A New Automated Redistricting Simulator Using Markov Chain Monte Carlo[*]

Benjamin Fifield[†] Michael Higgins[‡] Kosuke Imai[§]

July 20, 2014

Abstract

Decennial redistricting is a critical element of American representative democracy. A number of substantive scholars have used simulation methods to sample redistricting plans under various constraints in order to assess their impacts on partisanship and other aspects of representation. However, surprisingly few simulation methods exist in the literature, and the standard algorithm has no theoretical justification. To fill this gap, we propose a new automated redistricting simulator using Markov chain Monte Carlo. We formulate redistricting as a graph-cut problem and adopt the Swendsen-Wang algorithm for sampling contiguous districts. We then extend this basic algorithm to incorporate various constraints including equal population and geographical compactness. Finally, we apply simulated tempering and divide-and-conquer approaches to improve the mixing of the resulting Markov chain and scale the algorithm to states with a larger number of districts. The proposed algorithms, therefore, are designed to approximate the population of redistricting plans under various constraints. Through a small-scale validation study, we show that the proposed algorithm outperforms the existing standard algorithm. We also apply the proposed methodology to examine the partisan implications of redistricting under certain conditions in New Hampshire and Mississippi.

Keywords: gerrymandering, graph cuts, Metropolis-Hastings algorithm, simulated tempering, Swendsen-Wang algorithm

[*]We thank seminar participants at Dartmouth, Duke, and Microsoft Research (New York City Lab) for useful comments and suggestions.

[†]Ph.D. Candidate, Department of Politics, Princeton University, Princeton NJ 08544. Email: bfifield@princeton.edu

[‡]Postdoctoral Fellow, Department of Politics, Princeton University, Princeton NJ 08544. Email: mjh5@princeton.edu, URL: `http://www.princeton.edu/~mjh5`

[§]Professor, Department of Politics, Princeton University, Princeton NJ 08544. Phone: 609–258–6601, Email: kimai@princeton.edu, URL: `http://imai.princeton.edu`

1 Introduction

Decennial redistricting is a critical element of American representative democracy. Previous studies have found that redistricting influences turnout and representation (e.g., Abramowitz, 1983; Gelman and King, 1994; Ansolabehere *et al.*, 2000; McCarty *et al.*, 2003; Barreto *et al.*, 2004). From a public policy perspective, redistricting is potentially subject to partisan gerrymandering. After the controversial 2003 redistricting in Texas, for example, Republicans won 21 congressional seats in the 2004 election (Democrats won 11) whereas they had only 15 seats in 2002 (Democrats won 17). To address this concern, numerous remedies, including geographical compactness and partisan symmetry requirements, have been proposed (see Grofman and King, 2007; Fryer and Holden, 2011, and references therein).

The development of automated redistricting algorithms, which is the goal of this paper, began in the 1960s. Vickrey (1961) argued that such an "automatic and impersonal procedure" can eliminate gerrymandering (p. 110). After *Baker v. Carr* (1962) where the Supreme Court ruled that federal courts may review the constitutionality of state legislative apportionment, citizens, policy makers, and scholars became interested in redistricting. Weaver and Hess (1963) and Nagel (1965) were among the earliest attempts to develop automated redistricting algorithms (see also Hess *et al.*, 1965). Since then, a large number of methods have been developed to find an *optimal* redistricting plan for a given set of criteria (e.g., Garfinkel and Nemhauser, 1970; Browdy, 1990; Bozkaya *et al.*, 2003; Chou and Li, 2006; Fryer and Holden, 2011). These optimization methods serve as useful tools when drawing district boundaries (see Altman *et al.*, 2005, for an overview).

However, the main interest of substantive scholars has been to characterize the *distribution* of possible redistricting plans under various criteria for detecting instances of gerrymandering and understanding the causes and consequences of redistricting (e.g., Engstrom and Wildgen, 1977; O'Loughlin, 1982; Cirincione *et al.*, 2000; McCarty *et al.*, 2003; Chen and Rodden, 2013). In 42 of the 50 U.S. states, for example, state politicians control the redistricting process and approve redistricting plans through standard statutory means. Therefore, an important institutional and policy policy question is how to effectively constrain these politicians through means such as compactness requirements (e.g., Niemi *et al.*, 1990), in order to prevent the manipulation of redistricting for partisan ends. Simulation methods allow substantive scholars to answer these questions by approximating distributions of possible electoral outcomes under various institutional constraints.

Yet, surprisingly few simulation algorithms exist in the methodological literature. In fact, most, if not all, of these existing studies use essentially the same Monte Carlo simulation algorithm where a geographical unit is randomly selected as a “seed” for each district and then neighboring units are added to contiguously grow this district until it reaches the pre-specified population threshold (e.g., Cirincione *et al.*, 2000; Chen and Rodden, 2013). Unfortunately, no theoretical justification is given for these existing simulation algorithms, and some of them are best described as ad-hoc. A commonly used algorithm of this type is proposed by Cirincione *et al.* (2000) and implemented by Altman and McDonald (2011) in their open-source software. We hope to improve this state of the methodological literature.

To fulfill this methodological gap, in Section 2, we propose a new automated redistricting simulator using Markov chain Monte Carlo (MCMC). We formulate the task of drawing districting boundaries as the problem of graph-cuts, i.e., partitioning an adjacency graph into several connected subgraphs. We then adopt a version of the Swendsen-Wang algorithm to sample contiguous districts (Swendsen and Wang, 1987; Barbu and Zhu, 2005). We further extend this basic algorithm to incorporate various constraints commonly imposed on redistricting plans, including equal population requirements and geographical compactness. Finally, we apply simulated tempering (Marinari and Parisi, 1992; Geyer and Thompson, 1995) and divide-and-conquer approaches to improve the mixing of the resulting Markov chain and scale the algorithm to states with a larger number of districts. Therefore, unlike the existing algorithms, the proposed algorithms are designed to yield a representative sample of redistricting plans under various constraints.

In Section 3, we conduct a small-scale validation study where all possible redistricting plans under various constraints can be enumerated in a reasonable amount of time. We show that the proposed algorithms successfully approximate this true population distribution while the standard algorithm fails even in this small-scale redistricting problem. We also conduct an empirical study in realistic settings using redistricting and U.S. Census data from New Hampshire and Mississippi. In this case, the computation of the true population distribution is not feasible and so we evaluate the empirical performance of the proposed algorithms by examining several standard diagnostics of MCMC algorithms.

In Section 4, we conduct two kinds of empirical analyses using the New Hampshire and Mississippi data. First, we use the proposed algorithms to sample redistricting plans that are similar to the adapted redistricting plan. We then examine how the magnitude and direction of partisan bias changes as we switch more precincts from one district to another. This analysis reveals the partisan consequences of relatively small changes to the adapted plan. Second, we investigate the partisan implications of imposing a geographical compactness constraint on the redistricting process. We apply the proposed algorithms to study this topic, which is also examined in the literature by McCarty *et al.* (2003) and Chen and Rodden (2013) through the use of the existing simulation methods. Lastly, Section 5 gives concluding remarks.

2 The Proposed Methodology

In this section, we describe the proposed methodology. We begin by formulating redistricting as a graph-cut problem. We then propose a Markov chain Monte Carlo algorithm to uniformly sample redistricting plans with n contiguous districts. Next, we show how to incorporate various constraints such as equal population and geographical compactness. Finally, we scale our algorithm to states with a larger number of districts.

2.1 Redistricting as a Graph-cut Problem

Consider a typical redistricting problem where a state consisting of m geographical units (e.g., census blocks or voting precincts) must be divided into n contiguous districts. We formulate this redistricting problem as that of graph-cut where an adjacency graph is partitioned into a set of connected subgraphs (Altman, 1997; Mehrotra *et al.*, 1998). Formally, let $G = \{V, E\}$ represent an adjacency graph where $V = \{\{1\}, \{2\}, \ldots, \{m\}\}$ is the set of nodes (i.e., geographical units of redistricting) to be partitioned and E is the set of edges connecting neighboring nodes. This means that if two units, $\{i\}$ and $\{j\}$, are contiguous, there is an edge between their corresponding nodes on the graph, $(i, j) \in E$.

Given this setup, redistricting can be seen equivalent to the problem of partitioning an adjacency graph G. Formally, we partition the set of nodes V into n blocks, $\mathbf{v} = \{V_1, V_2, \ldots, V_n\}$ where each block is a non-empty subset of V, and every node in V belongs to one and only one block, i.e., $V_k \cap V_\ell = \emptyset$ and $\bigcup_{k=1}^{n} V_k = V$. Such a partition $\mathbf{v}$ generates an adjacency subgraph $G_{\mathbf{v}} = (V, E_{\mathbf{v}})$ where $E_{\mathbf{v}}$ is a subset of E. Specifically, an edge (i, j) belongs to $E_{\mathbf{v}}$ if and only if $(i, j) \in E$ and nodes $\{i\}$ and $\{j\}$ are contained in the same block of the partition, i.e., $\{i\}, \{j\} \in V_k$. Because $E_{\mathbf{v}}$ is obtained by removing some edges from E or "cutting" them, redistricting represents a graph cut problem. Finally, since each resulting district must be contiguous, a valid partition consists of only connected blocks where for any two nodes $\{i\}$ and $\{j\}$ in a connected block $V_k \in \mathbf{v}$, there exists a path of edges within V_k that joins these two nodes. Formally, there exists a set of nodes $\{\{i\} = \{i_0\}, \{i_1\}, \{i_2\}, \ldots, \{i_{m'-1}\}, \{i_{m'}\} = \{j\}\} \subset V_k$ such that, for all $\ell \in \{1, \ldots, m'\}$, $(i_{\ell-1}, i_\ell) \in E_{\mathbf{v}}$.

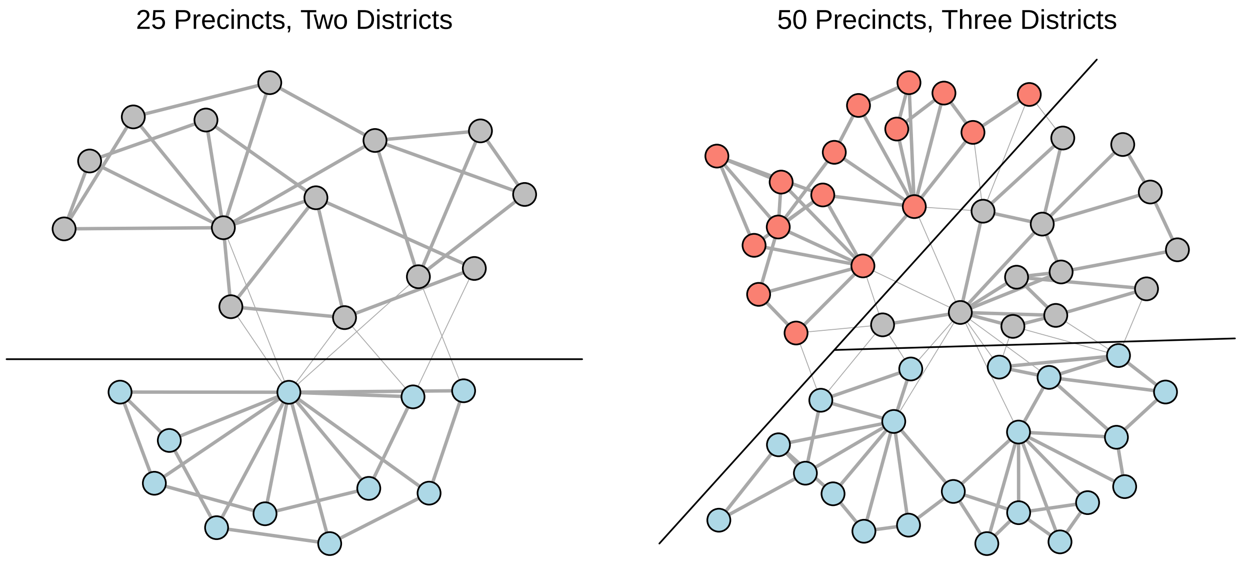


Figure 1: Redistricting as a Graph-cut Problem. A state is represented by an adjacency graph where nodes are geographical units and edges between two nodes imply their contiguity. Under this setting, redistricting is equivalent to removing or cutting some edges (light grey) to form connected subgraphs, which correspond to districts. Different districts are represented by different colors. Two illustrative examples, used in our validation study in Section 3.1, are given here.

Figure 1 presents two illustrative examples used in our validation study in Section 3.1. These examples are taken from actual Florida precinct data in an attempt to create realistic, albeit small, examples. A state is represented by an adjacency graph where nodes are geographical units and edges between two nodes imply their contiguity. The figure demonstrates that redistricting a state into n districts is equivalent to removing some edges of an adjacency graph (light grey) and forming n connected subgraphs.

2.2 The Basic Algorithm for Sampling Contiguous Districts

We propose a new automated simulator to uniformly sample valid redistricting plans with n contiguous districts. The contiguity of valid partitions dramatically increases the difficulty of developing such an algorithm. Intuitive methods for constructing partitions at random – e.g. randomly assigning precincts to districts – have a minuscule chance of yielding contiguous districts, and enumerating all partitions with contiguous districts is too large of a problem to be tractable in realistic redistricting settings. For more discussion, see Section 3.1.

Our algorithm is based on Markov chain Monte Carlo (MCMC) and is designed to obtain a dependent but representative sample from the uniform distribution of valid redistricting plans. In particular, we modify and extend Algorithm 1 of Barbu and Zhu (2005), which combines the Swendsen-Wang algorithm (Swendsen and Wang, 1987) with a Metropolis-Hastings step (Metropolis *et al.*, 1953; Hastings, 1970). This algorithm begins with a valid partition $\mathbf{v}_0$ (e.g., an actual redistricting plan adopted by the state) and transitions from a valid partition $\mathbf{v}_{t-1}$ to another partition $\mathbf{v}_t$ at each iteration t. Here, we describe the basic algorithm for sampling contiguous districts. Later in the paper, we extend this basic algorithm in a couple of important ways so that common constraints imposed on redistricting can be incorporated and the algorithm can be applied to states with a larger number of districts.

Figure 2 illustrates our algorithm using the 50 precinct example with 3 districts given in the right panel of Figure 1. Our algorithm begins by randomly "turning on" edges in $E_{\mathbf{v}_{t-1}}$; each edge is turned on with probability q. In the left upper plot of Figure 2, the edges that are turned on are indicated with darker grey. Next, we identify components that are connected through these "turned-on" edges and are on the boundaries of districts in $\mathbf{v}_{t-1}$. Each such connected component is indicated by a dotted polygon in the right upper plot. Third, among these, a subset of non-adjacent connected components are randomly selected as shown in the left lower plot (two in this case). These connected components are reassigned to adjacent districts to create a candidate partition. Finally, the acceptance probability is computed based on two kinds of edges from each of selected connected components, which are highlighted in the left lower plot: (1) "turned-on" edges, and (2) "turned-off" edges that are connected to adjacent districts. We accept or reject the candidate partition based on this probability.

