# Bayesian CART
# – Prior Specification and Posterior Simulation –

Yuhong Wu[*], Håkon Tjelmeland[†] and Mike West[*]

January 22, 2006

**Abstract**

We present advances in Bayesian modeling and computation for CART (classification and regression tree) models. The modeling innovations include a formal prior distributional structure for tree generation – the *pinball prior* – that allows for the combination of an explicit specification of a distribution for both the tree *size* and the tree *shape*. The core computational innovations involve a novel Metropolis–Hastings method that can dramatically improve the convergence and mixing properties of MCMC methods of Bayesian CART analysis. Earlier MCMC methods have simulated Bayesian CART models using very local MCMC moves, proposing only small changes to a "current" CART model. Our new Metropolis–Hastings move makes large changes in the CART tree, but is at the same time local in that it leaves unchanged the partition of observations into terminal nodes. We evaluate the effectiveness of the proposed algorithm in two examples, one with a constructed data set and one concerning analysis of a published breast cancer data set.

**Keywords:** Bayesian CART; Tree prior specification; Pinball prior; Restructure moves.

[*]Institute of Statistics and Decision Sciences, Duke University
[†]Department of Mathematical Sciences, Norwegian University of Science and Technology.

# 1 Introduction

Search methods for CART (classification and regression tree) models are traditionally based on greedy algorithms that generate trees by recursive partitioning of sample data into more and more homogeneous subsets, often followed by pruning to address over-fitting (Breiman et al., 1984; Clark and Pregibon, 1992), and greedy model search is also used in Bayesian tree analyses (E. Huang and Huang, 2003; J. Pittman and West, 2003, 2004a,b). In contrast, Bayesian Monte Carlo methods, introduced by Chipman et al. (1998) and Denison et al. (1998), aim to generate samples of tree models according to their posterior probabilities representing fit to the data. The current paper is concerned with innovations in Bayesian modeling and computation to move the simulation-based methodology to a more practical level. We introduce, with examples, a new

- prior specification – the "pinball prior" – that involves an explicit specification of a distribution for size and shape of a tree, and
- a Metropolis-Hastings method – a radical "tree-restructure" move – that may be embedded in MCMC methods for exploring in tree model space; this novel algorithm appears to resolve existing problems of slow movement and lack of convergence of earlier MCMC methods, to define practically efficient and effective simulation methods.

Section 2 defines notation and terminology for tree models and the methodology. The prior specification, including the novel prior on tree structure, is described in Section 3. Posterior analysis and the core MCMC methods, including the novel tree-restructure Metropolis-Hastings component, are discussed in Section 4. Section 5 presents two examples, and Section 6 presents some concluding comments.

# 2 CART Model Structure & Notation

## 2.1 Binary trees: Notation, node numbering & subtrees

The example tree in Figure 1 illustrates the unique numbering of nodes, from the root (0) to the set of terminal nodes (or leaves). An internal (non-terminal) $u$ has left child node $l(u) = 2u + 1$ and right child node $r(u) = 2u + 2$. Except for the root, each node $u$ has: *one parent node*, $p(u) = \lceil \frac{u}{2} \rceil - 1$, where $\lceil x \rceil$ is the smallest integer larger than or equal to $x$; and *one sister node*, $s(u) = u + 1$ if $u$ is odd, and $s(u) = u - 1$ if $u$ is even. Each node belongs to a certain level, or *floor*, in the tree: root node is at level 0, and then each node $u$ is in level $f(u) = \lfloor log_2(u + 1) \rfloor$ where $\lfloor x \rfloor$ is the largest integer less than or equal to $x$.

Formally, a (binary) tree $T$ is defined by a finite set of nodes from the index set $\mathcal{N} = \{0, 1, 2, \ldots\}$, such that $0 \in T$ and for any $u \in T \setminus \{0\}$ both $p(u), s(u) \in T$. Write $\mathcal{T}$ for the set of all possible finite binary trees. For any $T$, write $a(T)$ for the set of non-terminal nodes and $b(T)$ for the set of leaves; finally, write $m(T) = |b(T)|$, the number of leaves.

For any tree $T$, the subtree from node $u \in T$ is just the tree from (and including) $u$ on downwards to the leaves of $T$ below $u$, we denote it by $S_u(T) = \{v | v + u2^{f(v)} \in T\}$. We sometimes write $m_u(T)$ for the number of leaves in the subtree from $u$, i.e. $m_u(T) = m(S_u(T))$. Evidently, $S_0(T) = T$ and $m_0(T) = m(T)$.

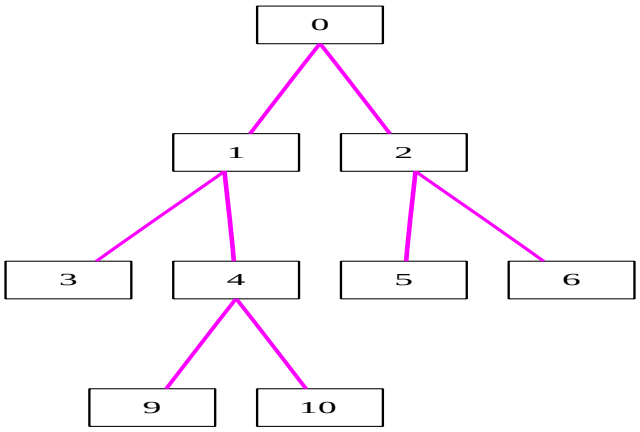



Figure 1: A (recursively defined, binary) tree with unique numbering of nodes.

## 2.2 CART splitting rules

We are interested in a CART model to predict variable $y \in \mathcal{Y}$ based on a set of candidate covariates (or predictor variables) $\boldsymbol{x} = (x^1, \ldots, x^p)'$. With sample spaces $x^j \in \mathcal{X}_j$ we have $\boldsymbol{x} \in \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_p$. A CART tree partitions $\mathcal{X}$ into regions by assigning a splitting rule to each internal node of a binary tree $T$. An internal node $u \in a(T)$ will be split as follows:

- choose a predictor variable index $k_T(u) \in \mathcal{P} = \{1, \ldots, p\}$ and a splitting threshold for that variable $\tau_T(u) \in \mathcal{X}_{k_T(u)}$;
- variables $(y, \boldsymbol{x})$ are assigned to the left child $l(u)$ of $u$ if $x^{k_T(u)} \leq \tau_T(u)$, otherwise to the right child $r(u)$.

Note that we allow the splitting threshold, $\tau_{T(u)}$, to take any value, they are not constrained to the observed values.

Writing $\mathsf{T} = (T, \boldsymbol{k}_T, \boldsymbol{\tau}_T)$, where $\boldsymbol{k}_T = \{k_T(u), u \in a(T)\}$ and $\boldsymbol{\tau}_T = \{\tau_T(u), u \in a(T)\}$, this recursively induces regions $R_{\mathsf{T}}(u)$ to each node $u \in T$, where

$$R_{\mathsf{T}}(u) = \begin{cases} \mathcal{X} & \text{if } u = 0, \\ R_{\mathsf{T}}(p(u)) \cap \{x \in \mathcal{X} | x^{k_T(u)} \leq \tau_T(u)\} & \text{if } u \text{ odd}, \\ R_{\mathsf{T}}(p(u)) \cap \{x \in \mathcal{X} | x^{k_T(u)} > \tau_T(u)\} & \text{if } u > 0 \text{ and even.} \end{cases} \tag{1}$$

In particular, the regions in the leaves $\{R_{\mathsf{T}}(u), u \in b(T)\}$ form a partition of $\mathcal{X}$.

Corresponding to the definition of subtrees $S_u(T)$ above, we let $S_u(\boldsymbol{k}_T)$ and $S_u(\boldsymbol{\tau}_T)$ denote the splitting variables and splitting thresholds in the subtree $S_u(T)$. For $\mathsf{T} = (T, \boldsymbol{k}_T, \boldsymbol{\tau}_T)$ and $u \in T$ we will also use the notation $S_u(\mathsf{T}) = (S_u(T), S_u(\boldsymbol{k}_T), S_u(\boldsymbol{\tau}_T))$.

## 2.3 Recursive Partitioning of Sample Observations

Suppose we have observations $(y_i, \boldsymbol{x}_i)$, $i \in \mathcal{I} = \{1, \ldots, n\}$. The CART specification (above) recursively partitions the data, assigning subsets of $\mathcal{I}$ to each node corresponding to the regions

$R_{\mathsf{T}}(u)$ defined above. Starting with the full data set at the root, $N_{\mathsf{T}}(0, \mathcal{I}) = \mathcal{I}$, the subset at any node $u$ is $N_{\mathsf{T}}(u, \mathcal{I}) = \{i \in \mathcal{I} | \boldsymbol{x}_i \in R_{\mathsf{T}}(u)\}$. The subsets in the leaves define a partition of the full data set, denoted by $\mathcal{L}_{\mathsf{T}}(\mathcal{I}) = \{N_{\mathsf{T}}(u), u \in b(T)\}$. One may express the partition of $\mathcal{I}$ in terms of partitions induced by subtrees. As an example, in the tree in Figure 2(a), we have $\mathcal{L}_{\mathsf{T}}(\mathcal{I}) = \{\{1,2,3\},\{4,5,6\},\{7,8,9\},\{10,11,12\}\}$, $\mathcal{L}_{S_1(\mathsf{T})}(N_{\mathsf{T}}(1,\mathcal{I})) = \{\{1,2,3\},\{4,5,6\}\}$, $\mathcal{L}_{S_2(\mathsf{T})}(N_{\mathsf{T}}(2,\mathcal{I})) = \{\{7,8,9\},\{10,11,12\}\}$, $\mathcal{L}_{S_3(\mathsf{T})}(N_{\mathsf{T}}(3,\mathcal{I})) = \{\{1,2,3\}\}$ and, finally, the set $\mathcal{L}_{S_4(\mathsf{T})}(N_{\mathsf{T}}(4,\mathcal{I})) = \{\{4,5,6\}\}$.

