SIAM J. APPL. MATH.
Vol. 83, No. 4, pp. 1366–1391

METROPOLIZED FOREST RECOMBINATION FOR MONTE CARLO SAMPLING OF GRAPH PARTITIONS*

ERIC AUTRY†, DANIEL CARTER‡, GREGORY J. HERSCHLAG†, ZACH HUNTER§, AND JONATHAN C. MATTINGLY¶

Abstract. We develop a new Markov chain on graph partitions that makes relatively global moves yet is computationally feasible to be used as the proposal in the Metropolis–Hastings method. Our resulting algorithm is able to sample from a specified measure on partitions or spanning forests. Being able to sample from a specified measure is a requirement of what we consider as the gold standard in quantifying the extent to which a particular map is a gerrymander. Our proposal chain modifies the recently developed method called recombination (ReCom), which draws spanning trees on joined partitions and then randomly cuts them to repartition. We improve the computational efficiency by augmenting the statespace from partitions to spanning forests. The extra information accelerates the computation of the forward and backward proposal probabilities which are required for the Metropolis–Hastings algorithm. We demonstrate this method by sampling redistricting plans on several measures of interest and find promising convergence results on several key observables of interest. We also explore some limitations in the measures that are efficient to sample from and investigate the feasibility of using parallel tempering to extend this space of measures.

Key words. Markov chain Monte Carlo, balanced graph partitioning, Metropolis-Hastings, spanning trees

MSC codes. 65C05, 91D10, 91D20, 91F10

DOI. 10.1137/21M1418010

1. Introduction.

Graph partition problems have become important in a variety of fields including clustering and detection of cliques in social networks (e.g., [18]), pathological and biological networks (e.g., [24]), infrastructural networks such as roads and airport controls (e.g., [23, 30]), image decomposition (e.g., [9]), and problems in redistricting (e.g., [13, 21, 15]) (for a review, see [8]). Typically, graph partition problems involve optimizing an objective function, such as minimizing the sum of

*Received by the editors May 6, 2021; accepted for publication (in revised form) January 12, 2023; published electronically July 5, 2023.

https://doi.org/10.1137/21M1418010

Funding: This work was funded by the NSF grant DMS-1613337. This work was initiated through the Discovering Research Mathematics Program between Duke Mathematics and the NC School of Science and Mathematics. The third and fifth authors thank the NSF grant DMS-1613337 for partial support of this work. The authors thank both SAMSI (DMS-1638521), TRIPODS (CFF-1934964), and the Rhodes Information Initiative for hosting workshops that were important in the development and dissemination of this work. The authors also thank the Rhodes Information Initiative and the Duke Provost and Deans office for financial support and creating a productive working environment.

†Department of Mathematics, Duke University, Durham, NC 27708 USA (autryeri@grinnell.edu, gjh@math.duke.edu).

‡North Carolina School of Science and Mathematics, Durham NC, 27707 USA, Currently Princeton University, Princeton, NJ 08554 USA.

§North Carolina School of Science and Mathematics, Durham NC, Currently Oxford University, Oxford OX1 4BH, UK (zwishking@gmail.com).

¶Department of Mathematics and Department of Statistical Science, Duke University, Durham, NC 27708-0320 USA (jonm@math.duke.edu).

edge weights spanning two partitions [8] or minimizing communication loads across partitions [20, 4].[1]

There is, however, a growing interest in inquiring into the properties of a 'typical' graph partition given certain partitioning criteria, particularly in spatial clustering and Bayesian statistics (e.g., [1, 5]). One begins by placing a probability measure on the space of graph partitions and then investigates partition properties under this distribution. Alternatively, one could place a specific distribution on spanning forests (a collection of disjoint spanning trees). We are interested in algorithms to sample in both settings. In practice, samples from the measure are drawn with techniques such as Monte Carlo algorithms.

Commonly, these methods invoke Markov Chain Monte Carlo (MCMC) methods in which one performs a random walk on the statespace, in our setting the collection of partitions or spanning forests. That is, one begins with some state (a partition or spanning forest) and then moves to one of its neighbors according to some random procedure. A simple formulation of such a walk, in the context of partitions, consists of choosing a single vertex on the boundary of a partition and proposing to exchange it to a neighboring partition [26, 28, 6, 33, 27, 13, 12]. Using ideas drawn from sampling spin glass models [32], one group has examined the possibility of using percolation clusters to accelerate vertex exchange [17].

One common constraint is to demand balanced partitions or forests, which is to say that the sums of node weights roughly match across partitions or trees. Although the boundary exchange methods mentioned above are provably ergodic under soft constraints, the chains can have difficulties mixing due to large energetic barriers (see, e.g., [31]; in addition, these methods can still prove useful in some practical settings [27]). As a potential remedy to these issues, a research group proposed a recombination (ReCom) algorithm which joins adjacent partitions or trees, draws a spanning tree on the joined partitions, and generates a new partition by cutting the new spanning tree [16].

The ReCom method has demonstrated positive mixing properties when sampling balance partitions or spanning forests on planar graphs. However, it samples from an unknown invariant distribution on the space of partitions.[2] Additionally, even if its sampling distribution was known, it cannot be modified, in its present form, to sample from a specified distribution as many applications require.

The ReCom algorithm has been shown to sample partitions from a measure that is 'close' to sampling a space of balanced partitions weighted according to the product of the number of spanning trees that can be drawn within each partition. However, it is unclear how to quantify this relationship generically. Additionally, it is computationally infeasible to use the original ReCom scheme as a proposal in a

[1]This article was originally released in October 2019 with the name "A Merge-Split Proposal for Reversible Monte Carlo Markov Chain Sampling of Redistricting Plans" to emphasize the imagery of merging and then splitting partitions and the initial motivation of generating redistricting plans. We changed the name to better indicate the connection to the ReCom algorithm and the feasibility of our algorithm to mesh with the Metropolis–Hastings algorithm and produce a sample from a specified target distribution. We also wanted to emphasis the broader application to graph partitioning as our appreciation for these directions has grown.

[2]It is worth mentioning that there are a series of other alternative algorithms to ReCom that algorithmically generate random graph partitions if one is not concerned with knowing the sampling measure or is not interested in having the flexibility of specifying the sampled measure. For example, the constructive algorithms of Chen and Rodden [11] randomly merge adjacent nodes and the genetic algorithms of Cho [25] involve merging two partitions to create a new partition. These procedures also do not have a known invariant measure nor are practical candidates for Metropolizing.

Metropolis–Hastings scheme which requires the calculation of both forward and backward probabilities. Hence ReCom cannot be effectively used in a Metropolizion scheme to sample from specified measures of interest. This limits its usefulness in many applications which require an MCMC scheme which sample from a specified measure of interest. This note shows how to extend the ideas in ReCom to make Metropolizing feasible.

In the current work, we use a modified ReCom Markov chain as a proposal in order to develop a reversible MCMC algorithm to sample a specified distribution on the space of balanced graph partitions. The global moves of ReCom promise faster mixing MCMC algorithms when used as the proposal chain, and the added (reversible) rejection step allows us to specify and alter the underlying measure being sampled. We alter the ReCom algorithm by extending graph partitions to track persistent spanning trees within each partition. We demonstrate that this expansion of the state space allows us to construct a reversible Markov chain that is able to completely propose redrawn pairs of adjacent partitions.

Although generic, a predominant application of sampling graph partitions is found in applications to redistricting, which is the case that we demonstrate our algorithm. In this problem, we consider a geographical region which is to be split into several districts (i.e., graph partitions). The region typically is comprised of a collection of small scale unit that will form the nodes of graph, and edges between nodes are prespecified, often via geographic adjacency.

We emphasis that we propose the algorithm in this paper precisely because ReCom is not feasible as a sampling method from a general distribution as typically desirable in the restricting context. While one can subsample to impose particular constraints or add weights to shift the ReCom algorithms distribution, the sampled distributions is still unknown under ReCom. From our earliest work, we have emphasized the need to sampled from a known, controllable distribution in the redistricting context. The algorithm presented here builds on the ReCom algorithm to fashion an algorithm which is computationally feasible to be used as the inner-loop in a Metropolis–Hastings sampling scheme. This means that the combined algorithm is able to sample from a specified measure on partitions or spanning forests. It is worth noting that sampling from a specified measure on spanning forests and the related measure on partitions may have dramatically different runtimes and algorithmic character. In either setting, the Metropolis algorithm we present here will have a specified invariant measure.

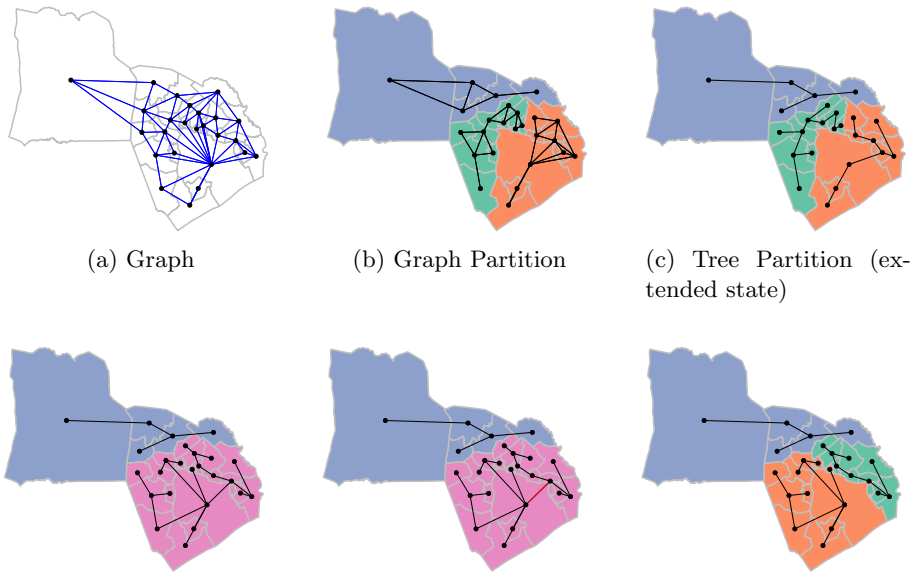
In general, graph partition problems, either in the context of optimization or sampling, are NP-hard or even NP-complete (e.g., see [8] and [31], respectively). However, it is not clear whether NP-completness is the best yardstick to measure the difficulty of this problem. First, it is not clear if the worst case analysis is most appropriate given that the typical graphs considered do not resemble the worst case examples. Second, many of these results are for uniform measures or with population balance across the partitions. Adding a Gibbs-like weighting term, as is our primary interest, can dramatically change the shape of the effective space to be sampled, thus the runtime of the algorithm. Last, many sampling problems are computationally intractable if full sampling distribution is the goal. In sampling, we are often not interested in recovering the full distribution on partitions, but rather several observables of interest. This is analogous to sampling on molecular dynamic problems, where instead of recovering every possible particle arrangement with the appropriate probability, one focuses on recovering observables of interest such as chemical potentials. Additionally, many useful and scientifically important algorithms are known to be NP-hard in the mot

general setting but are still quite successful in practice. Examples include the simplex algorithms or branch-and-bound optimization algorithms as well as many sampling algorithms. All perform poorly in a worst-case problem instance, but perform quite well in many applications. With this in mind, we present numerical evidence that the complexity of recovering robust statistics on low-dimensional observables of interest is significantly faster than attempting to recover the correct distribution on all partitions via sampling.

