Biometrika (2025), **112**, 2, asaf031 https://doi.org/10.1093/biomet/asaf031
Advance Access publication 15 April 2025

Exact sampling of spanning trees via fast-forwarded random walks

By EDRIC TAM

Department of Biomedical Data Science, Stanford University, 300 Pasteur Drive, Palo Alto, California 94304, U.S.A.
edrictam@stanford.edu

DAVID B. DUNSON

Department of Statistical Science and Department of Mathematics, Duke University, Box 90251, Durham, North Carolina 27708, U.S.A.
dunson@duke.edu

and LEO L. DUAN

Department of Statistics, University of Florida, 101C Griffin-Floyd Hall, P.O. Box 118545, Gainesville, Florida 32611, U.S.A.
li.duan@ufl.edu

Summary

Tree graphs are used routinely in statistics. When estimating a Bayesian model with a tree component, sampling the posterior remains a core difficulty. Existing Markov chain Monte Carlo methods tend to rely on local moves, often leading to poor mixing. A promising approach is to instead directly sample spanning trees on an auxiliary graph. Current spanning tree samplers, such as the celebrated Aldous–Broder algorithm, rely predominantly on simulating random walks that are required to visit all the nodes of the graph. Such algorithms are prone to getting stuck in certain subgraphs. We formalize this phenomenon using the bottlenecks in the random walk's transition probability matrix. We then propose a novel fast-forwarded cover algorithm that can break free from bottlenecks. The core idea is a marginalization argument that leads to a closed-form expression that allows for fast-forwarding to the event of visiting a new node. Unlike many existing approximation algorithms, our algorithm yields exact samples. We demonstrate the enhanced efficiency of the fast-forwarded cover algorithm, and illustrate its application in fitting a Bayesian dendrogram model on a Massachusetts crime and community dataset.

Some key words: Aldous–Broder algorithm; Bayesian modelling; Invariant distribution; Isoperimetric constant; Random walk; Spectral graph theory.

1. Introduction

Tree graphs are commonly encountered in statistical modelling. An undirected tree is an acyclic and connected graph $T = (V_T, E_T)$ with nodes $V_T = (1, \ldots, m)$ and edges

$E_T = \{(j, l)\}$ with $|E_T| = m - 1$. By designating a node $r \in V_T$ as the root, one can easily obtain from T a directed tree $\vec{T} = (V_{\vec{T}}, E_{\vec{T}}, r)$, with identical nodes $V_{\vec{T}} = V_T$ and directed edges $E_{\vec{T}} = \{(j \to l)\}$ obtained from E_T by pointing the edges away from r. Such tree structures provide succinct ways to capture complex dependencies that arise from a wide range of statistical applications.

In hierarchical modelling, directed trees represent multi-layer dependence structures underlying the observed data. From a generative perspective, we consider a $\vec{T}$ where each node v is equipped with a parameter μ_v. Using this tree, we define an augmented likelihood for data $y_1, \ldots, y_n$ and parameters $\mu_1, \ldots, \mu_m$ conditioned on assignment labels $z_i \in (1, \ldots, m)$:

$$L(y, \mu | \vec{T}, z) = \left\{ \prod_{i=1}^{n} \mathcal{F}(y_i | \mu_{z_i}) \right\} \left\{ \mathcal{R}(\mu_r) \prod_{(j \to l) \in E_{\vec{T}}} \mathcal{H}(\mu_l | \mu_j) \right\}. \tag{1}$$

Here $\mathcal{H}(\mu_l|\mu_j)$ is the transition probability kernel from μ_j to μ_l, $\mathcal{R}(\mu_r)$ is the marginal kernel for μ_r in the root and $\mathcal{F}$ is the conditional kernel of the data y_i given the assignment z_i for that observation. The $\vec{T}$ that we condition on includes both the edge set $E_{\vec{T}}$ and the root node r. There are potentially other parameters characterizing $\mathcal{F}$ and $\mathcal{H}$, including dependence on covariates via decision trees (Chipman et al., 1998; Castillo & Ročková, 2021) or related ensemble methods (Chipman et al., 2010; Linero & Yang, 2018), but we suppress these temporarily for ease of notation. Model (1) induces a partition on $(1, \ldots, n)$ via the latent assignments z. In contrast to traditional mixture models, which often assume each μ_k to be generated independently from a common distribution, (1) characterizes the dependence in μ_j and μ_l through an ancestry tree. Ancestors of v include the tree nodes in the path from the root to v. This type of dependence is well motivated in many application areas. The tree can be interpreted as an inferred evolutionary/phylogenetic history in certain biological settings (Huelsenbeck & Ronquist, 2001; Suchard et al., 2001; Neal, 2003), or alternatively as a multi-layer partitioning of a dataset (Heller & Ghahramani, 2005).

It is also common to use a collection, or forest, of trees for flexibility in modelling. Consider

$$L(y | \vec{T}_1, \ldots, \vec{T}_{\mathcal{K}}) = \prod_{\bar{k}=1}^{\mathcal{K}} \left\{ \mathcal{R}(y_{r^{(\bar{k})}}) \prod_{(j \to l) \in E_{\vec{T}_{\bar{k}}}} \mathcal{H}(y_l | y_j) \right\}, \tag{2}$$

where each $\vec{T}_{\bar{k}} = \{V_{\vec{T}_{\bar{k}}}, E_{\vec{T}_{\bar{k}}}, r^{(\bar{k})}\}$ is a component tree, and $(V_{\vec{T}_1}, \ldots, V_{\vec{T}_{\mathcal{K}}})$ gives a $\mathcal{K}$ partition of data index $(1, \ldots, n)$, where $n = \sum_{\bar{k}=1}^{\mathcal{K}} |V_{\vec{T}_{\bar{k}}}|$. The kernel $\mathcal{R}$ describes how the first point in a group arises, and $\mathcal{H}$ characterizes the conditional dependence of the subsequent points given the previous ones. The use of trees and forests in graphical modelling (Lauritzen, 1996) dates back at least to the single-linkage clustering algorithm (Gower & Ross, 1969), and is recently seen in dependence graph estimation (Duan & Dunson, 2023), contiguous spatial partitioning (Teixeira et al., 2019; Luo et al., 2021, 2024) and model-based spectral clustering (Duan & Roy, 2024). In addition, likelihood (2) has been extended to a mixture of overlapping trees in Bayesian network estimation (Meilă & Jordan, 2000; Meilă & Jaakkola, 2006; Elidan & Gould, 2008).

Although there is a rich literature on algorithmic approaches for obtaining point estimates of tree graphs (Kruskal, 1956; Prim, 1957), we are particularly interested in model-based Bayesian approaches. Such methods have the advantage of providing a characterization of uncertainty in estimating trees, while also inferring a generative probability model for the data. Quantification of uncertainty is crucial in this context. Algorithms that produce a single tree estimate are ripe for over-interpretation and lack of reproducibility, since in most applications there are many different trees that are almost equally plausible for the data. Naturally, the success of such a Bayesian approach hinges on whether one can conduct inferences based on the posterior distribution of trees in a computationally efficient way.

Sampling from posterior distributions for trees is generally a difficult problem. Current Markov chain Monte Carlo samplers often navigate tree spaces using local modifications, such as pruning and growing moves. Since tree spaces are combinatorial and large in size, these samplers often exhibit poor mixing. A natural alternative is to rely on conjugacy to employ block updates on $\vec{T}$ in Gibbs-type samplers. Our strategy is to view $\vec{T}$ as a *spanning tree* $\vec{\mathcal{T}}$ under a complete and weighted auxiliary graph $G = (V_G, E_G)$. Here, a spanning tree $\vec{\mathcal{T}}$ of G is simply a subtree of G that spans all the nodes of G with edges oriented away from some root node r. For the generative process in (1), one can consider a complete graph G with nodes $(1, \ldots, m)$, with a weighted adjacency matrix $Q \in [0, \infty)^{m \times m}$ of elements $q_{j,l} = \mathcal{H}(\mu_l | \mu_j)$ for every pair (j, l). Observe that using (1) and a uniform prior $\Pi_0(\vec{\mathcal{T}}) \propto 1$, we can conduct a full conditional update from $\Pi(\vec{\mathcal{T}}|-)$ by drawing from the two distributions

$$\Pr(r) = \frac{g_r/\rho_r}{\sum_{r'=1}^{m} g_{r'}/\rho_{r'}}, \qquad \Pr(\vec{\mathcal{T}}|r) = \rho_r \prod_{(j \to l) \in E_{\vec{\mathcal{T}}}} q_{j,l}. \tag{3}$$

Here, $\Pr(r)$ is a discrete probability distribution on $(1, \ldots, m)$ and $\Pr(\vec{\mathcal{T}}|r)$ is a distribution over the edges of $\vec{\mathcal{T}}$ given a root node r. We define $g_r = \mathcal{R}(\mu_r)$. The term $\rho_r = \{\sum_{\vec{\mathcal{T}} \text{ rooted at } r} \prod_{(j \to l) \in E_{\vec{\mathcal{T}}}} q_{j,l}\}^{-1}$ is a normalization constant in $\Pr(\vec{\mathcal{T}}|r)$ that could vary with r. The uniform prior $\Pi_0(\vec{\mathcal{T}}) \propto 1$ is a special case of a more general class of conjugate priors, consisting of a root term multiplied by a product on the edges of the spanning tree. We discuss the sampling of $\Pr(r)$ under various scenarios in § 2.4. Here, the imperative is to efficiently sample from spanning tree distributions of the form $\Pr(\vec{\mathcal{T}}|r)$.