### 2.4 Statistical model in leaves

We focus exclusively on CART models in which the subset of $y$ outcomes in any leaf $u$ is viewed as a random sample from a distribution with density $\phi(\cdot|\theta_u)$ (Chipman et al., 1998). Key examples are normal or Bernoulli distributions for $y$, the model indicating only that the parameters defining the distributions differ across leaves. We restrict attention to random sample models within leaves in this paper, though the new tree priors and MCMC innovations are of course applicable more generally, including to models with regressions within leaves. We write $\boldsymbol{\theta} \equiv \boldsymbol{\theta}_T = (\theta_u, u \in b(T))$ for the set of parameters in any tree.

## 3 Prior model

### 3.1 Prior components

Recall the notation $\mathsf{T} = \{T, \boldsymbol{k}_T, \boldsymbol{\tau}_T\}$. We consider hierarchical prior specifications, with implicit conditional independence assumptions, of the form

$$\pi(\mathsf{T}, \boldsymbol{\theta}) = \pi(T)\ \pi(\boldsymbol{k}_T|T)\ \pi(\boldsymbol{\tau}_T|T, \boldsymbol{k}_T)\ \pi(\boldsymbol{\theta}|\mathsf{T}). \tag{2}$$

For the final three components we adopt prior structures that are the same as, or minor modifications of, those used by previous authors (Chipman et al., 1998; Denison et al., 1998). Conditional on $T$ :

- Priors for selection of splitting variables $\pi(\boldsymbol{k}_T|T)$. We assume each $k_T(u)$ is independently generated from a fixed discrete distribution over the $p$ predictor variables, choosing variable $x^i$ with probability $\gamma(i)$, $i \in \mathcal{P}$. Thus $\pi(\boldsymbol{k}_T|T) = \prod_{u \in a(T)} \gamma(k_T(u))$. Note that the same predictor variable may be used as splitting variable in several internal nodes.

- Priors for splitting thresholds $\pi(\boldsymbol{\tau}_T|T, \boldsymbol{k}_T)$. We take the $\tau_T(u)$ as conditionally independent, and, for a chosen splitting variable $x^i$, specify a prior density $\delta_i(\tau)$ for the splitting threshold. Thus $\pi(\boldsymbol{\tau}_T|T, \boldsymbol{k}_T) = \prod_{u \in a(T)} \delta_{k_T(u)}(\tau_T(u))$.

- Finally, the leaf parameters are assumed independent with common prior density $\rho(\theta_u)$, so that $\pi(\boldsymbol{\theta}|\mathsf{T}) = \prod_{u \in b(T)} \rho(\theta_u)$.

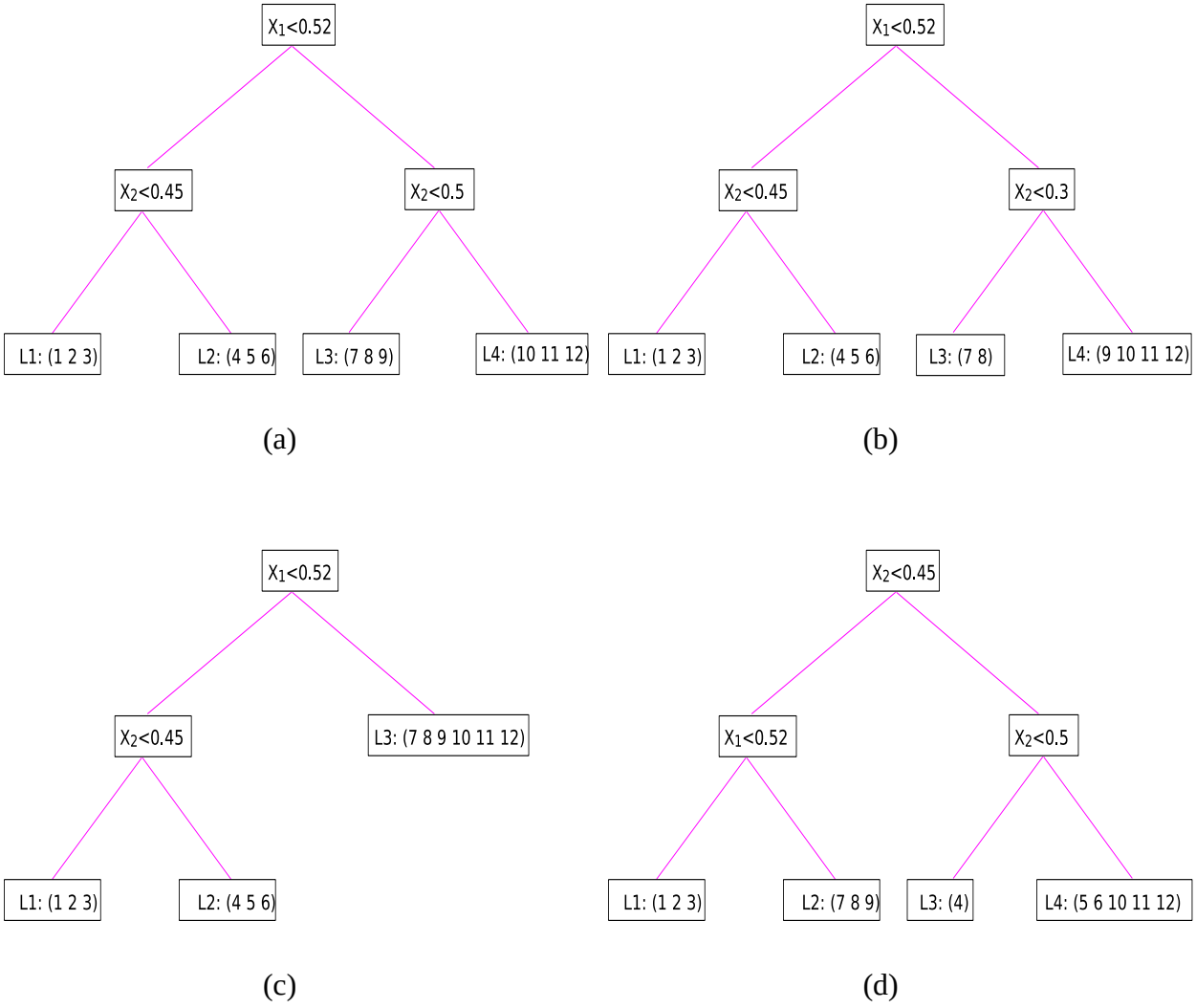



Figure 2: Illustration of the change, grow/prune and swap proposals with the constructed data set in Table 1. Change proposal: (a) shows current tree $T$ and (b) the potential new tree $T'$ after a change in node 2. Grow proposal: (c) and (a) show $T$ and $T'$, respectively. Prune proposal: (a) and (c) shows $T$ and $T'$, respectively. Swap proposal: (a) and (d) show $T$ and $T'$, respectively.

### 3.2 The pinball prior for tree generation

A key component is the prior $\pi(T)$ that defines control over the number and structure of nodes. Denison et al. (1998) specified a prior on the number of leaves, and then a uniform prior over trees with that number of leaves. Chipman et al. (1998) defined a tree-generating process within which the prior on the number of nodes and shape of the tree is implicit, but making it relatively difficult to incorporate prior structure directly on the number of leaves. The novel *pinball prior* here overcomes this and extends both earlier approaches: we focus on tree size ($m(T)$) and whether the tree layout is balanced or skewed.

The construction is simple and intuitive: the prior generates a number of terminal leaves $m(T)$, and then these leaves are cascaded down from the root node, randomly splitting left/right at any node according to a defined probability distribution at that node until they define individual leaves - the leaves fall down the tree as pinballs. Formally:

- Specify a prior density for tree size (number of leaves) $m(T) \sim \alpha(m(T))$. An obvious candidate is Poisson, say $m(T) = 1 + \text{Pois}(\lambda)$ for specified $\lambda$.
- Noting $m_0(T) = m(T)$, recall that $m_u(T)$ is the number of leaves in the subtree $S_u(T)$ below any node $u$. At this node, these leaves are distributed to the left/right according to some prior. Take $\beta(m_{l(u)}(T)|m_u(T))$ as a prior density governing the generation of this split: here $m_{l(u)}(T)$ is the number of leaves, of the current total $m_u(T)$, sent to the left node $l(u)$, the remainder going to $r(u)$. Note that the process ends when $m_u(T) = 1$ at any node $u$. Then
$$\pi(T) = \alpha(m_0(T)) \prod_{u \in a(T)} \beta(m_{l(u)}(T)|m_u(T)) \quad \text{for } T \in \mathcal{T}. \tag{3}$$

Different choices are possible for the function $\beta(i|m)$. Taking $\beta(i|m) = 1/\{(\#\text{ possible trees with } i \text{ leaves}) \cdot (\#\text{ possible trees with } m-i \text{ leaves})\}$ gives the uniform distribution over all trees of size $m$ adopted in Denison et al. (1998). Except for very small tree sizes, the number of unbalanced trees of a given size is much larger than the number of balanced trees of the same size, so this choice of $\beta(i|m)$ assigns a high prior probability to unbalanced trees. Natural alternatives are a uniform distribution for $\beta(i|m)$, i.e. $\beta(i|m) = 1/(m-1)$ for $i = 1, \ldots, m-1$, or $\beta(i|m) = 1 + \frac{1}{2}[\text{Bin}(i-1; m-2, p) + \text{Bin}(i-1; m-2, 1-p)]$, where $\text{Bin}(\cdot; n, p)$ denotes the binomial mass function with parameters $n$ and $p$. With $p = 1/2$ the latter gives a balanced tree with high probability, whereas with a $p$ close to zero the unbalanced trees get higher prior probabilities. Simulation from this symmetric prior is direct from its definition; see examples in Figure 3. Note that asymmetric priors could be used to encourage trees that are more skewed in overall shape.