The results of this paper were generalized in a multiscale setting [3]. That article builds on this and provides a framework for preserving structure in the graphs, for example, counties in the redistricting context.

2. Informal overview of the Metropolized Forest Recombination algorithm. Although our algorithm can sample from a general specified measure on graph partitions or spanning forests, we illustrate it in the context of redistricting. In the redistricting context typically, a region is partitioned by assigning small atomic geographic elements to a district or partition (see Figures 1a and 1b). At the highest level, the core of the algorithm is similar to ReCom [16]. We pick two adjacent partitions, merge them into a single subgraph, and randomly redivide the merged subgraph into two new balanced partitions. This process is then repeated.

In ReCom, the new partition is constructed by first generating a random spanning tree on the merged subgraph (Figure 1d). This spanning tree can then be used to efficiently divide the merged subgraph into two balance partitions. In the redistricting problem, for example, balance would mean that each partition contains a population



(a) Graph (b) Graph Partition (c) Tree Partition (extended state)

(d) Merge & Sample New Tree on Merged Subgraph (e) Find Edge to Cut (f) Split into Two Trees

FIG. 1. *As an illustrative example, we consider graph induced geometric regions* (a). *We then show how a partition of the graph induces subgraphs* (b), *how we extend the state to consider trees on the subgraphs* (c), *how a merge step might look* (d), *how edges are removed, and how a subsequent split step might look* (e, f).

that is equal up to some tolerance (Figure 1e). One such partition is chosen randomly from those which are possible by cutting the given spanning tree in two (Figure 1f).

We are interested in reimagining ReCom so that it is practical as the random proposal in a Metropolis–Hastings MCMC algorithm. To employ Metropolis–Hastings, we need to be able to calculate the probability of proposing a particular move from an initial state and also the probability of proposing the reverse move from the proposed state back to the initial state. In ReCom, these calculations are impractical without further insight.

To make the calculations feasible, we enlarge the state-space from graph partitions to spanning forests (Figure 1c). The extra information contained in the spanning forest representation greatly simplifies the calculation for the forward and reverse proposal probabilities (see section 4.3).[3]

3. The setting and target measure. We now lay the groundwork we need to define the algorithm sketched in the previous section more formally. We base our notation on that found in [7, 28] then expand upon it below. Let the graph G have vertices V and edges E. For example, each vertex may represent some subgeographic region in a redistricting; in this context, edges are placed between vertices that are either rook, queen, or legally adjacent.[4] Furthermore, in this context, we will be working with (mostly)[5] planar graphs; however, all of the ideas we will discuss are applicable to generic graphs.

We represent an n-partition on G as a function $\xi : V \to \{1, 2 \dots n\}$. Informally, $\xi(v) = i$ means v is in the ith partition. Given a partition ξ, we will denote by $V_i(\xi) = \{v \in V \mid \xi(v) = i\}$ and $E_i(\xi) = \{(v, u) \in E \mid \xi(v) = \xi(u) = i\}$, respectively, the set of vertices in the ith partition and the set of edges between vertices in the ith partition. We will define $\xi_i = (V_i(\xi), E_i(\xi))$ to be the subgraph induced by the ith partition.

We will also sometimes associate extra data with the vertices and edges. In the redistricting problem, this can include population, land area, and border length between vertices. The additional data is used to evaluate the districts on desired redistricting criteria, such as equal-population (i.e., balance between partitions) and geographical compactness. Of particular note, we define $\mathrm{pop}(v)$ to be the population of vertex v and

$$\mathrm{pop}(\xi_i) = \sum_{v \in V_i(\xi)} \mathrm{pop}(v) \tag{3.1}$$

to be the population of district ξ_i. More generically, $\mathrm{pop}(\xi_i)$ may be thought of a weighted node sum on a generic partition.

In many settings, partitions must be simply connected and hence the state space is a subset of the set of n-partitions of the vertex set of a graph, where n is the number of partitions. Using ξ_i as above to represent the subgraph associated to the ith partition, we can recast ξ as a map on vertices as the n subgraphs defining the partitions. That is,

[3]The methods in this work are not the only ones to make these calculations feasible; in particular, there is work in progress that adds self-loops in the Markov Chain of the ReCom scheme which allows one to determine the forward and backward proposal probabilities [10].

[4]Rook adjacency means that the geographical boundary between two regions has nonzero length; queen adjacency means that the boundaries touch, but may do so at a point. At times two regions may not be geographically adjacent, but may be considered adjacent for legal purposes; for example, an island may still be considered adjacent to regions on a mainland for the purposes of making districts.

[5]At times, certain regions that represent a node may not be connected. If a node represents such a region, it is possible for the graph to be nonplanar.

$$\xi = \{\xi_1, \xi_2, \ldots, \xi_n\}.$$

Since (up to the $n!$ equivalent labelings) there is a one-to-one correspondence between labeling functions ξ and partitions into n subgraphs $\{\xi_1, \xi_2, \ldots, \xi_n\}$, we will move between the two perspectives as is convenient and consider ξ to be both the labeling function and the partition.

We will see that in the context of Metropolizing our Forest ReCom proposal procedure it will be important to work on a space with more structure than the space of n-partitions. We choose to make our state space the set of *n-tree partitions* of a graph; this is the space of forests consisting of n disjoint trees whose union spans the vertices of the graph. We will use the term *spanning forest* interchangeably for such a collection of disjoint trees which span the graph. From this perspective the state space has elements of the form

$$T = \{T_1, T_2, \ldots, T_n\},$$

where each T_i is a spanning tree on the subgraph ξ_i with vertices $v_i = V_i(\xi)$ and edges $\varepsilon_i \subseteq E_i(\xi)$. The use of n-tree partitions of a graph rather than n-partitions only enlarges the state space. Thus, any distribution on the second can be represented on the first. However, we will see in section 4 that this additional richness will allow us to build a fast and feasible algorithm for calculating proposal probabilities. This extension is illustrated between Figure 1b and 1c. Henceforth, we will consider our state to be a collection disjoint spanning trees $\{T_i\}$ rather than a collection of disjoint graphs $\{\xi_i\}$. The one-to-one correspondence between partitions and labeling functions no longer holds, but since $\{T_i\}$ naturally induces $\{\xi_i\}$ (but not the converse), we will still consider the labeling function ξ corresponding to $T = \{T_i\}$ and denote it by $\xi(T)$ or just ξ if context makes the intent clear.

3.1. The target measure on spanning forests. We will now place the probability measure on this space spanning forest consisting of n disjoint trees $T = \{T_1, T_2, \ldots, T_n\}$. We take our measure to be of the form

$$P(T) \propto e^{-\beta J(\xi(T))} \tau(\xi(T))^{-\gamma}, \tag{3.2}$$

where J is a score function that evaluates how "good" a graph partition ξ is,[6] $\beta \in [0, 1]$ and $\gamma \in [0, 1]$ are tempering parameters used to change the importance of the factors $J(\xi)$ and $\tau(\xi)$, respectively, and

$$\tau(\xi) = \prod_{i=1}^{n} \tau(\xi_i), \tag{3.3}$$

where $\tau(\xi_i)$ is the total number of spanning trees on the graph ξ_i.[7] Since once the partitioning ξ is fixed one can choose the spanning tree for each subgraph independently, $\tau(\xi)$ counts the total number of spanning forests on ξ when viewed as a collection of disjoint graphs with the requirement that each partitioned subgraph is covered by

[6] Lower scores are "better" in the sense that a partition in question performs better when considering criteria included in the definition of J. Examples of criteria often incorporated in J in the context of redistricting are the compactness of the districts, the deviation from the ideal population of each district, and the degree to which administrative units (such as counties or cites) are fragmented.

[7] See the supplemental material (mergeSplit_supplement.pdf [local/web 486KB]) for an efficient way to compute $\tau(G)$ for any graph G.

a single spanning tree. In other words, $\tau(\xi)$ is the number of different states in our enlarged state space which correspond to the same partitions ξ.

The score function J encodes a preference for maps with lower scores. It also encodes absolute constraints, which is to say maps that are strictly not allowed in the ensemble, by setting $J(\xi) = \infty$ on those maps. For example, we may constrain the space if the partitions are not sufficiently balanced. It is worth noting that the structure of the spanning forest T does not explicitly enter the measure, as the measure only depends on the underlying partitioning, ξ. However, as already mentioned, we will see that considering our space to be T rather than ξ will be important in computing forward and reverse probabilities under Metropolis–Hastings.

Selecting different measures and score functions may have significant impact on our conclusions. For example, in [22], we sample the same redistricting graph on a measure that ignores municipal preservation and on another measure that prefers the preservation of municipalities, and we find that observables of interest may be nearly disjoint in distribution.

3.2. The structure of the measure. We now collect a number of observations about the structure of the measure P and various limiting cases in γ and β. We will write $P(T; \beta = b, \gamma = g)$ for the probability of seeing the spanning forest T in the distribution in (3.2) when $\beta = b$ and $\gamma = g$.

Uniform measure on spanning forests. When $\gamma = 0$ and $\beta \to 0$, $P(T)$ converges to the uniform measure on the spanning forest of n trees which satisfy the constraints described by the score function J; that is to say $J(\xi) < \infty$. If we were to use the convention that $0 \times \infty = 0$ in the exponent, when $\gamma = \beta = 0$ the measure becomes

$$P(T; \beta = 0, \gamma = 0) \propto 1, \tag{3.4}$$

which is to say we recover the uniform measure on the spanning forests, subject to no constraints.

Uniform on all graph partitions. When $\gamma = 1$, the distribution on graph partitions depends only on the factor involving J; that is, the probability of finding partition ξ no longer depends on $\tau(\xi)$. To see this, note that

$$P(\xi) \propto \sum_{T \in ST(\xi)} P(T) = e^{-\beta J(\xi)} \frac{\tau(\xi)}{\tau(\xi)^{\gamma}}, \tag{3.5}$$

where

$$ST(\xi) = ST(\xi_1) \times \cdots \times ST(\xi_n) \tag{3.6}$$

is the Cartesian product of all spanning trees, $ST(\xi_i)$, of subgraph ξ_i.

When $\gamma = 1$ and $\beta \to 0$, the measure becomes uniform on graph partitions subject to the absolute constraints given by the score function J. When $\gamma = 1$ and $\beta = 0$ (as before, using the convention that $0 \times \infty = 0$), the measure is uniform on all graph partitions.