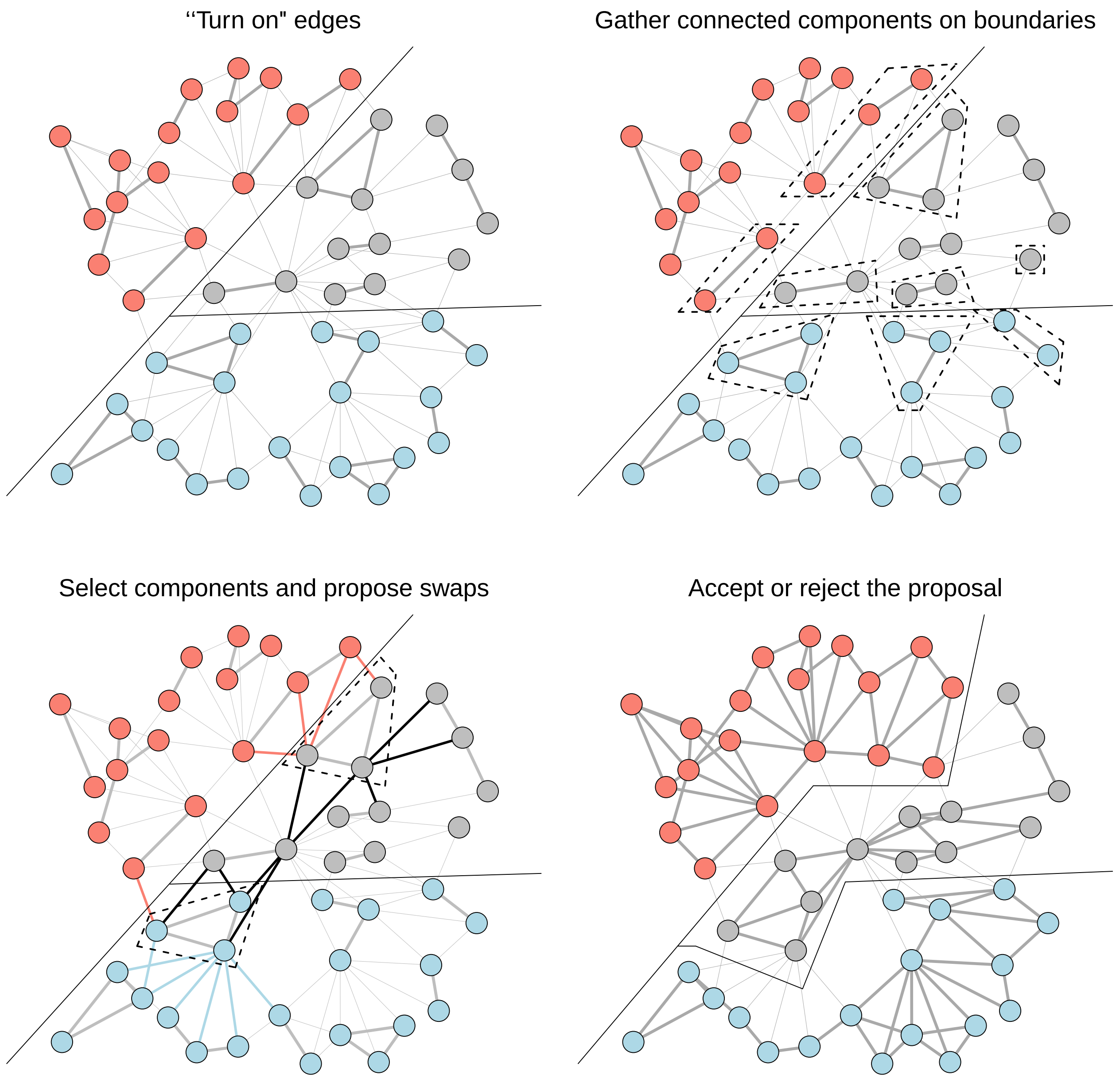
Our algorithm guarantees that its stationary distribution is equal to the uniform distribution of all valid partitions, thereby yielding a uniformly sampled sequence of redistricting plans with contiguous districts. We now formally describe this algorithm.

Algorithm 1 (Sampling Contiguous Redistricting Plans) *We initialize the algorithm by obtaining a valid partition* $\mathbf{v}_0 = \{V_{10}, V_{20}, \ldots, V_{n0}\}$ *and then repeat the following steps at each iteration* t,

Step 1 ("Turn on" edges): *From the partition* $\mathbf{v}_{t-1} = \{V_{1,t-1}, V_{2,t-1}, \ldots, V_{n,t-1}\}$, *obtain the adjacency graph* $G_{\mathbf{v}_{t-1}} = (V, E_{\mathbf{v}_{t-1}})$. *Obtain the edge set* $E^*_{\mathbf{v}_{t-1}} \subset E_{\mathbf{v}_{t-1}}$ *where each edge* $e \in E_{\mathbf{v}_{t-1}}$ *is independently added to* $E^*_{\mathbf{v}_{t-1}}$ *with probability* q.

Step 2 (Gather connected components on boundaries): *Find all components that are connected within* E^*_{t-1} *and adjacent to another block in the partition* $\mathbf{v}_{t-1}$. *Let* C *denote this set of connected components where for all* $C_\ell \in C$, *there exists* $k \in \{1, 2, \ldots, n\}$ *such that* $C_\ell \cap V_{k,t-1} = \emptyset$ *and* $(i, j) \in E$ *for some* $\{i\} \in C_\ell$ *and* $\{j\} \in V_{k,t-1}$.

Step 3 (Select non-adjacent connected components): *Randomly select a set of* r *non-*



‘‘Turn on" edges
Gather connected components on boundaries
Select components and propose swaps
Accept or reject the proposal

Figure 2: The Basic Algorithm for Sampling Contiguous Districts. The plots illustrate the proposed algorithm (Algorithm 1) using the 50 precinct data given in the right panel of Figure 1. First, in the left upper plot, each edge other than those which are cut in Figure 1 is “turned on” (dark grey) independently with certain probability. Second, in the right upper plot, connected components on the boundaries are identified (dashed polygons). Third, in the left lower plot, a certain number of non-adjacent connected components on boundaries are randomly selected (dashed polygons) and the acceptance ratio is calculated by counting certain edges (colored edges). Finally, in the right lower plot, the proposed swap is accepted using the Metropolis-Hastings ratio.

adjacent connected components C^* from C such that $\mathbf{v}_{t-1} \setminus C^*$ is a valid partition where each block of nodes $V_{\ell,t-1} \setminus C^*$ is connected in $G_{\mathbf{v}_{t-1}}$. The sampling is done such that each eligible subset of C is selected with equal probability.

Step 4 (Propose swaps): *Initialize a candidate partition $\mathbf{v}_t^* = (V_{1t}^*, V_{2t}^*, \dots, V_{nt}^*)$ by setting $V_{kt}^* = V_{k,t-1}$. For each component $C_\ell^* \in C^*$ with $\ell \in \{1, \dots, r\}$, find the block $V_{k,t-1} \in \mathbf{v}_{t-1}$ that contains C_ℓ^*, and let $A(C_\ell^*, \mathbf{v}_{t-1})$ denote the set of blocks in $\mathbf{v}_{t-1}$ that are adjacent to C_ℓ^*, not including the block that contains C_ℓ^*. Propose to assign C_ℓ^* from block $V_{k,t-1}$ to an adjacent block $V_{j',t-1}$ with probability $1/|A(C_\ell^*, \mathbf{v}_{t-1})|$. Set $V_{k't}^* = V_{k',t-1} \cup C_\ell^*$ and $V_{kt}^* = V_{k,t-1} \setminus C_\ell^*$. If $V_{kt}^* = \emptyset$, go back to Step 3. Observe that, after each proposed swap, $\mathbf{v}_t^*$ remains a connected set partition.*

Step 5 (Accept or reject the proposal): *Set*

$$\mathbf{v}_t \quad = \quad \begin{cases} \mathbf{v}_t^*, & \text{with probability} \quad \alpha(\mathbf{v}_{t-1} \to \mathbf{v}_t^*), \\ \mathbf{v}_{t-1}, & \text{with probability} \quad 1 - \alpha(\mathbf{v}_{t-1} \to \mathbf{v}_t^*). \end{cases} \tag{1}$$

The acceptance probability is given by

$$\alpha(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) \quad = \quad \min\left(1,\ (1-q)^{|B(C^*, E_{\mathbf{v}_t^*})| - |B(C^*, E_{\mathbf{v}_{t-1}})|}\right) \tag{2}$$

where $B(C^*, E_\mathbf{v}) = \{(i,j) \in E_\mathbf{v} : \exists C_\ell^* \in C^*, C_\ell^* \subset V_k \in \mathbf{v} \text{ s.t.} \{i\} \in C_\ell^*, \{j\} \in V_k \setminus C_\ell^*\}$ denotes the set of edges in $E_\mathbf{v}$ that need to be cut to form connected components C^*.

In Appendix A.1, we prove the following theorem, which states that if the Markov chain produced by the proposed algorithm is ergodic, then the stationary distribution of the chain is uniform on the population of all valid partitions $\Omega(G, n)$ (Tierney, 1994).

Theorem 1 *If every valid partition can be obtained through a sequence of moves given by Algorithm 1, then the stationary distribution of the resulting Markov chain is uniform on all valid partitions.*

The acceptance ratio given in equation (2) is based on the Metropolis-Hastings detailed balance condition (Metropolis *et al.*, 1953; Hastings, 1970),

$$\alpha(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) \quad = \quad \min\left(1,\ \frac{\pi(\mathbf{v}_t^* \to \mathbf{v}_{t-1})}{\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*)}\right), \tag{3}$$

where $\pi(\mathbf{v} \to \mathbf{v}^*)$ denote the probability that, starting from partition $\mathbf{v}$, an iteration of Algorithm 1 described above obtains a candidate partition $\mathbf{v}^*$ through Steps 1–4. Computing numerators and denominators of this ratio separately is combinatorially expensive. However, following Barbu and Zhu (2005), we show in Appendix A.1 that substantial calculation occurs, yielding a simple expression given in equation (2). Indeed, we only need to find all edges within $E_{\mathbf{v}_{t-1}}$ and $E_{\mathbf{v}_t^*}$ that join a node in a connected component of $C^*_\ell \in C^*$ to a node not contained in the block. Since components in C^* are not adjacent, this will ensure that the node not contained in C^*_ℓ will not be contained in a block in C^*.

Several additional remarks are in order. First, when implementing this algorithm, Step 2 requires the three operations: (1) identify all nodes that form a boundary of multiple partitions by comparing $G_{\mathbf{v}_{t-1}}$ with the original adjacency graph G, (2) identify all connected components that include at least one such node via the breadth-first or depth-first search algorithm, and (3) identify the partition to which each connected component belongs.

Second, in Step 3, we typically choose a positive integer r by randomly sampling it from a distribution with $\Pr(r = 1) > 0$ at each iteration. If $r = 1$, then the ergodicity of the Markov chain is guaranteed but the algorithm moves slowly in the sample space. When $r > 1$, the algorithm can mix faster by proposing multiple swaps. However, depending on the adjacency graph G, the algorithm may fail to reach some valid partitions. Thus, we allow r to take a value greater than 1 while keeping the probability of $r = 1$ positive (e.g., a truncated poisson distribution).

Third, in the original algorithm of Barbu and Zhu (2005), r is set to 1 and instead the authors use a small value of q to create larger connected components. This alternative strategy to improving mixing of the algorithm, though sensible in other settings, is not applicable to the current case. The reason is that larger connected components typically include more units from the interior of each block this in turn dramatically lowers the acceptance probability.

Finally, while this basic algorithm yields a sample of redistricting plans with contiguous districts, it does not incorporate common constraints imposed on redistricting process, including equal population, compactness, and political boundaries. In addition, our experience shows that the algorithm does not scale for states with a medium or larger number of districts. Therefore, we now describe two important modifications to the basic algorithm.

2.3 Constraints, Reweighting, and Simulated Tempering

In a typical redistricting process, several additional constraints are imposed. Two most commonly applied constraints are equal population and geographical compactness. We first consider the equal population constraint. Suppose that we use p_i to denote the population size for node $\{i\}$ where the population parity for the state is given by $\bar{p} \equiv \sum_{i=1}^{m} p_i / n$. Then, the population equality constraint can be written as,

$$P_{\mathbf{v}} \;=\; \max_{1 \le k \le n} \left| \frac{\sum_{i \in V_k} p_i}{\bar{p}} - 1 \right| \;\le\; \delta \qquad (4)$$

where δ determines the degree to which one wishes to impose the constraint. For example, $\delta = 0.03$ implies that the population of all districts must be within 3% of the population parity.

Next, we consider the geographical compactness. No consensus exists about the exact meaning of compactness and several alternative definitions have been proposed in the literature (see Niemi *et al.*, 1990). Here, we adopt the measure recently proposed by Fryer and Holden (2011). Let w_i be the population density of node $\{i\}$ and d_{ij} represent the distance between the centroids of nodes $\{i\}$ and $\{j\}$. The measure, which is called the relative proximity index, is based on the sum of squared distances among voters in each district relative to its minimum value. Then, the compactness constraint can be written as,

$$R_{\mathbf{v}} \;=\; \frac{\sum_{k=1}^{n} \sum_{i,j \in V_k, i<j} w_i w_j d_{ij}^2}{\min_{\mathbf{v}' \in \Omega(G,n)} \sum_{k=1}^{n} \sum_{i,j \in V_k', i<j} w_i w_j d_{ij}^2} \;\leq\; \epsilon \tag{5}$$

where $V_k' \in \mathbf{v}'$, ϵ determines the strength of this constraint, and $\Omega(G,n)$ is the set of all redistricting plans with n contiguous districts. Fryer and Holden (2011) develops an approximate algorithm to efficiently compute the minimum of the sum of squared distances, i.e., the denominator of equation (5). The authors also show that this measure is invariant to geographical size, population density, and the number of districts of a state, thereby allowing researchers to compare the index across different states and time periods.

How can we uniformly sample redistricting plans under these additional constraints? One possibility is to discard any candidate partition that does not satisfy the desired constraints. In Algorithm 1, after Step 4, one could check whether the candidate partition $\mathbf{v}_t^*$ satisfies the constraints and if not go back to Step 3. However, such a strategy often dramatically slows down the algorithm and worsens mixing. Alternatively, researchers could run Algorithm 1 without any modification and then simply discard any sampled redistricting plans that do not meet the constraints. The problem of this approach is that many sampled plans may be discarded when strong constraints are imposed.

