Bridging trees for posterior inference on Ancestral Recombination Graphs

K. Heine[*], A. Beskos[†], A. Jasra[‡], D. Balding[§] and M. De Iorio[¶]

Abstract

We present a new Markov chain Monte Carlo algorithm, implemented in software Arbores, for inferring the history of a sample of DNA sequences. Our principal innovation is a bridging procedure, previously applied only for simple stochastic processes, in which the local computations within a bridge can proceed independently of the rest of the DNA sequence, facilitating large-scale parallelisation.

Keywords: Ancestral Recombination Graph, Markov chain Monte Carlo, Bayesian inference, coalescent, tree scanning

1 Introduction

A central problem in population genetics is to infer genealogical histories over a set of homologous DNA sequences, including e.g. shared lineages of genome segments back to their most recent common ancestors (MRCAs), or recombination, mutation and other genomic events that underlie the observed sequence variation. A Bayesian approach judiciously combines structured probability models for the evolution of genealogies backwards in time (i.e. prior probabilities) with information in the DNA sequences to provide posterior probabilities of lineages. It also gives rise to the highly complex computational challenge of probing the deduced posterior distributions. We make considerable progress towards achieving this goal by developing a new Markov chain Monte Carlo (MCMC) algorithm, and accompanying software Arbores. Our methodology adopts a data augmentation direction and implements a bridging procedure to impute the latent sequence of genealogical trees between given trees at two genome sites.

A prominent probabilistic evolutionary model for DNA sequences is the *coalescent with recombination* — an extension of the single-locus (Kingman's) coalescent [10, 11]. The *ancestral recombination graph* (ARG) [6, 5] is a graphical representation of this evolutionary model. Essentially, ARG is graph representation of gene genealogies and the centre of our attention in this work. ARG is central in population genetics and,

[*]Corresponding author, email:k.m.p.heine@bath.ac.uk, Department of Mathematical Sciences, University of Bath, Claverton Down, Bath, BA2 7AY, UK

[†]Department of Statistical Science, University College London, Gower Street, London, WC1E 6BT, UK

[‡]Department of Statistics and Applied Probability, National University of Singapore, 6 Science Drive 2, Singapore, 117546, SG

[§]Centre for Systems Genomics, School of BioSciences and School of Mathematics & Statistics, University of Melbourne, Vic 3010, Australia

[¶]Department of Statistical Science, University College London, Gower Street, London, WC1E 6BT, UK, and Yale-NUS College, 16 College Avenue West, Singapore, 138527, SG

more generally, in biology, and describes the relationship between sequences undergoing recombination. Recombination is one of the most important evolutionary forces as it increases genetic diversity and promotes adaptation through exchange of genetic material [1]. Conscientious modelling and learning of recombination is fundamental to gaining a better understanding of many biological processes, e.g. genome structure [1], phenotypic diversity [26], or mapping of disease genes. See [1] for a review of possible applications of ARG-based models. In Bayesian inference, ARG is necessary for determining the likelihood function of a sample of observed chromosomes in a population genetic model, and, as such, for parameter estimation and hypothesis testing. Unfortunately, due to the complexity of the ARG representation and the computational difficulties associated to inferring an ARG given a sample of chromosomes, ARGs have not been widely used and inference is often based on summary statistics and univariate techniques, with inevitable loss of information.

More formally, a coalescent with recombination model can be thought of as specifying a prior law for an ARG. We follow the original definition and treat the ARG as a random graph, whereas some authors use this term to refer to a fixed graph, usually the one representing the true but unknown underlying genealogy or an estimate of it. The inference from an observed set of DNA sequences can then, in theory, proceed by deriving the corresponding posterior ARG distribution. We adopt a discrete approximation to the infinite-sites mutation model, which implies that one of at most two alleles can be found at any given genomic site (in practice, triallelic DNA sites exist but are rare). It follows that an observed set of N homologous DNA sequences, each comprised of S genomic sites, can be represented as a binary matrix D with $D_{i,j} \in \{0, 1\}$ for $1 \leq i \leq N, 1 \leq j \leq S$. Inferring the ARG conditional on the data D is notoriously difficult and has been a renowned challenge in computational genetics. Firstly, the posterior distribution is highly concentrated compared to the prior, so that elementary approaches based on importance or rejection sampling from the prior distribution are unacceptably inefficient. Secondly, the ARG is defined on a large and complex space involving a high-dimensional product of continuous and finite spaces, such that an exhaustive iteration even over the finite spaces is infeasible. Standard MCMC methods can nowadays be usefully applied in some high dimensional models (e.g. large hierarchical models, or inverse problems [1]), but small changes in the random variables that specify an ARG can give substantially different graphs (i.e. the likelihood surface is highly discontinuous), so that standard random-walk type MCMC methods are very difficult to implement for ARG inference.

Earlier attempts to address this sampling task have involved mainly importance sampling [5, 4] or MCMC [16, 13]. In [16], the primary motivation is the estimation of the 'global' recombination rate, rather than the full ARG, enabling the use of less informative observations which makes the problem substantially simpler than the one considered here. The method of [13] is based on proposing a replacement for a part of the ARG in proportion to its prior probability. These methods scale badly with S and N. More recently, [15] introduced the sequentially-Markov coalescent (SMC) approximation of the ARG which assumes that the sequence of genealogical trees at genome sites forms a Markov process. The authors demonstrated that the approximation is accurate over a wide range of recombination rates. The SMC, coupled with the observed sequences, can be viewed as a hidden Markov model, with coalescent trees as hidden states, from which the data D are generated under the mutation model. The most recent advance in this field, [17], exploits the Markovian structure of the SMC approximation and proceeds by re-simulating the sequence of coalescent subtrees that correspond to a specific subset of observed haplotypes. With some resemblance to the method of [13],

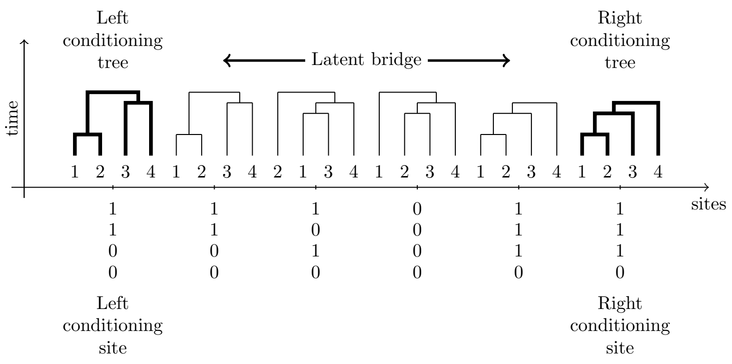


Figure 1: The data augmentation scheme for the SMC model. Given left and right conditioning trees our method samples an intervening bridge of coalescent trees, which is consistent with the relevant observed sites. In this example, the trees are comprised of 7 nodes (4 of which, at the bottom of the graph, are the leaves).

under the Markovian assumption, the resulting algorithm is a Gibbs sampler capable of performing inference on a moderately sized sample of sequences representing the complete genome.