Intermediate values of γ. As can be seen in (3.5), as γ becomes smaller, we favor partitions that have a larger product of tree counts on the subgraphs. In particular, when $\gamma = 0$ the chance of finding a partition, ξ, is proportional to the product of the number of spanning trees on the subgraphs in ξ.

For moderately sized graphs with a few hundred or thousand vertices, the number of spanning trees is extremely large. In fact, this number grows faster than exponentially with the number of vertices in the graph, assuming the graph has average degree larger than 2 [19]. This rapid growth may cause large disparities between the relative probabilities of different partitions, ξ, as this ratio will be proportional to the product of spanning tree ratios

$$\frac{P(\xi)}{P(\xi')} \propto \frac{\tau(\xi)}{\tau(\xi')} \frac{\tau(\xi')^{\gamma}}{\tau(\xi)^{\gamma}} e^{-\beta[J(\xi)-J(\xi')]}. \tag{3.7}$$

When taking a random walk through the state space using the Metropolis–Hastings algorithm, proposed states will usually have either far fewer or far more trees than the prior state, and the acceptance probability will be dominated by this ratio. This issue may be alleviated by tempering on γ in the interval $[0, 1]$. In a sequential or parallel tempering framework, a chain will spend some time with high acceptance rates at $\gamma = 0$, and some time sampling from the partition space at $\gamma = 1$. We discuss the plausibility of tempering below in section 7.

Induced measure on partitions. It is instructive to pause and consider the relative structure of the measures on spanning forests and partitions. The following lemma shows that all spanning forests which correspond to a given partition are equally likely to be sampled. In other words, the measure conditioned on a given partition is uniform on the spanning trees which correspond to that partition.

LEMMA 3.1. *If two spanning forests T and T' represent the same partition, then their corresponding states have equal probability under the measure P. Given our notation, we may write that if $\xi(T) = \xi(T')$, then $P(T) = P(T')$.*

Proof. This follows from the fact that for a given spanning forest T, both the score function J and the number of spanning trees τ only depend on the partition $\xi(T)$. Since these are the only occurrences of T in the definition of P, the result follows. ∎

4. Sampling from the measure P. As already discussed, we will use a global merge and split algorithm to propose moves to the standard Metropolis–Hastings algorithm. Our Forest ReCom algorithm is not itself reversible, but the resulting Markov chain given by the Metropolis–Hastings algorithm, with Forest ReCom as a proposal, will be. Additionally, by Metropolizing, we are able to sample from a wide range of target measures. Although one can always, in theory, use Metropolis–Hastings to create a reversible chain from any proposal method able to sample from a specified target measure, Metropolis–Hastings will fail in practice if the rejection probabilities are too large or if calculating the necessary transition probabilities is computationally infeasible. Previously, a similar algorithm described in [16] failed to create reversible chains nor could it easily calculate forward and backward probabilities between states. Our algorithm manages to efficiently compute the forward and backwards proposal probabilities, due to the extension of the state space, which allows for the Metropolized version to sample from a specified measure on either partition space or spanning-forest space.

In the next section, we review the Metropolis–Hastings algorithm in this setting. In section 4.2, we give a full description of our Metropolized Forest ReCom algorithm and many of the implementation details. We also explain what is gained computationally by working on the space of spanning forests rather than the space of partitions.

4.1. Metropolis–Hastings algorithm. To sample from the measure P defined previously, we use the Metropolis–Hastings algorithm with our Forest ReCom algorithm as the proposal method. We will denote by $Q(T,T')$ the probability of starting from the spanning forest T and proposing spanning forest T' using the Forest ReCom algorithm. In other words, if the current state of chain is the spanning forest T, the measure $Q(T,\,\cdot\,)$ is the distribution of the next proposed move of the chain. Then, following the Metropolis–Hastings prescription, this move is accepted with a probability $A(T,T')$ defined by

(4.1)
$$A(T,T') = \min\left(1, \frac{P(T')}{P(T)}\frac{Q(T',T)}{Q(T,T')}\right) = \min\left(1, e^{-\beta[J(\xi')-J(\xi)]}\left[\frac{\tau(\xi)}{\tau(\xi')}\right]^{\gamma}\frac{Q(T',T)}{Q(T,T')}\right),$$

and rejected with probability $1 - A(T,T')$. If the step is accepted, the next state is the proposed state; if the step is rejected, the next state does not change. It is an open problem whether it is possible to transition between any two points in the state space. If this is true, one can prove that this process will converge to sampling from the measure P if it is run for sufficiently many steps.

4.2. The Forest ReCom proposal. We now describe the Forest ReCom proposal Markov Chain Q introduced in the previous section. As already mentioned, this algorithm is specifically designed to have both forward and backward transition probabilities which can be efficiently computed. From (4.1), we see that this is critical if it is to be used in the Metropolis–Hastings algorithm as a proposal.

The proposal step shares many features with the ReCom algorithm [16]. The differences come in because we are evolving spanning forests rather than graph partitions. We provide a brief outline of this proposal, assuming that the current state of the chain is the spanning forest T. Our goal is to produce a new spanning forest, T', which corresponds to merging two adjacent spanning trees in T, then redividing the merged tree into two new spanning trees which satisfy constraints given by J, and then calculate $Q(T,T')$ and $Q(T',T)$. In this subsection, we provide a high-level outline of this procedure.

Given spanning forest $T = (T_1, \ldots, T_n)$, we

1. Choose two trees T_i and T_j from T which correspond to adjacent partitions.
2. Draw a new spanning tree T'_{ij} uniformly at random on the subgraph $\xi_{i,j}$ induced by the union of vertices in T_i and T_j and the edges connecting these vertices. In other words, this induced graph is $\xi_{ij} = (V_{ij}, E_{ij})$, where $V_{ij}(\xi) = \{v \in V \mid \xi(v) \in \{i,j\}\}$ and $E_{ij}(\xi) = \{(v,u) \in E \mid \xi(v), \xi(u) \in \{i,j\}\}$.
3. Determine the edges of the newly drawn tree T'_{ij} such that, once removed, they would split the spanning tree into two trees that each comply with some subset of the constraints. If there are no such edges, remain in the current state and the proposal is finished; otherwise, continue.
4. Select one such edge and remove it from the new spanning tree T'_{ij}, leaving two new trees T'_i and T'_j.

In detailing the implementation of these steps, there are a number of ways that the first step may be carried out. For example, we might chose uniformly from all pairs of adjacent partitions; alternatively, we can weigh the choice by some property of the shared boundary between partitions such as the sum of edge weights or we can account for the the values of the score function J.

The second step is achieved by Wilson's algorithm which employs loop-erased random walks. For the third step, the most pertinent constraint is equal population,

so the third step involves a simple depth-first search along the tree with exit criteria based on the remaining population within a search branch. Choosing the specific edge to cut in the fourth step may be done uniformly or with a weighted distribution that might, for example, favor more equal populations.

As a note, it is possible that certain hard constraints in the energy may disconnect the state space. Theoretically, one could replace all hard constraints with soft constraints which would reconnect the space, although potentially lead to significant energetic barriers in the space. In practice, we launch multiple chains with various initial conditions and compare the distributions on observables of interest (e.g., see below in section 6). Such studies do not guarantee mixing but do provide a way to test the hypothesis that mixing has occurred.

To make the algorithm reversible through Metropolis–Hastings, we must compute the probability of proposing state T' from state T, along with computing the reverse probability of proposing state T from state T'. Once the pair of spanning trees T_i and T_j, to be are merged and then split, is chosen, the remainder of the algorithm is summarized by the set of mappings shown in (4.2). They summarize steps 2–4 above. The annotations will help to explain why the choice of a forest of spanning trees as states space and the particular structure of (4.2) is important for calculating the probability of forward and backward probabilities.

$$\{T_i, T_j\} \xrightarrow[\text{deterministic}]{\text{many-to-one}} \xi_{ij} \xrightarrow[\text{random}]{\text{one-to-many}} T'_{ij} \xrightarrow[\text{random}]{\text{one-to-a-few}} \{T'_i, T'_j\}. \tag{4.2}$$

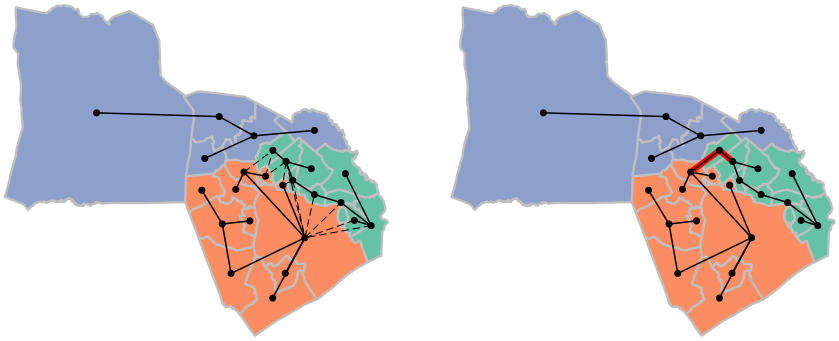
There are some indications in the above mappings that suggest calculating the forward and backward probabilities should be tractable. The initial mapping is deterministic and all the random choices to come only depend on ξ_{ij}. Though the next step is a one-to-many random map, we will choose it to be uniform on a set whose size we can calculate relatively efficiently, and, from which, we can draw uniformly. Since the next map is onto a relatively small set, it will be possible calculate the forward probabilities and produce a random draw. Because in this step the forward possibilities are limited, we will see that identifying the backward possibilities will also be tractable.

Calculating the forward and backward proposal probabilities, $Q(T, T')$ and $Q(T', T)$. We begin by calculating the chance that the merging of $\{T_i, T_j\}$ and subsequent splitting produces the replacement spanning trees $\{T'_i, T'_j\}$. To do this, we must examine all possible spanning trees on the induced graph of ξ_{ij} that could have been drawn in step 2 and then cut in step 4 to result in T'_i and T'_j. We must then sum the probability that we found T'_i and T'_j across all such choices to compute the probability of proposing the new state.

The set of all possible spanning trees that could result in T'_i and T'_j is simply the trees defined by $T'_i \cup T'_j$ along with each edge in G connecting the two graphs (see Figure 2a). The edges in G that connect the spanning trees are $\{(v, u) \in E \mid \xi'(u) = i, \xi'(v) = j\}$ and we will denote this set as $E(T'_i, T'_j)$. Together with the new trees T'_i and T'_j, each edge $e \in E(T'_i, T'_j)$ induces a spanning tree on the induced graph ξ_{ij}, which we will denote $T_{(T'_i, T'_j, e)}$. This new spanning tree is one of the trees that could have been drawn in step 2 and cut in step 4 to yield T'_i and T'_j.