The celebrated Aldous–Broder algorithm (Broder, 1989; Aldous, 1990) provides a tractable way to draw exact samples from $\Pr(\vec{\mathcal{T}}|r)$. Figure 1 provides an illustration. The algorithm proceeds by taking a random walk $X_t = (r, x_1, x_2, \ldots, x_t)$ on V_G and stops at the cover time of the walk when all nodes have been visited. By collecting the first entrance edges of X_t, which are the set of edges via which the walk first visited each node, one obtains a spanning tree $\vec{\mathcal{T}}$ rooted at r. By separately specifying a distribution on the root node r, a distribution is obtained on the directed spanning trees $\vec{\mathcal{T}}$. For a graph G containing m nodes, Broder & Karlin (1989) showed an expected cover time of $O(m \log m)$ in well-connected graphs and $O(m^3)$ in the worst case.

The Aldous–Broder algorithm can be empirically slow. In practice, even if the number of nodes m in G is small, the algorithm can get stuck in certain subgraphs for a long time. This issue is inherent to the nature of random walks. Other popular algorithms, such as Wilson's algorithm based on the loop-erased random walk (Wilson, 1996), also suffer from the same problem. Beyond random walks, there are Laplacian-based algorithms that sample uniform spanning trees from unweighted graphs (Guenoche, 1983; Kulkarni, 1990; Harvey & Xu, 2016). There are also approximate random spanning tree samplers (Madry et al.,

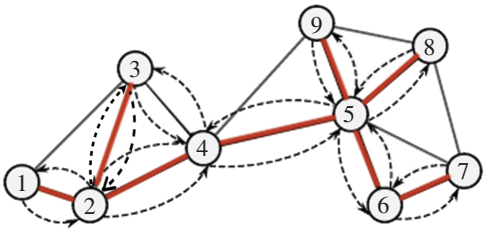

Fig. 1. Illustration for sampling a spanning tree. The edges of the underlying connected graph are shown with solid black lines. We start from a root node r (drawn proportional to g_r/ρ_r) and perform a random walk (dashed arrows) until we visit all nodes. The nodes are labelled by the order in which they are visited. The first entrance edges (red lines) of each node together form a spanning tree $\vec{\mathcal{T}}$.

2015; Schild, 2018) that rely on Laplacian solvers. In this article, we focus on algorithms based on random walks for their more general applicability.

We formalize the pathological phenomenon of the Aldous–Broder algorithm using eigenvalues of the normalized Laplacian to characterize bottlenecks in the graph. We then propose a fast-forwarded cover algorithm for exact sampling from $\Pr(\vec{\mathcal{T}}|r)$, while bypassing wasteful random walk steps. The key observation is that we do not need to simulate the entire random walk trajectory to obtain the first entrance edges to each node. We derive a closed-form expression that allows for direct, fast-forwarded sampling of these first entrance edges via a marginalization argument. In simulations, our fast-forwarded sampler enjoys much faster empirical performance against competitors when bottlenecks are present.

Existing spanning tree samplers often perform transitions directly according to the underlying graph's adjacency matrix Q, restricting their applicability to symmetric/undirected cases. We show that, by using an auxiliary matrix W, which is generated from Q under certain transformations, to specify the random walk transition probabilities, our fast-forwarded cover algorithm can be extended to more general scenarios, including cases where the underlying graph has an asymmetric adjacency matrix or the induced Markov chains are irreversible. These results are of independent interest. We illustrate our sampler by fitting a Bayesian dendrogram model on a Massachusetts crime and community dataset. The resulting Gibbs sampler exhibits drastically improved mixing performance compared to a reversible-jump sampler based on local moves and a sampler based on subtree prune and regraft moves.

2. Method

2.1. *Background on the Aldous–Broder algorithm*

We review and summarize results of the Aldous–Broder algorithm (Broder, 1989; Aldous, 1990) in this section. This algorithm samples a random spanning tree $\vec{\mathcal{T}}$ from the underlying graph G based on a random walk. Consider a weight matrix $W \in [0,\infty)^{m\times m}$ that is used to specify the probabilities of random walk transitions on G. For simplicity, it suffices for now to consider W as the graph's weighted adjacency matrix Q in (3), an important special case that applies when Q is symmetric. However, our results hold for more general cases where W is some specific transformation of Q. A detailed discussion is given in § 2.4 below.

We use W to specify a random walk over the state space $V_G = (1,\ldots,m)$. This random walk is a discrete-time Markov chain $X = (x_0, x_1, \ldots)$ with transition probabilities

$$p_{j,l} = \Pr(x_{t+1} = l | x_t = j) = \frac{w_{j,l}}{d_j}, \qquad d_j = \sum_{v=1}^{m} w_{j,v},$$

where $d_j > 0$ and w denotes entries of W. We assume that the Markov chain is irreducible: for any pair of (j, l), there exists $t > 0$ such that $\Pr(x_t = l | x_0 = j) > 0$. We denote the random walk up to time t by $X_t = (x_0, x_1, \ldots, x_t)$, and the invariant distribution of $x_t \colon t \to \infty$ by $(\pi_1, \ldots, \pi_m)$. The result of Broder (1989) further assumes reversibility of the Markov chain, $\pi_j p_{j,l} = \pi_l p_{l,j}$; however, this condition is not needed here.

The Aldous–Broder algorithm proceeds as follows: initialize the walk x_0 at root r; perform the random walk until all nodes are visited at the cover time $\hat{t}$; construct a directed spanning tree $\vec{\mathcal{T}}$ with the edge set as the set of first entrance edges all pointed away from r. Formally,

$$E_{\vec{\mathcal{T}}} = \bigcup_{j \in (V_G \setminus r)} \left\{ (x_{(t_j - 1)} \to x_{t_j}) \colon t_j = \min_{1 \leqslant t \leqslant \hat{t}} (t \colon x_t = j) \right\}.$$

The following theorem characterizes the induced distribution of $\vec{\mathcal{T}}$.

Theorem 1 (Extended Aldous–Broder algorithm). *Let $\vec{\mathcal{T}}$ be generated as above; then*

$$\Pr(\vec{\mathcal{T}} | r) \propto \left\{ \prod_{(j \to l) \in E_{\vec{\mathcal{T}}}} \frac{p_{j,l} \pi_j}{\pi_l} \right\} \frac{1}{\pi_r} \propto \prod_{(j \to l) \in E_{\vec{\mathcal{T}}}} p_{j,l} \pi_j, \tag{4}$$

where the probability is normalized over all directed spanning trees $\vec{\mathcal{T}}$ rooted from r.

If the underlying graph G is directed, the choice of root r affects the transition probabilities involved in sampling $\vec{\mathcal{T}}$. We discuss details on sampling r in § 2.4 below. The first term in (4) is based on an adaptation of Theorem 3 of Fredes & Marckert (2023), where a directed tree with edges pointed towards the root is used, as is standard in the probability literature. While this is the opposite of our choice of orientation, the mathematical results carry over. The second term is based on the fact that $\{\prod_{(j \to l) \in E_{\vec{\mathcal{T}}}} \pi_l\} \pi_r = \prod_{j=1}^{m} \pi_j$, which is invariant to $E_{\vec{\mathcal{T}}}$ and hence can be omitted.

In § 2.4 below, we also describe how one can choose W as some appropriate transformation of Q so that (4) becomes the product form shown in (3) in general settings without requiring matrix symmetry for W. For now, in the following subsection we focus on computational aspects and discuss bottleneck effects that impact the cover time of the Aldous–Broder algorithm.

2.2. *Bottlenecks in the random walk cover algorithm*

The Aldous–Broder algorithm is only efficient if there is a short cover time $\hat{t}$ to reach all nodes in V_G. However, when there are bottlenecks in the graph that lead to close-to-zero probabilities to move from the visited nodes U to the unvisited nodes $\bar{U} = V_G \setminus U$, the random walk can spend a long time within U. We now characterize the consequences of this curse of bottlenecks.

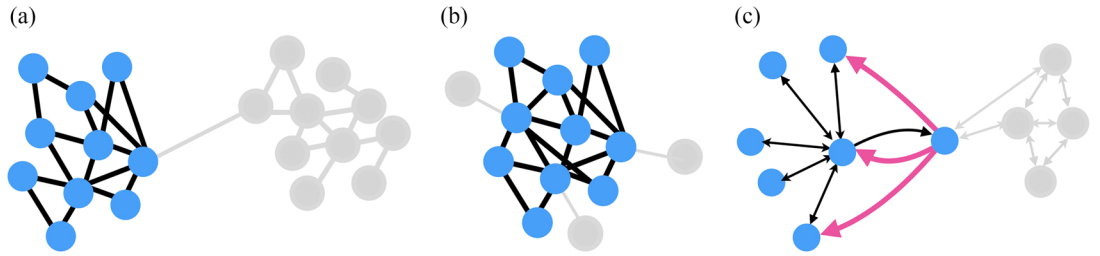

Fig. 2. Illustration of some commonly encountered graphs with the curse of bottlenecks: low transition probability to move from the visited nodes (blue) to those unvisited (grey). (a) Low edge weight between visited nodes (blue) and unvisited nodes (grey). (b) Unvisited nodes (grey) that are isolated. (c) Large probability (magenta lines) to return to visited nodes (blue) in a directed graph.