# 4 Posterior Analysis & Markov chain Monte Carlo

Write $\boldsymbol{y}$ for the observed response data $\boldsymbol{y} = \{y_i\}_{i=1}^n$. Following previous authors, we note that, given $\mathsf{T}$ and $\boldsymbol{\theta}$, the likelihood function depends on $\mathsf{T}$ only through the induced partition of observations to leaves, $\mathcal{L}_{\mathsf{T}}(\mathcal{I})$. The data within leaves are independent random samples, and we make the assumption that the prior $\rho(\theta_u)$ is such that we can analytically compute (or accurately approximate) the implied marginal likelihoods, so delivering the overall marginal likelihood over tree

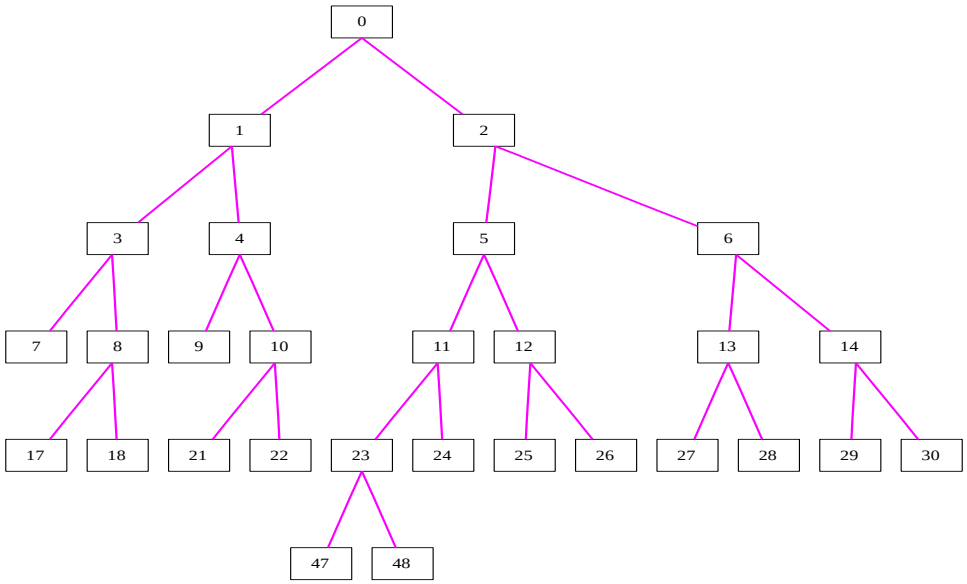



Figure 3: A sample from the pinball prior with $\alpha(m) = 1 + \text{Pois}(m-1; 15)$ and $\beta(i; m) = 1 + \text{Bin}(i-1; m-1, 0.5)$.

structures, namely $f(\boldsymbol{y}|\mathsf{T}) = \int f(\boldsymbol{y}|\boldsymbol{\theta}_T, \mathsf{T})\pi(\boldsymbol{\theta}_T)d\boldsymbol{\theta}_T$. Thus we have the posterior on tree model space,

$$\pi(\mathsf{T}|\boldsymbol{y}) \propto \pi(\mathsf{T})f(\boldsymbol{y}|\mathsf{T})$$

with the prior $\pi(\mathsf{T}) = \pi(T)\pi(\boldsymbol{k}_T|T)\pi(\boldsymbol{\tau}_T|T, \boldsymbol{k}_T)$ as already defined.

Metropolis–Hastings MCMC methods aim to generate samples from this posterior. A technical point is that we actually need to restrict to trees with non-empty leaves; more generally, we may decide to require a minimum of, say, $h$ data points in a leaf, so aim to sample from

$$\widetilde{\pi}(\mathsf{T}|\boldsymbol{y}) \propto \pi(\mathsf{T}|\boldsymbol{y}) \cdot \prod_{u \in b(T)} I[|N_{\mathsf{T}}(u, \mathcal{I})| \geq h]. \tag{4}$$

The Metropolis–Hastings methods in Chipman et al. (1998) and Denison et al. (1998) are quite similar and form our starting point. These two key articles also share the same honest conclusion: their MCMC methods, based on "local moves" around the tree space, are extraordinarily slow to converge. As a remedy they recommend restarts of the MCMC algorithm combined with ad-hoc weighting of the generated trees. Our major contribution here is to introduce novel MCMC moves that seem to solve the convergence problem. Thereby, there is no need for restarts or for weighting of the realizations. For model averaging all trees (after convergence) should be assigned the same weight. Moreover, if the distribution has multiple modes, the probability mass contained in a specific mode is easily estimated by the fraction of time the Markov chain spends in this mode.

Our MCMC method uses four Metropolis–Hastings proposals, namely "change", "grow/prune", "swap" and "(radical) restructure". The first three proposals are those of Chipman et al. (1998) and Denison et al. (1998). In the notation, we use primes to indicate proposal/potential new values, so that, for example, $T$ and $T'$ are the current and the proposed/candidate new tree structures, respectively. The proposals are as follows.

| Obs | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $x^1$ | 0.00 | 0.10 | 0.20 | 0.30 | 0.40 | 0.50 |
| $x^2$ | 0.10 | 0.30 | 0.40 | 0.48 | 0.56 | 0.57 |
| Obs | 7 | 8 | 9 | 10 | 11 | 12 |
| $x^1$ | 0.55 | 0.60 | 0.65 | 0.70 | 0.80 | 0.90 |
| $x^2$ | 0.00 | 0.20 | 0.40 | 0.60 | 0.80 | 1.00 |

Table 1: Example data, with $n = 12$, used to illustrate the Metropolis–Hastings proposals.

- *Change proposal:*
  Propose to change $k(u)$ and $\tau(u)$ for a randomly selected internal node $u \in a(T)$; leave everything else unchanged. For the constructed data set in Table 1, an illustration of the move is shown in Figures 2(a) and (b). Figure 12 (Appendix) contains pseudo code for the proposal mechanism.

- *Grow/prune proposal:*
  Propose to split a randomly selected leaf into two, or to prune the tree by merging two randomly selected sibling leaves. For the constructed data set in Table 1, Figures 2(a) and (c) illustrates the move. Figure 13 (Appendix) gives pseudo code for the move. One should note that this proposal changes the number of (continuously distributed) splitting thresholds and is thereby a reversible jump type of move (Green, 1995). However, as we propose the new splitting threholds independently of everything else, the associated Jacobian will always equal unity.

- *Swap proposal:*
  Propose to swap the splitting rules in a randomly selected parent-child pair that are both internal nodes. For the constructed data set in Table 1, Figures 2(a) and (d) illustrates the move. Figure 14 (Appendix) contains the pseudo code for the proposal.

## 4.1 The restructure proposal

We aim to propose large changes in tree structure $\mathsf{T} = (T, \boldsymbol{k}, \boldsymbol{\tau})$ without changing the number of leaves nor the partition of observations into leaves. Figure 4 illustrates the move for the data in Table 1.

Figure 4(a) shows a current tree, $\mathsf{T}$, and the corresponding partition of observations into leaves, $\mathcal{L}_{\mathsf{T}}(\mathcal{I})$. To propose a candidate $\mathsf{T}' = (T', \boldsymbol{k}', \boldsymbol{\tau}')$ with $\mathcal{L}_{\mathsf{T}'}(\mathcal{I}) = \mathcal{L}_{\mathsf{T}}(\mathcal{I})$ we first identify possible pairs of splitting variable and splitting thresholds for node 0. We require at least $h = 1$ leaf in each of the two subtrees $S_1(T')$ and $S_2(T')$. In Figure 4(b) the possible intervals for $\tau'(0)$ for $k'(0) = 1$ and for $k'(0) = 2$ are shaded. Three possible intervals exist for $k'(0) = 1$, two intervals for $k'(0) = 2$. We propose values for the pair $(k'(0), \tau'(0))$ by first sampling one of the five possible intervals uniformly at random and thereafter generating a value for $\tau'(0)$ uniformly at random within the chosen interval. This produces, for example, the situation shown in Figure 4(c); $\mathsf{T}'$ is now partly specified by $(k'(0), \tau'(0))$, $\mathcal{L}_{S_1(\mathsf{T}')}(\mathcal{I})$ and $\mathcal{L}_{S_2(\mathsf{T}')}(\mathcal{I})$. The right subtree $\mathcal{L}_{S_2(\mathsf{T}')}(\mathcal{I})$ contains only one leaf and is therefore already completely specified. The left subtree