Let the probability from step 4 that we cut the tree $T_{(T'_i, T'_j, e)}$ at edge e' be $P_{cut}(e' \mid T_{(T'_i, T'_j, e)})$. Note that the probability that we cut $T_{(T'_i, T'_j, e)}$ into T'_i and T'_j is $P_{cut}(e \mid T_{(T'_i, T'_j, e)})$ (see Figure 2b). Finally, note that the probability of drawing each of the spanning trees induced by some edge in $E(T'_i, T'_j)$ is simply $1/\tau(\xi_{ij})$.

Putting all of this together, we now find that the probability of the proposing T'_i and T'_j from T_i and T_j is



(a) Edges $E(T_i', T_j')$ (dashed) that complete a merged tree

(b) After selecting one edge from (A), we highlight all edges that could have been cut to provide a new balanced partition

FIG. 2. *When calculating the proposal probability of drawing T_i' (orange) and T_j' (green), we must examine all edges that could have made a spanning tree on the joined space* (a). *For each edge we must examine all edges that could have been cut and then compute the probability that we cut the edge that leads to the observed partition. In* (b) *we show the choice of one of the conflicted edges in $e \in E(T_i', T_j')$ and mark it along with a second edge in red; both of these edges could have been cut with probability $P_{cut}(e' \mid T_{(T_i', T_j', e)})$, where e' is either e or the second edge highlighted in red.*

$$q(\{T_i, T_j\}, \{T_i', T_j'\}) = \frac{1}{\tau(\xi_{ij})} \sum_{e \in E(T_i', T_j')} P_{cut}(e \mid T_{(T_i', T_j', e)}). \tag{4.3}$$

This calculation assumes that we have already chosen T_i and T_j. Given the spanning forest T let us denote by $p(\{i,j\} \mid T)$ the probability from step one of picking the pair of adjacent spanning trees T_i and T_j to merge. This probability is simple to calculate for most reasonable choices of how to perform step 1. Then the probability of proposing state T' from state T is given as

$$Q(T, T') = p(\{i,j\} \mid T) q(\{T_i, T_j\}, \{T_i', T_j'\}), \tag{4.4}$$

where $q(\{T_i, T_j\}, \{T_i', T_j'\})$ is the transition calculated in (4.3).

We summarize our algorithm in Algorithm 4.1 which summarizes the above details.

Properties of the Forest ReCom proposal Q. First, observe that when the Forest ReCom proposal from (4.4) is inserted into the formula for the acceptance probability from (4.1), we obtain

$$A(T, T') = \min\left(1, e^{-\beta(J(\xi') - J(\xi))} \left[\frac{\tau(\xi)}{\tau(\xi')}\right]^{\gamma} \frac{p(\{i,j\} \mid T')}{p(\{i,j\} \mid T)} \frac{q(\{T_i', T_j'\}, \{T_i, T_j\})}{q(\{T_i, T_j\}, \{T_i', T_j'\})}\right). \tag{4.5}$$

From this we see that if $\gamma = 0$, one does not need to calculate the $\tau(\xi)$ and $\tau(\xi')$ factors, which reduces the computational costs. Furthermore, as mentioned before, the difference between $\tau(\xi)$ and $\tau(\xi')$ is a principal reason for a low acceptance rate when γ is closer to 1.

Inserting (4.3) into (4.5), we see that the ratios of the Forest ReCom proposal probabilities can be written as

Algorithm 4.1 Metropolis–Hastings step.

Input districting plan ξ, extended by spanning forest T
Randomly select adjacent trees T_i, T_j, with probability $p(i,j|\xi)$
Generate a uniform random spanning tree on $\xi_{i,j}$: T'_{ij}
Find removable edges of T'_{ij} and select one, e' with probability $P_{cut}(e'|T_{ij})$
The cut provides two new trees: T'_i and T'_j
Find probability of proposing (T'_i, T'_j) from (T_i, T_j):
$\frac{1}{\tau(\xi_{i,j})} \sum_{e \in E(T'_i, T'_j)} P_{cut}(e|T(T'_i, T'_j, e))$
Find probability of proposing (T_i, T_j) from (T'_i, T'_j):
$\frac{1}{\tau(\xi_{i,j})} \sum_{e \in E(T_i, T_j)} P_{cut}(e|T(T_i, T_j, e))$
Find the acceptance probability $A(T'|T)$: $\frac{p(i,j|\xi')P(T') \sum_{e \in E(T_i,T_j)} P_{cut}(e|T(T_i,T_j,e))}{p(i,j|\xi)P(T) \sum_{e \in E(T'_i,T'_j)} P_{cut}(e|T(T'_i,T'_j,e))}$
Here $P(T)$ is the prespecified probability distribution on spanning forests
With probability $A(T'|T)$, replace T_i and T_j with T'_i and T'_j
return T

$$\frac{q(\{T'_i, T'_j\}, \{T_i, T_j\})}{q(\{T_i, T_j\}, \{T'_i, T'_j\})} = \frac{\sum_{e \in E(T_i,T_j)} P_{cut}(e \mid T_{(T_i,T_j,e)})}{\sum_{e \in E(T'_i,T'_j)} P_{cut}(e \mid T_{(T'_i,T'_j,e)})}. \tag{4.6}$$

In particular, we see that the $\tau(\xi_{ij})$ factors in each q expression cancel and hence, need not be computed.

To better understand the structure of this ratio, notice that when there is only a single edge that could possibly be cut for any spanning tree induced by the edges $E(T_i, T_j)$, we may write the proposal ratio as

$$\frac{q(\{T'_i, T'_j\}, \{T_i, T_j\})}{q(\{T_i, T_j\}, \{T'_i, T'_j\})} = \frac{|E_{ij}(T_i, T_j)|}{|E_{ij}(T'_i, T'_j)|}. \tag{4.7}$$

Hence, in this case we see that the ratio of probabilities is equal to the ratio of the number of edges which connect the two trees, or, in other words, the graph-theoretic length of the boundary. With this calculation in mind it is reasonable to define an effective boundary between T_i and T_j by

$$\partial(T_i, T_j) = \sum_{e \in E(T_i,T_j)} P_{cut}(e|T_{(T_i,T_j,e)}). \tag{4.8}$$

Returning to general γ and with this notation, the acceptance ratio becomes

$$A(T, T') = \min\left(1, e^{-\beta[J(\xi')-J(\xi)]} \frac{p(\{i,j\} \mid T')}{p(\{i,j\} \mid T)} \frac{\partial(T_i, T_j)}{\partial(T'_i, T'_j)} \left[\frac{\tau(T_i)\tau(T_j)}{\tau(T'_i)\tau(T'_j)}\right]^{\gamma}\right), \tag{4.9}$$

where we have used the fact that the spanning forests T and T' only differ in the ith and jth trees so

$$\frac{\tau(T)}{\tau(T')} = \frac{\tau(T_i)\tau(T_j)}{\tau(T'_i)\tau(T'_j)}.$$

As mentioned previously, the ratio between spanning tree products, τ, may be large between partitions, ξ and ξ'. This disparity is eliminated when $\gamma = 0$; however,

setting $\gamma = 0$ favors sampling partitions with higher values of τ. The parameter γ is presented as a smoothly varying parameter because it demonstrates how one may use a tempering (e.g., simulated or parallel tempering) scheme to vary γ from 0 to 1 across multiple chains in an extended product measure. We have chosen a form of the measure which essentially depends only on the partition $\xi(T)$ induced by the spanning forest T. Additionally, when $\gamma = 0$ there is no need to compute the number of spanning trees on the new partitions, as the products τ do not explicitly appear in the measure nor the proposal ratio.

4.3. Why lift from partitions to tree partitions?. We remark now on why it was useful to expand our state space on the space of all spanning forests. Consider our algorithm where given a partition $\xi = \{\xi_1, \dots, \xi_n\}$, we produce a new partition $\xi' = \{\xi'_1, \dots, \xi'_n\}$. ξ' is the same as ξ except that two adjacent elements of the partition have been merged and then split into two to create two new elements. This is essentially the Forest ReCom proposal described in section 4.2 and given by the probability distribution Q. Our algorithm begins by erasing the initial spanning trees on the two partitions chosen, merging the vertices, then drawing a new spanning tree on the induced graph. We could, however, also view the algorithm as moving between pairs of partition elements in which we draw a spanning tree on the merged induced graph and precede exactly as in section 4.2 only without knowledge of the extended state space. Hence the two perspectives only differ in that one takes a spanning forest and transitions to a new spanning forest, whereas the second perspective takes a partition and transitions into a new partition.

The difference comes when one tries to calculate the probability $Q(\xi, \xi')$. We have already seen that in the spanning forest space $Q(T, T')$ is tractable. However, we now explain why, to the best of our understanding, calculating $Q(\xi, \xi')$ is much more difficult.

Looking back at (4.3), we begin by remarking that the transition kernel $q(\{T_i, T_j\}, \{T'_i, T'_j\})$ only depends on $\{T_i, T_j\}$ through the union of their vertex set which we have denoted ξ_{ij}. Hence, we can equally view q as a transition from the partitions $\{\xi_i, \xi_j\}$ to the spanning trees to the $\{T'_i, T'_j\}$ denoted by $q(\{\xi_i.\xi_k\}, \{T'_i, T'_j\})$. This is simply because both $\{\xi_i, \xi_k\}$ and $\{T_i, T_j\}$ determine ξ_{ij} and hence can be used as input to calculate the probability. This last fact is evident from (4.2) and from (4.11) given below. With this observation, we have that

$$
\begin{aligned}
(4.10) \qquad Q(\{\xi_i, \xi_j\}, \{\xi'_i, \xi'_j\}) &= \frac{1}{\tau(\xi_{ij})} \sum_{T'_i \in ST(\xi'_i)} \sum_{T'_j \in ST(\xi'_j)} \sum_{e \in E(T'_i, T'_j)} P_{cut}(e \mid T_{(T'_i, T'_j, e)}) \\
&= \sum_{T'_i \in ST(\xi'_i)} \sum_{T'_j \in ST(\xi'_j)} q(\{\xi_i.\xi_j\}, \{T'_i, T'_j\}),
\end{aligned}
$$

where we have used the same notation as in section 4.2. We now see that if we want to calculate $Q(\{\xi_i, \xi_j\}, \{\xi'_i, \xi'_j\})$, one needs to calculate $|ST(\xi'_i)| \times |ST(\xi'_j)|$ transition probabilities of the form $q(\{\xi_i.\xi_j\}, \{T'_i, T'_j\})$ instead of the one needed for $q(\{T_i, T_j\}, \{T'_i, T'_j\})$. This saves considerable computational effort.