Figure 2 illustrates three common challenges through example graphs. The first two graphs are either directed or undirected, while the third is directed. First, as in panel (a), there may be disjoint sets of nodes that are only weakly connected, in the sense that the transition probability between these sets is small. Second, as in panel (b), there may be nodes that are isolated with few edges incident to them, with each incident edge having low transition probability. In this case, the random walk cover algorithm may visit most of the nodes within a short time, but faces difficulties reaching the last few. Third, as in panel (c), in a directed graph, due to edge weight asymmetry, there may be a much higher probability to remain in a node set that has been visited.

Regardless of the specific scenario, the curse of bottleneck phenomenon can be understood as a close-to-zero probability to move to the unvisited set, marginalized over both the current node and the potential arrival node amongst unvisited nodes.

We now formally characterize this phenomenon. Consider a strictly increasing node set $(r) = U_1 \subset U_2 \subset \cdots \subset U_m = V_G$, recording the history of nodes visited by the random walk, with $U_k = \bigcup_{t=0}^{\hat{t}_k} X_t$ and $\hat{t}_k = \min(t \geqslant 0 : |\bigcup_{\tilde{t}=0}^{t} X_{\tilde{t}}| = k)$, where we refer to $\hat{t}_k$ as the partial cover time for k nodes. The probability of transitioning outside of U_k at time $t+1$, conditional on the walk remaining within U_k from the beginning through time t, is

$$\begin{aligned}\Pr(x_{t+1} \in \bar{U}_k | x_{\tilde{t} \leqslant t} \in U_k) &= \sum_{j \in U_k} \left\{ \frac{\sum_{l \in \bar{U}_k} w_{j,l}}{d_j} \Pr(x_t = j | x_{\tilde{t} \leqslant t} \in U_k) \right\} \\ &\leqslant \max_{j \in U_k} \left(\frac{\sum_{l \in \bar{U}_k} w_{j,l}}{d_j} \right) \\ &= \left(\min_{j \in U_k} \frac{\sum_{l' \in U_k} w_{j,l'}}{\sum_{l \in \bar{U}_k} w_{j,l}} + 1 \right)^{-1},\end{aligned}$$

where the inequality uses the fact that the weighted average over $j \in U_k$ is less than or equal to the maximum, and the last equality is based on decomposing $d_j = \sum_{l \in \bar{U}_k} w_{j,l} + \sum_{l' \in U_k} w_{j,l'}$. Therefore, if the total outgoing weights to $\bar{U}_k$ are dominated by the weights to stay within

U_k, in the sense that $\sum_{l\in\bar{U}_k} w_{j,l} \ll \sum_{l'\in U_k} w_{j,l'}$, it is unlikely that $x_{t+1} \in \bar{U}_k$. With the above ingredients, we are ready to quantify the expected cover time.

THEOREM 2. *For a history of visited nodes $(U_1, \ldots, U_m)$, the corresponding expected cover time $\hat{t} = \hat{t}_m$ has the lower bound*

$$\mathbb{E}(\hat{t}|U_1, \ldots, U_m) \geqslant (m-1) + \sum_{k=1}^{m-1} \min_{j\in U_k} \frac{\sum_{l'\in U_k} w_{j,l'}}{\sum_{l\in\bar{U}_k} w_{j,l}}.$$

Remark 1. The lower bound applies to any $W \in [0,\infty)^{m\times m}$ that leads to an irreducible Markov chain. In practice, when the random walk cover algorithm appears to be stuck at a certain U_k, one can directly use $\min_{j\in U_k}(\sum_{l'\in U_k} w_{j,l'})/(\sum_{l\in\bar{U}_k} w_{j,l})$ as an estimate of the expected time for the walk to leave U_k.

The above result is a quantification for a specific node history of $(U_1, \ldots, U_m)$. One can obtain a worst-case expected cover time by maximizing over all possible sequences of $(U_1, \ldots, U_m)$. We are mainly interested in W matrices that satisfy a *circulation condition*: $\sum_{l=1}^m w_{j,l} = \sum_{k=1}^m w_{k,j}$, $j = 1, \ldots, m$. This condition can be applied to the directed graph setting, and includes symmetric W with $w_{j,l} = w_{l,j}$ as a special case.

THEOREM 3. *For a random walk X based on weight matrix W satisfying the circulation condition, the worst-case expected cover time $\hat{t}$ satisfies*

$$\max_{\{U_*\subset V_G:\, r\in U_*\}} \mathbb{E}\{\hat{t}|(U_1, \ldots, U_m), U_* \in (U_1, \ldots, U_m)\} \geqslant \frac{1}{M\{\lambda_2(\mathcal{L})\}^{1/2}} + m - 2$$

with $M = \sup_{t\geqslant 0} \sup_j \{\Pr(x_t = j)/\pi_j\}$, and $\lambda_2(\mathcal{L})$ the second smallest eigenvalue of the normalized graph Laplacian $\mathcal{L}$.

Remark 2. The above worst-case scenario corresponds to when the sampler visits a certain node subset $\hat{U}_* \subset V_G$ and gets stuck. Although the above lower bound on expected cover time is inspired by earlier work (Matthews, 1988; Lovász & Winkler, 1993; Levin & Peres, 2017), our result is distinguished by a directly computable lower bound. For example, the constant M^{-1} has a lower bound $\min_j \pi_j$ for any root distribution on $x_0 = r$.

2.3. *Skipping bottlenecks via fast-forwarding*

With the curse of bottlenecks on the cover time of random walks established, we now exploit a useful fact: when transforming the random walk trajectory $X_{\hat{t}}$ into the spanning tree edges $E_{\vec{T}}$, one only needs the first entrance edges to each node. It is unnecessary to simulate entire random walk trajectories until reaching a new node if the marginal distribution of the next first entrance edge can be directly sampled. We formalize this idea below.

We set up the notation here. At any time t, suppose we have visited the nodes in U, and let $e_j^t = 1$ if the walk is at node $j \in U$ at time t and the first exits U at time $t+1$. Let $e_j^t = 0$ otherwise. Intuitively, e_j^t represents the decision made at node j and time t on whether to leave U at the next time point $t+1$. These decision probabilities can differ with j.

Given a starting location $x_{t_0} = j_0 \in U$ at some time t_0, drawing the first entrance edge into $\bar{U}$ at $t_0 + \delta$ for some fixed $\delta \in \mathbb{Z}_+$ can be understood as the result of the events

$$(e_{j_0}^{t_0} = 0) \Rightarrow (x_{t_0+1} = j_1 \in U) \Rightarrow (e_{j_1}^{t_0+1} = 0) \Rightarrow \cdots \Rightarrow (x_{t_0+\delta} = j' \in U) \Rightarrow (e_{j'}^{t_0+\delta} = 1),$$

where $A \Rightarrow B$ represents the event B occurring after A. The trajectory of the walk stays within U from time t_0 until $t_0 + \delta$ and exits from node $j' \in U$ to node $l' \in \bar{U}$ at time $t_0 + \delta + 1$. This implies that the first entrance edge $j' \to l'$. We write

$$\eta_j = \Pr(e_j^t = 1 | x_t = j), \qquad 1 - \eta_j = \Pr(e_j^t = 0 | x_t = j) = \frac{\sum_{k \in U} w_{j,k}}{d_j}.$$

Collect the η into vector form $\eta_U = (\eta_j : j \in U)$. Let $P_{U,U} = \{p_{j,l} : j \in U, l \in U\}$ denote the submatrix of transition probabilities from nodes in U back into U. Writing a state vector $s_t = \{\Pr(x_t = j' | x_{t_0} = j_0, x_{\tilde{t}} \in U, \tilde{t} < t)\}_{j' \in U}$, we have the recursive relation $s_{t+1} = P_{U,U}^{\mathrm{T}} s_t$, when the walk has not left U by time $t + 1$. The initial state s_{t_0} is a $|U|$-dimensional vector with the element corresponding to j_0 set to 1 and all others equal to 0. For notational ease, we omit $x_{\tilde{t}} \in U$, $\tilde{t} < t_0 + \delta$, in the conditioning below. We then have the vector

$$\{\Pr(x_{t_0+\delta} = j', x_{t_0+\delta+1} \in \bar{U} | x_{t_0} = j_0)\}_{j' \in U} = \mathrm{diag}(\eta_U)(P_{U,U}^{\mathrm{T}})^{\delta-1} s_{t_0}.$$

Here, we started at the initial state at time t_0, transitioned within the visited nodes U for $\delta - 1$ times and visited $\bar{U}$ at time $t_0 + \delta + 1$ via $\mathrm{diag}(\eta_U)$. This expression yields a distribution on the nodes j' that $x_{t_0+\delta}$ can take. We can further marginalize the above by summing over $\delta \in \mathbb{Z}_{\geqslant 0}$. Since the summation involves a Neumann series, it has a closed form for the marginal if the series converges. The following theorem shows a sufficient condition.