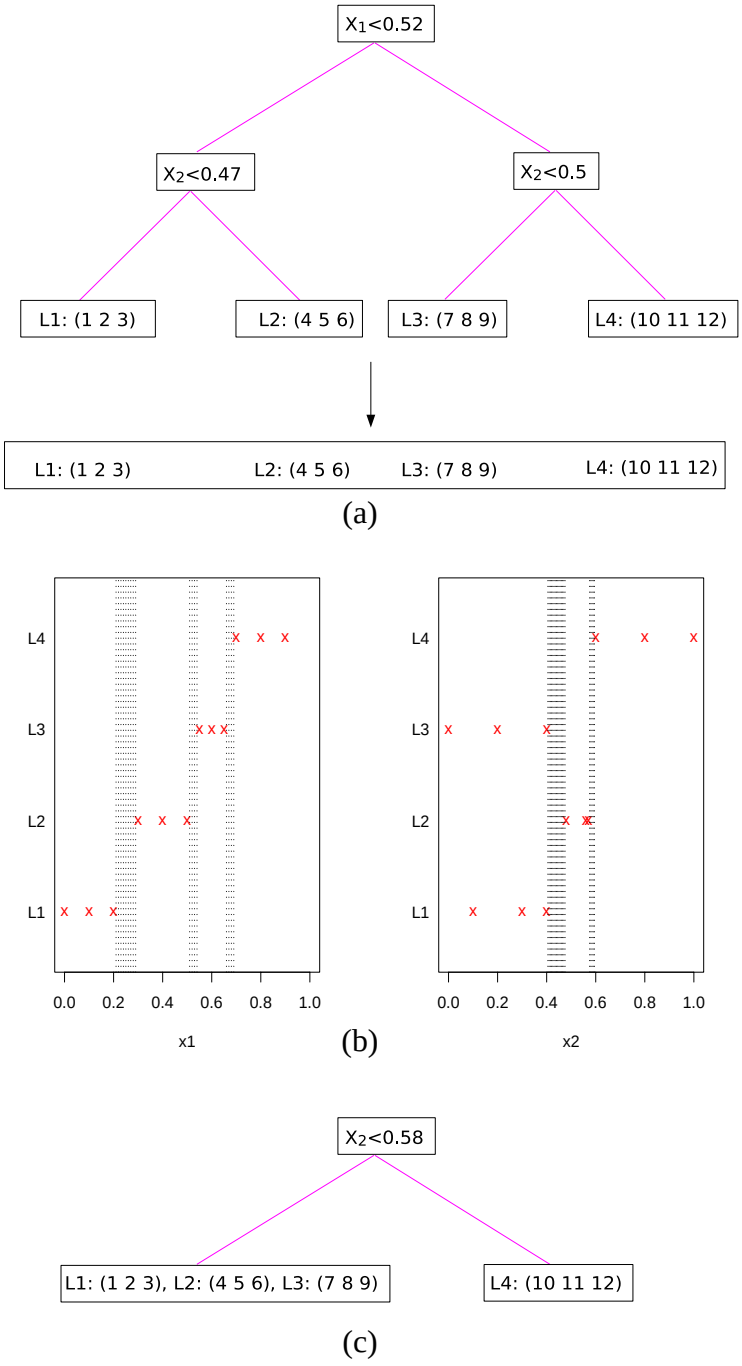
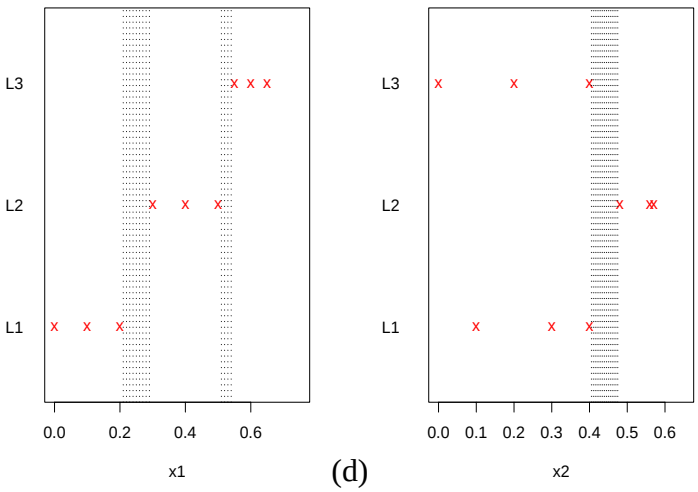
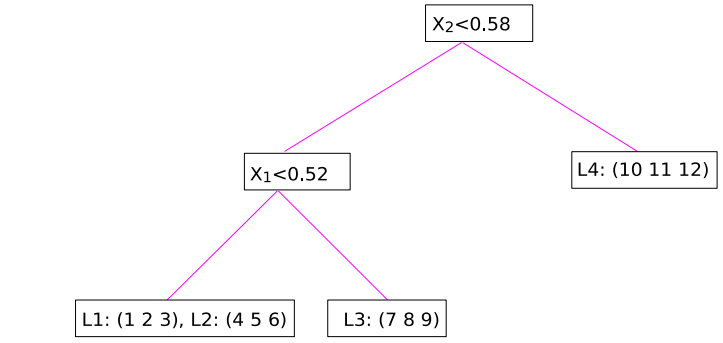


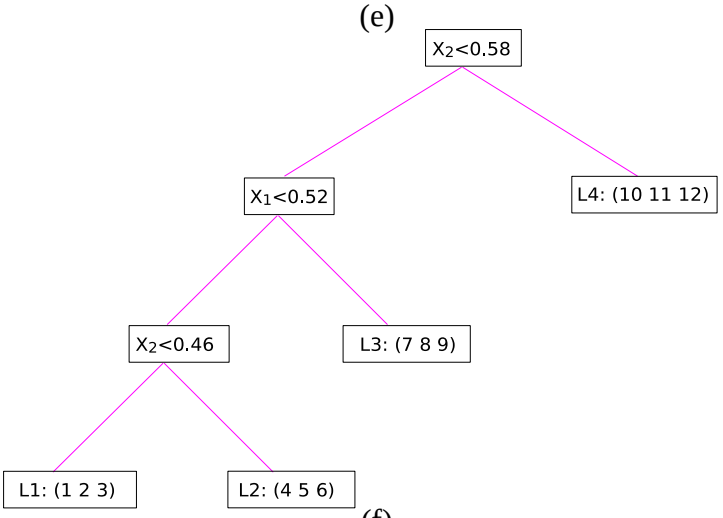

Figure 4: Illustration of restructure move with the constructed data set in Table 1. Note that the figure continues on the next page. (a) Current tree and the corresponding partition of observations into leaves; (b) Possible splits for the root node; (c) Set the first split.




(d)




(e)




(f)

Figure 4: (cont.) Illustration of restructure move with the constructed data set in Table 1. (d) Possible splits for node 1; (e) Set the split for node 1; (f) The candidate tree proposed by the restructure move.

$\mathcal{L}_{S_1(\mathsf{T}')}(\mathcal{I})$ contains three leaves and is split into two subtrees, $\mathcal{L}_{S_3(\mathsf{T}')}(\mathcal{I})$ and $\mathcal{L}_{S_4(\mathsf{T}')}(\mathcal{I})$, by repeating the process just described. Figure 4(d) shows the possible intervals for $\tau'(1)$ for $k'(1) = 1$ and $k'(1) = 2$. A possible result is shown in Figure 4(e). Thus, $\mathsf{T}'$ is now partially specified by $\{(k'(u), \tau'(u)), u = 1, 2\}$, $\mathcal{L}_{S_2(\mathsf{T}')}(\mathcal{I})$, $\mathcal{L}_{S_3(\mathsf{T}')}(\mathcal{I})$ and $\mathcal{L}_{S_4(\mathsf{T}')}(\mathcal{I})$. The subtree $\mathcal{L}_{S_3(\mathsf{T}')}(\mathcal{I})$ still contains more than one leaf and must therefore be split into two subtrees by repeating the above process once more. The possible intervals for $\tau'(3)$ are identified (this step is not shown in Figure 4) and used to generate values for $(k'(3), \tau'(3))$. The resulting completely specified $\mathsf{T}'$ is shown in Figure 4(f).

One should note that with the above proposal procedure there is a unique way to generate $\mathsf{T}'$ from $\mathsf{T}$, and a corresponding unique way to propose $\mathsf{T}$ from $\mathsf{T}'$. Computation of the associated proposal probability $q(\mathsf{T}'|\mathsf{T})$, and the probability $q(\mathsf{T}|\mathsf{T}')$ for the corresponding reverse move, follows directly from the generating procedure. The proposal probability $q(\mathsf{T}'|\mathsf{T})$ is a product with two factors associated with each internal node of $\mathsf{T}'$; correspondingly, $q(\mathsf{T}|\mathsf{T}')$ has two factors for each internal node of $\mathsf{T}$. The first factor comes from drawing an interval and is one divided by the number of possible intervals. The second factor, associated with drawing a value for $\tau'(u)$, is one divided by the length of the allowed interval.

Figure 15 (appendix) gives pseudo code for the proposal mechanism. One should note that in the process of splitting $\mathcal{L}_{S_u(\mathsf{T}')}(\mathcal{I})$ into $\mathcal{L}_{S_{l(u)}(\mathsf{T}')}(\mathcal{I})$ and $\mathcal{L}_{S_{r(u)}(\mathsf{T}')}(\mathcal{I})$ one risk is that no possible values exist for the pair $(k'(u), \tau'(u))$. If this happens in any of the splits we just set $\mathsf{T}' = \mathsf{T}$, i.e. keep the tree unchanged and increment the iteration variable by one. One should note that even if the tree is kept unchanged in this case, it is essential to count it as an iteration of the Metropolis–Hastings algorithm; otherwise it would be necessary to compute the probability for this event to happen, which in practice would be impossible except for very small trees.

## 4.2 Variations of the restructure proposals

In the restructure move one needs to map all possible splitting variables (and corresponding splitting intervals) for each internal node in the new tree. The computational cost of this is proportional to the number of candidate predictor variables $p$. If $p$ is very large, a complete mapping of possible splitting variables becomes computationally expensive. One may therefore select only a randomly chosen subset of the predictor variables. Letting $C \subseteq \mathcal{P}$ denote the subset of predictor variables mapped, a simple alternative is to let

$$\mathrm{P}(i \in C|\mathsf{T}) = \begin{cases} 1 & \text{if } i \in \{k_T(u) : u \in \mathsf{T}\}, \\ \alpha & \text{otherwise,} \end{cases} \tag{5}$$

independently for each $i \in \mathcal{P}$. The parameter $\alpha \in (0, 1)$ can be used to control the typical size of $C$. The restructure move can then be started by sampling a set $C$, and it should be noted that the Metropolis–Hastings acceptance probability must be modified by including also the probability of sampling $C$.

A second possible modification allows for changes also in the partition of observations in leaves. In the ProposeRestructure function (Figure 15 in Appendix) a new partition $\mathsf{L}'$ can be proposed just after the leaves, $\mathsf{L}$, have been picked. Different possibilities exist for how to sample $\mathsf{L}'$ given $\mathsf{L}$. An alternative based on the likelihood function is given in Figure 16, where we use the fact that the likelihood function only depends on the partition of observations into leaves.

# 5 Examples

We present simulation results for two examples. The first is for a small synthetic data set designed to give a posterior distribution with two distinct modes. This toy example is included to illustrate how the restructure move generates well-mixing Markov chains by enabling direct jumps between modes. The second example considers a breast cancer data set and provides a more complex and challenging posterior distribution for a comparison with previous MCMC approaches.