To overcome this difficulty, we propose to modify Algorithm 1 in the following manner. We first oversample the redistricting plans that are likely to meet the constraints. This means that fewer sampled plans are discarded due to the failure to satisfy the constraints. We then reweight the remaining valid redistricting plans such that they together approximate the uniform sampling from the population of all valid redistricting plans under the constraints. To do this, we consider the following model, which is known as the Potts model in statistical physics,

$$P(\mathbf{v}) \quad = \quad \frac{1}{z(\beta)} \exp\left(\beta \sum_{k=1}^{n} \psi(V_k)\right) \tag{6}$$

where $z(\beta)$ is the normalizing constant. The function $\psi(\cdot)$ is chosen so that it reflects the constraint of interest. For example, we use $\psi(V_k) = |\sum_{i \in V_k} p_i/\bar{p} - 1|$ and $\psi(V_k) = \sum_{i,j \in V_k} w_i w_j d_{ij}^2$ for the equal population and geographical compactness, respectively.

Algorithm 1 can be modified easily to sample from the non-uniform stationary distribution given in equation (6). In particular, we only need to change the acceptance probability in equation (2) of Step 5 to,

$$\alpha(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) \quad = \quad \min\left(1, \ \frac{g_\beta(\mathbf{v}_t^*)}{g_\beta(\mathbf{v}_{t-1})} \cdot q^{|B(C^*,\mathbf{v}_t^*)| - |B(C^*,\mathbf{v}_{t-1})|}\right) \tag{7}$$

where $g_\beta(\mathbf{v}) \equiv \exp\left(\beta \sum_{k=1}^n \psi(V_k)\right)$. Lastly, we reweight the sampled plans by the unnormalized inverse of equation (6) to approximate the uniform sampling from the population of all possible valid redistricting plans. If we resample the sampled plans with replacement using this importance weight, then the procedure is equivalent to the sampling/importance resampling (SIR) algorithm (Rubin, 1987).

One major drawback of this reweighting approach is that when each plan is weighted according to equation (6) the algorithm may have a harder time moving through the sample space. We use simulated tempering to improve the mixing of Algorithm 1 in such situations (Marinari and Parisi, 1992; Geyer and Thompson, 1995). Recall that we want to draw from the distribution given in equation (6). We initialize a sequence of *temperatures* $\{\beta^{(\ell)}\}_{\ell=0}^r$ where $\beta^{(0)}$ is called the *cold temperature*, which is the target parameter value for inference, and $\beta^{(r)} = 0$ is the *hot temperature* with $|\beta^{(0)}| > |\beta^{(1)}| > \cdots > |\beta^{(r)}| = 0$ and $\operatorname{sgn}(\beta^{(0)}) = \operatorname{sgn}(\beta^{(1)}) = \cdots = \operatorname{sgn}(\beta^{(r-1)})$. After many iterations, we keep the MCMC draws obtained when $\beta = \beta^{(0)}$ and discard the rest. By sampling under warm temperatures, simulated tempering allows for greater exploration of the target distribution. We then reweight the draws by the importance weight $1/g_{\beta^{(0)}}(\mathbf{v})$.

Specifically, we perform simulated tempering in two steps. First, we run an iteration of Algorithm 1 using the modified acceptance probability with $\beta = \beta^{(l)}$. We then make another Metropolis-Hastings decision on whether to change to a different value of β. The details of the algorithm are given below.

Algorithm 2 (Simulated Tempering) *Given the initial valid partition* $\mathbf{v}_0$ *and the initial temperature value* $\beta_0 = \beta^{(\kappa_0)}$ *with* $\kappa_0 = r$*, the simulated tempering algorithm repeats the following steps at each iteration* t*,*

Step 1 (Run the basic algorithm with the modified acceptance probability): *Using the current partition $\mathbf{v}_{t-1}$ and the current temperature $\beta_{t-1} = \beta^{(\kappa_{t-1})}$, obtain a valid partition $\mathbf{v}_t$ by running one iteration of Algorithm 1 with the acceptance probability given in equation (7).*

Step 2 (Choose a candidate temperature): *We set $\kappa_t^* = \kappa_{t-1} - 1$ with probability $u(\kappa_{t-1}, \kappa_{t-1}-1)$ and set $\kappa_t^* = \kappa_{t-1}+1$ with probability $u(\kappa_{t-1}, \kappa_{t-1}+1) = 1-u(\kappa_{t-1}, \kappa_{t-1}-1)$ where $u(\kappa_{t-1}, \kappa_{t-1} - 1) = u(\kappa_{t-1}, \kappa_{t-1} + 1) = 1/2$ when $1 \leq \kappa_{t-1} \leq r - 1$, and $u(r, r-1) = u(0,1) = 1, u(r, r+1) = u(0, -1) = 0$.*

Step 3 (Accept or reject the candidate temperature): *Set*

$$\kappa_t = \begin{cases} \kappa_t^*, & \text{with probability } \gamma(\kappa_{t-1} \to \kappa_t^*), \\ \kappa_{t-1}, & \text{with probability } 1 - \gamma(\kappa_{t-1} \to \kappa_t^*) \end{cases} \tag{8}$$

where

$$\gamma(\kappa_{t-1} \to \kappa_t^*) = \min\left(1, \frac{g_{\beta^{(\kappa_t^*)}}(\mathbf{v}_t)\ u(\kappa_t^*, \kappa_{t-1})\ w_{\kappa_t^*}}{g_{\beta^{(\kappa_{t-1})}}(\mathbf{v}_t)\ u(\kappa_{t-1}, \kappa_t^*)\ w_{\kappa_{t-1}}}\right) \tag{9}$$

where w_ℓ is an optional weight given to each $l \in \{0, 1, \ldots, r\}$.

2.4 Divide and Conquer

While they can, in theory, handle redistricting with an arbitrary number of districts, in practice Algorithm 1 and 2 introduced above do not easily scale to a state with a medium or large number of districts. On the other hand, the algorithms appear to efficiently sample redistricting plans with two contiguous districts. Therefore, we use a divide-and-conquer strategy when sampling redistricting plans with more than two districts. Specifically, at each iteration, we randomly choose a pair of contiguous districts and then run Algorithm 1 or 2 for a reasonably large number of iterations within these two districts to obtain a new candidate two-block partition. This candidate partition is then accepted with appropriate probability. The details of this new algorithm is given here.

Algorithm 3 (Divide and Conquer) *The algorithm begins with an initial valid partition $\mathbf{v}_0$ and repeats the following steps at each iteration t,*

Step 1 (Select an adjacent pair of blocks): *From the current partition $\mathbf{v}_{t-1}$, find all adjacent pairs of blocks:*

$$\mathbf{P}_{\mathbf{v}_{t-1}} = \{\{V_{k,t-1}, V_{\ell,t-1}\} : \exists\, (i,j) \in E_{\mathbf{v}_{t-1}} \text{ where } i \in V_{k,t-1}, j \in V_{\ell,t-1}\}. \tag{10}$$

Randomly choose a pair $\{V_{k',t-1}, V_{\ell',t-1}\} \in \mathbf{P}_{\mathbf{v}_{t-1}}$ for $k' < \ell'$ with probability $1/|\mathbf{P}_{\mathbf{v}_{t-1}}|$.

Step 2 (Run the basic algorithm for a large number of iterations): *Run Algorithm 1 or 2 for a sufficiently large number of iterations on* $V_{k',t-1} \cup V_{\ell',t-1}$ *(possibly with simulated tempering) starting from the initial partition* $\{V_{k',t-1}, V_{\ell',t-1}\}$. *Let* $\{V^*_{k't}, V^*_{\ell't}\}$ *denote the candidate partition of* $V_{k',t-1} \cup V_{\ell',t-1}$ *obtained at the end of the iterations. Then, the candidate partition is given by,*

$$\mathbf{v}^*_t \quad = \quad (V_{1,t-1}, V_{2,t-1}, \ldots, V^*_{k't}, \ldots, V^*_{\ell't}, \ldots, V_{n,t-1}). \tag{11}$$

Step 3 (Accept or reject the candidate partition): *Set*

$$\mathbf{v}_t = \begin{cases} \mathbf{v}^*_t, & \textit{with probability} \quad \lambda(\mathbf{v}_{t-1} \to \mathbf{v}^*_t), \\ \mathbf{v}_{t-1}, & \textit{with probability} \quad 1 - \lambda(\mathbf{v}_{t-1} \to \mathbf{v}^*_t), \end{cases} \tag{12}$$

where the acceptance probability is given by,

$$\lambda(\mathbf{v}_{t-1} \to \mathbf{v}^*_t) \quad = \quad \min\left(1, \frac{|\mathbf{P}_{\mathbf{v}_{t-1}}|}{|\mathbf{P}_{\mathbf{v}^*_t}|}\right). \tag{13}$$

Notice that the acceptance probability given in equation (13) can be computed by simply counting the number of all adjacent pairs of districts in each partition. In Appendix A.2, we prove the following theorem, which states that our divide-and-conquer algorithm yields a Markov chain whose stationary distribution is uniform on all valid partitions.

Theorem 2 *If every valid partition can be obtained through a sequence of moves given by Algorithm 3, then the stationary distribution of the Markov chain is approximately uniform on all valid partitions.*

In addition, the divide-and-conquer algorithm, as it is written, assumes that all adjacent pairs of blocks contain at least one valid proposal. For a small-scale redistricting problem such as the one considered in Section 3.1, however, it is possible to have a pair of adjacent blocks which has no valid swap proposal (e.g., any swap can either eliminate or shatter a district). problems. In such cases, one must only consider the pairs of adjacent blocks that have valid proposals. Fortunately, this scenario is extremely unlikely in more realistic redistricting problems.

The divide-and-conquer strategy described here can also incorporate the common constraints of redistricting process described in Section 2.3. In particular, we consider the model given in equation (6). In Appendix A.2, we also prove the generalization of Theorem 2 and show that the divide-and-conquer algorithm requires no modification even when the target distribution is non-uniform and is equal to the model given in equation (6).

2.5 Comparison with the Existing Algorithms

A number of substantive researchers used Monte Carlo simulation algorithms to sample possible redistricting plans under various criteria in order to detect the instances of gerrymandering and understand the causes and consequences of redistricting (e.g., Engstrom and Wildgen, 1977; O'Loughlin, 1982; Cirincione *et al.*, 2000; McCarty *et al.*, 2003; Chen and Rodden, 2013). Most of these studies use a similar Monte Carlo simulation algorithm where a geographical unit is randomly selected as a "seed" for each district and then neighboring units are added to contiguously grow this district until it reaches the pre-specified population threshold. A representative of such algorithms, proposed by Cirincione *et al.* (2000) and implemented by Altman and McDonald (2011) in their open-source `BARD` package, is given here.

Algorithm 4 (The Standard Redistricting Simulator (Cirincione *et al.*, 2000)) *For each district, we repeat the following steps.*

Step 1: *From the set of unassigned units, randomly select the seed unit of the district.*

Step 2: *Identify all unassigned units adjacent to the district.*

Step 3: *Randomly select one of the adjacent units and add it to the district.*

Step 4: *Repeat Steps 2 and 3 until the district reaches the predetermined population threshold.*

Additional criteria can be incorporated into this algorithm by modifying Step 3 to select certain units. For example, to improve the compactness of the resulting districts, one may choose an adjacent unassigned unit that falls entirely within the minimum bounding rectangle of the emerging district. Alternatively, an adjacent unassigned unit that is the closest to emerging district can be selected (see Chen and Rodden, 2013).

Nevertheless, the major problem of these simulation algorithms is their adhoc nature. For example, as the documentation of `BARD` package warns, the creation of earlier districts may make it impossible to yield contiguous districts. More importantly, the algorithms come with no theoretical result and are not even designed to uniformly sample redistricting plans even though researchers have a tendency to assume that they are. In contrast, the proposed algorithms described in Sections 2.2–2.4 are built upon the well-known theories and strategies developed in the literature on the Markov chain Monte Carlo methods. The disadvantage of our algorithms, however, is that they yield a dependent sample and hence their performance will hinge upon the degree of mixing. Thus, we now turn to the assessment of the empirical performance of the proposed algorithms.

3 Evaluating the Performance of the Proposed Algorithms

In this section, we assess the performance of the proposed algorithms in two ways. First, we conduct a small-scale validation study where, due to its size, all possible redistricting maps can be enumerated in a reasonable amount of time. We show that our algorithms can approximate the target distribution well when the standard algorithm commonly used in the literature fails. Second, we use the actual redistricting data to examine the convergence behavior of the proposed algorithms in more realistic settings using the actual redistricting data from New Hampshire (two districts) and Mississippi (four districts). For these data, the computation of the true population distribution is not feasible. Instead, we evaluate the empirical performance of the proposed algorithms by examining the standard diagnostics of MCMC algorithms.

To conduct these analyses, we integrate precinct-level data from two sources. We utilize precinct-level shape files and electoral returns data from the Harvard Election Data Archive to determine precinct adjacency and voting behavior. We supplement this data with basic demographic information from U.S. Census Bureau P.L. 94-171 summary files, which are compiled by the Census Bureau and disseminated to the 50 states in order to obtain population parity in decennial redistricting.

3.1 A Small-scale Validation Study

We conduct a validation study where we analyze the convergence of our algorithm to the target distribution on the 25 and 50 precinct sets shown as adjacency graphs in Figure 1. Due to the small size of these sets, all possible redistricting plans can be enumerated in a reasonable amount of time. We begin by considering the problem of partitioning each of these graphs into two districts. We apply the proposed algorithm (Algorithm 1) with the starting map obtained randomly by running the standard algorithm (Algorithm 4) once. In addition, we apply the standard algorithm (Algorithm 4), which is implemented through the `BARD` package (Altman and McDonald, 2011), to compare its performance with that of our proposed algorithm. We then consider partitions of the 25 precinct set into three districts. The results of the proposed algorithm are based on a single chain of 10,000 draws while those of the standard algorithm are based on the same number of independent draws.

Before we give results, it should be noted that, even for this small-scale study, the enumeration of all valid partitions is a non-trivial problem. In the case of partitioning 50 precincts into two districts, of the $(2^{50} - 2)/2 \approx 5.63 \times 10^{14}$ possible partitions, only $1,773,447$ form two contiguous districts, and only $620,434$ of these partitions have districts that have an equal population constraint within 20% of parity. For partitions of 25 precincts into three districts, of the roughly $3^{25}/6 \approx 1.41 \times 10^{11}$ possible partitions, $82,623$ have three contiguous districts, and $3,617$ have district populations within 20% of parity. Hence, enumerating all possible partitions to obtain those that are valid is intractable.