THEOREM 4. *If $P_{U,U}$ is irreducible (cannot be rearranged to a block upper triangular matrix by row and column permutations) and there exists at least one $j \in U$: $\eta_j > 0$, then*

$$\{\Pr(x_{(\hat{t}_{|U|+1}-1)} = j' | x_{t_0} = j_0)\}_{j' \in U} = \mathrm{diag}(\eta_U)(I - P_{U,U}^{\mathrm{T}})^{-1} s_{t_0}, \tag{5}$$

where $(\hat{t}_{|U|+1} - 1)$ is the time point before moving to a node in $\bar{U}$.

The above irreducibility condition is satisfied when $w_{j,l} > 0$ for all $j \neq l$. This theorem allows us to sample the first exit edge (j', l') from U to $\bar{U}$ in a straightforward manner. We first draw $j' \in U$ according to the distribution specified by the vector in (5). We then take a random step from the drawn node j' via the transition specified by

$$\Pr(x_{\hat{t}_{|U|+1}} = l' | x_{(\hat{t}_{|U|+1}-1)} = j', e_j^{(\hat{t}_{|U|+1}-1)} = 1) = \frac{w_{j',l'}}{\sum_{k \in \bar{U}} w_{j',k}} \tag{6}$$

to obtain $l' \in \bar{U}$. We refer to steps (5) and (6) as the *fast-forwarding steps*.

Remark 3. The direct calculation of the matrix inverse $(I - P_{U,U}^{\mathrm{T}})^{-1}$ has $O(|U|^3)$ cost, but solving the linear equation x: $(I - P_{U,U}^{\mathrm{T}})x = s_{t_0}$ using iterative numerical methods can lead to a lower cost $O(\tilde{K}|U|)$, with $\tilde{K}$ the number of iterations until convergence (Saad, 2003).

A natural algorithm for drawing a spanning tree performs Aldous–Broder-type random walk steps until a bottleneck is reached, and then uses fast-forwarding steps; this is then repeated until all nodes are visited. We call this approach the *fast-forwarded cover algorithm*.

Algorithm 1. The fast-forwarded cover algorithm.

Initialize $x_0 = r$. Initialize the first entrance step tracker $\alpha = 1$. Initialize $\tau = 1$.
1. Simulate one random walk step to x_τ.
2. If x_τ has not been visited before, update $\alpha \leftarrow \tau$.
3. If the steps taken since the first entrance step $(\tau - \alpha) \geqslant \kappa_0$, a preset threshold,
   update $x_{\tau+1}$ to j' sampled according to (5),
   update $x_{\tau+2}$ to l' sampled according to (6),
   update $\alpha \leftarrow \tau + 2$ and $\tau \leftarrow \tau + 3$. Go to step 1.

If $(\tau - \alpha) < \kappa_0$, update $\tau \leftarrow \tau + 1$. Go to step 1.
Repeat the above until all $\alpha = m$, where m is the number of nodes of the underlying graph.
Collect and return the first entrance edges.

We use a new index τ above, since the iteration number will be different from the underlying random walk time index t as soon as a fast-forwarding step is used. The algorithm completes when all nodes have been visited, and the number of iterations required is at most $\kappa_0(m-1)$. In this article, we set $\kappa_0 = 1000$, which leads to excellent performance in all our experiments. More generally, since the fast-forwarding step has complexity $O(\tilde{K}|U|)$, it is natural to consider a varying threshold $\kappa_\tau \propto m_\tau$, where m_τ is the number of visited nodes $|U|$ by iteration τ.

Before presenting our empirical results, we generalize the random walk cover and our fast-forwarded modifications to be broadly applicable to sampling any directed spanning tree with probability in the form of (3). This is accomplished through a careful specification of the transition matrix P, or, equivalently, W up to row-wise normalization.

2.4. *Sampling from the root distribution and applicability of cover algorithms*

Given a graph with weighted adjacency matrix Q, our goal is to draw samples from (3) using cover algorithms, a terminology we use to include random walk cover and our fast-forwarded modifications. We now discuss how to sample $\Pr(r) \propto g_r/\rho_r$. Two cases are of interest.

Case 1 *(Circulation)*. We start with the canonical case in which a simple condition $\sum_{l=1}^m q_{j,l} = \sum_{k=1}^m q_{k,j}$ holds for $j = 1, \ldots, m$. If we view $q_{j,l}$ as a flow from node j to l, these equalities describe a type of flow conservation, with Q describing a circulation matrix (Chung, 2005). This includes the special case where Q is symmetric. Under this circulation condition, setting the random walk transition probabilities $w_{j,l}$ as equal to the underlying graph's edge weights $q_{j,l}$ is sufficient for cover algorithms to produce samples satisfying

$$\Pr(\vec{\mathcal{T}}|r) \propto \prod_{(j\to l)\in E_{\vec{\mathcal{T}}}} q_{j,l},$$

corresponding to $\rho_r \propto 1$. Therefore, we can simply draw the root from $\Pr(r) \propto g_r$, and use a cover algorithm with $W = Q$ to obtain samples distributed according to (3).

Case 2 *(General form)*. For a general and irreducible $Q \in [0,\infty)^{m\times m}$, the stationary distribution vector π is a deterministic transform of a left eigenvector of the transition probability

matrix P. Here, $(p_{j,l}\pi_j) \propto q_{j,l}$ may not hold for all (j,l). To solve this problem, we introduce an auxiliary Markov chain as a mathematical tool for calculating P. Consider a chain $Y = (y_0, y_1, \ldots)$ over V_G with transition kernel

$$\tilde{p}_{l,j} = \Pr(y_{t+1} = j | y_t = l) = \frac{q^*_{l,j}}{\sum_{k=1}^m q^*_{l,k}}, \qquad q^*_{l,j} = q_{j,l}.$$

We denote the invariant distribution of $y_t \colon t \to \infty$ by $(\pi^*_1, \ldots, \pi^*_m)$, which can be numerically computed as a left eigenvector of the transition matrix formed by the $\tilde{p}_{l,j}$.

We now form the chain $X = (x_0, x_1, \ldots)$ with the transition probability specified as

$$p_{j,l} = \frac{\tilde{p}_{l,j}\pi^*_l}{\pi^*_j}. \tag{7}$$

Reversibility ($p_{j,l} = \tilde{p}_{j,l}$) is not needed; however, we can see that $\sum_j p_{j,l}\pi^*_j = \pi^*_l$ for all l. Therefore, X has the same invariant distribution $(\pi_1, \ldots, \pi_m) = (\pi^*_1, \ldots, \pi^*_m)$ as Y. As a result, we can run a cover algorithm using P from (7). The probability in (4) from Theorem 1 becomes

$$\Pr(\vec{\mathcal{T}}|r) \propto \prod_{(j\to l)\in E_{\vec{\mathcal{T}}}} \tilde{p}_{l,j}\pi^*_l = \prod_{(j\to l)\in E_{\vec{\mathcal{T}}}} \frac{q_{j,l}}{\sum_{k=1}^m q_{k,l}}\pi^*_l \propto \left(\prod_{(j\to l)\in E_{\vec{\mathcal{T}}}} q_{j,l}\right)\left(\frac{\sum_{k=1}^m q_{k,r}}{\pi^*_r}\right),$$

using the invariance to $\vec{\mathcal{T}}$ implied by

$$\frac{\pi^*_r}{\sum_{k=1}^m q_{k,r}} \prod_{(j\to l)\in E_{\vec{\mathcal{T}}}} \frac{\pi^*_l}{\sum_{k=1}^m q_{k,l}} = \prod_{l=1}^m \frac{\pi^*_l}{\sum_{k=1}^m q_{k,l}}.$$

The normalizing constant is now varying with r, $\rho_r \propto (\sum_{k=1}^m q_{k,r}/\pi^*_r)$. Therefore, we draw the root from $\Pr(r) \propto g_r/\rho_r$, and then use a cover algorithm to draw from $\Pr(\vec{\mathcal{T}}|r)$.

3. SIMULATIONS

To investigate the gain in computational efficiency of our fast-forwarded cover algorithm over competitors, we conduct several simulations in `R` and compare the run times of various spanning tree sampling algorithms on a machine equipped with an Apple M4 Max chip.