1.1 Overview of the proposed method

A key contribution of our methodology is to reduce the computational complexity of the MCMC algorithm by dividing the inference task into a large number of subtasks, amenable to parallel computations. For each pre-specified genome segment, the coalescent trees at the initial and terminal sites are fixed, while a new sequence of trees is proposed at the intervening sites, compatible with the data D and the terminal trees. The proposed sequence is accepted or rejected according to the Metropolis-Hastings (MH) acceptance test. Due to the Markovian structure of the SMC process, these calculations require no information from outside the chosen genome segment, see Figure 1. This is crucial to the algorithm's ability to localise the computations and control the combinatorial complexity.

This approach is similar in principle to bridging methods for discretely-observed diffusion processes (see e.g. [19]) or Markov jump-processes (see e.g. [2]). In recognition of this conceptual similarity, we refer to the proposed sequence of coalescent trees over a genome segment as a *bridge*, and to our algorithm as the *tree-bridging MCMC sampler*. We are not aware of previous uses of bridging in such a complex state space. The major challenge for our algorithm involves developing an efficient way of generating bridges, which requires three main steps:

1. **Tree Scanning:** We construct a bridge by a method inspired by the *tree scanning* algorithm [22, 23], [7, page 328]. Starting with the left conditioning tree (see Figure 1), the tree at each subsequent site is either the same, corresponding to an identity operation, or a subtree is cut off (*pruned*) and then reattached (*regrafted*) onto another branch, which is called a subtree-prune-and-regraft (SPR) operation [22, 20, 21]. We iterate exhaustively over all such operations at each site, until all possible paths of coalescent trees have been generated up to the right conditioning site.

Tree scanning delivers paths of trees, each endowed with a total time ordering of the nodes (i.e. the coalescences or branch merges), but the exact times associated with the nodes are not yet fixed. Moreover, most tree paths will typically not match the right terminal condition.

2. **Time Adjustment:** Node times at the leftmost tree are fixed by the conditioning. Each non-identity SPR operation within a tree path introduces a new node. For each path, we check whether the times of the new nodes can be set so that the times in the right terminal tree are realised. If not, the path is discarded, otherwise the times of the nodes that are present in the right conditioning trees are fixed to their value in that conditioning tree.

3. **Sampling:** The last step is the generation of the complete bridge with all nodes affixed (i.e. their times determined). We choose a coalescent tree path at random from the set resulting after Step 2. For this path, we generate times for each node not present in either terminal tree, from the uniform or exponential distribution (see later sections for details) in the permitted interval that respects the tree structures and node orderings.

The process is repeated for a number of bridges spanning the genome region, and overlapping so that each genome site is non-terminal in at least one bridge, thus is able to vary.

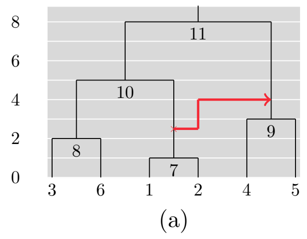
The remainder of this paper is organised as follows. Section 2 reviews the Markovian dynamics of the SMC model, and the corresponding likelihood of the observed DNA sequences. Section 3 presents the MCMC algorithm for sampling from the ARG posterior distribution. Section 4 shows some heuristics for reducing the computational cost of the MCMC proposal. Section 5 gives numerical results from an example application of the new algorithm. We conclude with some remarks in Section 6.

2 Hidden Markov model for ARG inference

The adoption of the SMC approximation for the law of the ARG enables us to formulate the inference problem in the hidden Markov model framework as described in this section.

2.1 Hidden tree process

A realisation of the SMC process is a sequence (or path) of coalescent trees $T = (T_i)_{1\leq i\leq S}$, where each T_i is a rooted binary tree. A tree branch is a directed edge identified by an ordered pair of nodes. Branches are named after the child node, so (u, v) is referred to as branch v. Each tree has a unique *root* node that has no parents. Any other node v has a parent $\mathrm{pa}(v)$. The children of v are denoted by $\mathrm{ch}(v)$. Associated with each observed sequence is a leaf node v such that $\mathrm{ch}(v) = \emptyset$. We identify the nodes of each T_i with integers $\{1, \ldots, 2N-1\}$, the first N nodes being the leaves. Each tree T_i is fully specified by its topology $\mathbf{C}_i$ and node times $\mathbf{t}_i = (t_{i,1}, \ldots, t_{i,2N-1})$ as exemplified in Figure 2a. The topology $\mathbf{C}_i$ can be represented as a $(N-1) \times 2$ matrix whose kth row includes the two elements of $\mathrm{ch}(N+k)$. The two elements of each row of $\mathbf{C}_i$ are placed in increasing order. We have $t_{i,n} = 0$ for the leaf nodes, while non-leaf nodes are indexed in increasing time order.



$$\mathbf{C}_i = \begin{bmatrix} 1 & 2 \\ 3 & 6 \\ 4 & 5 \\ 7 & 8 \\ 9 & 10 \end{bmatrix}$$

(b)

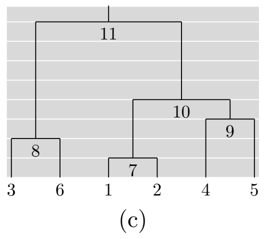


Figure 2: (a) A graph representation of a coalescent tree T_i having the topology $\mathbf{C}_i$ shown in (b), with node times $\mathbf{t}_i = (0,0,0,0,0,0,1,2,3,5,8)$, $u_i = 7$, $v_i = 9$, $r_i = 2.5$, $w_i = 4$. (c) The resulting tree after applying the SPR operation (u_i, v_i, r_i, w_i) to the coalescent tree T_i.