Another perspective is that computing $Q(\{\xi_i, \xi_j\}, \{\xi'_i, \xi'_j\})$ involves the one-to-many map of taking a partition into all possible spanning trees followed by the many-to-one map of erasing the spanning tree to obtain the partition. This is expensive. In $Q(\{T_i, T_j\}, \{T'_i, T'_j\})$ the order is reversed. The many-to-one operation of erasing the trees to arrive at a partition comes first followed by the one-to-many operation of drawing the trees. This computation is relatively easy. This is summarized in the diagram in (4.11), which should be compared with (4.2):

(4.11)
$$\{\xi_i, \xi_j\} \xrightarrow[\text{deterministic}]{\text{many-to-one}} \xi_{ij} \xrightarrow[\text{random}]{\text{one-to-many}} T'_{ij} \xrightarrow[\text{random}]{\text{one-to-a-few}} \{T'_i, T'_j\} \xrightarrow[\text{random}]{\text{many-to-one}} \{\xi'_i, \xi'_j\}.$$

When comparing with (4.2), we see that the first maps are essentially equivalent basically encapsulating the already mentioned fact that all of the random choices only depend on the initial state though ξ_{ij}. The next two random maps are the same as for the algorithm on spanning trees; and hence, are relatively easily to draw from and compute the backwards and forwards probabilities. The complication comes from the last mapping. It is many-to-one. This means that there are many tree pairs $\{T'_i, T'_j\}$ in the preimage of a single partition pair $\{\xi'_i, \xi'_j\}$. This is represented by the two outer sums in (4.10) which combined are over $|ST(\xi'_i)| \times |ST(\xi'_j)|$ terms.

Remark 4.1. Up until this point, we have insisted that the state space be defined as a spanning forest. However, it is possible to use Metropolized Forest ReCom to sample from the space of partitions while preserving most of the computational improvements.

To see this, observe that each spanning tree within each partition is uniform conditioned on that partition, thus conditioned on each partition, each tree in the partition is equally likely. Due to this, we can project the state space down the space of partitions without losing critical information. We then modify the algorithm as follows: When we merge and split on adjacent partitions we can draw new random trees on each partition. There is a small amount of wasted effort on redrawing the trees within current partitions; Metropolized Forest ReCom does not waste this effort since it holds the trees within the state. These trees are then used when calculating the reverse probability. When we either accept or reject, the old or new trees can be deleted and we can project back down to the space of partitions.

5. Implementation of Forest ReCom proposal Q. We detail the implementations of steps within the proposal. There are many choices when picking the adjacent pair probability pair $P(\{i,j\}|T)$—one may uniformly choose amongst adjacent partitions or weigh the choice by the sum of edge weights (e.g., shared border length for redistricting), the shared number of conflicted edges, or some heuristic of the acceptance probability. In the current work we make this choice in two different ways. In one method, we uniformly pick a random district and then uniformly pick a random district neighbor, so that

$$\begin{aligned} P(\{i,j\}|T) &= P(\{i,j\}|\xi(T)) \\ &= P(i|\xi)P(j|i,\xi) + P(j|\xi)P(i|j,\xi) \\ &= \frac{1}{n}\left(\frac{1}{N_i(\xi)} + \frac{1}{N_j(\xi)}\right), \end{aligned} \tag{5.1}$$

where n is the number of partitions, and $N_i(\xi(T))$ is the number of partitions neighboring partition ξ_i. In the second method, we randomly select an edge that is not confined to a single partition, meaning that we preferentially choose partition pairs with more shared edges. Formally,

$$P(\{i,j\}|T) = P(\{i,j\}|\xi(T)) = \frac{\partial_e(\xi_i, \xi_j)}{\sum_{\ell<k} \partial_e(\xi_\ell, \xi_k)}, \tag{5.2}$$

where $\partial_e(\xi_\ell, \xi_k)$ is the number of edges on the graph that have one node in partition ξ_ℓ and another in partition ξ_k. We denote the former method as "uniform partition-uniform neighbor" and the latter as "boundary weighted."

As already mentioned, uniform spanning trees will be drawn using Wilson's algorithm. There are several implementations of this algorithm (for details, see the supplemental material (mergeSplit_supplement.pdf [local/web 486KB])). If $\gamma \neq 0$, we must compute the number of spanning trees on each subgraph induced by ξ_i; this is accomplished via Kirchoff's theorem and is, algorithmically, the slowest step of the algorithm (see the supplemental material (mergeSplit_supplement.pdf [local/web 486KB])).

When choosing what edge to cut, we must specify the probability of cutting edge e given tree T_{ij}, $P_{cut}(e|T_{ij})$. Perhaps the simplest implementation is to uniformly choose an edge from the set of edges such that the the cut leads to two trees with populations within the constraints set out by J. Let $E_c(T)$ denote the edges, such that if a single edge was removed from T, the remaining two trees would have population within the specified constraints. Then

$$P_{cut}(e|T) = \begin{cases} |E_c(T)|^{-1}, & e \in E_c(T), \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

We adopt this approach in the present work. We detail how we determine $E_c(T)$ both for the initial cut, and when computing the proposal probability given in (4.3) in the supplemental material (mergeSplit_supplement.pdf [local/web 486KB]).

6. Validation on an enumerable example. We implement and test our algorithm by considering the graph displayed in Figures 1 and 2. This graph is derived from a realistic redistricting example that contains three partitions. Balance across the partitions is found by ensuring the three partitions have roughly equal population (see the supplemental material (mergeSplit_supplement.pdf [local/web 486KB]) for more details). We are able to explicitly enumerate the possible partitions of this graph, meaning that we can compare our sampling methods with a ground truth.

We run two batches of 10 independent chains. The two batches are for the two methods of choosing adjacent district pairs to merge and split (see (5.1) and (5.2) in section 5). Within each batch, each of the chains begins with a unique initial condition and runs for one million steps with $\gamma = 1$ and J only accounting for the population constraints (which is constrained via the proposal).[8]

6.1. Convergence on two low dimensional observables. With the samples collected from the chains, we consider two sets of observables on this space, which are related to evaluating partisan gerrymandering: The number of Democrats elected under a given set of votes, and the order statistics of the margins of victory (i.e., the Democratic vote fractions). Details are provided in the supplemental material (mergeSplit_supplement.pdf [local/web 486KB]). For the purpose of validating our algorithm we note that we are less concerned about *which* low dimensional statistic we evaluate than we are with taking some relevant observable of interest. Alternatively, we could have examined the distribution of some measure on compactness across the three partitions (i.e., districts) or have taken some other relevant measure on the space.

In our experiments, we find that the methods of selecting adjacent partitions lead to similar results. Under the "uniform district-uniform neighbor" method (see (5.1)), an average of 185,884.1 proposals per chain are accepted over the 10 chains of 10^6 steps. Under the "boundary weighted" method (see (5.2)), an average of 52,537.2 proposals per chain are accepted over the 10 chains of 10^6 steps. For each batch of

[8]The code we used to run these chains is available at https://git.math.duke.edu/gitlab/gjh/mergesplitcodebase.git. See the supplemental material (mergeSplit_supplement.pdf [local/web 486KB]) for details on how we determine unique initial conditions for each chain.

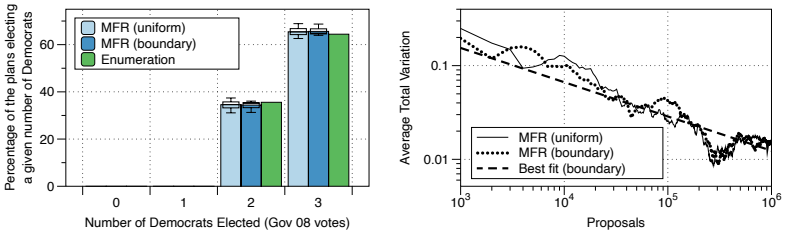


FIG. 3. *We display the distribution of the number of Democrats elected in the small graph from Figure* 1 *with a fixed vote count (left). We use two methods of picking adjacent districts when sampling. We find that the two Metropolized Forest ReCom (MFR) sampling methods (uniform district-uniform neighbor, and boundary weighted) come very close to the exact enumerated distribution (shown in green) after running* 10 *chains of one million proposals. We use box plots to demonstrate the variation across chains. We then plot the averaged total variation in each of the* 10 *distributions as a function of the number of proposals (right). We fit a power law to the observations and find that the error decays like* $\propto x^{-0.32}$ *and* $\propto x^{-0.37}$ *in the uniform district-uniform neighbor and boundary weighted methods, respectively. (Figure in color online.)*

10 chains we plot the sampled distribution of Democrats elected in Figure 3 (left). We also estimate the rate of convergence by examining the average total variation[9] between each of the 10 chains and the enumerated ensemble. We take a best fit power law over both methods and find an order of convergence of to be 0.32 for the "uniform district-uniform neighbor" method and 0.37 for the "boundary weighted" method (see the right side of Figure 3). At the final step, we find that the total variation between the ensemble of aggregated chains and the enumerated results has reduced to 0.0155 for the uniform district-uniform neighbor method and 0.0154 for the boundary weighted method. In both cases, the exact and sampled distributions are qualitatively very close after one million proposals.

To study the convergence of the ordered marginal distributions, we examine the rate of convergence as a function of the number of proposals. For all pairs of chains within a batch of 10 chains, we examine the total variation distance between each of the three marginal ensembles and then average this distance. We then take the maximum averaged total variation across all pairs of chains. To measure the total variation distance, we establish histograms in each ordered marginal with a bin width of 0.2%. We plot the averaged total variation as a function of the number of proposals in Figure 4. The average total variation decreases (roughly) according to a power law with order 0.39 for the uniform-district uniform-neighbor cases and with an order of 0.34 for the boundary weighted case. However, the two plots are indistinguishable by eye and we only display one. After one million proposals the averaged total variation has decreased to less than 0.044% for both methods and the overall features of all three distributions are extremely similar.

6.2. Lack of convergence on the distribution of partitions. We examine the underlying distribution on the full redistricting space: namely, the uniform distribution on the 17,653 enumerated partitions. We plot the total variation distance between the chains and the exact result as a function of the number of proposals in Figure 5. We find a power law relationship with order 0.26 along with a smaller constant of proportionality—this metric converges significantly slower than the previous two estimates. By the end of the million proposals, the total variation across the 10 independent chains is large after one million proposals, with an average value of

[9]Here and throughout this paper, we use the normalized version of the total variation so that it takes on values between 0 and 1.