We compare the fast-forwarded cover algorithm against other random walk-based competitors, namely, the Aldous–Broder algorithm and Wilson's algorithm. First, we assess the impact of the bottleneck size, quantified via $1/\{\lambda_2(\mathcal{L})\}^{1/2}$, on wall-clock runtime. We simulate a graph with $m = 500$ nodes, where we partition the nodes V into two blocks with $|V_1| = |V_2| = 250$. The graph is represented by a symmetric weight matrix $W \in \mathbb{R}^{500\times 500}$ with each $w_{j,l} = u_{j,l}b_{j,l}$. The factor $u_{j,l}$ representing the size of the weight is simulated from $\mathrm{Un}(0,1)$. We set $b_{j,l} = |V_1|^2$ for those j and l that are in the same node partition, and $b_{j,l} \sim \mathrm{Ber}(\zeta)$ for those j and l that are in different partitions. This creates two complete subgraphs connected by a small number of edges having small weight. We perform experiments at different edge densities with ζ from $(0.5, 0.1, 0.05, 0.01)$, corresponding to a range of bottleneck values near $1/\{\lambda_2(\mathcal{L})\}^{1/2} = (249, 560, 786, 1738)$, corresponding to graphs 1 to 4 in Fig. 3(a). Under each value of ζ, we draw 10 spanning trees from each of the three

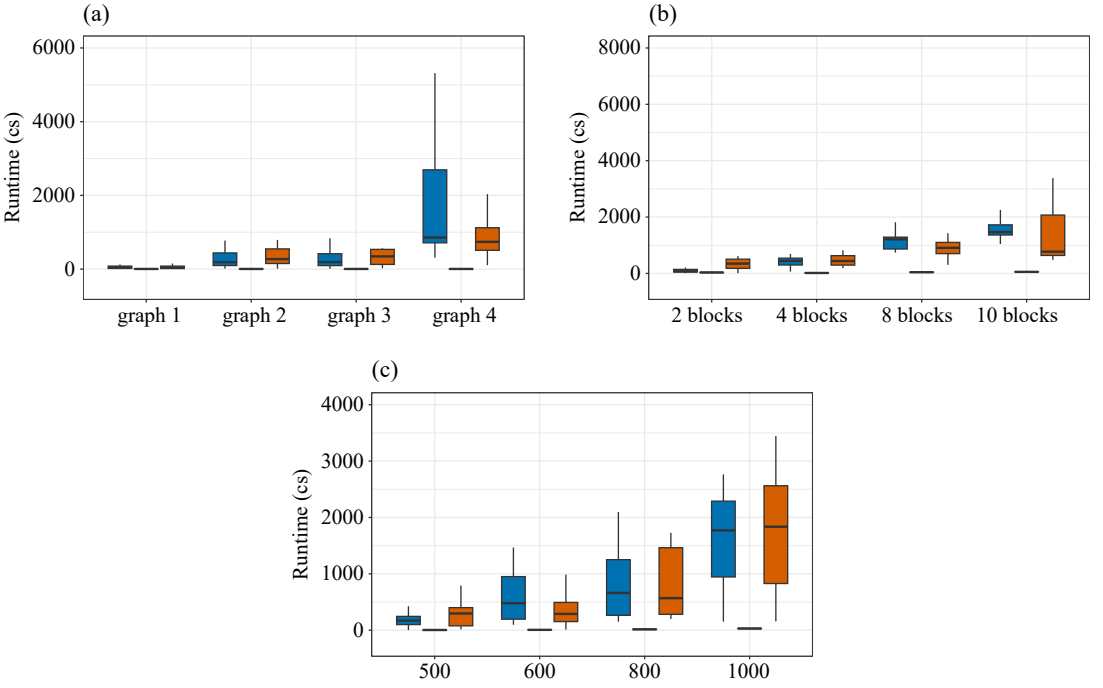

Fig. 3. Comparisons of the runtime between the Aldous–Broder algorithm (blue), fast-forwarded cover algorithm (grey) and Wilson's algorithm (orange). In each setting, the fast-forwarded algorithm runtime has a small mean and variance, so each grey box appears close to a thin line. (a) Runtime for different bottleneck sizes $1/\{\lambda_2(\mathcal{L})\}^{1/2}$. (b) Runtime over different numbers of blocks and hence bottlenecks. (c) Runtime over different numbers of nodes.

algorithms under comparison. Figure 3(a) shows that while the runtimes for the Aldous–Broder algorithm and Wilson's algorithm increase rapidly as the bottleneck size increases, the fast-forwarded algorithm runtime is almost unaffected by bottleneck size. Similar plots are shown in the Supplementary Material, where we compare the Aldous–Broder algorithm, Wilson's algorithm and the fast-forwarded algorithm under the same setting, but measure time by the number of random walk steps taken.

Second, we assess the effects of the number of blocks on the runtime; more blocks implies more bottlenecks. We simulate $W \in \mathbb{R}^{600\times 600}$ according to $w_{j,l} = u_{j,l}b_{j,l}c_{j,l}$. The factors $u_{j,l}$ and $b_{j,l}$ are simulated as above, except that we partition the 600 nodes into K blocks $(V_1, \ldots, V_K)$ each of node size $(600/K)$, and use $b_{j,l} = b_{l,j} \sim \text{Ber}(0.001)$ and $c_{j,l} = 0.005/K$ for those $(j,l)\colon j \in V_k, l \in V \setminus V_k$. The factors $c_{j,l}$ are chosen so that empirically the size of $1/\{\lambda_2(\mathcal{L})\}^{1/2}$ remains roughly the same. We vary K in $(2, 4, 8, 10)$, and draw 10 spanning trees using each of the algorithms under comparison. Figure 3(b) shows that runtimes of the Aldous–Broder and Wilson's algorithms increase quickly as the number of blocks increases, while the fast-forwarded algorithm is again almost unaffected.

Third, we assess scalability by running the samplers on graphs with numbers of nodes ranging within $(500, 600, 800, 1000)$ in the two-block case using $b_{j,l} = b_{l,j} \sim \text{Ber}(0.1)$. Figure 3(c) shows that the runtimes for the Aldous–Broder and Wilson's algorithms increase much faster than the runtime of the fast-forwarded cover algorithm. In the Supplementary Material, we present a comparison with a Laplacian-based sampler outlined in Algorithm 1 of Harvey & Xu (2016). While the Laplacian-based sampler offers improved mixing against the reversible-jump sampler, the fast-forwarded cover algorithm enjoys substantially better mixing.

4. BAYESIAN DENDROGRAM INFERENCE FOR CRIME AND COMMUNITY DATA

We demonstrate the use of our algorithm in inferring a Bayesian dendrogram. We consider a hierarchical model in the form of (1) for continuous $y_i \in \mathbb{R}^{\tilde{d}}$ with $\vec{\mathcal{T}}$ rooted at node 1. We choose

$$\mathcal{F}(y_i|\mu_{z_i}) = \phi(y_i; \mu_{z_i}, \Sigma), \qquad \mathcal{H}(\mu_l|\mu_j, \Sigma) = \phi(\mu_l; \mu_j, \lambda^{-1}\Sigma).$$

By symmetry of kernel $\mathcal{H}$, the underlying graph G is undirected. This leads to the hierarchical model

$$L(y|\mu, z) = \prod_{i=1}^{n} \phi(y_i; \mu_{z_i}, \Sigma), \qquad \Pi(\mu|E_{\vec{\mathcal{T}}}, r) = R(\mu_r) \prod_{(j\to l)\in E_{\vec{\mathcal{T}}}} \phi(\mu_l; \mu_j, \lambda^{-1}\Sigma)$$

with priors $\Pi_0(\vec{\mathcal{T}}|r = 1) \propto 1$, $z_i \overset{\text{i.i.d.}}{\sim} \text{Categorical}(\tilde{\pi}_1, \ldots, \tilde{\pi}_{\tilde{K}})$, $(\tilde{\pi}_1, \ldots, \tilde{\pi}_{\tilde{K}}) \sim \text{Dir}(\alpha, \ldots, \alpha)$, $\Sigma \sim \text{W}^{-1}(\nu, \Sigma_0)$, $R(\mu_1 = 0) = 1$. Since $\mathcal{F}$ and $\mathcal{H}$ have a shared covariance Σ up to a scale change $\lambda > 0$, a conjugate normal-inverse Wishart prior can be chosen for μ and Σ.

Our interest is inferring the posterior on $\vec{\mathcal{T}}$, but we also have uncertainty in allocations z_i and component-specific parameters (μ_k). The resulting joint posterior distribution is highly complex. Because of computational challenges, the literature tends to avoid characterizing the uncertainty in $\vec{\mathcal{T}}$. For example, for Bayesian hierarchical clustering, Heller & Ghahramani (2005) used a hypothesis testing criterion to iteratively merge clusters, while Heard et al. (2006) combined clusters based on a metric that captures the closeness between clusters. Alternatively, one can use a Markov chain Monte Carlo sampling algorithm. Classical algorithms rely on making local changes in $\vec{\mathcal{T}}$ using reversible-jump Metropolis–Hastings (Chipman et al., 1998; Denison et al., 1998), which tends to be very inefficient. Wu et al. (2007) considered adding a move that allows a larger tree to be restructured to improve mixing, but with high per iteration expense for large graphs.

Our new spanning tree sampler can bypass these computational challenges. We take an overfitted modelling approach, considering an encompassing tree $\vec{\mathcal{T}}$ with $\tilde{m}$ nodes rooted at 1, with $\tilde{m}$ sufficiently large to provide an upper bound on the true value $\tilde{m} \geqslant m$. This allows us to build a blocked Gibbs sampler based on drawing from $\Pi(z|y, \mu, \vec{\mathcal{T}})$, $\Pi(\mu|y, z, \vec{\mathcal{T}})$ and $\Pi(\vec{\mathcal{T}}|\mu)$, with $\vec{\mathcal{T}}$ a spanning tree for $\tilde{m}$ nodes, in addition to the steps of updating other parameters.