Given $T_i = (\mathbf{C}_i, \mathbf{t}_i)$, for some $1 \le i \le S-1$, the next tree T_{i+1} is determined by the occurrence (or not) of a recombination between sites i, $i+1$, and the specification of such recombination. A recombination is represented here via an SPR operation characterised by a quadruple consisting of two nodes, u_i, v_i, and two positive real times r_i, w_i. If no recombination happens between sites i and $i+1$ then by convention $(u_i, v_i, r_i, w_i) = (0,0,0,0)$, representing the identity SPR operation, and consequently $T_i = T_{i+1}$. If a recombination does occur, then the subtree[1] rooted at node u_i is pruned from the tree at time r_i and subsequently regrafted back into the tree onto branch v_i at time $w_i > r_i$, as shown in Figure 2.

We proceed to a detailed description of the dynamics of the hidden tree process. We need to provide a few definitions. For measurable spaces $(\mathbb{X}, \mathcal{X})$, $(\mathbb{Y}, \mathcal{Y})$, $(\mathbb{Z}, \mathcal{Z})$ and probability kernels $K_1 : \mathbb{X} \times \mathcal{Y} \to [0,1]$ and $K_2 : \mathbb{Y} \times \mathcal{Z} \to [0,1]$ we define the probability kernel $K_1 K_2 : \mathbb{X} \times (\mathcal{Y} \otimes \mathcal{Z})$ so that for $x \in \mathbb{X}$, $A \in \mathcal{Y} \otimes \mathcal{Z}$

$$(K_1 K_2)(x, A) = \int K_1(x, \mathrm{d}y) \int K_2(y, \mathrm{d}z) \mathbb{I}_A(y, z).$$

For a kernel $K : \mathbb{X} \times \mathcal{Y} \to [0,1]$ and a probability measure $\mu : \mathcal{X} \to [0,1]$ we define the probability measure $\mu K : \mathcal{X} \otimes \mathcal{Y} \to [0,1]$ such that, for $A \in \mathcal{X} \otimes \mathcal{Y}$

$$(\mu K)(A) = \int \mu(\mathrm{d}x) \int K(x, \mathrm{d}y) \mathbb{I}_A(x, y).$$

For probability measures μ_1, μ_2 defined on $(\mathbb{X}, \mathcal{X})$, $(\mathbb{Y}, \mathcal{Y})$, respectively, we let $\mu_1 \otimes \mu_2$ denote the product measure on $(\mathbb{X} \times \mathbb{Y}, \mathcal{X} \otimes \mathcal{Y})$. The k-fold iterate of a kernel K is written as K^k.

The SMC process is initiated with a standard coalescent tree, followed by Markovian transitions defined via SPR operations. Formally, we define the initial distribution μ and the kernel K via the coordinate-wise decompositions

$$\mu = \mu_{\mathbf{C},\mathbf{t}} K_u K_r K_w K_v, \quad K = K_{\mathbf{C},\mathbf{t}} K_u K_r K_w K_v,$$

from which we obtain the law of the SMC process as μK^{S-1}. We now define the probability measure $\mu_{\mathbf{C},\mathbf{t}}$ and kernels $K_{\mathbf{C},\mathbf{t}}$, K_u, K_r, K_w, K_v. First, we note that $\mu_{\mathbf{C},\mathbf{t}}$

[1]Without exception we use *subtree* to refer to a tree whose leaves are also leaves in the original tree.

simply corresponds to the law of the standard coalescent tree [11, 10, 8] with a formal definition as follows. The law $\mu_{\mathbf{C}}$ of the initial canonical form $\mathbf{C}_1$ is defined such that for all $1 \leq i \leq N-1$ the pair $(\mathbf{C}_1(i,1), \mathbf{C}_1(i,2))$ – where $\mathbf{C}_1(i,j)$ denotes the element on the ith row and jth column of $\mathbf{C}_1$ – is uniformly distributed over the set

$$\left\{ (j,k) \in \mathcal{N}_{i-1} \times \mathcal{N}_{i-1} : j < k \right\},$$

starting from the set of leaves $\mathcal{N}_0 = \{1, \ldots, N\}$, and then having, for $1 \leq i < N-1$,

$$\mathcal{N}_i = (\mathcal{N}_{i-1} \setminus \{\mathbf{C}_1(i,1), \mathbf{C}_1(i,2)\}) \cup \{N+i\}.$$

To define the law $\mu_{\mathbf{t}}$ of the coalescence times $\mathbf{t}_1$, we write $f(x;\beta) = \beta e^{-\beta x}$, $x \geq 0$, for the probability density of the exponential distribution with parameter $\beta > 0$ and define $\mu_{\mathbf{t}} = \mu_{t_1} \otimes \cdots \otimes \mu_{t_{2N-1}}$, where

$$\mu_{t_i}(\mathrm{d}x) = \begin{cases} \delta_0(\mathrm{d}x), & 1 \leq i \leq N \\ f\left(x;\beta_0\right)\mathrm{d}x, & i = N+1 \\ f\left(x - t_{i-1};\beta_{i-(N+1)}\right)\mathrm{d}x, & i > N+1 \end{cases}$$

and $\beta_i = \binom{N-i}{2}$, $0 \leq i < N-1$. The initial distribution is simply $\mu_{\mathbf{C},\mathbf{t}} = \mu_{\mathbf{C}} \otimes \mu_{\mathbf{t}}$. Conditionally on a given coalescent tree $T_i = (\mathbf{C}_i, \mathbf{t}_i)$, $1 \leq i \leq S-1$, with some probability there will be no recombination between sites i and $i+1$, or there will be a recombination at a position chosen uniformly along the branches of the tree, with involved recombination rate parameter $\rho > 0$. Formally, the conditional law of u_i is

$$K_u(T_i, \mathrm{d}x) = \left(1 - e^{-\rho L_i}\right) \sum_{n=1}^{2N-2} \frac{l_{i,n}}{L_i} \delta_n(\mathrm{d}x) + e^{-\rho L_i} \delta_0(\mathrm{d}x),$$

where the total branch length L_i and branch lengths $l_{i,n}$ of T_i are defined as

$$L_i := \sum_{n=1}^{2N-2} l_{i,n}. \quad \text{where } l_{i,n} := t_{i,\mathrm{pa}(n)} - t_{i,n}. \tag{1}$$