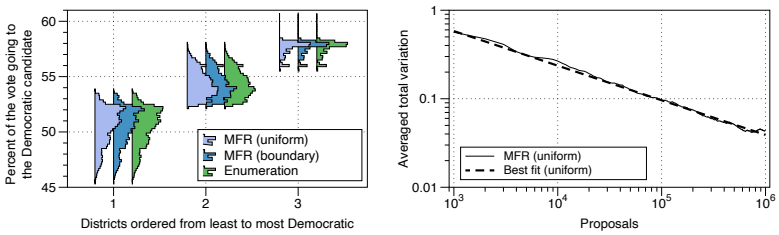


FIG. 4. *We display the ordered marginal distributions for the percent of the vote received by the Democrat in the enumerated ensemble (green) and over the batches of* 10 *chains of one million proposals of the two MFR algorithms (dark and light blue) (left). After one million proposals, the two distributions become extremely similar. We examine the difference between the marginal distributions as a function of the number of proposals (right). We omit the weighted boundary method as the line nearly perfectly overlaps with the uniform-district uniform-neighbor method displayed in the plot and labled "uniform." (Figure in color online.)*

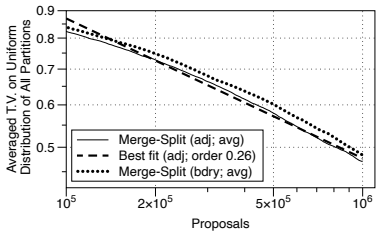


FIG. 5. *We display the total variation between the MFR chain and the uniform distribution on all partitions of Duplin–Onslow. On average, the* 10 *chains hold a total variation of roughly* 0.47 *after one million proposals. This distribution decays according to a power law with order* 0.26 *for the uniform district-uniform neighbor method and* 0.25 *for the boundary weighted method.*

0.47. This may reflect the recently proven result [31], which states that sampling the uniform measure on graph partitions is likely NP-hard.

If we place any faith in the power law relations above, then we may predict that the chain should have sampled the uniform distribution with a total variation of roughly 1% after 2.6×10^{12} proposals. However, the above experiment also demonstrates that the chains may accurately predict observables of interest far faster than they are able to recover the full underlying distribution. To make this comparison explicit, using the best fit power law for the distribution of elected Democrats predicts that roughly two million proposals would be needed to achieve a total variation of roughly 1% (which is in the realm of what we observe). Similarly, we would need roughly 32 million proposals for the total variation in the averaged marginal distributions to be roughly 1%. These predictions suggest that one needs a factor of roughly 10^6 or 10^5 more samples to approximate the distribution on all partitions than the partisan outcomes considered.

We also investigate which of the enumerated partitions went unsampled. Of the 10 chains in the weighted boundary method, each chain samples roughly two thirds of the 17,653 enumerated partitions at least once. When looking over all chains, only 795 of the enumerated partitions were not sampled by any of the 10 chains after the one million proposals. We examine the histogram of the product of spanning trees, $\tau(\xi)$ for the enumerated partitions and the sampled partitions in Figure 6. We find the distributions are nearly identical; however, there is some variation for partitions with fewer total spanning forests. To confirm this observation, we plot the histogram

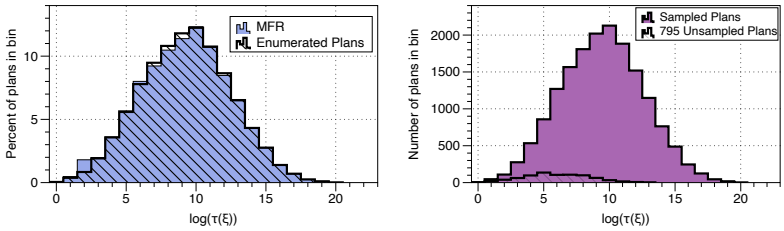


FIG. 6. *On the left, we plot the histogram of the number of spanning forests, $\tau(\xi)$, for the enumerated partitions and the sampled partitions. On the right, we examine the histogram of the partitions that were sampled at least once and compare it with the partitions that we not sampled within the boundary weighted MFR batch of* 10 *chains.*

FIG. 7. *We compare the ordered marginal distributions across two distributions on the Duplin–Onslow graph. The distributions are* (i) *uniform on forests ($\gamma = 0$) and* (ii) *uniform on partitions ($\gamma = 1$). The observable distributions show significant differences.*

of enumerated partitions that were sampled at least once and compare it to the histogram of enumerated partitions that were not sampled and see that, in general, the unsampled partitions have fewer associated spanning forests (see Figure 6).

6.3. The impact of the distribution. Next, we demonstrate how changing the measure may change the observables. We again sample the Duplin–Onslow graph with uniform spanning forests ($\gamma = 0$) and compare the result in Figure 7. There are significant differences between these measures. For example, when we consider a measure that is uniform on spanning forests, there is a 26% chance that three Democrats are elected. In contrast, under a measure that is uniform on partitions, there is a 64% chance that three Democrats are elected. In short, the choice of measure will effect our conclusions on what is fair in this case, and our method allows us to have control over which measure is being sampled.

6.4. Acceptance rate dependence on γ. The discussion above has centered around numerical investigations of mixing times. Another tool often used to study the efficiency of Metropolis–Hastings chains is to examine the acceptance rate of the algorithm. We conclude our discussion of the Duplin–Onslow cluster by examining the acceptance rates as a function of γ. We generate eight additional chains on the uniform district-uniform neighbor method, the ith of which has a $\gamma = (i-1)/8$. The score function on these chains only considers the population deviation, which is handled via the proposal; for Duplin–Onslow we again use a threshold of 5%. We run the chains for one million proposals and estimate the acceptance rates in Figure 8. As discussed previously, we see the highest acceptance rates for low γ and the lowest acceptance rates for large γ. The relationship between acceptance with γ appears to be monotonically decreasing. For the Duplin–Onslow graph, it yields an acceptance of just over 40% for $\gamma = 0$, and just under 20% for $\gamma = 1$.

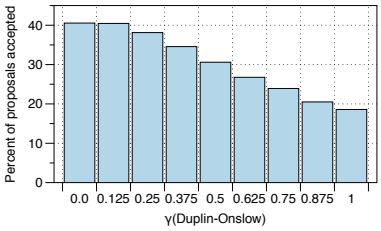


FIG. 8. *We display the acceptance rate in the MFR algorithm as a function of γ in the Duplin–Onslow county cluster under the uniform district-uniform neighbor pair proposal.*

Remark 6.1. In this work, we fix a certain measure; however, selecting different measures and score functions may have a significant impact on our conclusions. For example, in [22], the present methods were used to sample the same redistricting graph on a measure that ignores municipal preservation and on another measure that prefers the preservation of municipalities. In this setting we have found that certain observables of interest are nearly disjoint in distribution.

We also show the difference between sampling at $\gamma = 0$ and $\gamma = 1$ in the above example in the supplemental material (mergeSplit_supplement.pdf [local/web 486KB]).

7. Evidence of convergence on an innumerable example. We continue by examining the MFR algorithm applied to a larger problem. We examine the precinct graph of North Carolina and consider 13 partitions, which represent the 13 congressional districts of the 2010 redistricting cycle. See the supplemental material (mergeSplit_supplement.pdf [local/web 486KB]) for the precinct map and specific details on this example.

Without constraints, partitions sampled from the space of uniform (population compliant) partitions may have jagged boundaries and form highly noncompact districts that look like space filling curves (see, e.g., [16]). Under the ReCom Markov chain, districts tend to be significantly more compact; however, built in rejections for noncompact districts may be included [16]. As noted above, when $\gamma = 0$, our MFR is a Metropolized version of ReCom with an expanded state space; because of this ReCom should focus sampling on districts that are comparable to the forest distribution with $\gamma = 0$. We therefore begin by examining convergence properties of MFR for $\gamma = 0$ with no additional constraints (other than population) and with a constraint that ensures no district within a plan has an isoperemetric ratio of more than 110 (or Polsby–Popper less than 0.114; this threshold allows districts to be slightly less compact than the least compact district in the 2016 North Carolina congressional districts).

7.1. Convergence on two low dimensional observables ($\gamma = 0$). We examine the convergence properties of both cases by running a batch of 10 chains for each method. Each chain is run for one million steps, starts with randomized initial conditions, and uses the "boundary-weighted" method of choosing adjacent districts. When bounding the district compactness, we ensure that the initial states contain districts that are all more compact than the allowable limit. When unconstrained, we find an acceptance rate of just over a fourth: Each chain accepts more than 260,000 proposals and accepts 265,146.3 proposals on average. When ensuring compact districts, we find an acceptance rate of just over a fifth: Each chain accepts more than 210,000 proposals and accepts 211,966.3 proposals on average.

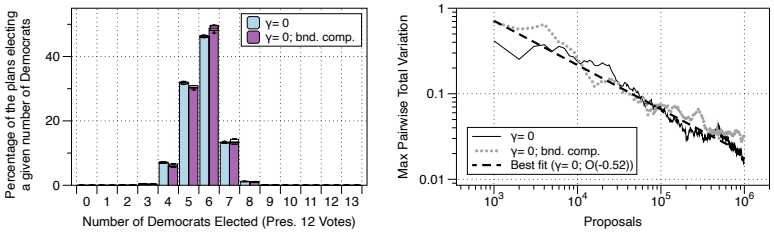


FIG. 9. *We display the distribution of the number of Democrats elected in the* 13 *districts in the North Carolina congressional districts in the case that we change the districts but fix the votes to be those of the* 2012 *Presidential race (left). We use box plots to demonstrate the variation across chains becomes small after one million proposals (they are very small, demonstrating the agreement across chains, and appear around the top of each bar). We plot the maximal pairwise total variation between the* 10 *distributions as a function of the number of proposals (right). We fit a power law to the observations and find that the error decays like* $\propto x^{-0.52}$ *when* $\gamma = 0$ *and* $O(0.4)$ *when* $\gamma = 0$ *and with constrained compactness.*

To test the convergence, we use votes cast in the 2012 presidential race and plot the histogram of the number of Democrats that would have been elected in the ensemble of districting plans under these votes (see Figure 9). We find that the distribution of elected Democrats converges to nearly the same distribution, independent of the starting location. We demonstrate the small variance across the chains by placing box plots on top of our histograms in Figure 9. Without bounding the compactness we see a variance across the chains that is less than 1.4% in total variation; with bounded compactness, we see a total variation of less than 3% across the histograms in the 10 chains. We also examine the rate of convergence by examining the maximum pairwise total variation across all histograms of elected Democrats after a given number of proposals. Taking a best fit power law, we find that the order of convergence is $O(0.52)$ without any constraint on compactness and $O(0.4)$ with the constraint (see the right of Figure 9).

We continue by examining the ordered marginal distributions of Democratic vote fractions across the ensemble of plans. We create histograms for the marginal distributions with a bin width of 0.2%. We then compute the average total variation distance across all 13 marginal ensembles between each pair of chains and find the pair with the maximum average total variation distance across all pairs. We display the two most distinct chains after one million proposals along with the distribution comprised of all 10 chains in Figure 10 for the batch that constrains compactness; we do this because the chain that does not constrain compactness has even smaller differences across chains. The 10 chains converge to nearly identical distribution, both when constraining the compactness or not. We take a best fit power law by examining the maximum pairwise distribution as a function of the number of proposals and find a convergence rate of 0.5 when the compactness is unconstrained and 0.44 when the compactness is bounded.

