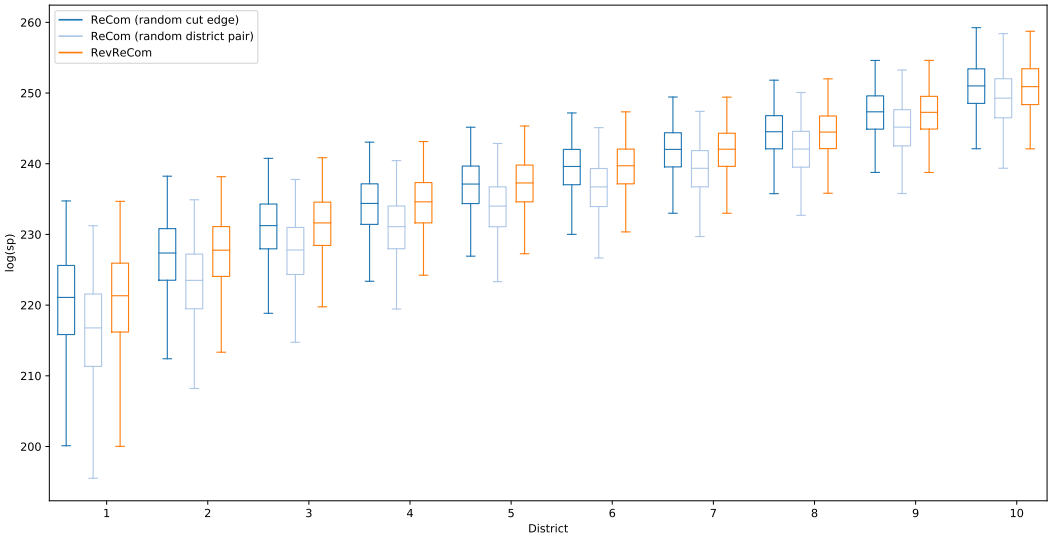

FIGURE 9. Distribution of district spanning tree counts on a 50×50 grid with uniform population.

We see that ReCom-A resembles RevReCom to a degree that is nearly indistinguishable to the eye, while ReCom-B is producing districts that are consistently less compact. Because partisan breakdown is often not very sensitive to the shapes of districts, it is entirely likely that a plot of partisan summary statistics would look identical.

APPENDIX D. BENCHMARKING DETAILS

This brief appendix contains benchmarking stats from the Python and Rust implementations of various simple recombination variants. The intent is to facilitate comparisons between implementations so that other authors can make apples-to-apples speed comparisons; this means that the collection of variants does not need to be comprehensive. For all benchmarking runs, we use the Virginia precinct dual graph (`github.com/mggg-states`) with the CD12 seed (Congressional districts from 2012), $\epsilon = .05$, and minimal updaters. We set $m = 30$ for RevReCom runs. The runs were conducted on an Apple M1 Pro laptop (10 cores) with 16 GB RAM.

GerryChain (main branch, Python 3.10.2): accepted 1000 plans in 51.56 seconds (command: `python gerrychain_benchmark.py`) → 19.4 accepted plans / second

Rust ReCom-A (`cut-edges-rmst`), 1 core: 1,000,000 proposals → 485,987 accepted plans in 107.79 seconds → 4,509 accepted plans / second

Rust ReCom-A (`cut-edges-rmst`), 4 cores: 1,000,000 proposals → 485,983 accepted plans in 71.08 seconds → 6,837 accepted plans / second

Rust ReCom-A (`cut-edges-rmst`): 1,000,000 proposals → 487,239 accepted plans in 89.68 seconds → 5,443 accepted plans / second

Rust RevReCom, 1 core: 1,000,000 proposals → 2,136 accepted plans in 43.791 seconds → 48.8 accepted plans / second

Rust RevReCom, 4 cores, batch size of 1: 1,000,000 proposals → 2,033 accepted plans in 25.537 seconds → 79.6 accepted plans / second

Rust RevReCom, 4 cores, batch size of 32: 1,000,000 proposals → 2,105 accepted plans in 9.331 seconds → 225.6 accepted plans / second

Rust RevReCom, 10 cores, batch size of 12: 1,000,000 proposals → 2,022 accepted plans in 5.928 seconds → 341.1 accepted plans / second

Rust RevReCom, 10 cores, batch size of 16: 1,000,000 proposals → 1,715 accepted plans in 5.401 seconds → 317.5 accepted plans / second