For a given coalescent tree T_i and a pruned node u_i, the conditional law of the pruning time r_i is defined as

$$K_r(T_i, u_i, \mathrm{d}x) = \mathbb{I}(u_i \neq 0) \frac{\mathbb{I}(x \in [t_{i,u_i},\ t_{i,\mathrm{pa}(u_i)}])}{l_{i,u_i}} \mathrm{d}x + \mathbb{I}(u_i = 0)\delta_0(\mathrm{d}x).$$

The indicator functions $\mathbb{I}(u_i = 0)$ and $\mathbb{I}(u_i \neq 0)$ specify the conditional distributions of r_i in the two cases: recombination occurs ($u_i \neq 0$) or it does not ($u_i = 0$). Next, the SMC model assumes that branch u_i is pruned at time r_i, so the segment of the branch above this time is deleted. The remaining part of the separated branch is extended backwards in time from r_i and is re-attached to the main body of the tree according to the standard coalescent tree dynamics, i.e. according to a Poisson process with rate equal to the number of existing branches at any time instance. Formally, given (T_i, u_i, r_i), let k be the number of nodes above the pruning time r_i, including the parent node $\mathrm{pa}(u_i)$, and $\tilde{t}_1 < \cdots < \tilde{t}_k$ their times with the convention $\tilde{t}_0 = r_i$, $\tilde{t}_{k+1} = \infty$. The conditional law of the coalescence time w_i given (T_i, u_i, r_i) is

$$K_w(T_i, u_i, r_i, \mathrm{d}x) = \mathbb{I}(u_i \neq 0) f_{T_i,u_i,r_i}(x)\mathrm{d}x + \mathbb{I}(u_i = 0)\delta_0(\mathrm{d}x),$$

where, for $F(\,\cdot\,;\,\beta)$ denoting the cumulative distribution function of $\mathrm{Exp}(\beta)$, $\beta > 0$, and $\bar{F} = 1 - F$,

$$f_{T_i,u_i,r_i}(x) = \sum_{j=0}^{k} f(x - \tilde{t}_j; \beta_j)\,\mathbb{I}\,[\,x \in (\tilde{t}_j, \tilde{t}_{j+1})\,]\left[\prod_{l=0}^{j-1} \bar{F}(\tilde{t}_{l+1} - \tilde{t}_l; \beta_l)\right],$$

with

$$\beta_j = \begin{cases} k+1-j, & \tilde{t}_j \geq t_{i,\mathrm{pa}(u_i)} \\ k-j, & \text{otherwise.} \end{cases}$$

The conditional law of the regraft node v_i given (T_i, u_i, w_i) is

$$K_v(T_i, u_i, w_i, \mathrm{d}x) = \frac{\mathbb{I}(u_i \neq 0)}{|A(T_i, u_i, w_i)|} \sum_{n \in A(T_i,u_i,w_i)} \delta_n(\mathrm{d}x) + \mathbb{I}(u_i = 0)\delta_0(\mathrm{d}x),$$

where

$$A(T_i, u_i, w_i) = \left\{n \in \{1, \ldots, 2N-1\} \setminus \{u_i\} : w_i \in (t_{i,n}, t_{i,\mathrm{pa}(n)})\right\}.$$

The variables u_i, v_i, w_i specify an SPR operation that applies to a coalescent tree T_i to determine the next one which we denote by $\mathrm{SPR}(T_i, u_i, v_i, w_i)$, hence we formally write

$$K_{\mathbf{C},\mathbf{t}}(T_i, u_i, v_i, w_i, \mathrm{d}x) = \delta_{\mathrm{SPR}(T_i,u_i,v_i,w_i)}(\mathrm{d}x).$$

2.2 Observation process

Information about the tree process $T = (T_i)_{1 \leq i \leq S}$, is obtained through the data $D = (D_{i,j}) = (D_i)$, with columns $D_i \in \{0,1\}^N$. The mutation model generating D given T is a discrete version of the infinite-sites model [6], and involves a mutation rate parameter $\theta > 0$. Given T_i, for a site $1 \leq i \leq S$, the model specifies that with probability $\exp(-\theta L_i)$ no mutations occur at site i, for L_i is as defined in (1), otherwise exactly one mutation arises, and the place where the mutation occurs is uniformly distributed over the branches of T_i. If a mutation has occurred at a site i, we can now infer from D_i the branch on which it arose: it is the unique branch b_i such that $D_{i,n} = 1$ if either $n = b_i$ or n is a descendant of b_i, otherwise $D_{i,n} = 0$. The assumption of at most one mutation per site plays a role in our method as it implies that the sequences of trees inconsistent with it are not permitted. Under this assumption, the likelihood at site i is

$$\mathcal{L}_i(T_i) = P(D_i|T_i) = \begin{cases} e^{-\theta L_i}, & \sum_n D_{i,n} = 0, \\ \left(1 - e^{-\theta L_i}\right) \frac{l_{i,b_i}}{L_i}, & \sum_n D_{i,n} > 0. \end{cases} \tag{2}$$

Given $T = (T_i)_{1 \leq i \leq S}$, mutations occur independently at each site, so the joint likelihood over any set of sites is the product of $\mathcal{L}_i(T_i)$ over i in the set. If $\mathcal{L}_i(T_i) > 0$, then we say that $T_i = (\mathbf{C}_i, \mathbf{t}_i)$ is compatible with D_i, and since the compatibility of T_i depends only on the topology $\mathbf{C}_i$ and not on $\mathbf{t}_i$, we also say that $\mathbf{C}_i$ is compatible with D_i. If $\mathbf{C}_i$ is compatible with D_i for all $i \in \{1, \ldots, S\}$, then we say that T is compatible with D.