A brief description of our enumeration algorithm is as follows. In the case of two districts, we choose an initial starting node and form a partition where one district is that initial node and the other district is the complement, provided the complement is connected. We then form connected components of two nodes comprised of that starting node and and nodes that are adjacent to that node. We identify all valid partitions where one district is a two-node component and the other district is the complement of the component. We continue forming connected components of incrementally increasing sizes and finding valid partitions until all possible partitions are found. In the case of three precincts, if the complement of a connected component is comprised of two additional connected components, we store that partition as valid. If the complement is a single connected component, we apply the two-district algorithm on the complement. After this enumeration, we identify which partitions have districts with populations within a certain percentage of parity.

Figure 3 presents the results of the validation study with two districts. The upper (bottom) row presents the analysis of the 25 (50) precinct sets, and the left (right) column displays the results without (with) the equal population constraint where up to 20% deviation from the parity is imposed. These plots show that, using the Republican dissimilarity index as a statistic, the proposed algorithm (Algorithm 1; black solid lines) can provide a good approximation of true population distribution of valid redistricting maps (grey histograms) while the standard algorithm (Algorithm 4; red dashed lines) may not. These results hold for both the 25 and 50 precinct sets as well as with and without the equal population constraint. In contrast, we find that the performance of the standard algorithm deteriorates as we impose the equal population constraint.

Figure 4 presents the results of the validation study with three districts and 25 precincts; enumerating all valid three-district redistricting plans for the 50 precinct set is currently too computationally difficult to be tractable. As before, we apply the proposed algorithm (Algorithm 1) with the starting map obtained randomly from the standard algorithm (Algorithm 4) and examine the simulated tempering algorithm (Algorithm 2). These algorithms are also implemented with (the lower panel) and without (the upper panel) the divide-and-conquer strategy (Algorithm 3).

The plots of the left column show that when no constraint is imposed the proposed algorithm,

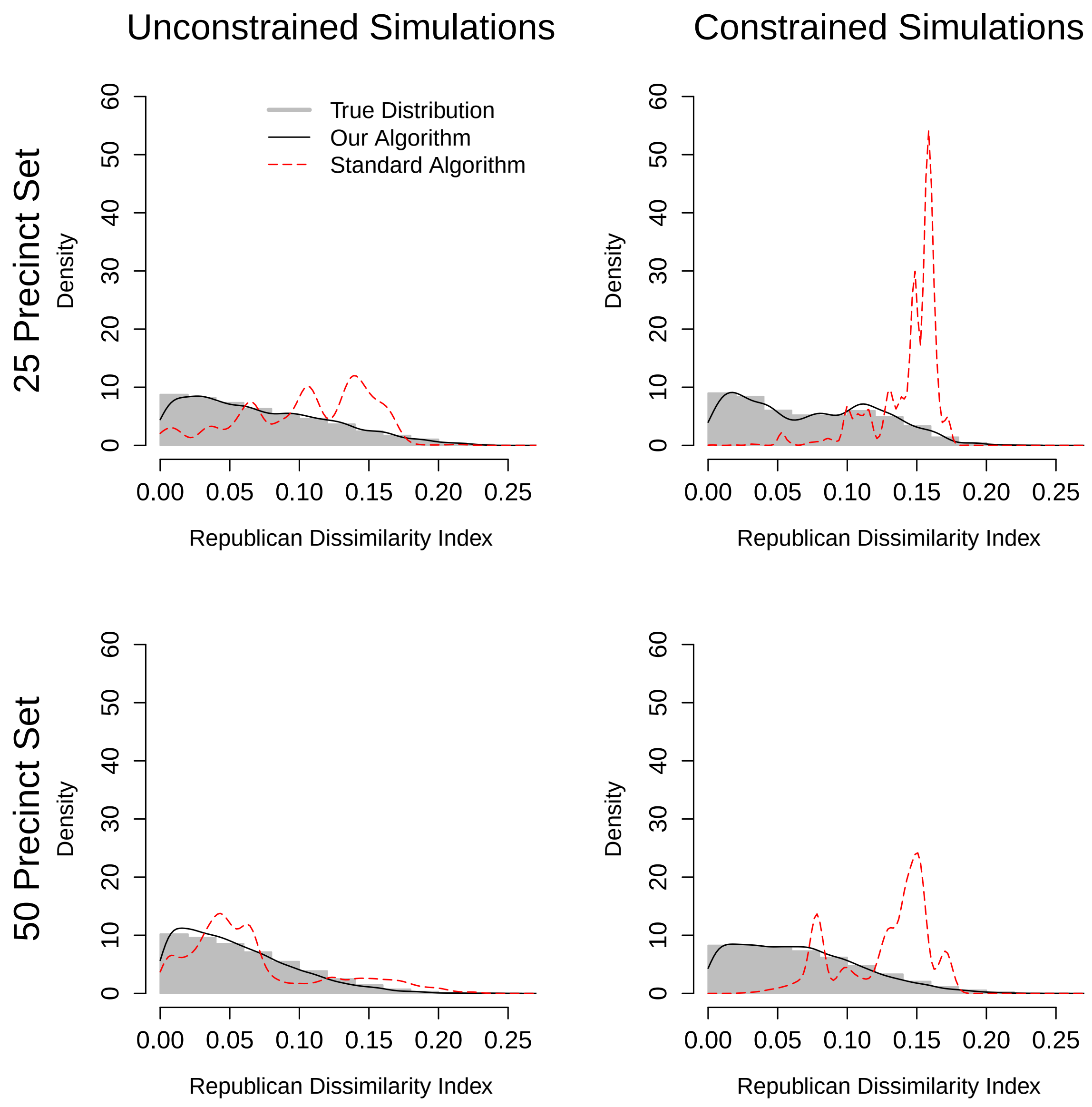


Figure 3: A Small-scale Validation Study with Two Districts. The plots show that the proposed algorithm (Algorithm 1; solid black lines) approximates well the true population distribution (grey histograms)) while the standard algorithm (Algorithm 4; red dashed lines) fails. The upper (bottom) row presents the results of the 25 (50) precinct set with two districts. The graph representation of these precinct sets are given in Figure 1 though the 50 precinct set in this validation study also has two districts rather than three. The left (right) column presents the redistricting results without (with) a population equality constraint where up to 20% deviation from the parity is imposed. The standard algorithm has a harder time, approximating the target distribution when the population constraint is imposed. The result for each algorithm is based on a single chain of 10,000 draws.

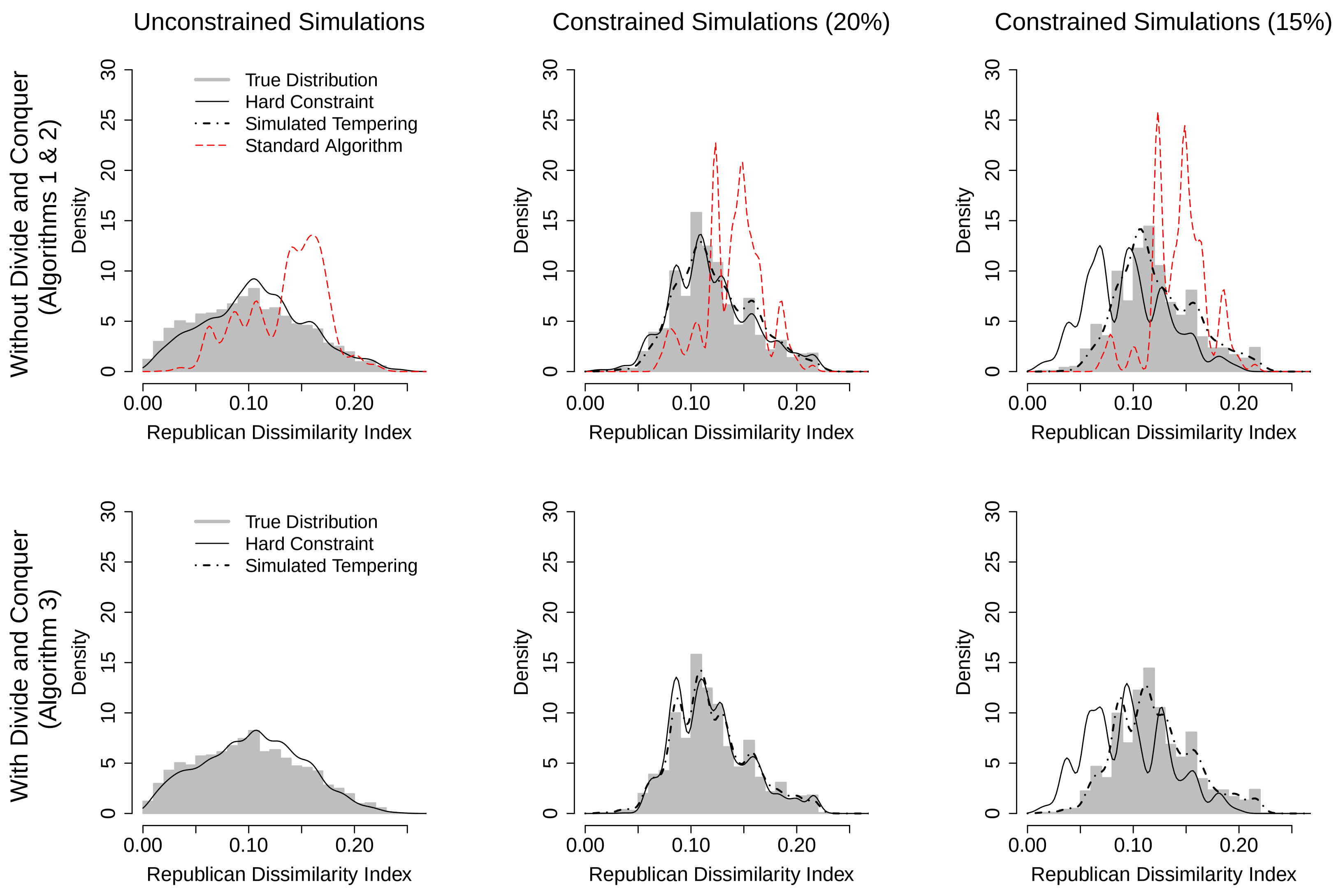


Figure 4: A Small-scale Validation Study with Three Districts. The underlying data is the 25 precinct set shown in the left plot of Figure 1. The plots in the first row show that the proposed algorithm (Algorithm 1; solid black lines) approximates well the true population distribution (grey histograms)) when no (left plot) or weak (middle plot) equal population constraint is imposed. However, the algorithm exhibits poor performance when a stronger equal population constraint (right plot) is imposed. In contrast, the proposed algorithm with simulated tempering (Algorithm 2; black dot-dashed line) approximates the true population distribution well even when a stronger constraint is placed. Finally, the standard algorithm (Algorithm 4; red dashed lines) fails to approximate the target distribution in all cases. In the plots of the second row, the same exact pattern is observed for the divide-and-conquer algorithms (Algorithm 3; black solid and dot-dashed lines). The results for each algorithm is based on a single chain of 10,000 draws.

with or without divide-and-conquer, approximates the target distribution well while the sample from the standard algorithm is far from being representative of the population. This finding is consistent with the results in Figure 3. In the plots of the middle and right columns, we impose the equal population constraint where only up to 20% and 15% deviation from the population parity is allowed, respectively.

We also display the results from the proposed algorithm with the simulated tempering algorithm (Algorithm 2; black dot-dashed lines). To implement this algorithm, we specify a sequence of temperatures $\{\beta^{(\ell)}\}_{\ell=0}^{r}$, where $\beta^{(0)} = 0$, and $\beta^{(r)} = -20$. After a preliminary analysis, we chose a target temperature of $\beta^{(\ell)} = -4$ for the target deviations of 20% and 15%. We choose this setup so that the rejection ratio is in the recommended 20–40% range (Geyer and Thompson, 1995) and the target temperature value is chosen based on the number of plans that meet the population constraint. The results are not particularly sensitive to this choice. For example, any target temperature between -2 and -8 can approximate the underlying distribution well. In both cases, we use a subset of draws taken under the target temperature. We then resample the remaining draws using the importance weights $1/g_{\beta^{(\ell)}}(\mathbf{v})$, and finally subset down to the set of remaining draws that fall within the population target. As in the plots of the left column, we present the results with (bottom panel) and without (upper panel) divide-and-conquer (Algorithm 3; black solid and dot-dashed lines).

We find that when a weak constraint, i.e., 20% equal population constraint, is imposed (in the middle plot), the proposed algorithms with and without simulated tempering approximates the true population distribution well. However, when a stronger constraint, i.e., 15%, is placed, the proposed algorithm with simulated tempering (Algorithm 2) maintains a good approximation whereas the proposed algorithm without it (Algorithm 1) fails to yield a representative sample. Finally, the standard algorithm (Algorithm 4; red dashed lines) continues to exhibit poor performance. The same pattern is observed with divide-and-conquer where the algorithm with simulated tempering performs well even when the strong constraint is imposed.

Finally, Figure 5 compares the runtime between the proposed basic algorithm (Algorithm 1; solid black lines) and the standard algorithm (Algorithm 4; red dashed lines) under various validation study settings. Each algorithm is run until it yields 10,000 draws using a node on a Linux server with 2.66 GHz Nehalem processors and 3GB RAM (no parallel computing is used). We find that under all settings we consider here the runtime for the proposed algorithm is at least 50 times shorter than that for the standard algorithm. This difference increases as the number of

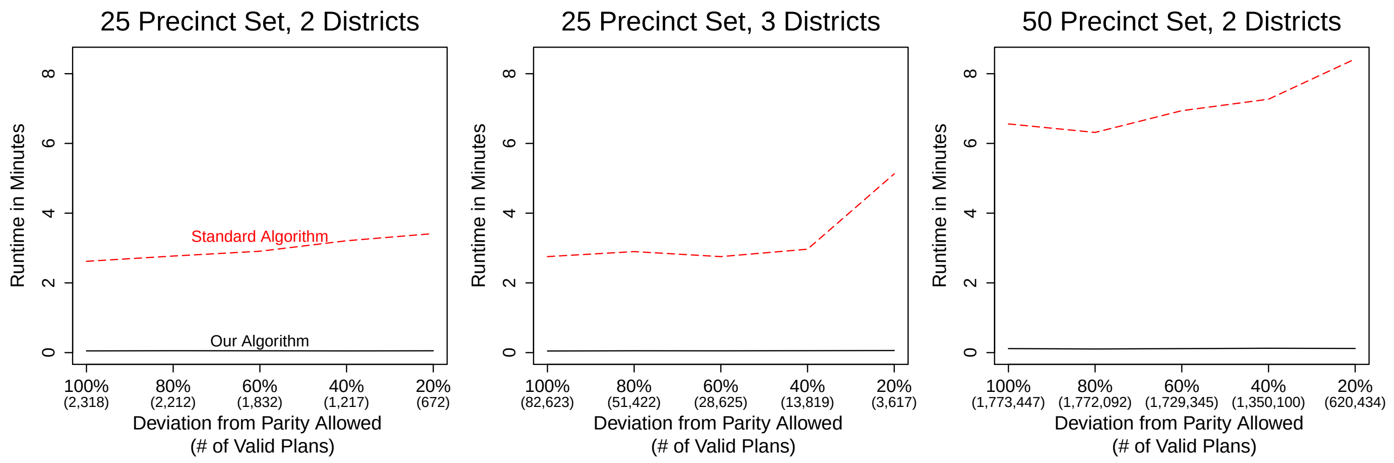


Figure 5: Runtime Comparison between the Proposed and Standard Algorithms in the Small-scale Validation Study. The runtime is compared between the proposed basic algorithm (Algorithm 1; solid black lines) and the standard algorithm (Algorithm 4; red dashed lines) under various settings. Each algorithm is run until it yields 10,000 draws. The runtime is much greater for the standard algorithm than the proposed algorithm. It also increases much more quickly for the former as the number of precincts and the strength of equal population constraint increase.

precincts and the strength of equal population constraint (x-axis) increases. In sum, in terms of computational speed, the proposed algorithm scales much better than the standard algorithm.