## 5.1 A synthetic example

We first discuss MCMC convergence and mixing properties using a synthetic data set having $p = 3$ predictors and $\mathcal{Y} = \mathbb{R}$. We generated $n = 300$ sets of $(x^1, x^2, x^3)$ , where $x_i^1 \sim \text{Unif}(0.1, 0.4)$ for $i = 1 \ldots 200$, $x_i^1 \sim \text{Unif}(0.6, 0.9)$ for $i = 201 \ldots 300$, $x_i^2 \sim \text{Unif}(0.1, 0.4)$ for $i = 1 \ldots 100$, $x_i^2 \sim \text{Unif}(0.6, 0.9)$ for $i = 101 \ldots 200$, $x_i^2 \sim \text{Unif}(0.1, 0.9)$ for $i = 201 \ldots 300$, $x_i^3 \sim \text{Unif}(0.6, 0.9)$ for $i = 1 \ldots 200$ and $x_i^3 \sim \text{Unif}(0.1, 0.4)$ for $i = 201 \ldots 300$. Given these predictor values, $y$ values were simulated independently as

$$y = \begin{cases} 1 + \text{N}(0, 0.25) & \text{if } x^1 \leq 0.5 \text{ and } x^2 \leq 0.5, \\ 3 + \text{N}(0, 0.25) & \text{if } x^1 \leq 0.5 \text{ and } x^2 > 0.5, \\ 5 + \text{N}(0, 0.25) & \text{if } x^1 > 0.5. \end{cases} \tag{6}$$

The partition of observations with respect to $x^1$, $x^2$ and $x^3$ is shown in Figure 5, where the symbols refer to the three regions defined in (6). The mean values of $y$ in the three regions are well separated and it is easy to check that the two trees in Figure 6 are the only trees consistent with the regions.

The analysis assumes a prior within-leaf model where the $y_i$'s are independent and $y_i \sim \text{N}(\mu_u, \sigma_u^2)$ when observation $i$ is assigned to leaf $u$, i.e. $\theta_u = (\mu_u, \sigma_u^2)$. We use conjugate priors: $1/\sigma_i^2 \sim \text{Gam}(0.5, 1.5)$ and $(\mu_i|\sigma_i^2) \sim \text{N}(0, \sigma_i^2)$. The pinball prior has $\alpha(m) = 1 + \text{Pois}(m - 1; 10)$ and $\beta(i|m) = 1 + \text{Bin}(i - 1; m - 2, \frac{1}{2})$. The splitting variables are chosen uniformly via $\gamma(i) = \frac{1}{3}$, and splitting thresholds come, for all $k$, from $\delta_k(\cdot) = \text{N}(0.5, 2)$ truncated to $[0, 1]$.

We tried two algorithms: first, the MCMC using only the grow/prune, swap and change proposals; second, including the basic restructure proposal too. In the first case, we call an "iteration" a series of $60$ change, $60$ grow/prune and $60$ swap moves; when including the restructure proposal, we take one iteration to mean $50$ change moves, $50$ grow/prune moves, $50$ swap moves and $1$ restructure move. With our implementation these two types of iterations require essentially the same amount of computing time. We start the chain from the left tree shown in Figure 6 and run each of the algorithms for $8,000$ iterations. We use the last $4,000$ iterations to estimate, for each predictor, the posterior probability that this predictor is used as a splitting variable. In the tree samples produced with the algorithm without restructure move, both $x^1$ and $x^2$ are used in all the $4000$ tree samples while $x^3$ is used only twice. Actually, by examining the tree samples, we find that $3,997$ out of the $4,000$ samples have exactly the same structure and splitting variables as the starting tree. The remaining three trees have one split more than the starting tree. Thus, the tree shown in the right panel in Figure 6 has never been visited. Since the implementations of the grow/prune, swap and change proposals are the same as those of Chipman et al. (1998) and Denison et al. (1998), one can see that the algorithm in Chipman et al. (1998) and Denison et al. (1998) will hardly find the tree shown in the right panel in Figure 6. Restarts of the algorithm can help, but again one needs

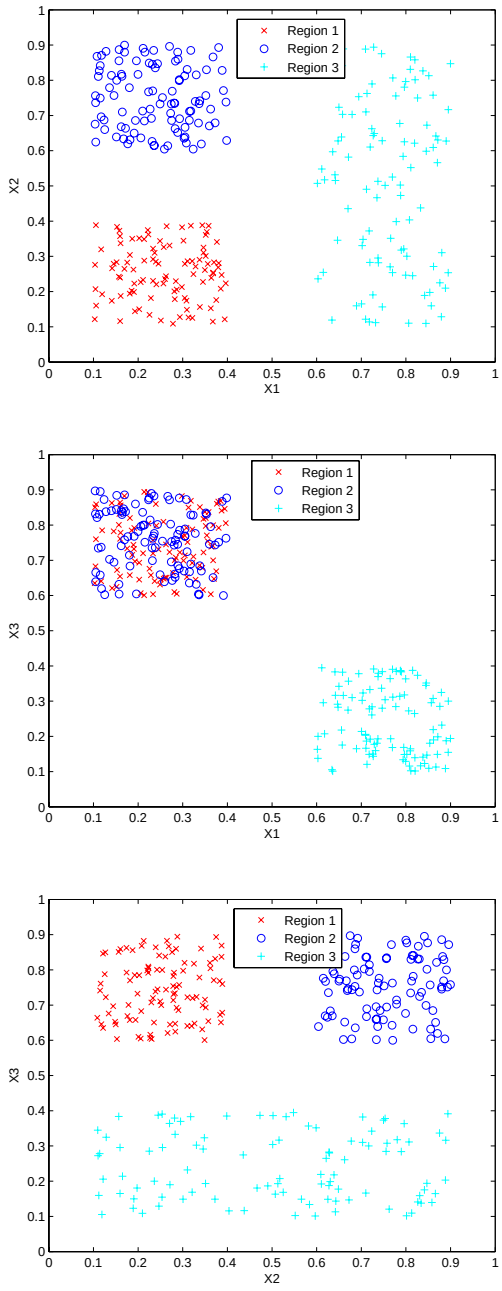



Figure 5: Simulated data example: Partition of observations with respect to $x^1$, $x^2$ and $x^3$. The symbols used refer to the three regions defined in (6). Cross: $x^1 \leq 0.5$ and $x^2 \leq 0.5$, circle: $x^1 \leq 0.5$ and $x^2 > 0.5$; plus: $x^1 > 0.5$.

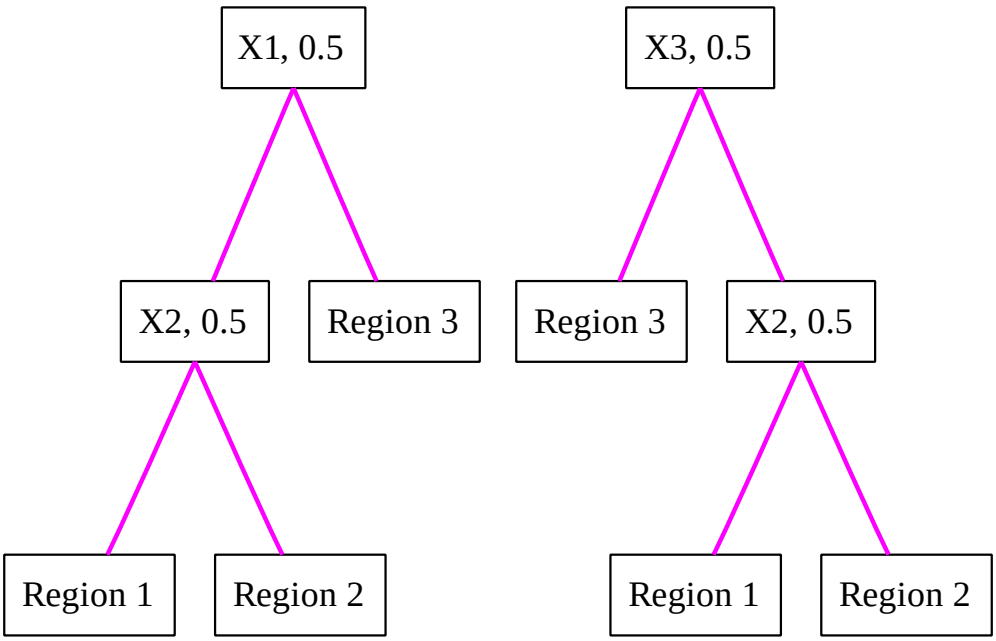



Figure 6: The two trees consistent with the three regions defined in (6).

to weight all the generated trees. In contrast, using the algorithm with the restructure move, $x^1$ is used $1290$ times, $x^2$ is used $4000$ times and $x^3$ is used $2712$ times. By examining the tree samples, we find that both the two trees in Figure 6 are very frequently visited.

Finally, to check robustness with respect, particularly, to the key Poisson prior $\alpha(\cdot)$, we reran the analysis after replacing the $y$ data with pure noise - a standard normal random sample. This produced a posterior distribution with $\Pr(m_0(T) = 1|\boldsymbol{y}) \approx 0.37$, compared to the prior probability of 0.05, supporting the view that the analysis is relatively robust to the assumed Poisson form; this has been borne out in other examples, including the following real-data example.

## 5.2 Breast cancer data example

A second example analyzes the breast cancer data set used in Chipman et al. (1998). The data were obtained from the University of California, Irvine repository of machine learning databases (ftp://ftp.ics.uci.edu/pub/machine-learning-databases) and originated in Wolberg and Mangasarian (1990). The same data set is also used for a Bayesian CART model search, but without convergence of the MCMC algorithm, in Chipman et al. (1998). The data set has 9 cellular cancer characteristics, all ordered numeric variables, and the response is binary, indicating benign $(0)$ and malignant $(1)$ tumors, i.e. $\mathcal{Y} = \{0, 1\}$. We deleted missing values in the data set and so use 683 of the original 699 observations.

For $\mathsf{T}$ we use exactly the same prior distribution as in Section 5.1. For each of the predictor variables we use a linear transformation to place all values of $x^i$ in the unit interval. The within-leaf sampling model is Bernoulli with probability (of malignance) $\theta_u$ in leaf $u$ having independent uniform priors. MCMC analysis was performed using the same two algorithms as above, with and without the restructure move.