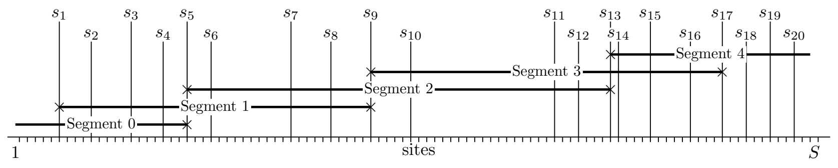


Figure 3: Segregating sites $s_1, \ldots, s_{20}$ are highlighted. Sites within jth segment ($j = 0, \ldots, 4$) are illustrated with conditioning sites marked as $\times$.

3 Tree-bridging MCMC algorithm

We develop a Metropolis-Hastings MCMC algorithm for sampling from $\pi = P(T|D)$, the distribution of the tree process T over a genome interval, given the observation process D. Convergence of the algorithm is assured from any initial state for T, given regularity conditions on the chain (e.g. Harris recurrent, see e.g. [18, page 221]). However, a choice of initial state that is realistic given D can give an important reduction in the time to convergence. One way to generate a good initial state is to use an existing algorithm from the literature to obtain a point estimate ARG by seeking to minimise the number of recombinations. Arbores uses a variant of the SHRUB algorithm of [24] (see also [7, page 312]) for this purpose.

3.1 Segment selection

The bridges are constructed over genome segments, which can be selected in various ways. All sites should be non-fixed, i.e. not the conditioning sites, in at least one segment, allowing them to vary. The two end sites of the genome must be treated separately as they will involve only one-sided conditioning. By default, Arbores defines the segments so that, for a chosen $m \geq 1$, the first segment stops at and conditions upon the $(m + 1)$th *segregating site* (site where at least one 0 and one 1 are observed). Other segments contain $2m + 1$ consecutive segregating sites (including the left/right conditioning sites, which are always segregating) of which the leftmost $m{+}1$ are shared with the preceding segment. The last segment (with a left-side only conditioning) includes $m + 1$ segregating sites from the preceding segment, plus up to m additional segregating sites to reach the right end site of the genome. An example of the segment selection procedure is given in Figure 3. More formally, the segments are defined as follows. Let m_S denote the number of segregating sites and let $s = (s_1, \ldots, s_{m_\mathrm{S}})$, be the indices of the segregating sites in a strictly increasing order. We call $2m + 1$ the *bridge length* and assume that $m \ll m_\mathrm{S}$. For each jth segment, where $j = 0, \ldots, J$, the initial and terminal site indices (α_j, β_j) are defined as follows. For $j = 0$, we set $(\alpha_0, \beta_0) = (1, s_{m+1})$, and thereafter

$$\alpha_j = s_{m(j-1)+1}, \quad \beta_j = s_{m(j+1)+1}, \quad 1 \leq j < J,$$

where

$$J := \max \{j \in \mathbb{N} : mj + 1 < m_\mathrm{S}\},$$

and $(\alpha_J, \beta_J) = (s_{m(J-1)+1}, S)$. Figure 3 shows an example illustration of the deduced segments for a case when $m = 4$ and $m_\mathrm{S} = 20$.

3.2 Generation of bridge proposals

We now focus on a specific bridge, so let T and π here denote the restrictions of the tree process and target distribution onto this chosen bridge. We denote by $q(T)$ the proposal distribution for the bridge tree process given its current state. A bridge is updated by first sampling $T' \sim q(T)$, and then, with probability

$$\min\left(1, \frac{\pi(T')q(T)}{\pi(T)q(T')}\right),$$

the current state T is replaced with T'; otherwise the current bridge T is retained. We now describe the mechanism for generating bridge proposal from $q(T)$. Each quadruple (u_i, v_i, r_i, w_i) specifying an SPR operation for coalescent tree T_i, $L \leq i \leq R$, must either be equal to $(0, 0, 0, 0)$, representing the identity operation, or satisfy the following conditions: i) u_i is not the root, ii) $v_i \notin \{u_i, \mathrm{pa}(u_i)\}$ (this follows simply by our definition of v_i), iii) $w_i > r_i$.

3.2.1 Step 1: Tree scanning

The first step is to generate the set of all possible sequences of tree topologies over the bridge, compatible with the data D and the assumption of at most one recombination between adjacent sites. We start with $\mathbf{C}_L$, the topology of the left conditioning tree T_L, and subsequently construct a sequence, or a path, of topologies $(\mathbf{C}_i)_{L<i\leq R}$ according to all possible choices of SPR operation nodes, (u_i, v_i). Paths that include a topology not compatible with the data are discarded. For the first (leftmost) segment, we have conditioning from the right but not from the left, and so we proceed in reverse direction, from right to left, but otherwise the construction is the same. The tree scanning step is theoretically straightforward, but can be computationally intensive. Each topology at site i can give rise to $\mathcal{O}(N^2)$ topologies at site $i + 1$ (see [21]); the complexity of constructing these topology paths over the segment can be exponential in its length. Heuristics to make the algorithm practically feasible are described later in Section 4.

3.2.2 Step 2: Time adjustment

By construction, the topology paths generated in Step 1 are consistent with $\mathbf{C}_L$, but many of them may not be consistent with $\mathbf{C}_R$ (this is not required for the rightmost segment, for which this time adjustment step can be skipped). Given a topology path $(\mathbf{C}_i)_{L\leq i\leq R}$, one can associate any non-identity transition from $\mathbf{C}_i$ to $\mathbf{C}_{i+1}$ with the deletion of a node in $\mathbf{C}_i$ and the creation of a new one in $\mathbf{C}_{i+1}$. We view this pair of events as a *move* of a single node: the composition of all such moves from site L to site R for a node make up the *trajectory* of that node. Thus, when $\mathbf{C}_i \neq \mathbf{C}_{i+1}$, one node in $\mathbf{C}_i$ moves to a new position, while all other nodes are unchanged. Figure 4 shows an example of the node trajectories from L to R. The dashed lines depict node trajectories over consecutive sites. The leaf nodes never move and are therefore omitted from Figure 4.