After obtaining a posterior sample, we can marginalize redundant nodes and change each sampled spanning tree $\vec{\mathcal{T}}$ to a reduced dendrogram $\vec{T}$, while maintaining an equivalent generative model for the data. Given a sample of $(z_1, \ldots, z_n)$, and $\vec{T}$ initialized at $\vec{\mathcal{T}}$, we use the following pruning procedures corresponding to integrating out the densities related to j.

(i) If j is an empty leaf node, without any downstream edge $(j \to l)$ and with $\sum_{i=1}^{n} 1(z_i = j) = n_j = 0$, we remove j and $(k \to j)$ from $\vec{T}$.
(ii) If j only has two edges $(k \to j)$ and $(j \to l)$, and $n_j = 0$, we remove j and replace the two edges by $(k \to l)$ in $\vec{T}$.

We iterate the above pruning steps on the tree's nodes until we cannot reduce the size of $\vec{T}$ any further. The spanning tree $\vec{\mathcal{T}}$ can be viewed as a latent variable that facilitates the specification of the model and posterior computation for the dendrogram $\vec{T}$. We refer to this approach as a *spanning tree-augmented dendrogram.*

To illustrate this model, we consider an application to the community and crime dataset from the University of California Irvine Machine Learning Repository. The dataset consists of socioeconomic attributes from the 1990 U.S. census, crime statistics from the 1995 Uniform Crime Report and law enforcement attributes from the 1990 Law Enforcement Management and Administrative Statistics Survey. We focus on economic data from the state of Massachusetts, where there are $n = 123$ relevant entries, each corresponding to a community in the state, with $d = 2$ continuous attributes: the community's median income and median rent. These attributes are log transformed and standardized to have sample mean 0 and marginal sample variance 1. Our focus is on characterizing variability in these economic attributes by grouping the communities hierarchically. A dendrogram is natural for this purpose.

We choose priors to favour node parameters μ_j that are broadly spread across the support of the data, with relatively few data points associated with each μ_j. This is achieved by choosing $\nu = n$, $\Sigma_0 = 0.2^2 I_{\tilde{d}}$, $\lambda = 0.25$ and $\tilde{m} = \lfloor n/4 \rfloor$. To favour effective elimination of unnecessary clusters, we follow common practice in the literature on overfitted mixtures (Van Havre et al., 2015) and choose a symmetric Dirichlet with a small concentration parameter ($\alpha = 0.1$) for weights $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\tilde{m}}$. Since the data have been centred, we fix root choice $r = 1$ and $\mu_1 = 0$. There are implicit Bayesian Occam's razor effects that favour the induced dendrogram $\vec{T}$ to be small. As in other Bayesian mixture models, the marginal likelihood will tend to decrease if data are overclustered, favouring setting $n_k = 0$ to automatically remove some of the clusters. This tendency is furthered by our symmetric Dirichlet prior with precision close to zero for the cluster weights. In addition, the uniform prior on $\vec{\mathcal{T}}$ leads to higher weight on dendrograms with few nodes as such dendrograms have more ways to be marginalized (pruned) from an $\tilde{m}$-node $\vec{T}$.

Using a Gibbs sampler, we can update each term above using closed-form full conditional distributions. We provide the details in the Supplementary Material. To show computational advantages of this Gibbs sampler, we compare with sampling the posterior for a directly specified dendrogram using a reversible-jump Markov chain Monte Carlo sampler. The model is almost the same as our spanning tree-augmented version, with the same choice of $\mathcal{F}$ and $\mathcal{G}$, priors for the parameters and upper bound $\tilde{m}$ on the number of nodes in $\vec{T}$. However, we allow the number of nodes in $\vec{T}$ to vary; therefore, some nodes amongst $(1, \ldots, \tilde{m})$ might not be in $\vec{T}$. In the likelihood, we replace $\mathcal{F}(y_i|\mu_k)$ by zero if k is not a node of $\vec{T}$, and use the prior $\Pi_0(\vec{T}) \propto 0.01^{|V_{\vec{T}}|}$ to favour small dendrograms. Accordingly, we use birth/death proposals to add/remove nodes from $\vec{T}$, and a Metropolis–Hastings criterion to accept or reject each proposal. We provide details in the Supplementary Material.

We run each sampler for 5000 iterations, with a burn-in of 3500 iterations. In wall-clock time, both the Gibbs sampler using our fast-forward cover algorithm and the reversible-jump Markov chain Monte Carlo sampler run for approximately 0.5 to 1 h. However, there are dramatic differences in mixing performance. The reversible-jump Markov chain Monte Carlo sampler tends to be stuck in certain states for a long time, while the spanning tree–based Gibbs sampler shows excellent mixing. Figure 4 compares the mixing for the number of nonempty leaves of the dendrogram. The Gibbs sampler for a spanning tree-augmented dendrogram model shows much faster mixing, compared to a reversible-jump sampler for a directly specified dendrogram model. The posterior distributions targeted by the Gibbs sampler and reversible-jump sampler are slightly different due to how the complexity of the tree is regulated. Results for other summaries are shown in the Supplementary Material. We compare effective sample sizes per iteration in Table 1. We further conducted posterior

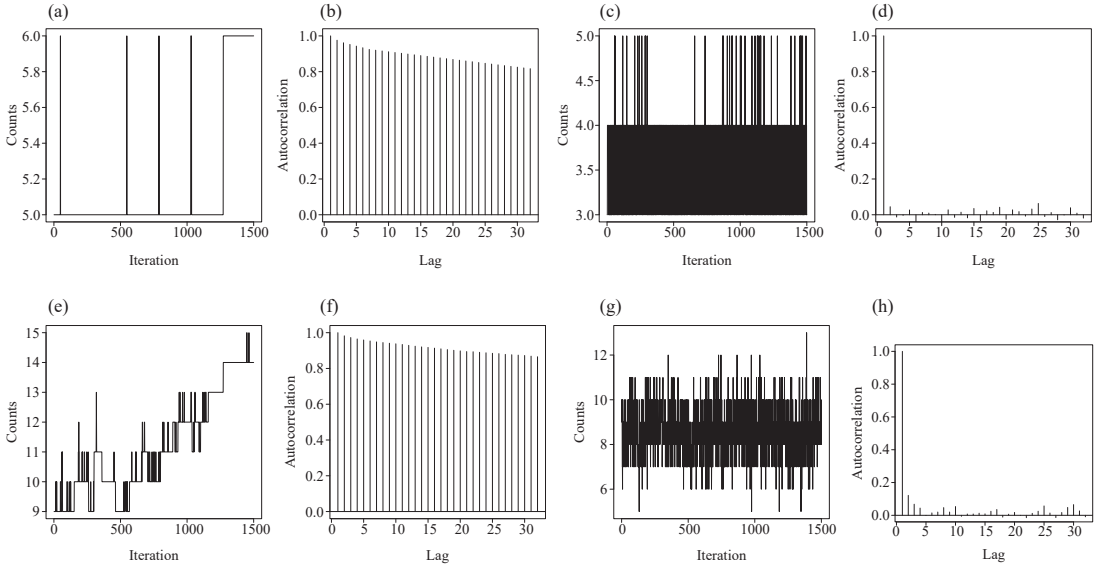

Fig. 4. (a) Trace of the maximum degree from the reversible-jump sampler. (b) Autocorrelation of the maximum degree for the reversible-jump sampler. (c) Trace of the maximum degree from the Gibbs sampler. (d) Autocorrelation of the maximum degree from the Gibbs sampler. (e) Trace of the number of nonempty leaves for the reversible-jump sampler. (f) Autocorrelation of the number of nonempty leaves for the reversible-jump sampler. (g) Trace of the number of nonempty leaves from the Gibbs sampler. (h) Autocorrelation of the number of nonempty leaves from the Gibbs sampler.

Table 1. *Effective sample size per iteration for the inferred tree from the reversible-jump sampler and our proposed spanning tree–augmented dendrogram Gibbs sampler*

Parameter	Gibbs sampler for the spanning tree–augmented dendrogram	Reversible-jump sampler
Maximum degree	0.913	0.003
Maximum depth	0.29	0.007
Number of leaves	0.702	0.002

sampling using the subtree prune and regraft move (Evans & Winter, 2005; Song, 2006). The results are provided in the Supplementary Material.

To quantify the uncertainty around the obtained clusters and visualize the inferred hierarchical structures, we use the posterior similarity matrix (Fritsch & Ickstadt, 2009). We record whether communities share an ancestor node at depths 1, 2 and 3 of the sampled dendrogram, and average over the posterior samples to compute a probability for each such pairing. The results are shown in heat maps in Fig. 5. One can observe clear block-diagonal structures at all depths, which become increasingly noisy as the depth increases.