3.2 An Empirical Study

The scale of the validation study presented above is small so that we can enumerate all possible redistricting plans in a reasonable amount of time. This allowed us to examine how well each algorithm is able to approximate the true population distribution. However, the scale of the study is too small to be realistic. Below, we apply the proposed algorithms to the 2008 election data and conduct standard convergence diagnostics of MCMC algorithms. While we cannot compare the distribution of sampled maps with the true population distribution, this empirical study enables us to investigate the performance of the proposed methods in realistic settings.

We first consider New Hampshire. The state has two congressional districts and consists of 327 precincts, and so this is one of the simplest realistic redistricting problems. The left panel of Figure 6 shows the implemented statewide redistricting plan as of 2010. Under this plan, Democrats and Republicans won a single congressional seat each. In 2008, Obama won 54% of votes in this state while his 2012 voteshare was 52%. Redistricting in New Hampshire is determined by its state legislature and plans are passed as standard statutes, which makes them subject to gubernatorial veto. We apply the proposed basic algorithm (Algorithm 1) and simulated tempering algorithm

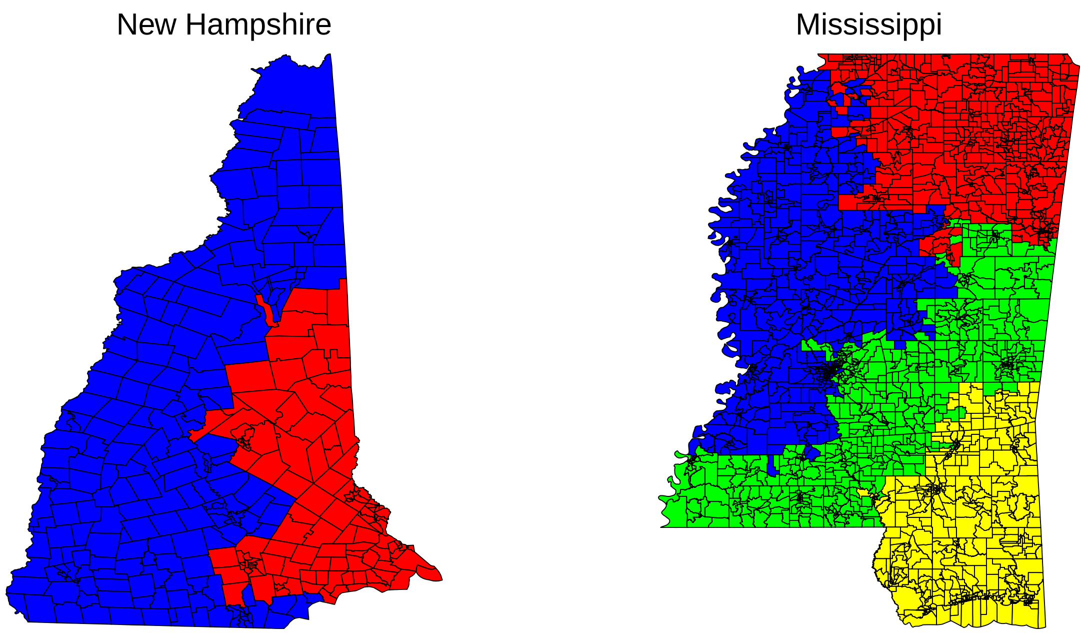


Figure 6: Precinct-level Maps of New Hampshire (327 precincts, two congressional districts) and Mississippi (1,969 precincts, four congressional districts). Colors correspond to precinct congressional district assignments in 2010. In New Hampshire, Democrats and Republicans each hold a single congressional seat. In Mississippi, Republicans hold three congressional seats while Democrats hold a single seat.

(Algorithm 2). The target population consists of all redistricting plans with contiguous districts and a maximum of 1% deviation from the population parity.

A total of 10 chains are run until 500,000 draws are obtained for each algorithm. Inference is based on a total of 91,630 draws, which is the number of draws with simulated tempering that both satisfy the population constraint and were drawn under the target temperature value, $\beta^{(\ell)} = -30$. For starting values, we use independent draws from the standard algorithm (Algorithm 4 as implemented in the `BARD` package). For the simulated tempering algorithm, after some preliminary analysis, we have decided to allow $\beta^{(\ell)}$ to take values between 0 and -100, in increments of 10, with the target temperature value of -30. As in the small-scale verification study, we only use draws taken under the target temperature, and then reweight according to the importance weights $1/g_{\beta^{(\ell)}}(\mathbf{v})$ before selecting all remaining draws that fall within the target parity deviation of 1%.

Figure 7 presents the results. The figure shows the autocorrelation plots (left column), the trace plots (middle column), and the Gelman-Rubin potential scale reduction factors (Gelman and Rubin, 1992; right column) for the basic algorithm (top panel) and the simulated tempering algorithm (bottom panel). We use the logit transformed Republican dissimilarity index for all

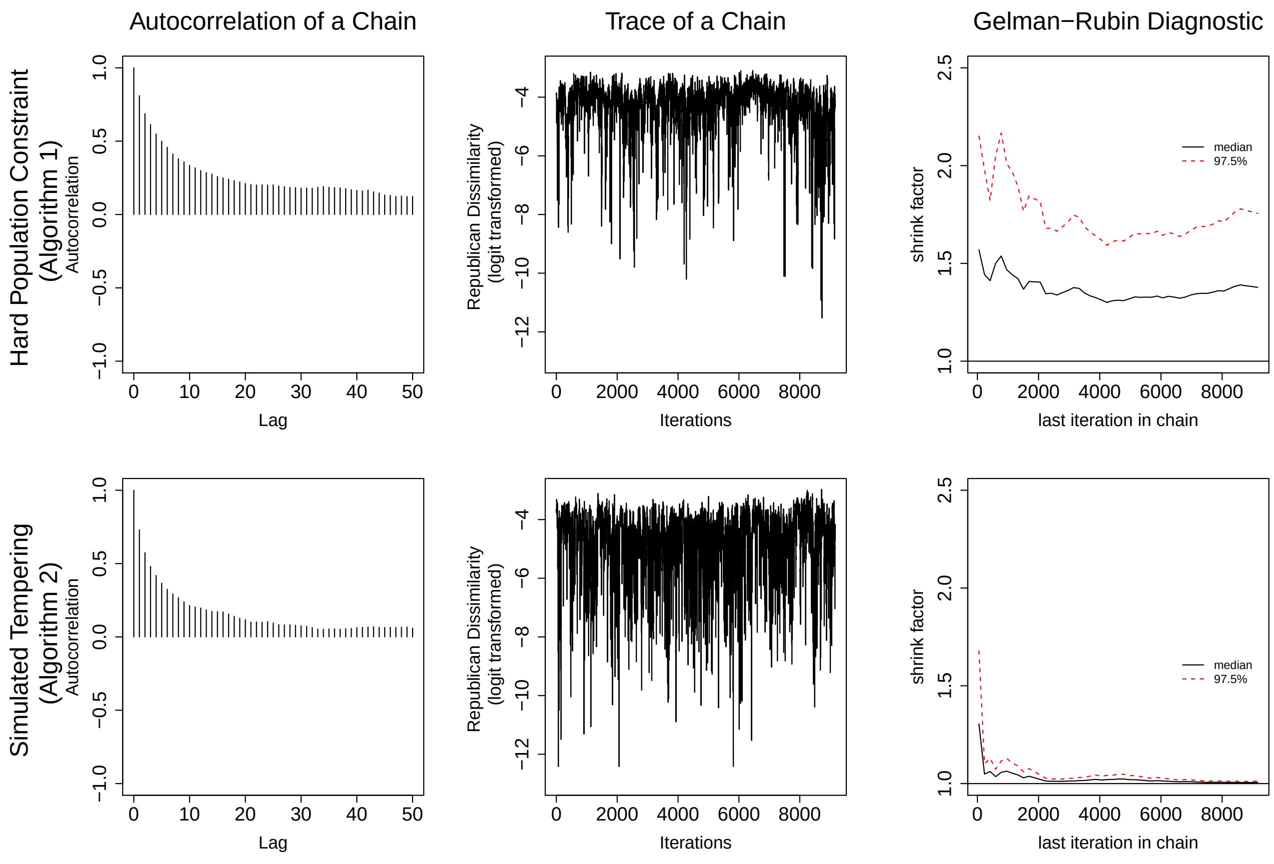


Figure 7: Convergence Diagnostics of the Proposed Algorithm for the 2008 New Hampshire Redistricting Data. The proposed basic algorithm (Algorithm 1; top panel) and simulated tempering algorithm (Algorithm 2; bottom panel) are applied to the New Hampshire data with 327 precincts and 2 congressional districts. The target population consists of all redistricting plans with contiguous districts and a maximum of 1% deviation from the population parity. A total of 10 chains are run with different starting maps for each algorithm until 500,000 draws are obtained, and inference is based on a total of 91,630 draws (the number of draws in the simulated tempering algorithm that are both drawn under the target temperature and satisfy the target population constraint). For the logit transformed Republican dissimilarity index, the autocorrelation plots (left column), the trace plots (middle column), and the Gelman-Rubin potential scale reduction factors (right column) are presented. The simulated tempering algorithm outperforms the basic algorithm across all three diagnostics.

diagnostics. The simulated tempering algorithm outperforms the basic algorithm. The former has a lower autocorrelation and mixes better. In addition, the potential scale reduction factor goes down quickly, suggesting that all the chains with different starting maps become indistinguishable from each other after approximately 2,000 draws.

Next, we analyze the 2008 election data from Mississippi. This state has a total of four congressional districts and 1,969 precincts, thereby providing a more challenging example when compared to New Hampshire. The right-hand panel of Figure 6 shows the implemented redistricting plan in Mississippi as of 2010. In 2008, 43% of the electorate voted for Obama while his voteshare in the 2012 election for this state was 44%. Redistricting in Mississippi is determined by its state legislature subject to gubernatorial veto. One important feature of Mississippi is its sizable African-American population, which consists of 37% of the population. This group is concentrated in the capitol city, Jackson, and in surrounding areas in the west of the state, which poses a special challenge to the algorithms. Democrats typically win this seat, shaded in blue in Figure 6, while Republicans typically win the other three seats in Mississippi. Mississippi is also one of the nine states fully covered by Section V of the Voting Rights Act, which obligates political officials to submit its proposed redistricting plan to the U.S. Department of Justice. However, following the Supreme Court's decision in *Shelby County v. Holder* (2013) to strike down the pre-clearance formula determining Section V coverage, Mississippi is no longer subject to Section V requirements by default.

We examine how the divide and conquer algorithm (Algorithm 3) performs in a state with more than two districts. For the divide and conquer algorithm, we run the standard algorithm for 2,000 iterations within a pair of randomly selected adjacent districts. A total of 10 chains are run using the independent draws from the standard algorithm (Algorithm 4) as starting values.

Figure 8 presents the results of this analysis based on a preliminary set of 2,000 draws (more draws are forthcoming). The same set of diagnostics are conducted for the Republican dissimilarity index (top row) and the African-American dissimilarity index (bottom row). The figure shows that although the Mississippi data pose a much more challenging application than the New Hampshire data, the divide-and-conquer algorithm still performs respectably. In particular, while the impact of starting values remains, the potential scale reduction factor (in the plots given in the right column) is low and continues to decrease, even after only 2,000 simulations. Because African American voters are geographically concentrated, the algorithm has a particularly hard time mixing for the African-American dissimilarity index, which is reflected in the scale reduction factor for African-American dissimilarity.

In the future, we hope to combine the divide-and-conquer strategy with the simulated tempering method to further improve the performance of the algorithm.