For each site i and non-leaf node n, we define the indicators $\kappa^-(i, n)$ and $\kappa^+(i, n)$ so that $\kappa^-(i, n) = 1$ (resp. 0) if node n at site i moved (resp. did not move) at the sites from L to i. Indicators $\kappa^+(i, n)$ are defined in a similar manner, but referring to sites from i to R. Given these definitions we proceed as follows. First, we remove paths that do not terminate at $\mathbf{C}_R$. Then, for each path $(\mathbf{C}_i)_{L\leq i\leq R}$ we identify the subset $\mathcal{F}$ of

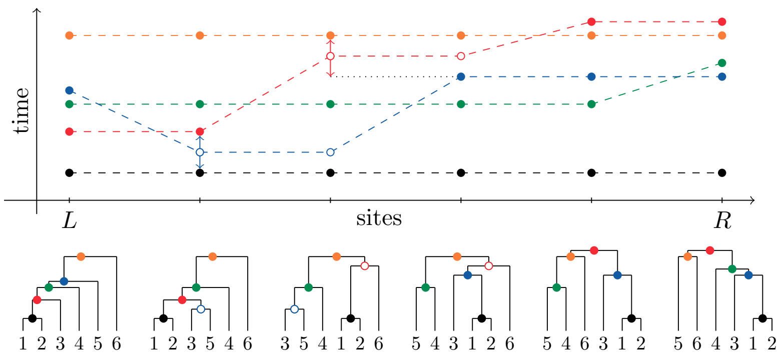


Figure 4: Node trajectories for $N = 6$ sequences and 5 recombinations. The underlying tree path is shown below the axes. Here $\kappa^-(L{+}2, 8) = \kappa^+(L{+}2, 8) = 1$ and $\kappa^+(R{-}2, 9) = 1$, but $\kappa^-(R{-}2, 9) = 0$. The nodes affixed by $\mathbf{t}_L$ and $\mathbf{t}_R$ are depicted by ●, ●, ●, ●, and ●, while the free nodes depicted by ○ and ○.

$\{L, \ldots, R\} \times \{N{+}1, \ldots, 2N{-}1\}$ defined as

$$\mathcal{F} = \big\{(i, n) : \kappa^-(i, n) = 0 \text{ or } \kappa^+(i, n) = 0\,\big\}. \tag{3}$$

We now take the times $\mathbf{t}_L$, $\mathbf{t}_R$ of the left and right conditioning trees, T_L and T_R, also under consideration. All nodes in $\mathcal{F}$ will be affixed to some time instance in $\mathbf{t}_L$ (if $\kappa^-(n, i) = 0$) or $\mathbf{t}_R$ (if $\kappa^+(n, i) = 0$). The nodes in $\mathcal{F}$ are termed *affixed* while all the other nodes are *free*. Free nodes can only appear in trajectories involving at least two node moves, otherwise they will be affixed to a time either in $\mathbf{t}_L$ or $\mathbf{t}_R$.

Not all topology paths will be compatible with T_L and T_R. In practice, we iterate over the entire set of topology paths and verify for each individual path whether the free nodes can be affixed according to T_L and T_R; if not, the path is discarded. Figure 5 shows two paths: one can be affixed while the other cannot.

3.2.3 Step 3: Sampling

After Step 2, we have a set of topology paths that are compatible with the two-sided conditioning. To fully specify the bridge proposal we sample one of these paths uniformly at random, then generate the times of its free nodes and the pruning times. In Figure 4, there are four free nodes but two free times to sample as the four nodes are comprised of two pairs, the paired nodes being positioned at the same time instance. There can be at most one free time per site and we generate them from left to right. The vertical two-headed arrows in Figure 4 depict the sampling domains for this example. Notice that for the domain of the free node at site $L{+}2$ one must also consider the affixed nodes on both sides, so that the obtained sequence of coalescent trees is compatible with the orderings implied by the chosen topology path.

To provide an explicit formula, suppose that $t_{i,n}$ has been identified as a free time, the first subscript i designating the site and the second subscript n designating the node. Once $t_{i,n}$ is affixed, we let $l \in \{0, \ldots, R{-}i{-}1\}$ denote the number of subsequent sites where this node will remain at the same time, and $n_1, \ldots, n_l \in \{N+1, \ldots, 2N-1\}$ the labelling of this node at these next sites. The sampling domain of $t_{i,n}$ is then

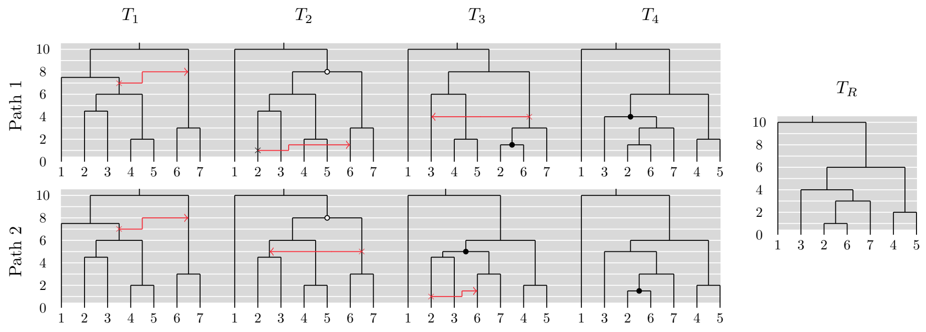


Figure 5: Two paths and their generating SPR operations. Path 1 is compatible with the right conditioning tree but Path 2 is not. For Path 2, the conditioning requires the node at time 5 to be adjusted to time 4, but this cannot be done as it is clear from the second tree of Path 2, that the node created by the second SPR operation must be in the interval $(4.5, 6)$. Hence Path 2 is not compatible with T_L, T_R. Nodes introduced by the SPR operations that are affixed are depicted by $\bullet$ and free nodes are depicted by $\circ$.