To interpret the inferred clusters, we visualize the communities in Massachusetts on a map. We identified the largest diagonal block obtained from the posterior similarity matrix at depth 1 and coloured the map by membership. We juxtapose the cluster membership map with another map coloured by whether the median rent in the community is above the threshold of $550 in Fig. 6. There is a nearly identical correspondence between the two maps, suggesting that the clustering faithfully captures the variation in the data. We also observed a visible concentration of such higher income and rent communities in eastern Massachusetts

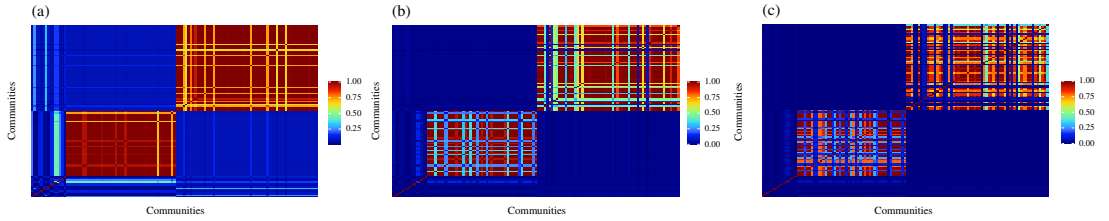

Fig. 5. Posterior similarity matrices at different depths for the crime and community data: (a) depth 1, (b) depth 2, (c) depth 3.

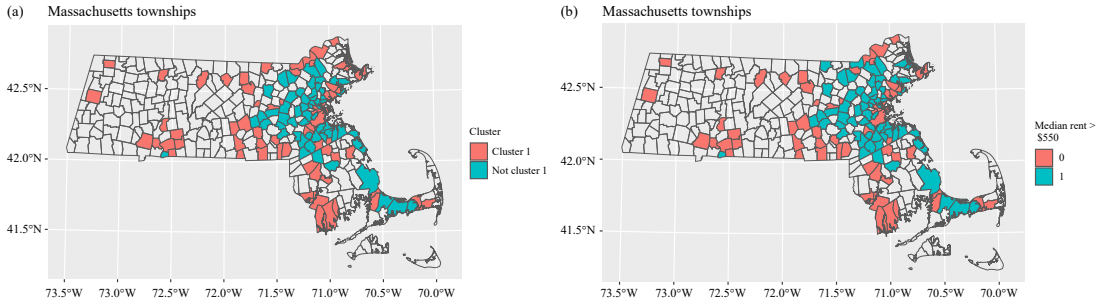

Fig. 6. (a) Map plot of Massachusetts communities, coloured by membership of a depth-1 cluster. (b) Map plot of Massachusetts communities, coloured by whether median rent is above threshold. The communities in the largest cluster, red in panel (a), at depth 1 of the estimated dendrogram roughly match with those with median rent above $550 in panel (b).

near the coast. Details for computing posterior similarity matrices and maps are provided in the Supplementary Material.

5. Discussion

This article is motivated by improving the computational efficiency of posterior inference for tree parameters. Expanding beyond the application to dendrograms, it is of interest to develop spanning tree-augmented models for more complex models, such as more elaborate discrete latent structure models (Zeng et al., 2023) or the treed Gaussian process (Gramacy & Lee, 2008; Payne et al., 2024). For broader classes of tree models that may not admit a product-over-edge probability distribution, one may modify our random walk cover algorithm to form a computationally efficient proposal-generating distribution. Related Metropolis–Hastings algorithms for sampling graph partitions have recently been studied by Autry et al. (2023). For a tree with an unknown number of nodes or structural dependence on a latent arrival process, such as diffusion trees (Neal, 2003) or Bayesian phylogenetic trees (Huelsenbeck & Ronquist, 2001), it would be interesting to explore extending the fast-forwarding algorithm to bypass wasteful random walks on an infinite graph or under time-varying transition probabilities.

Acknowledgement

This work was partially supported by Merck, the European Research Council, the Office of Naval Research, the National Institutes of Health and the National Science Foundation of the United States.

SUPPLEMENTARY MATERIAL

The Supplementary Material contains further simulations. All codes for this paper can be found at www.github.com/edrictam/fast_spanning_tree_sampler.

APPENDIX

Proof of Theorem 2. Considering $(\hat{t}_{k+1} - \hat{t}_k)$ as the interarrival time, we want to show that

$$\mathbb{E}(\hat{t}_{k+1} - \hat{t}_k) \geqslant 1/\tilde{q}, \qquad \tilde{q} = \left\{1 + \min_{j \in U_k} \frac{\sum_{l \in U_k} w_{j,l}}{\sum_{l' \in \bar{U}_k} w_{j,l'}}\right\}^{-1}.$$

Consider a sequence of independent Bernoulli events with success probabilities $\tilde{p}_1, \tilde{p}_2, \ldots$, with all $\tilde{p}_{\tilde{t}} \leqslant \tilde{q}$, and denote the index on the first success in this sequence by $\tilde{T}$. If $\mathbb{E}\tilde{T} < \infty$ then $\mathbb{E}\tilde{T} \geqslant 1/\tilde{q}$. Since $\tilde{T} \geqslant 0$, we know that $\mathbb{E}\tilde{T} = \sum_{t=0}^{\infty} \Pr(\tilde{T} > t) = \sum_{t=1}^{\infty} \prod_{\tilde{t}=1}^{t} (1 - \tilde{p}_{\tilde{t}}) \geqslant \sum_{t=1}^{\infty} \prod_{\tilde{t}=1}^{t} (1 - \tilde{q}) = 1/\tilde{q}$. Adding over $\hat{t} = \sum_{k=1}^{m} (\hat{t}_{k+1} - \hat{t}_k)$ with $\hat{t}_1 = 0$ yields the result. □

Proof of Theorem 3. We first state Cheeger's inequality for circulation graphs (Chung, 2005, Theorem 5.1). For a directed and weighted graph with nonnegative weight matrix W, having $\sum_{l=1}^{m} w_{j,l} = \sum_{l=1}^{m} w_{l,j}$ for all j, the second smallest eigenvalue $\lambda_2(\mathcal{L})$ satisfies

$$\{\lambda_2(\mathcal{L})\}^{1/2} \geqslant \min_{\{U:\, 1 \leqslant |U| \leqslant m-1\}} \frac{\sum_{j \in U, l \in \bar{U}} w_{j,l}}{\min(\sum_{j \in U} d_j, \sum_{j \in \bar{U}} d_j)}.$$

Letting $U = \bigcup_{\tilde{t} \leqslant t} X_{\tilde{t}}$, $|U| \leqslant m - 1$, the probability of exiting U at time $t + 1$ is

$$\Pr(x_{t+1} \in \bar{U} | x_{\tilde{t}} \in U \text{ for all } \tilde{t} \leqslant t) = \sum_{j \in U} \left\{ \frac{\sum_{l \in \bar{U}} w_{j,l}}{d_j} S_t(j) \right\},$$

where $S_t(j) = \Pr(x_t = j)$. Using $\tilde{e}_j = \sum_{l \in \bar{U}} w_{j,l}$, we obtain

$$\begin{aligned}
\left[\sum_{j \in U} \left\{ \frac{\tilde{e}_j}{d_j} S_t(j) \right\} \right] \sum_{j \in U} d_j &= \sum_{j \in U} \left\{ \tilde{e}_j S_t(j) \frac{\sum_{j' \in U} d_{j'}}{d_j} \right\} \\
&\leqslant \sum_{j \in U} \tilde{e}_j \max_{j \in U} \left\{ S_t(j) \frac{\sum_{j' \in U} d_{j'}}{d_j} \right\} \\
&\leqslant \left(\sum_{j \in U} \tilde{e}_j \right) \max_{j \in U \cup \bar{U}} \left\{ S_t(j) \frac{\sum_{j' \in U \cup \bar{U}} d_{j'}}{d_j} \right\}.
\end{aligned}$$

After rearranging terms, we obtain

$$\begin{aligned}
\Pr(x_{t+1} \in \bar{U} | x_{\tilde{t}} \in U \text{ for all } \tilde{t} \leqslant t) &\leqslant \frac{\sum_{j \in U} \tilde{e}_j}{\sum_{j \in U} d_j} \max_{j \in U \cup \bar{U}} \left\{ S_t(j) \frac{\sum_{j'} d_{j'}}{d_j} \right\} \\
&\leqslant \frac{\sum_{j \in U} \tilde{e}_j}{\min(\sum_{j \in U} d_j, \sum_{j \in \bar{U}} d_j)} \max_{j \in U \cup \bar{U}} \left\{ S_t(j) \frac{\sum_{j'} d_{j'}}{d_j} \right\}.
\end{aligned}$$

Using $\pi_j = d_j / \sum_{j'} d_{j'}$ and taking the minimum over both sides, we have

$$\min_{U:\, |U| \leqslant m-1} \Pr(x_{t+1} \in \bar{U} | x_{\tilde{t}} \in U \text{ for all } \tilde{t} \leqslant t) \leqslant \{\lambda_2(\mathcal{L})\}^{1/2} \max_j S_t(j)/\pi_j.$$

Letting U_{k^*} reach the minimum on the left-hand side, by a similar argument as in the proof of Theorem 2,

$$\mathbb{E}(\hat{t}_{k^*+1} - \hat{t}^*_k) \geqslant \frac{1}{M\{\lambda_2(\mathcal{L})\}^{1/2}}.$$

Adding the other $(m-2)$ steps, each with $\hat{t}_{k'+1} - \hat{t}_{k'} \geqslant 1$, leads to the result. □

Proof of Theorem 4. The Neumann series converges if the spectral radius $\lambda_1(P_{U,U}) < 1$ strictly. Since $P_{U,U}$ is a nonnegative matrix, by the Perron–Frobenius theorem, $\lambda_1(P_{U,U}) \leqslant \max_i \sum_j p_{i,j} \leqslant 1$.