4 Empirical Analyses

In this section, we answer some substantive questions through empirical analyses of the election data from New Hampshire and Mississippi introduced in Section 3.2. First, we conduct "local exploration" around the actual districts by sampling redistricting plans that are similar to the

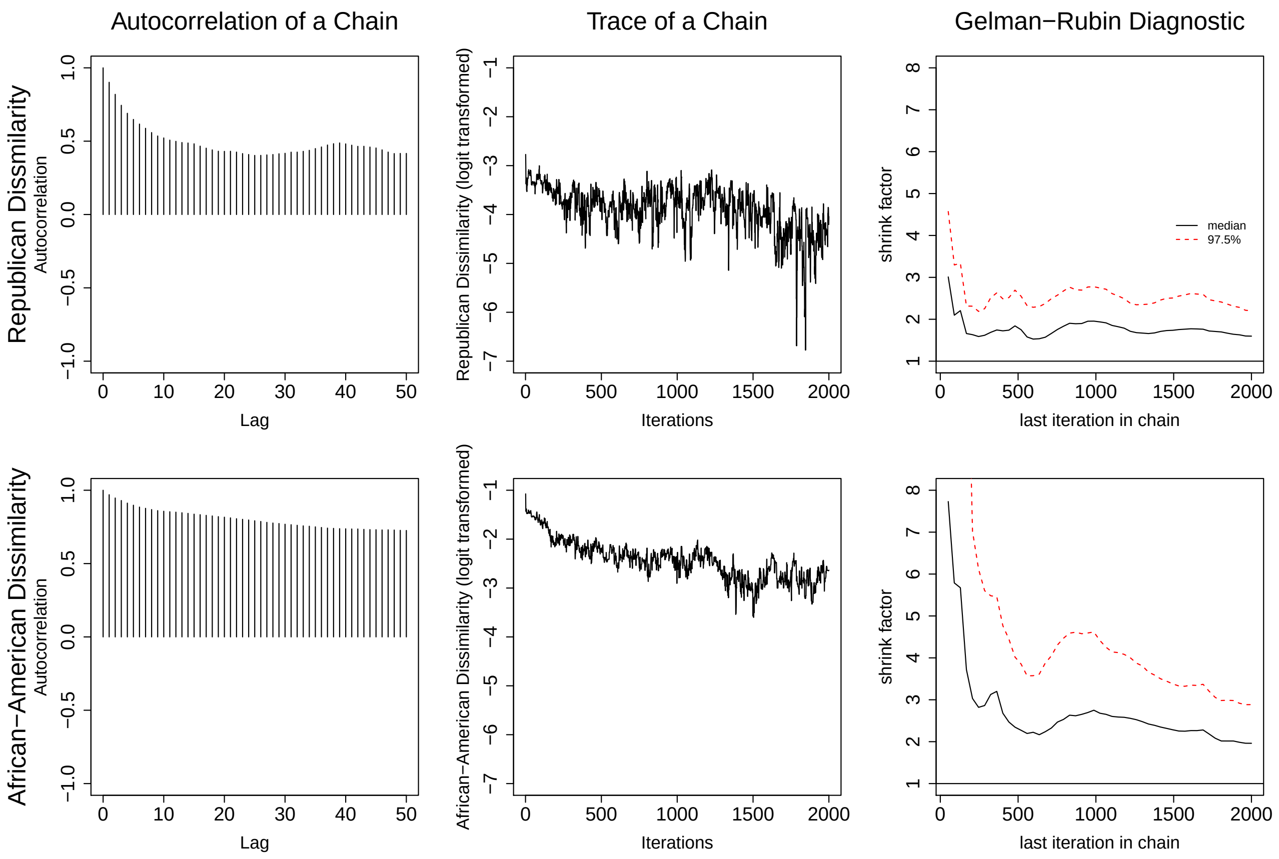


Figure 8: Convergence Diagnostics of the Proposed Algorithm for the 2008 Mississippi Redistricting Data. The information identical to that of Figure 7 is displayed here for two statistics, Republican dissimilarity index and African-American dissimilarity index (both logit transformed). See the caption of Figure 7 for details. The data is obtained from 2,000 simulations of the divide-and-conquer algorithm (Algorithm 3), where each draw is obtained from running the standard algorithm 2,000 time within an adjacent pair of congressional districts.

adapted plan. We then investigate the degree and direction of partisan bias that can be introduced by varying degrees of changes to the adapted plan. Next, we conduct "global exploration" by letting the algorithms sample redistricting maps from the distribution of all possible plans that meet an equal population constraint. Finally, we examine the partisan implications of imposing a geographical compactness requirement in each state.

4.1 Sampling Redistricting Plans that are Similar to the Adapted Plan

When redistricting is done, the new districts are often quite similar to the old districts. In fact, new district boundaries are typically drawn by using the old districts as a starting point and making some changes to them. To reflect this typical redistricting process, we conduct "local exploration" by sampling redistricting plans that are similar to the adapted plan. We then investigate how the degree and direction of partisan bias changes as we increase the number of precincts switched from one district to another. This analysis reveals the degree and direction of partisan bias that can be introduced by a realistic level of changes to the current districts.

To conduct this local exploration, we run a single chain of 100,000 draws with the actual districts as the starting map. We restrict exploration of the proposed algorithm to within the local neighborhood of the adapted plan through the reweighting procedure described in Section 2.3. We set $\psi(V_k)$ equal to the fraction of precincts switched within district k, and specify $\beta = -10$ without simulated tempering. Through reweighting and sampling with replacement (see Section 2.3), we approximate uniform sampling from the population of redistricting plans that are within a pre-specified level of similarity to the current redistricting plan. To assess the convergence of the chain, we use the Geweke diagnostic, which compares the means of different sections of the chain (typically the first 10% and the last 50%) to determine if the samples are drawn from the stationary distribution. For both the New Hampshire and the Mississippi simulations, we fail to reject the null hypothesis that the two means are equal, suggesting that the samples are drawn from the stationary distribution.

Following the literature (see Grofman and King, 2007, and references therein), we use deviation from partisan symmetry as a measure of partisan bias. To measure this, we vary the 2008 Democratic and Republican two-party voteshares uniformly across precincts from the actual voteshare by adding uniform swings of various sizes. We consider all vote swings where the aggregate two-party voteshares across the entire state are between 30% and 70%. These voteshares are aggregated according to each redistricting plan, yielding the two-party voteshares at the district level. These district-level voteshares in turn can be used to determine the number of hypothetical Democrat and Republican winners of the election. The resulting information is summarized as the seats-votes curve (Tufte, 1973), which in this case is an empirical step-function $s^*(x)$ evaluated within the range of voteshares x from 30% to 70%. For example, for a value of the Republican voteshare corresponding to a jump in this step-function, when the statewide voteshare for Republicans is just smaller than this value, there will be one less Republican winner than if the statewide voteshare was just above this value.

We then find a step-function $s(x)$, symmetric across the 50%-50% voteshare, that is "closest" to the empirical step function through a brute force algorithm. Specifically, we find the symmetric step-function $s(x)$ that minimizes the absolute area between the symmetric step-function $s(x)$ and the empirical step-function $s^*(x)$. This minimized difference between the two step-functions defines

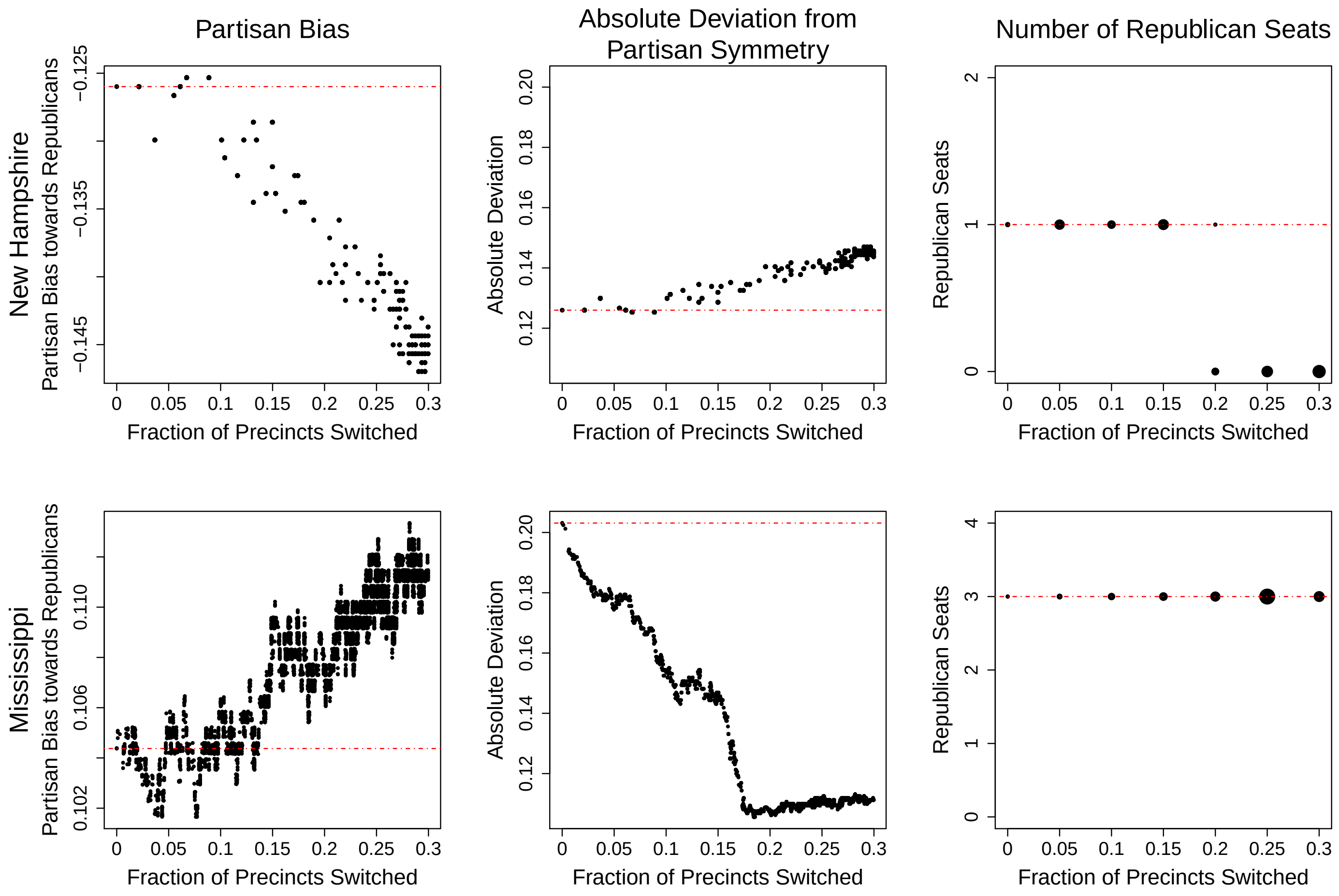


Figure 9: Redistricting Plans that are Similar to the Adapted Plan (dashed lines) for New Hampshire (top panel) and Mississippi (bottom panel). The plots in the left column show how the degree and direction of Republican bias change as we switch more precincts from one district to another. The plots in the middle column show the absolute deviation from partisan symmetry. Finally, the plots in the right column show the number of Republican legislators who would win a seat under various redistricting plans. The area of solid circles are proportional to the number of corresponding redistricting plans.

the absolute deviation from partisan symmetry, and is formally expressed as follows,

$$\text{deviation from partisan symmetry} \equiv \int_{0.3}^{0.7} |s(x) - s^*(x)| dx \tag{14}$$

The partisan bias is defined as the difference between the area below the symmetric step function and the area below the empirical step function,

$$\text{partisan bias} \equiv \int_{0.3}^{0.7} s(x)dx - \int_{0.3}^{0.7} s^*(x)dx = \int_{0.3}^{0.7} (s(x) - s^*(x))\, dx. \tag{15}$$

Figure 9 presents the results for New Hampshire (top panel) and Mississippi (bottom panel). The plots are based on a sample of redistricting plans that are similar to the adapted plan (red dashed lines). In the left column, we show how the degree of partisan bias changes as we switch more precincts from one district to another. In New Hampshire, the adapted redistricting plan was slightly biased towards Democrats but this bias steadily increases once we switch more than 10% of precincts. The deviation from partisan symmetry also increases as the redistricting plans deviate from the adapted plan. That is, the empirical seats-votes curve is further away from the symmetric step-function when more precincts are switched. Finally, the increase in partisan bias towards Democrats is also reflected by the fact that no Republican would be elected under a large proportion of redistricting plans when we switch 20% of precincts and more.

A different pattern emerges for Mississippi (bottom panel). Here, the actual districts exhibits a positive Republican bias and the magnitude of this bias appears to grow on average as we switch more precincts from one district into another. However, there exists a considerable variation especially when a small fraction of precincts are switched. Indeed, up to about 10% of precincts switched, the partisan bias stays at a similar level on average. Moreover, unlike New Hampshire, as the proportion of precincts switched increases, the deviation from partisan symmetry declines even though the partisan bias is growing. This means that the empirical seats-votes curve is becoming closer to the symmetric seats-votes curve (i.e., partisan symmetry) even though the bias for Republicans remain. Finally, the number of Republican seats stays at 3 under all sampled redistricting plans, which again contrasts with the results for New Hampshire.

4.2 Assessing the Partisan Implications of a Compactness Constraint

Next, we assess the partisan implications of a geographic compactness constraint. This has been of interest to many scholars including McCarty *et al.* (2003), Fryer and Holden (2011), and Chen and Rodden (2013). Currently, we analyze only the New Hampshire data; in the future, we intend to include Mississippi in our analysis.

To assess the effect of compactness, we run our algorithm under two different specifications. Under each specification, we iterate our algorithm repeatedly until obtaining 50,000 plans, where each plan has districts with population within 1% of parity. We repeat this process 10 times, thereby obtaining 500,000 redistricting plans in total. We then sample 12,000 plans with replacement for our plots in Figure 10. Under one specification, we use the basic algorithm (Algorithm 1) without reweighting or simulated tempering. Under another, we follow the reweighting procedure for geographical compactness described in Section 2.3, and use simulated tempering where the cold temperature $\beta = -1000$ and $\beta^{(\ell)}$ varies from 0 to $-1,000$ by 100.

Figure 10 presents the results for New Hampshire. Simulations corresponding to the bottom three panels apply the geographic compactness constraint and those for the top three panels do not.