$\mathcal{D}(t_{i,n}) = [t_i^{\downarrow}, t_i^{\uparrow}]$, where

$$t_i^{\downarrow} := \max\{t_{i,n-1}, t^{\downarrow}_{i+1,n_1}, \dots, t^{\downarrow}_{i+l,n_l}\},$$
$$t_i^{\uparrow} := \min\{t_{i,n+1}, t^{\uparrow}_{i+1,n_1}, \dots, t^{\uparrow}_{i+l,n_l}\},$$

and $t^{\downarrow}_{i+s,n_s}$ (resp. $t^{\uparrow}_{i+s,n_s}$) denotes the largest (resp. smallest) affixed time below (resp. above) node n_s at site $i+s$, for $1 \le s \le l$. Note that some of the times involved in the $\min$ can be $+\infty$ if there is no node with affixed time above the node of interest, so the domain $\mathcal{D}(t_{i,n})$ may or may not be bounded from above. For bounded domains we choose a uniform distribution and for unbounded domains we choose the shifted exponential distribution with density $\exp(-x+s)$, for all $x > s$ where $s = \inf\{x \in \mathcal{D}(t_{i,n})\}$. The sampling of the pruning times is easier as, once all nodes are affixed, we simply have the sampling domains

$$\mathcal{D}(r_i) = \big[\, t_{i,u_i},\ \min(t_{i,\mathrm{pa}(u_i)}, w_i)\,\big], \qquad L \le i < R.$$

Again, we choose a uniform distribution. The generation of the bridge proposals is thus completed.

3.3 Acceptance probability

We introduce the notation $\tilde{T}_i = \{T_i, u_i, v_i, r_i, w_i\}$ to refer to the coalescent tree T_i endowed with its pruning and regraft nodes and times. We write $T' = (\tilde{T}'_L, \dots, \tilde{T}'_R)$ for the bridge proposal and $T = (\tilde{T}_L, \dots, \tilde{T}_R)$ for the current state, with $T_L = T'_L$, $\tilde{T}_R = \tilde{T}'_R$. As the tree process is Markovian and the observations independent (given the trees), the target distribution ratio in the Metropolis-Hastings acceptance probability can be written as

$$\frac{\pi(T')}{\pi(T)} = \frac{\prod_{i=L}^{R-1} K(\tilde{T}'_{i-1}, \tilde{T}'_i) \prod_{i=L+1}^{R-1} \mathcal{L}_i(T'_i)}{\prod_{i=L}^{R-1} K(\tilde{T}_{i-1}, \tilde{T}_i) \prod_{i=L+1}^{R-1} \mathcal{L}_i(T_i)}, \tag{4}$$

under the convention that $K(\tilde{T}_0', \tilde{T}_1') = \mu(\tilde{T}_1')$. Let $Q_{\mathcal{P}}$ denote the proposal probability mass function of the topology paths after Step 2, let $(\mathbf{C}_L', \ldots, \mathbf{C}_R')$ denote the topology paths corresponding to T', and let M' denote the number of free times $(f_1', \ldots, f_{M'}')$ for T' with joint density $Q_f(f_1', \ldots, f_{M'}')$. Also, we let $Q_r(r_L', \ldots, r_{R-1}' \mid f_1', \ldots, f_{M'}')$ denote the conditional joint density for the pruning times. The corresponding quantities for the current position are defined analogously in the obvious way by omitting the prime. We complete the specification of the acceptance probability in our bridging algorithm via the calculation

$$\frac{q(T)}{q(T')} = \frac{Q_{\mathcal{P}}(\mathbf{C}_L, \ldots, \mathbf{C}_R)}{Q_{\mathcal{P}}(\mathbf{C}_L', \ldots, \mathbf{C}_R')} \times \frac{Q_f(f_1, \ldots, f_M)}{Q_f(f_1', \ldots, f_{M'}')} \times \frac{Q_r(r_L, \ldots, r_{R-1} \mid f_1, \ldots, f_M)}{Q_r(r_L', \ldots, r_{R-1}' \mid f_1', \ldots, f_{M'}')}. \tag{5}$$

The above algorithm corresponds to an Independence Metropolis-Hastings algorithm. It is easy to check that, under reasonable choices for the proposal, $\sup_T \pi(T)/q(T) < \infty$, with the supremum taken over the support of $\pi(\cdot)$, thereby guaranteeing uniform ergodicity for the bridge sampler. To see that, notice that the number of the permitted topology paths is finite, so one only needs to assign non-zero probability to each one of them (e.g. via the discrete uniform mentioned earlier in the text); then, the number of free regraft and pruning times is also finite, and one needs to select a lower bounded density for the times of finite support (e.g. the continuous uniform referred to earlier) and a proper density for the unbounded times (e.g. the 'prior' exponential density chosen above will dominate the posterior density). In terms of the complete method, the use of overlapping blocks implies that uniform ergodicity will also hold as long as all topologies over the complete genome supported by the posterior can be visited by the proposal during the execution of the algorithm. We conjecture that this is true due to the flexibility of the method, but a rigorous proof requires elaborate work on a theme exceeding the scope of the paper. Such ergodicity results are of qualitative nature, and the efficiency of the method is determined by more practical considerations, e.g. the computing cost for the realisation of the proposal or the size of the acceptance probability.

3.4 Coalescent time sampling

Arbores implements an additional MCMC step that proposes the movement only of the coalescence times. This time sampling is scheduled for execution after each complete execution of all bridge segment steps. The sampling is done via a standard Metropolis-Hastings step for each coalescence time separately in a manner that preserves the coalescence time ordering throughout the entire sequence. The sampling distribution used in this step is a truncated version the conventional exponential distribution for coalescence times (see e.g. the discussion following equation (4) in [9]). Truncation is needed to ensure the preservation of the coalescent time ordering.

4 Enabling heuristics

The above 'idealised' algorithm outlined can be too computationally expensive to be implemented in practice. This is mainly due to the worst case exponential size of the set of topology paths in the number of sites within the bridge segments. Therefore, we need to introduce heuristics, some of them approximative, to deliver a practical algorithm. The main principle underlying the choice of heuristics is the one of *minimum-recombination*. That is, we allow the data to enforce recombinations when required, and we avoid placing recombinations at positions not supported as such by the data. At the same time we require the algorithm to traverse the space of different recombinations, at different sites, so we perform another heuristic to also allow for this. One can think of our algorithm as one that follows principles from deterministic minimum-recombination algorithms used in a separate stream of the subject literature to reduce computing costs, while still being a proper (if approximate) posterior sampling MCMC algorithm. Due to this minimum recombination approach, we expect the approximations to improve as the ratio of mutation rate over recombination rate per site increases. We also wish to acknowledge that, due to some of the heuristics, our approximation is not guaranteed to be reversible. However, problems due to the irreversibility are expected to be rare and hence this is considered an acceptable approximation. The heuristics are described below.