7.2. Changing the invariant measure. When $\gamma = 0$, the measure places more importance on plans with higher spanning forest counts. Depending on the application, one may wish to remove this relative importance between plans (which is to set $\gamma = 1$) and then focus on partitions of interest via the score function $J(\xi)$.

We investigate the algorithm at $\gamma = 1$, and begin with no constraints or additions to the score function. In this case, we see no evidence of convergence after 10 million

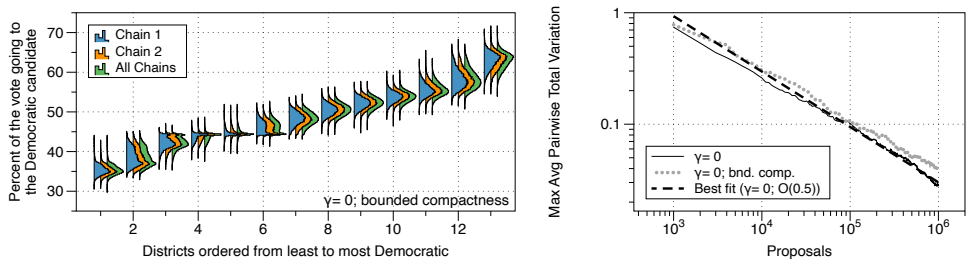


FIG. 10. *We display the ordered marginal distributions for the percent of the vote received by the Democrat in all* 10 *chains (green) and over the two chains that are most distinct after one million proposals (orange and blue) for the batch with* $\gamma = 0$ *and constrained compactness (left). After one million proposals, the distributions become extremely similar. We then take the maximum pairwise total variation averaged across all marginal distributions as a function of the number of proposals for the batches with and without compactness constrains with* $\gamma = 0$ *(right). We find a rate of convergence on the order of* 0.5 *when there is no constraint on compactness and* 0.44 *when there is a constraint. (Figure in color online.)*

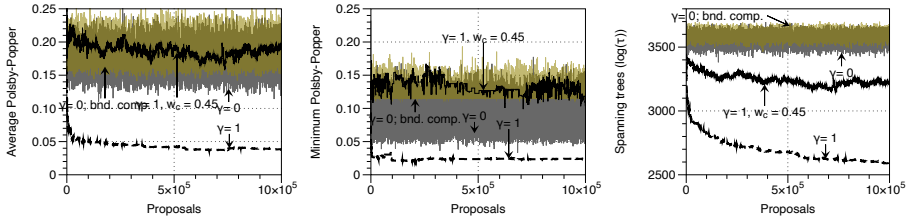


FIG. 11. *We display the average Polsby–Popper scores over a plan, the minimal Polsby–Popper score over all districts within a plan, and the total number of spanning forests as a function of the number of proposals on the left, center, and right, respectively. The gold line represents the* $\gamma = 0$ *case with bounded compactness; it is semitransparent and is displayed over the gray line that represents the* $\gamma = 0$ *case without bounded compactness. In the center, the gold line is strictly bounded below and thus appears flat.*

steps as the total variation distance between chains remains very high in both observables. The reason for this may be that uniformly drawn partitions on this graph may have undulating boundaries with highly noncompact districts (e.g., see [16]); the consequence would be that when districts are merged along a boundary, the proposal will tend to smooth the boundary which will lead to a higher number spanning trees in the proposed districts and lead to a higher chance of rejection. Indeed, we see that the compactness greatly decreases as the chain evolves in Figure 11 which supports this hypothesis.

We continue to investigate the $\gamma = 1$ case by adjusting the score function $J(\xi)$ to include a compactness term. We do this by adding a sum of the isoperimetric ratios of the districts and weight this sum by a factor w_c so that now

$$J(\xi) = J_{pop}(\xi) + w_c J_c(\xi), \quad \text{where} \tag{7.1}$$

$$J_c(\xi) = \sum_{d=1}^{D} P_d^2 / A_d, \tag{7.2}$$

and $J_{pop}(\xi)$ is infinite when the population deviation of a district is beyond an allowable threshold (i.e., leads to an unbalanced partition), P_d is the perimeter, or boundary length, of district d, and A_d is the area of district d.[10]

We make the supposition that the level of compactness in the isoperimetric ratio associated with the $\gamma = 0$ measure is, on average, appropriate. However, sampling from $\gamma = 0$ will bias such compact plans towards those with more spanning trees. In undoing this bias, we set $\gamma = 1$ and tune w_c so that we are likely to sample districts that are as compact as those sampled with $\gamma = 0$ (i.e., have a similar isoperimetric ratio). We find $w_c = 0.45$. We show the similarity of the average compactness scores between the two $\gamma = 0, w_c = 0$ chains (one with constraints, one without) and $\gamma = 1, w_c = 0.45$ in Figure 11. In the same figure, we also examine the evolution of the least compact district in these runs and find that the run with $\gamma = 1, w_c = 0.45$ leads to more-compact least-compact districts than the run with $\gamma = 0, w_c = 0$. When $\gamma = 1$ and $w_c = 0$ we see the plans become extremely noncompact. We also display the number of spanning trees. Changing w_c from 0 to 0.45 reduces the discrepancy in the number of spanning trees when compared to the $\gamma = 0$ chains. However, the number of spanning trees found in the $\gamma = 1, w_c = 0.45$ chain are still significantly smaller than those of the $\gamma = 0$ chains. When $\gamma = 1$ and $w_c = 0.45$, we see no evidence of convergence after one million steps.

7.2.1. Potential for parallel tempering. Having observed convergence for $\gamma = 0$ and a lack of convergence at $\gamma = 1$, we conclude by examining the potential for a tempering algorithm to traverse between these two regions. There are two primary methods of tempering: simulated and parallel. In parallel tempering, we consider the product measure of N chains. One chain has a marginal distribution that mixes quickly, and another samples from the target distribution. The remaining chains are used to allow chains to swap states via a swap proposal that is made reversible through Metropolis–Hastings. The idea is that a state may start in the chain that quickly mixes, but may gradually swap chains until it finds itself in the chain with the target distribution. When it is in the chain with the target distribution, it will be used to collect observables of interest.

In the current context, we could temper by expanding the state space to include a variety of γ's and w_c's that linearly interpolate between a known fast mixing chain ($\gamma = w_c = 0$) and a target distribution ($\gamma = 1$, $w_c = 0.45$). One must provide a sufficient number of intermediate values so that the states can swap chains and move to and from both extremes.

To investigate the potential efficacy of tempering, we examine nine batches of 10 chains, each run for one million proposals, and linearly interpolate between parameters so that $\gamma = i/8$ and $w_c = 0.45 \cdot i/8$ for $0 \le i \le 8$. We first examine the convergence properties of the nine batches chains by examining the maximum pairwise average total variation distance between all chains within a batch. As above, the average is taken over all 13 ordered marginals. We plot the result in Figure 12, and find that the distributions gradually (close to linearly) diverge as we simultaneously increase γ and w_c until the deviation is close to 1 at $(\gamma, w_c) = (1, 0.45)$. To examine the possible efficacy of tempering, we examine how the distributions of spanning trees and average

[10] See the supplemental material (mergeSplit_supplement.pdf [local/web 486KB]) for an investigation of the difference between using spanning trees and the isoperimetric ratios in this example problem. Specifically, we find low correlation between these quantities which demonstrates that focusing around plans at $\gamma = 0$ will carry different weights on the plans in the ensemble than those sampled from $\gamma = 1$ and $w_c = 0.45$, despite the similarities in the compactness scores between these measures.

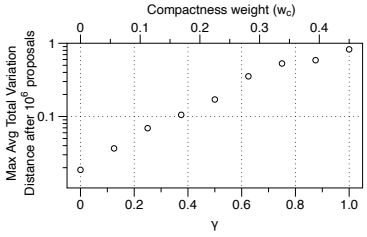


FIG. 12. *We display the maximum pairwise average total variation distance across* 10 *chains after one million proposals for different values of* γ. *The total variation is taken across each ordered marginal distribution from most Republican to Democratic districts.*

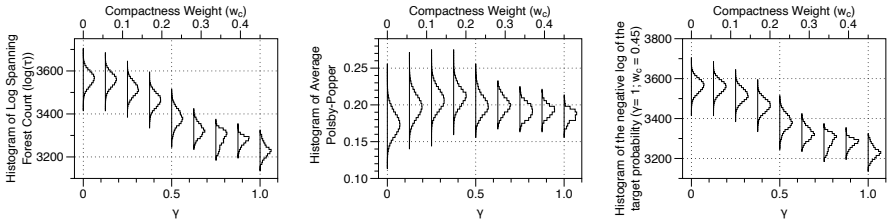


FIG. 13. *We display the distributions of the spanning trees (left) and compactness (right) over a variety of parameters* (γ, w_c) *that linearly interpolate between* $(\gamma, w_c) = (0, 0)$ *and* $(\gamma, w_c) = (1, 0.45)$. *Finally, we also display the probability of the sampled plans from each chain with respect to the target measure of* $(\gamma, w_c) = (1, 0.45)$.

Polsby–Popper score overlap within a single chain in each batch. We only consider elements in the chains after 200,000 proposals and plot both sets of distributions in Figure 13. We find that both of the distributions of spanning trees and distributions of Polsby–Popper have significant overlap across successive increments as γ and w_c increase/decrease. We also examine the probability of seeing the samples from each chain according to the target measure with $\gamma = 1$ and $w_c = 0.45$; we see that the chains at different parameters sample from much different locations in the space, but that these regions overlap as the parameters vary.

Each of the chains shown in Figure 13 has the same initial state. Given the lack of evidence in convergence at higher values of γ and w_c, it is possible that the chains are sampling around some local minimum. If the chains are stuck around this local minimum, then the overlap of distributions we see may not hold for other minima; furthermore, we have only visualized marginal distributions related to the energy (in spanning forest counts and compactness scores) rather than joint distributions of these values.

To further probe the viability of a tempering scheme, we analyze how the chains would mix in the context of parallel tempering. Using the nine batches of 10 chains from above, with the parameters of each chain defined as $(\gamma = i/8, w_c = 0.45 \cdot i/8)$, for $0 \le i \le 8$, we introduce a tempering proposal that swaps states across chains. For example, if the state space of the nine chains is given as $(\xi_0, \xi_1, \ldots, \xi_i, \xi_{i+1}, \ldots \xi_8)$, we may propose a new state $(\xi_0, \xi_1, \ldots, \xi_{i+1}, \xi_i, \ldots \xi_8)$. Under Metropolis–Hastings, the acceptance probability of swapping states across chains may is calculated as

$$A((\ldots, \xi_i, \xi_{i+1}, \ldots), (\ldots, \xi_{i+1}, \xi_i, \ldots)) = \min\left(1, \frac{\pi_i(\xi_{i+1})\pi_{i+1}(\xi_i)}{\pi_i(\xi_i)\pi_{i+1}(\xi_{i+1})}\right), \quad \text{where}$$
$$\pi_i(\xi) = e^{-0.45(i/8)J_c(\xi)}\tau^{-i/8}.$$

For each $i < 8$, we randomly sample 10^5 pairs of plans from the batches of i and $(i+1)$ parameter values from the second half of the walks (i.e., after 500,000 proposals). We compute the acceptance probability of swapping the two states. As a heuristic for mixing, we examine probability of accepting a proposed swap. We find that, on average, 23.9% of the swap proposals will be accepted. All adjacent chains have more than a 13% chance of accepting a proposed swap with the exception of the $i = 4$ and 5 chains which have a 6.1% chance of accepting a proposed swap. Although implementing a tempering algorithm is beyond the scope of this work, these final investigations demonstrate that such a scheme should be reasonable to implement.