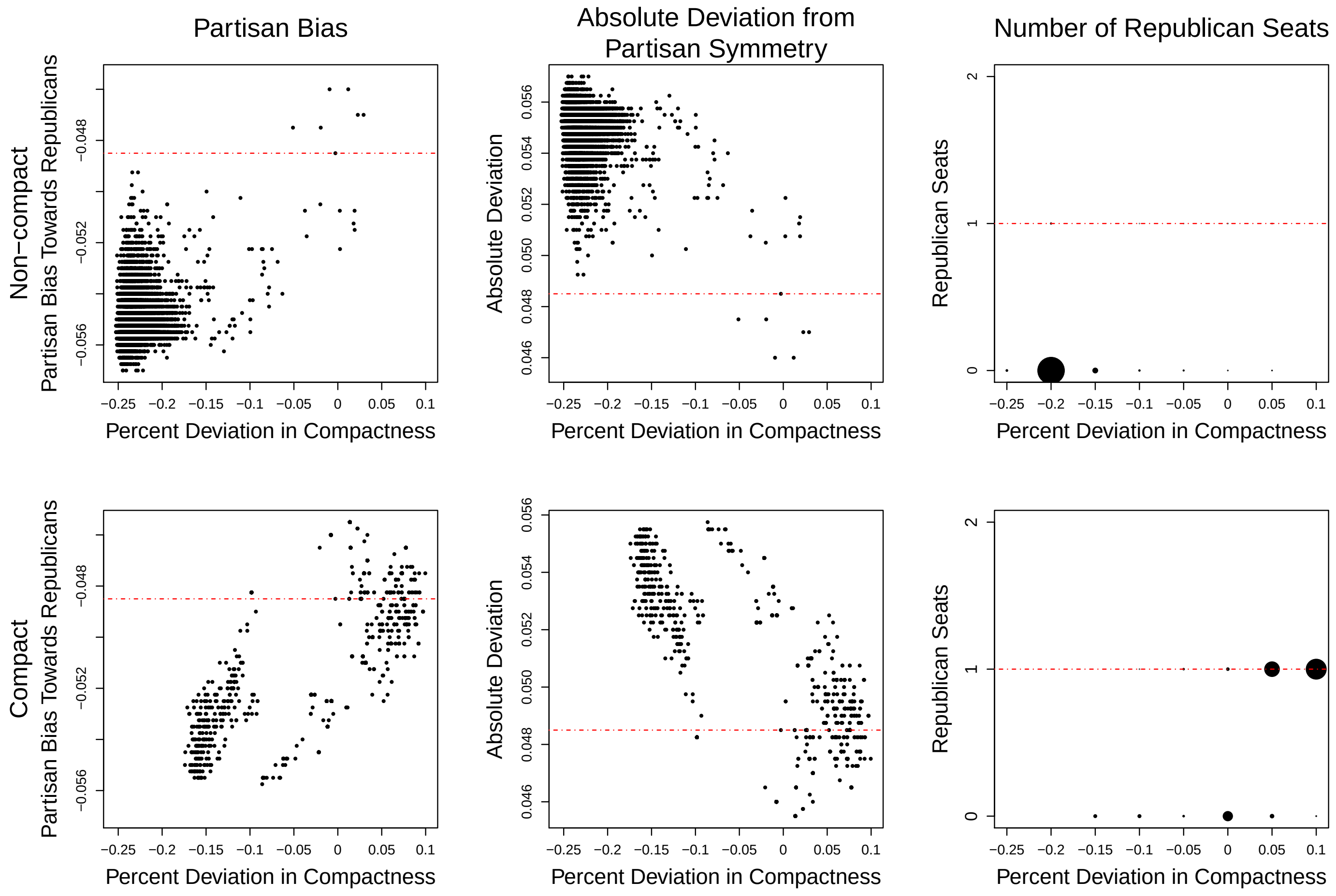


Figure 10: New Hampshire Redistricting Plans Obtained With (bottom panel) and Without (top panel) a Compactness Constraint. The plots in the left column show how the degree and direction of Republican bias change as we switch more precincts from one district to another. The plots in the middle column show the absolute deviation from partisan symmetry. Finally, the plots in the right column show the number of Republican legislators who would win a seat under various redistricting plans. The area of solid circles are proportional to the number of corresponding redistricting plans.

When no compactness constraint is imposed, the implemented redistricting plan is among the most compact plans recovered by the algorithm. As geographic compactness decreases relative to the true plan, the redistricting plans are increasingly biased in favor of Democrats (left column). This is reflected in the expected number of Republican seats won (right column). In nearly all of these plans, Democrats are predicted to win both Congressional seats. Lower geographic compactness also corresponds to an increase in distance from partisan symmetry.

In contrast, a different pattern of representation emerges when we apply a compactness constraint. First, the algorithm successfully discovers redistricting plans that are more compact than the implemented plan. Whereas the median redistricting plan in non-compact simulations is 23 percentage points less compact than the implemented plan, the median redistricting plan recovered when applying a compactness constraint is 6 percentage points more compact than the implemented plan. Furthermore, as plans become more compact, the recovered plans are less biased in favor of Democrats (left column). This can also be seen in the expected number of seats won by Republicans. In contrast to the non-compact simulations, where few redistricting plans gave Republicans an expected Congressional seat, most compact redistricting plans give both the Democrats and the Republicans a Congressional seat (right column). Finally, increasing geographic compactness appears to keep plans from deviating too severely from partisan symmetry (middle column).

5 Concluding Remarks

Over the last half century, a number of automated redistricting algorithms have been proposed in the methodological literature. Most of these algorithms have been designed to find an optimal redistricting plan given a certain set of criteria. However, many substantive researchers have been interested in characterizing the distribution of redistricting plans under various constraints. Unfortunately, few such simulation algorithms exist and even the ones that are commonly used by applied researchers have no theoretical justification.

In this paper, we propose a new automated redistricting simulator using Markov chain Monte Carlo. Unlike the existing standard algorithm, the proposed algorithms has a theoretical justification and approximates the target distribution well in a small-scale validation study. Even in more realistic settings where the computational challenge is greater, our initial analyses shows a promising performance of the proposed algorithms. Nevertheless, it is still unclear whether these algorithms scale to those states with a greater number of districts than those considered here. In the future, we plan to investigate whether combining simulated tempering and divide-and-conquer strategies can overcome the computational challenge posed by those large states.

Another promising line of research we are currently undertaking is to examine the factors that predict the redistricting outcome. For example, substantive researchers are interested in how the institutional features of redistricting process (e.g., bipartisan commission vs. state legislature) determines the redistricting process. Such an analysis requires inferences about the parameters that are underlying our generative model. In contrast, in this paper we restricted our attention to the question of how to simulate redistricting plans given these model parameters. Therefore, a different methodological approach is required to address this and other methodological challenges.

A Proofs of Theorems

A.1 Proof of Theorem 1

Let $\Gamma(C^*, G_{\mathbf{v}})$ denote all sets of connected components C obtainable through "turning on" edges in $E_{\mathbf{v}}$ such that $C^* \subset C$. Let $p(C \mid G_{\mathbf{v}})$ denote the probability that C is obtained through Steps 1 and 2 of Algorithm 1. Let $p(C^* \mid C)$ denote the probability that, given C, its particular subset C^* is selected at Step 3. Note that this probability does not depend on the partition $\mathbf{v}$. Then, it follows that

$$\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) = \sum_{C' \in \Gamma(C^*, G_{\mathbf{v}_{t-1}})} p(C^* \mid C') p(C' \mid G_{\mathbf{v}_{t-1}}) \prod_{\ell=1}^{r} \frac{1}{|A(C_\ell^*, \mathbf{v}_{t-1})|} \tag{16}$$

$$\pi(\mathbf{v}_t^* \to \mathbf{v}_{t-1}) = \sum_{C' \in \Gamma(C^*, G_{\mathbf{v}_t^*})} p(C^* \mid C') p(C' \mid G_{\mathbf{v}_t^*}) \prod_{\ell=1}^{r} \frac{1}{|A(C_\ell^*, \mathbf{v}_t^*)|} \tag{17}$$

We now simplify equations (16) and (17) to identify common terms, which then cancel each other in equation (3). First, we show

$$|A(C_\ell^*, \mathbf{v}_{t-1})| = |A(C_\ell^*, \mathbf{v}_t^*)| \tag{18}$$

for any connected component $C_\ell^* \in C^*$ where $l \in \{1, \ldots, r\}$.

Suppose that, without loss of generality, C_ℓ^* is adjacent to blocks $V_{1,t-1}, V_{2,t-1}, \ldots, V_{|A(C_\ell^*,\mathbf{v}_{t-1})|,t-1} \in \mathbf{v}_{t-1}$, and C_ℓ^* is contained in block $V_{|A(C_\ell^*,\mathbf{v}_{t-1})|+1,t-1} \in \mathbf{v}_{t-1}$. The check that $V_{kt}^* \neq \emptyset$ in Step 4 of the algorithm ensures that $C_\ell^* \neq V_{|A(C_\ell^*,\mathbf{v}_{t-1})|+1,t-1}$. Since $\mathbf{v}_{t-1}$ is a connected set partition, there must exist $\{i_{|A(C_\ell^*,\mathbf{v}_t^*)|+1}\} \in C_\ell^*$ and $\{j_{|A(C_\ell^*,\mathbf{v}_t^*)|+1}\} \in V_{|A(C_\ell^*,\mathbf{v}_{t-1})|+1,t-1} \setminus C_\ell^*$ that are adjacent in $G_{\mathbf{v}_{t-1}}$. Moreover, there exist pairs of adjacent nodes $(\{i_1\}, \{j_1\}), \ldots, (\{i_{|A(C_\ell^*,\mathbf{v}_{t-1})|}\}, \{j_{|A(C_\ell^*,\mathbf{v}_{t-1})|}\})$ with $\{i_k\} \in C_\ell^*, \{j_k\} \in V_{k,t-1}$ where $1 \leq k \leq |A(C_\ell^*, \mathbf{v}_{t-1})|$. Since C^* is comprised of non-adjacent connected components, it follows that nodes $\{j_1\}, \ldots, \{j_{|A(C_\ell^*,\mathbf{v}_{t-1})|}\}, \{j_{|A(C_\ell^*,\mathbf{v}_{t-1})|+1}\}$ do not change block assignment when transitioning from $\mathbf{v}_{t-1}$ to $\mathbf{v}_t^*$, and thus, are contained in distinct blocks in $\mathbf{v}_t^*$. Thus, the connected component C_ℓ^* is adjacent to all blocks corresponding to a node in $\{\{j_1\}, \ldots, \{j_{|A(C_\ell^*,\mathbf{v}_t^*)|}\}, \{j_{|A(C_\ell^*,\mathbf{v}_t^*)|+1}\}\}$ except for the block containing C_ℓ^*: $|A(C_\ell^*, \mathbf{v}_{t-1})|$ blocks in total. Hence, $|A(C_\ell^*, \mathbf{v}_t^*)| \geq |A(C_\ell^*, \mathbf{v}_{t-1})|$. Moreover, for any block $V_{k,t-1} \notin A(C_\ell^*, \mathbf{v}_{t-1})$ such that $C_\ell^* \not\subset V_{k,t-1}$, the corresponding block $V_{k,t}^*$ obtained by swapping connected components in C^* will not be contained in $A(C_\ell^*, \mathbf{v}_t^*)$; by definition, for any $\{i\} \in C_\ell^*$, $\{j\} \in V_{k,t-1}$, $(i,j) \notin E$, and since connected components in C^* are not adjacent, it follows that no edge connects a vertex in $V_{k,t}^*$ to a vertex in C_ℓ^*. This proves equation (18).

Next, through a proof by contradiction, we show that

$$\Gamma(C^*, G_{\mathbf{v}_{t-1}}) = \Gamma(C^*, G_{\mathbf{v}_t^*}). \tag{19}$$

By showing this, we also conclude that $\mathbf{v}_{t-1}$ can be a candidate partition when starting from $\mathbf{v}_t^*$, i.e., $\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) > 0$ implies $\pi(\mathbf{v}_t^* \to \mathbf{v}_{t-1}) > 0$. Suppose that there exists a set of connected components $C' \in \Gamma(C^*, G_{\mathbf{v}_{t-1}})$ such that $C' \notin \Gamma(C^*, G_{\mathbf{v}_t^*})$. This means that there exists $C'_\ell \in C'$ that can be formed by turning on edges in $E_{\mathbf{v}_{t-1}^*}$ but not in $E_{\mathbf{v}_t^*}$. Thus, there exists $\{i\}, \{j\} \in C'_\ell$ such that $(i,j) \in E_{\mathbf{v}_{t-1}}$ and $(i,j) \notin E_{\mathbf{v}_t^*}$. However, according to Step 4 of the algorithm, the only edges deleted in the transition between $\mathbf{v}_{t-1}$ and $\mathbf{v}_t^*$, are those connecting a vertex in $\{i\}$ in C^* to a vertex $\{j\} \notin C^*$. Since $C^* \subset C' \in \Gamma(C^*, G_{\mathbf{v}_{t-1}})$, $\{i\}$ and $\{j\}$ cannot be contained in the same component of C', a contradiction. An analogous argument shows that there is no connected component $C' \in \Gamma(C, \mathbf{v}_t^*)$ such that $C' \notin \Gamma(C, \mathbf{v}_{t-1})$. This proves equation (19).

Third, we decompose $p(C \mid G_{\mathbf{v}})$. For a partition $\mathbf{v}$, let $\Lambda(C, E_{\mathbf{v}})$ denote all subsets of edges of $E_{\mathbf{v}}$ such that, when only those edges in a subset are turned on, the set of connected components C is formed (Step 2). Note that C can be formed if and only if the partition $\mathbf{v}$ satisfies $E_C \subset E_{\mathbf{v}}$, and $\Lambda(C, E_{\mathbf{v}})$ is identical for all such partitions. Specifically, $\Lambda(C, E_{\mathbf{v}_{t-1}}) = \Lambda(C, E_{\mathbf{v}_t^*})$. To see this, observe that every set of edges $E^*_{\mathbf{v}} \in \Lambda(C, E_{\mathbf{v}})$ must connect nodes within each connected component in C, and must not include any edges joining a connected component to a node not included in the connected component. For any connected component $C_\ell \in C$, there must be a block $V_k \in \mathbf{v}$ such that $C_\ell \subset V_k$. Since $E_{\mathbf{v}}$ contains all edges joining two nodes in V_k, it follows that any set of edges connecting nodes in C is contained in $E_{\mathbf{v}}$.

Given a set of "turned-on" edges $E^*_{\mathbf{v}} \in \Lambda(C, E_{\mathbf{v}})$, define $\overline{E}^*_{\mathbf{v}} \equiv E_{\mathbf{v}} \setminus E^*_{\mathbf{v}}$ as the set of "turned-off" edges. Observe that, for $E^*_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}})$, $E^*_{\mathbf{v}_t^*} \in \Lambda(C, E_{\mathbf{v}_t})$ with $E^*_{\mathbf{v}_{t-1}} = E^*_{\mathbf{v}_t}$ $\overline{E}^*_{\mathbf{v}_{t-1}}$ may be different from $\overline{E}^*_{\mathbf{v}_t^*}$. That is, if the candidate partition $\mathbf{v}^*$ is obtained from $\mathbf{v}_{t-1}$ by assigning connected component $C' \in C$ from block V_ℓ to block $V_{\ell'}$, $\overline{E}^*_{\mathbf{v}_t^*}$ may contain an edge that connects a node in C' to an adjacent node in $V_{\ell'}$, whereas this edge cannot occur in $\overline{E}^*_{\mathbf{v}_{t-1}}$. Define

$$\begin{aligned} B(C^*, \overline{E}^*_{\mathbf{v}}) &\equiv \{(i,j) \in \overline{E}^*_{\mathbf{v}} : \{i\} \in C^*, \{j\} \notin C^*\} \\ &= \{(i,j) \in \overline{E}^*_{\mathbf{v}} : \exists C^*_\ell \in C^*, C^*_\ell \subset V_k \in \mathbf{v} \; s.t. \{i\} \in C^*_\ell, \{j\} \in V_k \setminus C^*_\ell\} \end{aligned} \tag{20}$$

as the set of edges in $\overline{E}^*_{\mathbf{v}}$ that connect a block of nodes in C^* to a vertex not in C^*, i.e., those edges that need to be "cut" to form blocks of vertices C^*. Since $C^* \subset C$, for partition $\mathbf{v}$, $B(C^*, E_{\mathbf{v}})$ is the same for every set of turned-on edges in $\Lambda(C, E_{\mathbf{v}})$, and is the same across all sets of connected components in $\Gamma(C^*, G_{\mathbf{v}})$. Then, we can write $p(C \mid G_{\mathbf{v}})$ as:

$$p(C \mid G_{\mathbf{v}_{t-1}}) = \prod_{e \in B(C^*, E_{\mathbf{v}_{t-1}})} (1 - q_e) \sum_{E^*_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}})} \prod_{e \in E^*_{\mathbf{v}_{t-1}}} q_e \prod_{e \in \overline{E}^*_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})} (1 - q_e) \tag{21}$$

where we allow the edge cut probability to differ across edges.