Since $P_{U,U}$ is irreducible, by the Perron–Frobenius theorem, there exists a unique vector $P^{\mathrm{T}}_{U,U}\phi_* = \lambda_1(P_{U,U})\phi_*$, with ϕ_* all positive and $1^{\mathrm{T}}\phi_* = 1$. We follow Chapter 8 of Meyer (2023), and let Q be a nonnegative matrix such that $P^{\mathrm{T}}_{U,U} + Q$ has each column summable to 1; hence, $1^{\mathrm{T}}(P^{\mathrm{T}}_{U,U} + Q)\phi \leqslant 1$ for any ϕ all positive and $1^{\mathrm{T}}\phi = 1$. Since there exists at least $\eta_j > 0$, we know that at least one $Q_{j,l} > 0$.If $\lambda_1(P_{U,U}) = 1$, we would have

$$1^{\mathrm{T}}(P^{\mathrm{T}}_{U,U} + Q)\phi_* = 1^{\mathrm{T}}\phi_* + 1^{\mathrm{T}}Q\phi_* > 1,$$

which is a contradiction. Therefore, we know that $\lambda_1(P_{U,U}) < 1$. □

References

Aldous, D. J. (1990). The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM J. Discrete Math.* **3**, 450–65.

Autry, E., Carter, D., Herschlag, G. J., Hunter, Z. & Mattingly, J. C. (2023). Metropolized forest recombination for Monte Carlo sampling of graph partitions. *SIAM J. Appl. Math.* **83**, 1366–91.

Broder, A. Z. (1989). Generating random spanning trees. In *30th Ann. Symp. Foundat. Comp. Sci.*, pp. 442–47. Piscataway, NJ: IEEE Press.

Broder, A. Z. & Karlin, A. R. (1989). Bounds on the cover time. *J. Theor. Prob.* **2**, 101–20.

Castillo, I. & Ročková, V. (2021). Uncertainty quantification for Bayesian CART. *Ann. Statist.* **49**, 3482–509.

Chipman, H. A., George, E. I. & McCulloch, R. E. (1998). Bayesian CART model search. *J. Am. Statist. Assoc.* **93**, 935–48.

Chipman, H. A., George, E. I. & McCulloch, R. E. (2010). BART: Bayesian additive regression trees. *Ann. Appl. Statist.* **4**, 266–98.

Chung, F. (2005). Laplacians and the Cheeger inequality for directed graphs. *Ann. Combinat.* **9**, 1–19.

Denison, D. G., Mallick, B. K. & Smith, A. F. (1998). A Bayesian CART algorithm. *Biometrika* **85**, 363–77.

Duan, L. L. & Dunson, D. B. (2023). Bayesian spanning tree: estimating the backbone of the dependence graph. *J. Mach. Learn. Res.* **24**, 1–44.

Duan, L. L. & Roy, A. (2024). Spectral clustering, Bayesian spanning forest, and forest process. *J. Am. Statist. Assoc.* **119**, 2140–53.

Elidan, G. & Gould, S. (2008). Learning bounded treewidth Bayesian networks. In *Advances in Neural Information Processing Systems*, vol. **21**, Ed. D. Koller, D. Schuurmans, Y. Bengio and L. Bottou, pp. 1–8. Red Hook, NY: Curran Associates.

Evans, S. N. & Winter, A. (2005). Subtree prune and regraft: a reversible real tree-valued Markov process. *Ann. Prob.* **34**, 918–61.

Fredes, L. & Marckert, J.-F. (2023). A combinatorial proof of Aldous–Broder theorem for general Markov chains. *Random Struct. Algor.* **62**, 430–49.

Fritsch, A. & Ickstadt, K. (2009). Improved criteria for clustering based on the posterior similarity matrix. *Bayesian Anal.* **4**, 367–91.

Gower, J. C. & Ross, G. J. (1969). Minimum spanning trees and single linkage cluster analysis. *J. R. Statist. Soc. C* **18**, 54–64.

Gramacy, R. B. & Lee, H. K. H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. *J. Am. Statist. Assoc.* **103**, 1119–30.

Guenoche, A. (1983). Random spanning tree. *J. Algor.* **4**, 214–20.

Harvey, N. J. A. & Xu, K. (2016). Generating random spanning trees via fast matrix multiplication. In *LATIN 2016: Theoretical Informatics Lecture Notes Comp. Sci.* ***9644***), Ed. E. Kranakis, G. Navarro and E. Chávez, pp. 522–35. Berlin: Springer.

Heard, N. A., Holmes, C. C. & Stephens, D. A. (2006). A quantitative study of gene regulation involved in the immune response of anopheline mosquitoes: an application of Bayesian hierarchical clustering of curves. *J. Am. Statist. Assoc*. **101**, 18–29.
Heller, K. A. & Ghahramani, Z. (2005). Bayesian hierarchical clustering. In *Proc. 22nd Int. Conf. Mach. Learn*., pp. 297–304. New York: Association for Computing Machinery.
Huelsenbeck, J. P. & Ronquist, F. (2001). MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* **17**, 754–5.
Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Am. Math. Soc*. **7**, 48–50.
Kulkarni, V. G. (1990). Generating random combinatorial objects. *J. Algor*. **11**, 185—207.
Lauritzen, S. L. (1996). *Graphical Models*. New York, NY: Oxford University Press.
Levin, D. A. & Peres, Y. (2017). *Markov Chains and Mixing Times*, 2nd ed. Providence, RI: American Mathematical Society.
Linero, A. R. & Yang, Y. (2018). Bayesian regression tree ensembles that adapt to smoothness and sparsity. *J. R. Statist. Soc. B* **80**, 1087–110.
Lovász, L. & Winkler, P. (1993). A note on the last new vertex visited by a random walk. *J. Graph Theory* **17**, 593–6.
Luo, Z. T., Sang, H. & Mallick, B. (2021). A Bayesian contiguous partitioning method for learning clustered latent variables. *J. Mach. Learn. Res*. **22**, 1748–99.
Luo, Z. T., Sang, H. & Mallick, B. (2024). A nonstationary soft partitioned Gaussian process model via random spanning trees. *J. Am. Statist. Assoc*. **119**, 2105–16.
Madry, A., Straszak, D. & Tarnawski, J. (2015). Fast generation of random spanning trees and the effective resistance metric. In *Proc. 26th Ann. ACM-SIAM Symp. Discrete Algor.*, pp. 2019–36. Philadelphia, PA: Society for Industrial and Applied Mathematics.
Matthews, P. (1988). Covering problems for Markov chains. *Ann. Prob*. **16**, 1215–28.
Meilă, M. & Jaakkola, T. (2006). Tractable Bayesian learning of tree belief networks. *Statist. Comp*. **16**, 77–92.
Meilă, M. & Jordan, M. I. (2000). Learning with mixtures of trees. *J. Mach. Learn. Res.* **1**, 1–48.
Meyer, C. D. (2023). *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics.
Neal, R. M. (2003). Density modeling and clustering using Dirichlet diffusion trees. *Bayesian Statist*. **7**, 619–29.
Payne, R. D., Guha, N. & Mallick, B. K. (2024). A Bayesian survival treed hazards model using latent Gaussian processes. *Biometrics* **80**, ujad009.
Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell Syst. Tech. J*. **36**, 1389–401.
Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics.
Schild, A. (2018). An almost-linear time algorithm for uniform random spanning tree generation. In *Proc. 50th Ann. ACM SIGACT Symp. Theory Comp*., pp. 214–27. New York: Association for Computing Machinery.
Song, Y. S. (2006). Properties of subtree-prune-and-regraft operations on totally-ordered phylogenetic trees. *Ann. Combinat*. **10**, 147–63.
Suchard, M. A., Weiss, R. E. & Sinsheimer, J. S. (2001). Bayesian selection of continuous-time Markov chain evolutionary models. *Molec. Biol. Evol*. **18**, 1001–13.
Teixeira, L. V., Assunção, R. M. & Loschi, R. H. (2019). Bayesian space-time partitioning by sampling and pruning spanning trees. *J. Mach. Learn. Res.* **20**, 1–35.
Van Havre, Z., White, N., Rousseau, J. & Mengersen, K. (2015). Overfitting Bayesian mixture models with an unknown number of components. *PloS One* **10**, e0131739.
Wilson, D. B. (1996). Generating random spanning trees more quickly than the cover time. In *Proc. 28th Ann. ACM Symp. Theory Comp*., pp. 296–303. New York: Association for Computing Machinery.
Wu, Y., Tjelmeland, H. & West, M. (2007). Bayesian CART: prior specification and posterior simulation. *J. Comp. Graph. Statist*. **16**, 44–66.
Zeng, Z., Gu, Y. & Xu, G. (2023). A tensor-EM method for large-scale latent class analysis with binary responses. *Psychometrika* **88**, 580–612.

[*Received on* 5 *May* 2024. *Editorial decision on* 21 *March* 2025]