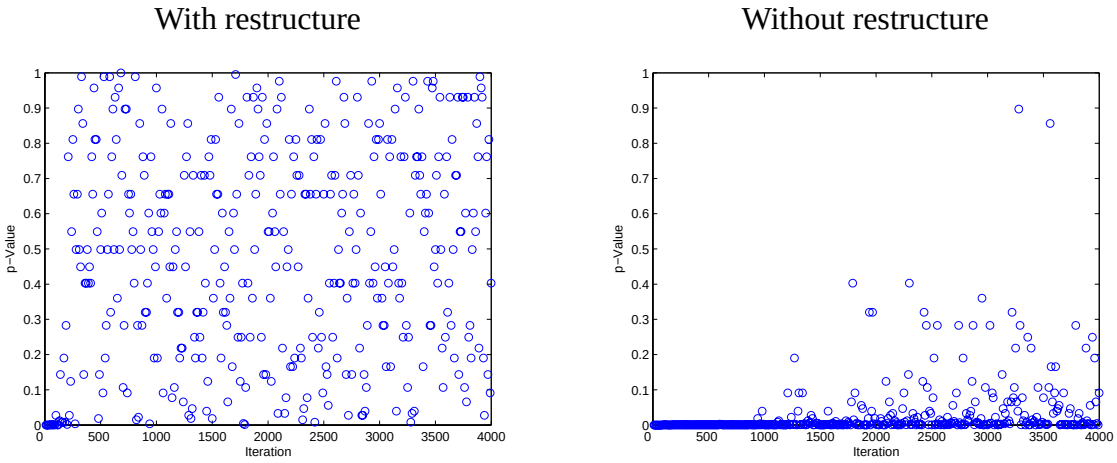



Figure 7: Breast cancer data example: P-values of K-S statistics for samples from algorithm with and without restructure move.

Convergence and mixing analysis becomes more difficult here than in the above toy example. There are numerous methods for exploring MCMC convergence (Cowles and Carlin, 1996; Brooks and Roberts, 1998). Here we simulate one long and $K$ short MCMC runs, initializing each run by sampling from the prior distribution. For each $i = 1, 2, \ldots$ we pick $K$ realizations from the long run and one realization from each of the short runs. From the long run we use the realizations after $k \cdot i$ iterations for $k = 1, 2, \ldots, K$, whereas for the short runs we use the realization after iteration number $i$. If the chain has converged before $i$ iterations and the mixing is sufficient for two realizations $i$ iterations apart to be independent, then the $K$ realizations from the short runs and the $K$ realizations from the long run are independent and all come from the same distribution. For any scalar function of the $2K$ realizations, we compute the Kolmogorov-Smirnov $p$-value (Conover, 1971) to provide some insight into whether this seems reasonable. Figure 7 shows the result when $K = 250$ and the scalar function is the log posterior density. When including the restructure move, the algorithm is converging in less than $500$ iterations, whereas without the restructure move convergence clearly takes more than $4,000$ iterations.[1] We have also generated this type of convergence diagnostics plots for various other scalar functions including, among others, the log integrated likelihood, tree size and number of times a particular predictor variable is used as a splitting variable in the tree. The convergence differences between the two algorithms are more or less striking dependent on which scalar function is used; the case shown in Figure 7 indicates the slowest convergence of the non-restructure move approach for this example.

The topology of the tree space is very complex and to get a clear picture of how the two algorithms work is difficult, but some insight can be gained from Figure 8. This shows, for ten runs with each of the two algorithms, trace plots of log integrated likelihood and the number of leaves. Each simulation is for $1,000$ iterations, of which the last $500$ iterations are shown in the plots. Vertical dashed lines mark the start of new runs. We note how some of the runs with the algorithm without the restructure seem to get trapped in local modes for the whole $500$ iteration

[1] With our implementation in early 2005, $500$ iterations for this data set took about $6$ seconds on an Intel Xeon $3.0Ghz$ with $2Gb$ memory.

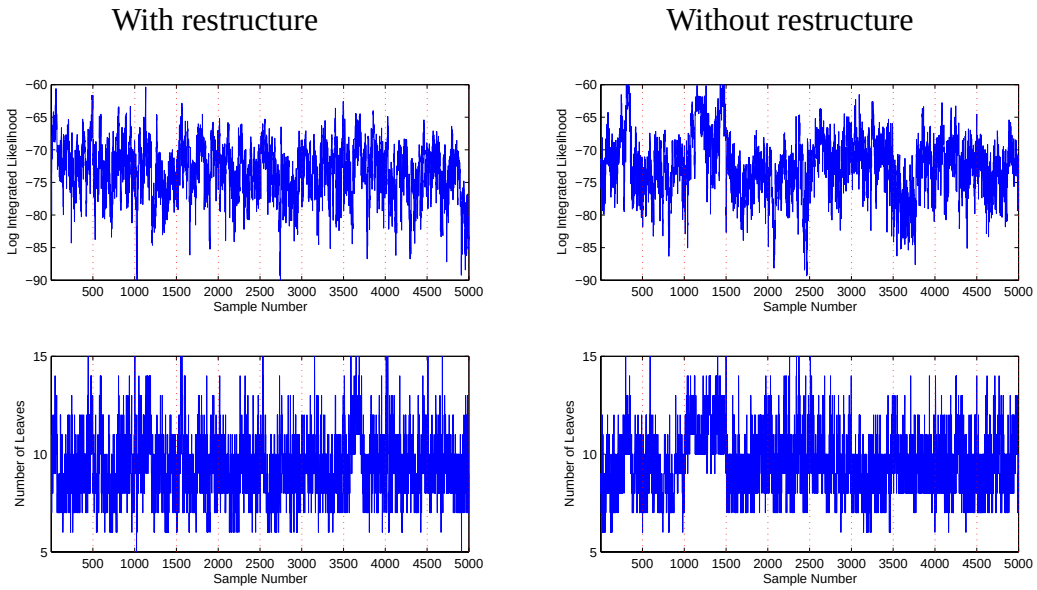



Figure 8: Breast cancer data example: Trace plots of log integrated likelihood (upper row) and number of leaves (lower row) for ten runs of the algorithm with the restructure proposal (right column) and without the restructure proposal (left column). In each simulation the algorithms is run for $1,000$, of which only the last $500$ are shown in the figure. The start of a new run is indicated by vertical dashed lines.

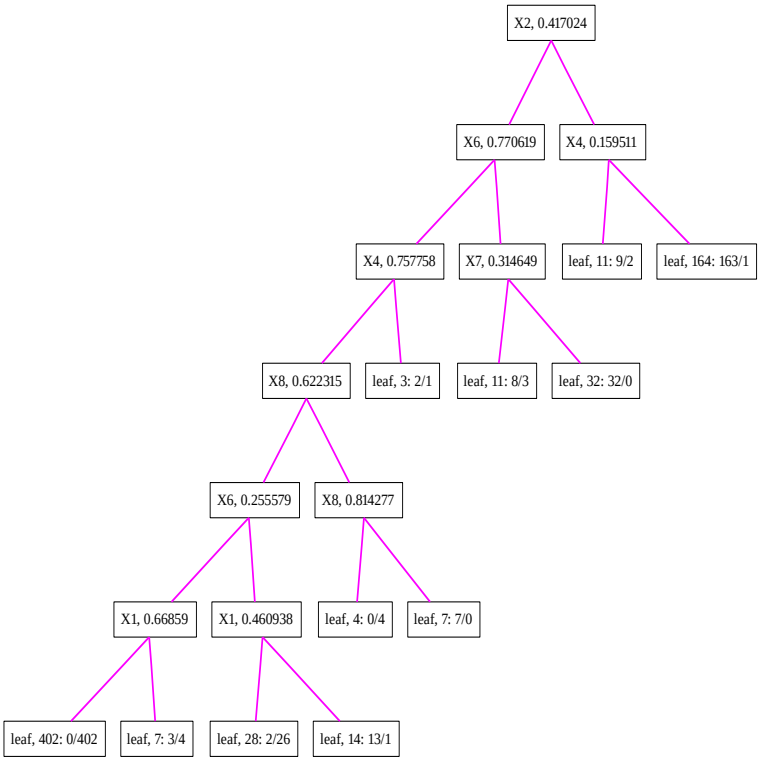
period. In particular this is true for runs $3$, $7$ and $8$. In the trace plots for the algorithm with the restructure move, no such effects were encountered. Thus, the use of the restructure move seems to improve the convergence and mixing properties of the Markov chain by preventing it from being trapped in local modes over long times. In turn this removes the need for restarts and the ad hoc weighting discussed in Chipman et al. (1998).

Exploration of posterior inferences involves inspecting trees generated from the sampler (as in Figure 9) and histograms of posterior samples of key quantities of interest, such as the tree size $m(T)$. In Figure 10 we see that the data give little support to trees with more than 15 terminal nodes, and an approximate posterior mode of 9 leaves. We have also tried pinball priors encouraging both smaller and larger trees, and the results are very insensitive to this choice. The middle panel in Figure 10 gives the posterior distribution for the log integrated likelihood. Most of the probability mass lies in the interval $(-81, -65)$, but the run visited trees with log integrated likelihood values up to $-60$. Chipman et al. (1998) reports log integrated likelihood values up to $-62.2$, with most lying below $-64$. As our run was longer than theirs, this is as one should expect. In addition to the tree size we also consider a statistic describing another property of the tree. Given a simulated tree, we first compute the posterior mean of $\theta$ in each of the leaves. If this value is larger than 0.5, subjects in this leaf have higher probability for being malign. Thus, we classify such leaves as malign and the remaining leaves as benign. We consider the fraction of the subjects where the subject status differ from the leaf status. We call this the "malign/benign mixing fraction". A low value indicates pure malign and benign leaves, whereas a high value results from more mixed leaves. The right panel in Figure 10 shows the histogram for this statistic. The mean value is $0.0248$, whereas the smallest value found in the run is $0.0132$. Thus, only a small degree of mixing occurs in the leaves. The "miss-classification rates" reported in Chipman et al. (1998) corresponds to mixing fractions down to $0.016$, though they do not report any corresponding mean value.

We may also easily summarize the posterior to assess relevance of each of the predictor variables. For each predictor we estimate the posterior probability that this predictor is used as a splitting variable (at least once), with results in Table 2. Most of the nine predictors have predictive relevance, which – given that one or a few may be truly predictive – is not surprising in view of the collinearity observed (the correlation between the predictors ranges from $0.34$ to $0.91$). The table provides the Monte Carlo estimates of posterior co-inclusion probabilities for pairs of variables as well as the marginal probabilities of inclusion for each.