8. Discussion. Graph partition problems have drawn long standing interest in mathematics which has only grown over recent decades. These problems typically involve optimizing partitions according to some objective function. More recently, however, there has been growing interest in how to sample the space of graph partitions when this space is coupled to a probability distribution. This endeavour has been used to determine fairness in redistricting and in detecting gerrymandering. To this end, we have developed an MFR algorithm capable of sampling graph partitions or random forest from a specified probability density on the space of readom forests or spanning trees. This is achieved my metabolizing so that after an acceptance/rejection step the resulting random algorithm has the specified measure as its invariant measure. This holds since the Markov chain generated by the algorithm satisfies detailed balance with respect to the specified measure.

Our algorithm will likely have favorable mixing properties when the measure is uniform on spanning forests as the transition proposal will entirely redraw adjacent pairs of districts. When there is a measure on forests ($\gamma = 0$), the proposal is accepted based on comparing the relative effective boundaries between the altered districts; when $\gamma > 0$, the proposal is also accepted based on comparing the product of the number of trees on the partition, $\tau(\xi(T))$ with $\tau(\xi(T'))$. As in [29, 15, 14], our proposal chain relies upon (i) uniformly sampling spanning trees on a simply connected subgraph derived from adjacent partitions, and (ii) counting the total number of spanning trees on the subgraph. These two elements are the most expensive part of the proposal and yield a computational complexity that is polynomial in the number of nodes. The complexity is reduced when sampling the uniform measure on the extended tree space as we do not need to consider the number of possible trees that may be drawn on a partition. We have written a python code base that is available at https://git.math.duke.edu/gitlab/gjh/mergesplitcodebase.git.

Although sampling from balanced graph partitions may be thought of as a general problem in applied mathematics, it has been applied to the field of fair redistricting. Thus we have demonstrated the efficacy of our method on two real world problems associated with redistricting: the three state house districts of the Duplin–Onslow county cluster in North Carolina, and the 13 North Carolina congressional districts. In the former (smaller) problem, we have demonstrated good mixing with $\gamma = 1$. In the latter we have demonstrated poor mixing for $\gamma = 1$, but good mixing for $\gamma = 0$. We have also shown how convergence varies as γ is adjusted from $\gamma = 0$ to $\gamma = 1$. In principle, one could use tempering to sample at $\gamma = 1$ while using the mixing properties of $\gamma = 0$ to decorrelate the chains. This is important, because in many applications the relevant measures are based on partitions rather than spanning forests.

Although partitioning preferences may be encoded within the energy function $J(\xi)$, the MFR procedure may not be able to efficiently explore certain constraints. However, we have provided an example where it is likely possible to use parallel

or simulated tempering to sample partitions in a way that is independent of the underlying tree counts.

Our algorithm has focused on the use of uniform spanning trees on merged pairs of adjacent partitions. However, it is not dependent on these choices and there are a number of immediate and intriguing extensions including the use of weighted edges to draw nonuniform spanning trees on the space or by merging three adjacent partitions to promote faster mixing. Furthermore, one could adapt a multiscale framework that strictly preserves higher level structures.[11] Such extensions go beyond the goals of the current paper but are promising avenues toward more sophisticated sampling procedures on graph partitions.

Acknowledgments. We thank Colin Rundel, Adam Graham-Squire, and Stephen Schecter for discussions around enumeration and providing the enumerations used here. We thank Dan Teague, Dana Randall, Debmalya Panigrahi, and Daryl DeFord for useful discussions.

REFERENCES

[1] C. Anderson, D. Lee, and N. Dean, *Spatial clustering of average risks and risk trends in bayesian disease mapping*, Biometrical J., 59 (2017), pp. 41–56.

[2] E. A. Autry, D. Carter, G. Herschlag, Z. Hunter, and J. C. Mattingly, *Multi-scale Merge-split Markov Chain Monte Carlo for Redistricting*, preprint, https://arxiv.org/abs/2008.08054, 2020.

[3] E. A. Autry, D. Carter, G. J. Herschlag, Z. Hunter, and J. C. Mattingly, *Metropolized multiscale forest recombination for redistricting*, Multiscale Model. Simul., 19 (2021), pp. 1885–1914, https://doi.org/10.1137/21M1406854.

[4] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, *Graph partitioning and graph clustering*, in Proceedings of the 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, 2012, Contemp. Math. 588, 2013.

[5] C. Balocchi, S. K. Deshpande, E. I. George, and S. T. Jensen, *Crime in Philadelphia: Bayesian Clustering with Particle Optimization*, preprint, arXiv:1912.00111, 2019.

[6] S. Bangia, B. Dou, S. Guo, and C. Vaughn, *Quantifying Gerrymandering Data+ Project*, 2015, https://services.math.duke.edu/projects/gerrymandering/.

[7] S. Bangia, C. V. Graves, G. Herschlag, H. S. Kang, J. Luo, J. C. Mattingly, and R. Ravier, *Redistricting: Drawing the Line*, preprint, https://arxiv.org/abs/1704.03360, 2017.

[8] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, *Recent advances in graph partitioning*, Algorithm Eng., (2016), pp. 117–158.

[9] K. S. Camilus and V. Govindan, *A review on graph based segmentation*, Int. J. Image, Graph. Signal Process., 4 (2012), pp. 1–13.

[10] S. Cannon, *Recombination, Reversibility, and Short Bursts*, 2020, https://bigdata.duke.edu/tripods/redistconf20.

[11] J. Chen and J. Rodden, *Unintentional gerrymandering: Political geography and electoral bias in legislatures*, Quart. J. Polit. Sci., 8 (2013), pp. 239–269.

[12] M. Chikina, A. Frieze, J. Mattingly, and W. Pegden, *Separating Effect from Significance in Markov Chain Tests*, preprint, https://arxiv.org/abs/1904.04052, 2019.

[13] M. Chikina, A. Frieze, and W. Pegden, *Assessing significance in a Markov chain without mixing*, Proc. Natl. Acad. Sci., 114 (2017), pp. 2860–2864, https://doi.org/10.1073/pnas.1617540114.

[14] D. DeFord, *Compactness profiles and reversible sampling methods for plane and graph partitions*, Workshop Event, Duke University, Durham, NC, 2018, https://www.samsi.info/programs-and-activities/research-workshops/quantitative-redistricting/.

[15] D. DeFord and M. Duchin, *Redistricting reform in Virginia: Districting criteria in context*, Virginia Policy Review, 12 (2019), pp. 120–146.

[16] D. DeFord, M. Duchin, and J. Solomon, *Recombination: A family of Markov chains for redistricting*, Harvard Data Sci. Rev., 3 (2021).

[11] A preprint of this work is available [2].

[17] B. Fifield, M. Higgins, K. Imai, and A. Tarr, *A New Automated Redistricting Simulator Using Markov Chain Monte Carlo*, Work. Paper, Princeton University, Princeton, NJ, 2015.
[18] S. Fortunato, *Community detection in graphs*, Phys. Rep., 486 (2010), pp. 75–174.
[19] C. Greenhill, M. Isaev, M. Kwan, and B. D. McKay, *The average number of spanning trees in sparse graphs with given degrees*, Eur. J. Combin., 63 (2017), pp. 6–25, https://doi.org/10.1016/j.ejc.2017.02.003.
[20] B. Hendrickson and T. G. Kolda, *Graph partitioning models for parallel computing*, Parallel Comput., 26 (2000), pp. 1519–1534.
[21] G. Herschlag, H. S. Kang, J. Luo, C. V. Graves, S. Bangia, R. Ravier, and J. C. Mattingly, *Quantifying gerrymandering in North Carolina*, Stat. Public Policy, 7 (2020), pp. 30–38.
[22] G. Herschlag and J. Mattinlgly, *Evaluating the* 2021 *Proposed General Assembly Redistricting Plan: Senate*, 2018–Present, 2021, https://sites.duke.edu/quantifyinggerrymandering/files/2021/11/senateNCGAReport_comparison.pdf.
[23] U. Lauther, *An Extremely Fast, Exact Algorithm for Finding Shortest Paths in Static Networks with Geographical Background*, 2006, pp. 219–230, http://www.gor-ev.de/wp-content/uploads/2016/08/LAUTHER.pdf.
[24] B. H. Junker and F. Schreiber, *Analysis of Biological Networks*, Vol. 2, John Wiley & Sons, 2011.
[25] Y. Y. Liu, W. K. T. Cho, and S. Wang, *Pear: A massively parallel evolutionary computation approach for political redistricting optimization and analysis*, Swarm Evol. Comput., 30 (2016), pp. 78–92.
[26] W. Macmillan, *Redistricting in a GIS environment: An optimisation algorithm using switching-points*, J. Geogr. Syst., 3 (2001), pp. 167–180.
[27] J. C. Mattingly, *Expert Report for Common Cause v. Lewis*, Common Cause v. Lewis, 2019.
[28] J. C. Mattingly and C. Vaughn, *Redistricting and the Will of the People*, preprint, https://arxiv.org/abs/1410.8796, 2014.
[29] MGGG, *Comparison of Districting Plans for the Virginia House of Delegates*, https://mggg.org/VA-report.pdf.
[30] R. H. Möhring, H. Schilling, B. Schütz, D. Wagner, and T. Willhalm, *Partitioning graphs to speedup Dijkstra's algorithm*, J. Exp. Algorithmics (JEA), 11 (2007), pp. 2–8.
[31] L. Najt, D. Deford, and J. Solomon, *Complexity and Geometry of Sampling Connected Graph Partitions*, preprint, https://arxiv.org/abs/1908.08881, 2019.
[32] R. H. Swendsen and J.-S. Wang, *Nonuniversal critical dynamics in monte carlo simulations*, Phys. Rev. Lett., 58 (1987), 86.
[33] L. C. Wu, J. X. Dou, D. Sleator, A. Frieze, and D. Miller, *Impartial Redistricting: A Markov Chain Approach*, preprint, https://arxiv.org/abs/1510.03247, 2015.