Finally, we show that, for any $E^*_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}})$, $E^*_{\mathbf{v}_t^*} \in \Lambda(C, E_{\mathbf{v}_t})$ with $E^*_{\mathbf{v}_{t-1}} = E^*_{\mathbf{v}_t}$,

$$E^*_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}}) = E^*_{\mathbf{v}_t^*} \setminus B(C^*, E_{\mathbf{v}_t^*}) \tag{22}$$

Consider any edge $e \in E_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})$. This edge can either join two nodes within a single connected component or joins two nodes in two distinct connected components. In the former case, both nodes are contained in a single block of $\mathbf{v}_{t-1}$, and since connected components are reassigned to form the candidate partition $\mathbf{v}_t^*$, it follows that both nodes are contained in a single block $V^* \in \mathbf{v}_t^*$. Hence, $e \in E_{\mathbf{v}_t^*}$, and since does not join a node in connected component in C^* to a node in a connected component that is not in C^*, it follows that $e \in E_{\mathbf{v}_t^*} \setminus B(C^*, E_{\mathbf{v}_t^*})$. In the latter case, observe that, since $e \in E_{\mathbf{v}_{t-1}}$, both connected components must be contained within the same block of $\mathbf{v}_{t-1}$. Since they do not belong to C^*, neither component is reassigned to a block, and hence, are contained within the same block $V_{kt}^* \in \mathbf{v}_t^*$. Thus, $e \in E_{\mathbf{v}_t^*}$, and since does not join a node in connected component in C^* to a node in a connected component that is not in C^*, it follows that $e \in E_{\mathbf{v}_t^*} \setminus B(C^*, E_{\mathbf{v}_t^*})$. In both cases, $e \in E_{\mathbf{v}_t^*} \setminus B(C^*, E_{\mathbf{v}_t^*})$. Thus, $E_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}}) \subset E_{\mathbf{v}_t^*} \setminus B(C^*, E_{\mathbf{v}_t^*})$. By the same argument, $E_{\mathbf{v}_t^*} \setminus B(C^*, E_{\mathbf{v}_t^*}) \subset E_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})$, and we have shown equation (22). By this observation, we can now write,

$$p(C \mid G_{\mathbf{v}_t^*}) \;=\; \prod_{e \in B(C^*, E_{\mathbf{v}_t^*})} (1-q_e) \sum_{E^*_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}})} \prod_{e \in E^*_{\mathbf{v}_{t-1}}} q_e \prod_{e \in E^*_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})} (1-q_e). \quad (23)$$

Using equation (21) and the fact that the set of edges $B(C^*, \mathbf{v}_{t-1})$ is identical across all sets of connected components $C_\ell \in C^*$, we can write as:

$$\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) \;=\; \prod_{e \in B(C^*, E_{\mathbf{v}_{t-1}})} (1-q_e) \sum_{C \in \Gamma(C^*, \mathbf{v}_{t-1})} \left(\sum_{E^*_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}})} \prod_{e \in E^*_{\mathbf{v}_{t-1}}} q_e \prod_{e \in \overline{E}^*_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})} (1-q_e) \right)$$
$$\times \;\; p(C^* \mid C) \prod_{\ell=1}^{r} \frac{1}{|A(C_\ell^*, \mathbf{v}_{t-1})|} \quad (24)$$

Similarly, we find that:

$$\pi(\mathbf{v}_t^* \to \mathbf{v}_{t-1}) \;=\; \prod_{e \in B(C^*, E_{\mathbf{v}_t^*})} (1-q_e) \sum_{C \in \Gamma(C^*, \mathbf{v}_{t-1})} \left(\sum_{E^*_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}})} \prod_{e \in E^*_{\mathbf{v}_{t-1}}} q_e \prod_{e \in \overline{E}^*_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})} (1-q_e) \right)$$
$$\times \;\; p(C^* \mid C) \prod_{\ell=1}^{r} \frac{1}{|A(C_\ell^*, \mathbf{v}_{t-1})|}. \quad (25)$$

Thus, many terms cancel out and we obtain the following expression for the acceptance probability:

$$\alpha(\mathbf{v} \to \mathbf{v}^*) \;=\; \min\left(1,\; \frac{\prod_{e \in B(C^*, \mathbf{v}_t^*)} (1-q_e)}{\prod_{e \in B(C^*, \mathbf{v}_{t-1})} (1-q_e)}\right). \quad (26)$$

In the special case that edges are turned on with equal probability, i.e., $q = q_e$ for all e, this ratio can be computed by counting the number of edges connecting nodes in blocks of C^* to nodes outside of those blocks:

$$\alpha(\mathbf{v} \to \mathbf{v}^*) \;=\; \min\left(1,\; (1-q)^{|B(C^*, \mathbf{v}_t^*)| - |B(C^*, \mathbf{v}_{t-1})|}\right). \quad (27)$$

□

A.2 Proof of Theorem 2

We proceed as before and compute the acceptance probability:

$$\lambda(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) \quad = \quad \min\left(1,\ \frac{\pi(\mathbf{v}_t^* \to \mathbf{v}_{t-1})}{\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*)}\right). \tag{28}$$

The probability of obtaining candidate partition $\mathbf{v}_t^*$ from $\mathbf{v}_{t-1}$ is:

$$\begin{aligned}\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) &= P(\text{Select } \{V_{k',t-1}, V_{\ell',t-1}\} \in \mathbf{P}_{\mathbf{v}_{t-1}}) \\ &\quad \times P(\text{Obtain } \{V_{k't}^*, V_{\ell't}^*\} \text{ from Algorithm 1 } \mid \text{Select } \{V_{k',t-1}, V_{\ell',t-1}\}) \\ &\approx \frac{1}{|\mathbf{P}_{\mathbf{v}_{t-1}}|} \frac{1}{|\mathbf{Q}(V_{k',t-1} \cup V_{\ell',t-1})|} \end{aligned} \tag{29}$$

where $\mathbf{Q}(V)$ represents the set of all valid partitions of V into two subgraphs. The first term in this probability comes from the fact that we are randomly sampling a pair of adjacent blocks, and the second term comes from the assumption that the number of iterations is large enough that the terminal partition is close in distribution to uniform over all valid partitions. Noting that $V_{k',t-1} \cup V_{\ell',t-1} = V_{k',t}^* \cup V_{\ell',t}^*$, we have,

$$\frac{\pi(\mathbf{v}_t^* \to \mathbf{v}_{t-1})}{\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*)} \quad = \quad \frac{|\mathbf{P}_{\mathbf{v}_{t-1}}|\ |\mathbf{Q}(V_{k',t-1} \cup V_{\ell',t-1})|}{|\mathbf{P}_{\mathbf{v}_t^*}|\ |\mathbf{Q}(V_{k't}^* \cup V_{\ell't}^*)|} \quad = \quad \frac{|\mathbf{P}_{\mathbf{v}_{t-1}}|}{|\mathbf{P}_{\mathbf{v}_t^*}|} \tag{30}$$

Next, we consider the generalization of this result by entertaining the model given in equation (6). We show that no modification is necessary for our divide-and-conquer algorithm even when the target distribution is given by this model, which is a non-uniform distribution (unless $\beta = 0$). Assume that the modified basic algorithm as discussed in Section 2.3 and applied in Step 2 of the algorithm converges to a distribution whose unnormalized density is given by $g_\beta(V_k \cup V_\ell) = \exp[\beta\{\psi(V_k) + \psi(V_\ell)\}] = g_\beta(V_k)g_\beta(V_\ell)$. Then, we can write:

$$\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*) \quad \approx \quad \frac{g_\beta(V_{k't}^*)g_\beta(V_{\ell't}^*)}{|\mathbf{P}_{\mathbf{v}_{t-1}}|} \tag{31}$$

Therefore, we have:

$$\frac{\pi(\mathbf{v}_t^* \to \mathbf{v}_{t-1})}{\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*)} \quad \approx \quad \frac{|\mathbf{P}_{\mathbf{v}_{t-1}}| g_\beta(V_{k',t-1}) g_\beta(V_{\ell',t-1})}{|\mathbf{P}_{\mathbf{v}_t^*}| g_\beta(V_{k',t}^*) g_\beta(V_{\ell',t}^*)} \tag{32}$$

Now, since $\mathbf{v}_{t-1}$ and $\mathbf{v}_t^*$ are identical except for blocks k' and ℓ', it follows that:

$$\frac{P(\mathbf{v}_t^*)}{P(\mathbf{v}_{t-1})} \quad = \quad \frac{\prod_{k=1}^n g(V_{kt}^*)}{\prod_{k=1}^n g(V_{k,t-1})} = \frac{g(V_{k't}^*)g(V_{\ell't}^*)}{g(V_{k',t-1})g(V_{\ell',t-1})} \tag{33}$$

Hence, for convergence to the stationary distribution $P(\mathbf{v})$, the acceptance probability must be:

$$\lambda(\mathbf{v} \to \mathbf{v}^*) \quad = \quad \min\left(1,\ \frac{\pi(\mathbf{v}_t^* \to \mathbf{v}_{t-1})P(\mathbf{v}_t^*)}{\pi(\mathbf{v}_{t-1} \to \mathbf{v}_t^*)P(\mathbf{v}_{t-1})}\right) \quad = \quad \min\left(1,\ \frac{|\mathbf{P}_{\mathbf{v}_{t-1}}|}{|\mathbf{P}_{\mathbf{v}_t^*}|}\right). \tag{34}$$

□

References

Abramowitz, A. I. (1983). Partisan redistricting and the 1982 congressional elections. *Journal of Politics* **45**, 3, 767–770.

Altman, M. (1997). The computational complexity of automated redistricting: Is automation the answer. *Rutgers Computer & Technology Law Journal* **23**, 81–142.

Altman, M., MacDonald, K., and McDonald, M. (2005). From crayons to computers: The evolution of computer use in redistricting. *Social Science Computer Review* **23**, 3, 334–346.

Altman, M. and McDonald, M. P. (2011). BARD: Better automated redistricting. *Journal of Statistical Software* **42**, 4, 1–28.

Ansolabehere, S., Snyder, J. M., and Stewart, C. (2000). Old voters, new voters, and the personal vote: Using redistricting to measure the incumbency advantage. *American Journal of Political Science* **44**, 1, 17–34.

Barbu, A. and Zhu, S.-C. (2005). Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **27**, 8, 1239–1253.

Barreto, M. A., Segura, G. M., and Woods, N. D. (2004). Mobilizing effect of majority-minority districts. *American Political Science Review* **98**, 1, 65–75.

Bozkaya, B., Erkut, E., and Laporte, G. (2003). A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research* **144**, 12–26.

Browdy, M. H. (1990). Simulated annealing: An improved computer model for political redistricting. *Yale Law & Policy Review* **8**, 1, 163–179.

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

Chou, C.-I. and Li, S. P. (2006). Taming the gerrymander — statistical physics approach to political districting problem. *Physica A: Statistical Mechanics and its Applications* **369**, 2, 799–808.

Cirincione, C., Darling, T. A., and O'Rourke, T. G. (2000). Assessing South Carolina's 1990s congressional districting. *Political Geography* **19**, 189–211.

Engstrom, R. L. and Wildgen, J. K. (1977). Pruning thorns from the thicket: An empirical test of the existence of racial gerrymandering. *Legislative Studies Quarterly* **2**, 4, 465–479.

Fryer, R. and Holden, R. (2011). Measuring the compactness of political districting plans. *Journal of Law and Economics* **54**, 3, 493–535.

Garfinkel, R. S. and Nemhauser, G. L. (1970). Political districting by implicit enumeration techniques. *Management Science* **16**, 8, B495–B508.

Gelman, A. and King, G. (1994). A unified method of evaluating electoral systems and redistricting plans. *American Journal of Political Science* **38**, 513–554.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulations using multiple sequences (with discussion). *Statistical Science* **7**, 4, 457–472.

Geyer, C. J. and Thompson, E. A. (1995). Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association* **90**, 909–920.

Grofman, B. and King, G. (2007). The future of partisan symmetry as a judicial test for partisan gerrymandering after *lulac v. perry*. *Election Law Journal* **6**, 1, 2–35.

Hastings, W. K. (1970). Monte Carlo sampling methods usings Markov chains and their applications. *Biometrika* **57**, 97–109.

Hess, S. W., Weaver, J. B., Siegfeldt, H. J., Whelan, J. N., and Zitlau, P. A. (1965). Nonpartisan political redistrictingn by computer. *Operations Research* **13**, 6, 998–1006.

Marinari, E. and Parisi, G. (1992). Simulated tempering: A new Monte Carlo scheme. *Europhysics Letters* **19**, 451–458.

McCarty, N., Poole, K. T., and Rosenthal, H. (2003). Does gerrymandering cause polarization? *Amerian Journal of Political Science* **53**, 3, 666–680.

Mehrotra, A., Johnson, E., and Nemhauser, G. L. (1998). An optimization based heuristic for political districting. *Management Science* **44**, 8, 1100–1114.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics* **21**, 1087–1092.

Nagel, S. S. (1965). Simplified bipartisan computer redistricting. *Stanford Law Journal* **17**, 5, 863–899.

Niemi, R. G., Grofman, B., Carlucci, C., and Hofeller, T. (1990). Measuring compactness and the role of a compactness standard in a test for partisan and racial gerrymandering. *Journal of Politics* **52**, 1155–1181.

O'Loughlin, J. (1982). The identification and evaluation of racial gerrymandering. *Annals of the Association of American Geographers* **72**, 2, 165–184.

Rubin, D. B. (1987). Comment: A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputation when fractions of missing information are modest:the SIR algorithm. *Journal of the American Statistical Association* **82**, 398, 543–546.

Swendsen, R. H. and Wang, J. S. (1987). Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters* **58**, 86–88.

Tierney, L. (1994). Markov chains for exploring posterior distributions (with discussion). *The Annals of Statistics* **22**, 1701–1762.

Tufte, E. R. (1973). The relationship between seats and votes in two-party systems. *American Political Science Review* **67**, 2, 540–554.

Vickrey, W. (1961). On the prevention of gerrymandering. *Political Science Quarterly* **76**, 1, 105–110.

Weaver, J. B. and Hess, S. W. (1963). A procedure for nonpartisan districting: Development of computer techniques. *Yale Law Journal* **73**, 2, 288–308.