Turning now to prediction, the entire analysis was repeated using a randomly selected 342 observations as training data, producing $10,000$ sampled tree samples. The posterior was used to make out-of-sample predictions on the remaining $341$ test observations. Figure 11 displays the results. With a simple threshold at 0.5 on the Monte Carlo estimates of the implied predictive probabilities, averaged over all 10,000 trees, the raw prediction error is $13$ out of $341$. By comparison, use of the greedy algorithm rpart in R (with default parameter settings) led to $23$ misclassifications in this 50:50 hold-out prediction assessment. Modifying rpart to permit any number of observations in each leaf led to $18$ misclassified cases.

To further assess prediction validity we also ran a full ten-fold cross validation analysis, each time randomly dividing the data set in ten parts with each part consisting of ten percent of the benign observations and ten percent of the malign observation. We repeated the ten fold cross validation ten times and this resulted in an average misclassification rate of $3.9\%$.



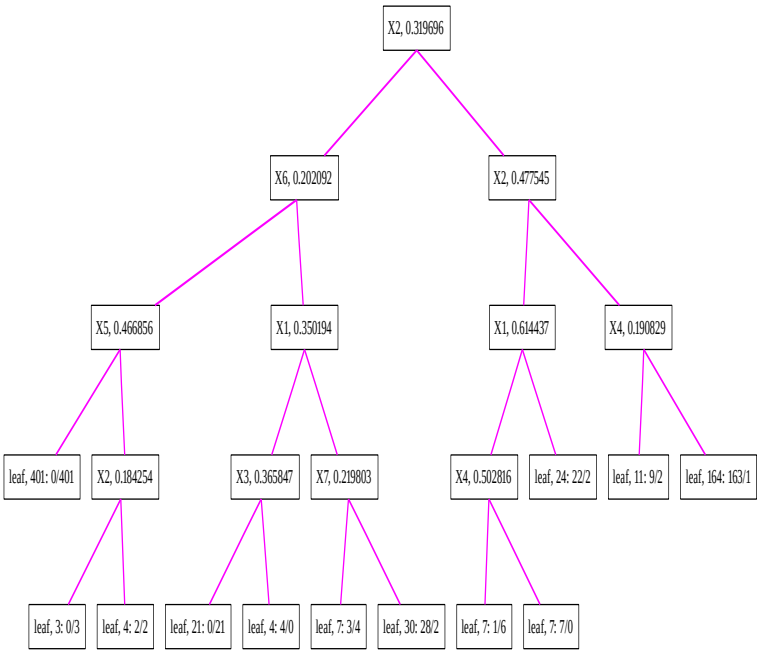




Figure 9: Breast cancer data example: Two trees chosen at random from the posterior distribution.

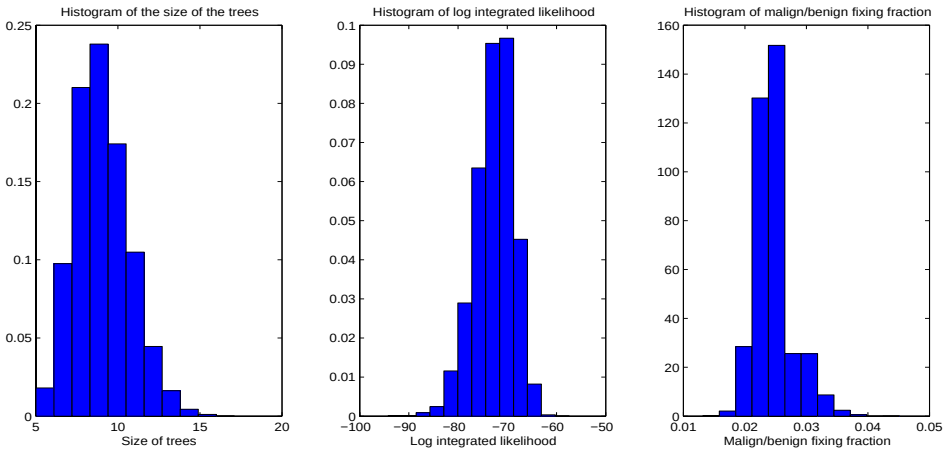



Figure 10: Breast cancer data example: Left: Posterior for tree size $m(T)$; Center: Posterior for log integrated likelihood; Right: Summary histogram of Malign/Benign mixing fraction.

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | *Name* |
|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | **0.98** | 0.96 | 0.61 | 0.47 | 0.51 | 0.98 | 0.38 | 0.60 | 0.26 | clump thickness |
| $x_2$ | 0.96 | **0.98** | 0.60 | 0.47 | 0.50 | 0.98 | 0.37 | 0.60 | 0.26 | uniformity of cell size |
| $x_3$ | 0.61 | 0.60 | **0.62** | 0.28 | 0.32 | 0.62 | 0.27 | 0.35 | 0.17 | uniformity of cell shape |
| $x_4$ | 0.47 | 0.47 | 0.28 | **0.48** | 0.24 | 0.48 | 0.15 | 0.28 | 0.14 | marginal adhesion |
| $x_5$ | 0.51 | 0.50 | 0.32 | 0.24 | **0.52** | 0.52 | 0.19 | 0.22 | 0.15 | single epithelial cell size |
| $x_6$ | 0.98 | 0.98 | 0.62 | 0.48 | 0.52 | **1.00** | 0.39 | 0.61 | 0.27 | bare nuclei |
| $x_7$ | 0.38 | 0.37 | 0.27 | 0.15 | 0.19 | 0.39 | **0.39** | 0.24 | 0.10 | bland chromatin |
| $x_8$ | 0.60 | 0.60 | 0.35 | 0.28 | 0.22 | 0.61 | 0.24 | **0.61** | 0.15 | normal nucleoli |
| $x_9$ | 0.26 | 0.26 | 0.17 | 0.14 | 0.15 | 0.27 | 0.10 | 0.15 | **0.27** | mitoses |

Table 2: Breast cancer data example: Posterior pairwise and marginal (on diagonal) model inclusion probabilities for the nine predictors.

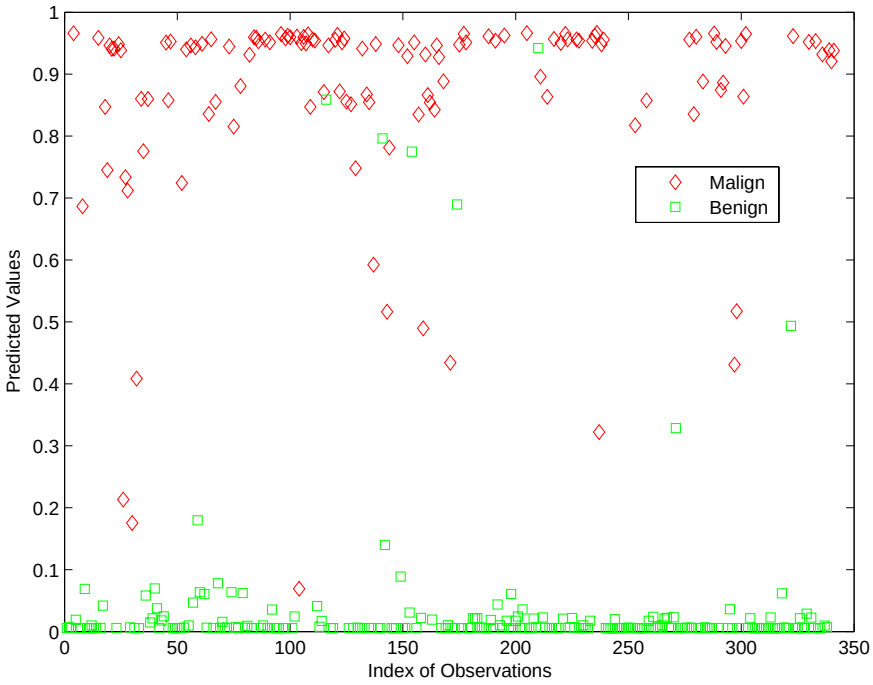



Figure 11: Breast cancer example: Predictive probabilities for the 50% sample test-set held out to assess predictive accuracy of the tree model fitted to the 50% training sample. The squares and diamonds represent subjects with benign and malign recurrence, respectively.

# 6 Closing Remarks

We have revisited the Bayesian formulation for CART models (Chipman et al., 1998; Denison et al., 1998) with two main contributions: a new prior specification for the tree structure which combines an explicit specification of the tree size distribution with the possibility of specifying a distribution for the tree shape, and a new "radical restructure" Metropolis–Hastings move for CART trees. We have explored aspects of robustness and prediction in Bayesian tree models under the new formulation, and demonstrated the very major improvements in the convergence and mixing properties of the posterior simulators using the new MCMC method. This moves Bayesian CART methodology to a position in which we can now generate realizations from CART tree posterior distributions, something not earlier possible. One should also note that even if our focus has been Bayesian CART trees, the same ideas can be used for any Bayesian tree models.

Convergence and mixing properties of an MCMC algorithm depends on the target distribution of interest. Thus, the number of observations and even the observed values may drastically influence the effectiveness of a Metropolis–Hastings algorithm. We have tried the new algorithm for two data sets. It is of interest to try the algorithm also for other and larger data sets. As the number of observations becomes larger, it is reasonable to believe that there will be fewer reconfigurations possible in the restructure move, and that the mixing properties will deteriorate correspondingly. Adopting a hierarchical prior for $\boldsymbol{\theta}$ as in Chipman et al. (2000) will most likely also have a negative effect on the mixing properties. The question is how large data sets the current algorithm is able to handle and whether other proposal distributions need to be considered. If convergence becomes a problem for large data sets, it is also possible to speed up the computations by adopting a Metropolis–Hastings algorithm where many potential new states are proposed in each iteration, see the discussion in Liu et al. (2000), Qin and Liu (2001) and Tjelmeland (2004). This situation is ideal for parallel computation and can, for example, be used to define a modified restructure move where one generates many trees consistent with the current partition of observations. Finally, we note that scale-up in terms of numbers of potential predictor variables is immediate, though there the general questions of collinearity and predictive similarity (or even equivalence) of multiple models raise other issues, as they do in all regression and prediction approaches.

# Acknowledgments

The authors acknowledge the support of grants from the NSF (DMS 0102227 and 0342172), and also the support of SAMSI and Duke University (H.T.) during the *Multiscale Model Development and Control Design* program during 2003-04. The authors also gratefully acknowledge the constructive comments of the editor, associate editor and three referees on an earlier version of the paper.

# References

Breiman, L., Friedman, J., Olshen, R., and Stone, C. *Classification and Regression Trees*. Belmont CA.: The Wadsworth statistics/probability series (1984).

Brooks, S. P. and Roberts, C. P. "Convergence assessment techniques for Markov chain Monte Carlo." *Statistics and Computing*, 8:319–335 (1998).

Chipman, H., George, E., and McCulloch, R. "Bayesian CART model search (with discussion)." *J. Am. Statist. Ass.*, 93:935–960 (1998).

—. "Hierarchical priors for Bayesian CART shrinkage." *Statistics and Computing*, 10:17–24 (2000).

Clark, L. and Pregibon, D. "Tree-based models." In Chambers, J. and Hastie, T. (eds.), *Statistical Models in S*, Wadsworth & Brooks/Cole computer science series. Wadsworth & Brooks/Cole Advanced Books & Software (1992).

Conover, W. *Practical Nonparametric Statistics*. New York: John Wiley & Sons (1971).

Cowles, M. K. and Carlin, B. P. "Markov chain Monte-Carlo convergence diagnostics: a comparative review." *J. Am. Statist. Ass.*, 91:883–904 (1996).

Denison, D., Mallick, B., and Smith, A. "A Bayesian CART algorithm." *Biometrika*, 85:363–377 (1998).

Green, P. J. "Reversible jump MCMC computation and Bayesian model determination." *Biometrika*, 82(4):711–732 (1995).

Huang, E., Chen, S., Dressman, H., Pittman, J., Tsou, M., Horng, C., Bild, A., Iversen, E., Liao, M., Chen, C., West, M., Nevins, J., and Huang, A. "Gene expression predictors of breast cancer outcomes." *The Lancet*, 361:1590–1596 (2003).

Liu, J. S., Liang, F. M., and Wong, W. H. "The multiple-try method and local optimization in Metropolis sampling." *J. Am. Statist. Ass.*, 95:121–134 (2000).

Pittman, J., Huang, E., Dressman, H., Horng, C., Cheng, S., Tsou, M., Chen, C., Bild, A., Iversen, E., Huang, A., Nevins, J., and West, M. "Integrated modeling of clinical and gene expression information for personalized prediction of disease outcomes." *Proceedings of the National Academy of Sciences*, 101:8431–8436 (2004a).

Pittman, J., Huang, E., Nevins, J., and West, M. "Prediction tree models in clinico-genomics." *Bulletin of the International Statistical Institute*, 54:76 (2003).

Pittman, J., Huang, E., Wang, Q., Nevins, J., and West, M. "Binary analysis of binary prediction tree models for retrospectively sampled outcomes." *Biostatistics*, 5:587–601 (2004b).

Qin, Z. S. and Liu, J. S. "Multi-point Metropolis method with application to hybrid Monte Carlo." *Journal of Computational Physics*, 172:827–840 (2001).

Tjelmeland, H. "Using all Metropolis–Hastings proposals to estimate mean values." Technical report, Statistics no. 4, Department of Mathematical Sciences, Norwegian University of Science and Technology, Trondheim, Norway (2004).

Wolberg, W. and Mangasarian, O. "Multisurface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology." *Proceedings of the National Academy of Sciences*, 87:9193–9196 (1990).

# Appendix: Pseudo code for MCMC moves

Function ProposeChange($\mathsf{T}$,$\mathcal{I}$)
  Set $T' = T$.
  Draw $u$ from a uniform distribution on $a(T')$.
  Draw $k_{T'}(u)$ from $\gamma(\cdot)$.
  Draw $\tau_{T'}(u)$ from $\delta_{k_{T'}(u)}(\cdot)$.
  Set $k_{T'}(v) = k_T(v)$ and $\tau_{T'}(v) = \tau_T(v)$ for $v \in a(T) \setminus \{u\}$.
  Return $\mathsf{T}'$
End.

Figure 12: Pseudo code for generating a potential new tree in the change move. Notation from Section 2 is used.

Function ProposeGrowPrune($\mathsf{T}$,$\mathcal{I}$)
  With probability $\frac{1}{2}I(m_0(T) > 1)$ set $I_{\text{grow}} = 0$, else set $I_{\text{grow}} = 1$.
  If $(I_{\text{grow}} = 1)$
    Draw $u$ from a uniform distribution on $b(T)$.
    Set $T' = T \cup \{l(u), r(u)\}$.
    Draw $k_{T'}(u)$ from $\gamma(\cdot)$.
    Draw $\tau_{T'}(u)$ from $\delta_{k_{T'}(u)}(\cdot)$.
    Set $k_{T'}(v) = k_T(v)$ and $\tau_{T'}(v) = \tau_T(v)$ for each $v \in a(T)$.
  Else
    Draw $u$ from a uniform distribution on the set $\{u \in a(T) | l(u), r(u) \in b(T)\}$.
    Set $T' = T \setminus \{l(u), r(u)\}$.
    Set $k_{T'}(v) = k_T(v)$ and $\tau_{T'}(v) = \tau_T(v)$ for each $v \in a(T')$.
  End.
  Return $\mathsf{T}'$.
End.

Figure 13: Pseudo code for generating a potential new tree in the grow/prune move. Notation from Section 2 is used.

Function ProposeSwap($\mathsf{T}$,$\mathcal{I}$)
  Set $T' = T$.
  Draw $(u_1, u_2)$ from a uniform distribution on $a(T')$
    that $(u_1, u_2)$ is a parent-child pair.
  Set $k_{T'}(u_1) = k_T(u_2)$, and $\tau_{T'}(u_1) = \tau_T(u_2)$.
  Set $k_{T'}(u_2) = k_T(u_1)$, and $\tau_{T'}(u_2) = \tau_T(u_1)$.
  Set $k_{T'}(v) = k_T(v)$ and $\tau_{T'}(v) = \tau_T(v)$ for $v \in a(T) \setminus \{u_1, u_2\}$.
  Return $\mathsf{T}'$
End.

Figure 14: Pseudo code for generating a potential new tree in the swap. Notation from Section 2 is used.

```
Function ProposeRestructure(T,I)
   Pick the leaves of the current tree, i.e. set L = L_T(I).
   [T', err] = DrawTree(L).
   If (err = 0)
      Return T'
   Else
      Return T
   End.
End.
```

```
Function DrawTree(L)
   Let m denote the number of leaves in L, i.e. m = |L|.
   If (m > 1)
      Find the set of all possible pairs of splitting variables
         and splitting intervals for L. Denote the result
         by {k_j, (τ_j^lower, τ_j^upper), j = 1, ..., n_I}.
      If (n_I > 0)
         Draw k(0) from a uniformly distribution on {1, ..., n_I}.
         Draw τ(0) from a uniform distribution on
            the interval (τ_{k(0)}^lower, τ_{k(0)}^upper).
         Set L_l and L_r equal to the subsets of L which have
            x^{k(0)} < τ(0) and x^{k(0)} ≥ τ(0), respectively.
         [S_1(T), err_1] = DrawTree(L_l); [S_2(T), err_2] = DrawTree(L_r).
         If (err_1 = 0 and err_2 = 0)
            Set err = 0.
            Return [T, err].
         End.
      End.
      Set T = ∅, T = [T, ·, ·] and err = 1.
      Return [T, err].
   Else                              /* Only one node in the tree */
      Set T = {0}, T = [T, ·, ·] and err = 0.
      Return [T, err].
   End.
End.
```

Figure 15: Pseudo code for generating a potential new tree, and the DrawTree function used. Notation from Section 2 is used.

Function ProposePartition($\mathsf{L}$,$r_{\text{iter}}$)
  Set $\widetilde{\mathsf{L}}^0 = \mathsf{L}$.
  Let $m$ denote the number of leaves in $\widetilde{\mathsf{L}}^0$ and let $\widetilde{\mathsf{L}}^0_l$ denote
    leaf number $l$, i.e. $\widetilde{\mathsf{L}}^0 = (\widetilde{\mathsf{L}}^0_1, \ldots, \widetilde{\mathsf{L}}^0_m)$.
  For $(i = 1 : r_{\text{iter}})$
    Draw $l_{\text{from}}$ from a uniform distribution on $\{1, \ldots, m\}$.
    Draw $l_{\text{to}}$ from a uniform distribution on $\{1, \ldots, l_{\text{from}} - 1,$
      $l_{\text{from}} + 1, \ldots, m\}$.
    Let $s$ denote the number of elements (observations) in $\widetilde{\mathsf{L}}^{i-1}_{l_{\text{from}}}$.
    If $(s > 1)$
      Use $\widetilde{\mathsf{L}}^{i-1}$ to define $s$ new leaf configurations by moving one
        observation from leaf number $l_{\text{from}}$ to leaf number $l_{\text{to}}$. Denote
        these configurations by $\breve{\mathsf{L}}^1, \ldots, \breve{\mathsf{L}}^s$.
      Set $\breve{\mathsf{L}}^0 = \widetilde{\mathsf{L}}^{i-1}$.
      Draw an integer $t \in \{0, 1, \ldots, s\}$ where $\mathrm{P}(t = j) \propto f(\boldsymbol{y}|\breve{\mathsf{L}}^j, \boldsymbol{\theta})$.
      Set $\widetilde{\mathsf{L}}^i = \breve{\mathsf{L}}^t$.
    Else
      Set $\widetilde{\mathsf{L}}^i = \widetilde{\mathsf{L}}^{i-1}$.
    End.
  End.
  Return $\widetilde{\mathsf{L}}^{r_{\text{iter}}}$
End.

Figure 16: Pseudo code for generating a potential new leaf configuration to be used in the restructure proposal.