4.1 Parsimonious SPR operation

The first heuristic aims to reduce the cardinality of the set of proposed tree paths, by switching from an exhaustive tree scan to one that adopts a parsimonious approach as regards to the number of SPR operations, taking under consideration the information available in the data.

For non-segregating sites any tree is compatible, so in the generation of the topology paths we omit non-segregating sites and perform SPR operations only between consecutive segregating sites – if necessary, i.e. if none of the currently generated topologies is compatible with the next segregating site. This leads to a substantial reduction in the computational cost. Some further considerations are needed here, as it may well be the case that more than one SPR operation is needed between two segregating sites to generate trees compatible to the data. Thus, we iterate – if necessary – over the number of SPR operations. In practice, Arbores attempts to construct the topology paths with none, one, or (at most) two SPR operations between each pair of consecutive segregating sites. Topologies available at the current step that require more SPR operations than others are removed (their path is deleted from the set of currently constructed paths). In the numerical applications we have looked at, requiring more that two SPR operations between consecutive segregating sites was a rare event; in the cases that this does occur, Arbores skips the update on that segment. Even if a segment update is skipped at an iteration of the MCMC sampler, it is likely that due to the updates of the other segments, the skipped segment can be updated the next time it is processed.

4.2 Extra recombinations

The parsimonious rules for allowing SPR operations described above come with a drawback: for a given bridge, by adopting, effectively, a minimal number of recombinations principle, one can deduce that all generated topology paths will have the same number of recombination sites along the bridge and, in fact, the same number of recombinations between all pairs of segregating sites contained in the bridge. To overcome this rigidity, we allow in a controlled way the topology paths to consist of more than the minimum number of SPR operations. This is done by first performing the tree scan as described in part (a), resulting in a set of tree topology paths. After this we repeat the tree scan in a sightly modified way. For the first pair of consecutive segregating sites that does not require an SPR operation, we nevertheless introduce one. For the remaining pairs of segregating sites the tree scan proceeds normally by introducing SPR operations only if needed. The resulting topology paths are added to the set of previously generated paths. We repeat this modified tree scan step for each remaining pair of consecutive segregating sites where no SPR is required. In this way we allow only one extra SPR operation for one pair of consecutive segregating sites at a time, with the aim of keeping the combinatorics manageable. At the same time, these additional SPR operations are enough to allow mobility in the number of recombinations and their positions.

4.3 Subtree search

The computationally most expensive part of the algorithm, even after implementing the heuristics described above, is the iteration over the possible SPR operations in the construction of the topology paths. In a naive implementation, one would simply apply all possible sequences of SPR operations and then check for each individual outcome whether the resulting tree is compatible with the data or not. Some computation can be saved by observing that certain operations are known in advance to produce an incompatible tree. We identify the operations that may produce a compatible tree as follows. Consider a tree at a segregating site i. We assign colours (black or white) to its leaf nodes, by specifying that the leaf n is black if the nth observation at the *next* segregating site, say j, equals 1 and white otherwise. Note, that we are considering the data at the next segregating site $j > i$ because the aim is to characterise the operations that produce a tree compatible with the data at site j. The colouring is extended to all nodes by recursively defining the colour of a node to be equal to the common colour of its children, if such colour exists, and grey otherwise, as demonstrated in Figure 6. Note also that the role of black and white nodes is not interchangeable, because for a tree to be compatible with the data, it must contain exactly one subtree whose all nodes are black and which is not a subtree of another subtree with black nodes only. Such condition is not imposed on the white nodes. We then have to consider only two types of SPR operation nodes (u_i, v_i):

(i) u_i is black and $\mathrm{pa}(u_i)$ is not black and v_i is black,

(ii) u_i is white and $\mathrm{pa}(u_i)$ is not white and v_i is white, gray or black and $\mathrm{pa}(v_i)$ is not black.

It is not guaranteed that these operations yield compatible trees, but in the case of single SPR operation between segregating sites, it can be proven that the set of operations yielding a compatible tree is a subset of such operations. The proof is included in the Appendix, Theorem 1 in Section A. This implies that this heuristic does not introduce any additional approximations. Indeed, one can see in Figure 6, showing all SPR operations yielding a compatible tree, that each of these operations are either of type (i) or (ii) defined above. For more than one SPR operation between segregating sites, this is not true and the reduction of the SPR operation search set to operations of type (i) or (ii) will result in an approximation.

4.4 Output

We note that – through the heuristics – the algorithm aims to provide a principled approximation to the idealised MCMC algorithm defined in the previous sections. Ul-

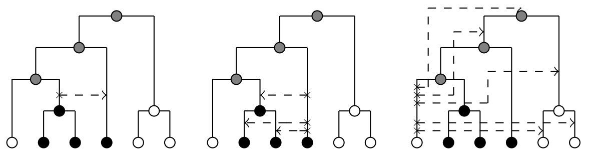


Figure 6: All SPR operations that yield a compatible tree. Each operation is of kind (i) or (ii).

timately, the proposal will provide samples from the support of the idealised posterior: e.g. when the proposal, using the heuristics, has determined a recombination between two successive segregating sites, the non-segregating sites are also taken under consideration and the recombination site is selected in some manner (e.g. at random) amongst the intermediate non-segregating sites and the right-side segregating one, yielding a topology path over all sites.

5 Numerical experiments

We carried out two numerical experiments with the proposed algorithm. In the first experiment, we ran the algorithm on the well-known Kreitman data [12] and in the second experiment we carried out a comparison with the recently proposed ARGweaver algorithm [17], sometimes perceived as the state-of-art method for this problem.

5.1 Kreitman data

The Kreitman data were first preprocessed by removing duplicate rows and columns with minor allele count less than two. The resulting data consisted of DNA polymorphisms of 9 sequences across 2,287 sites of which 30 were segregating. We set the mutation parameter to $\theta = 0.013$ corresponding to approximately 30 mutations on average for the data of the given size. The recombination rate was set to $\rho = 0.0035$ corresponding to approximately 8 recombinations in average. The minimum number of recombinations for these data is known to be 7 (see, e.g. [7, page 144]).

The chain was run for 2×10^5 iterations. Iteration here means either the processing of a single segment or sampling a single coalescent time. This number of iterations amounts to all segments having been sampled approximately 6,500 times, the exact number slightly varying according to the number of recombinations at a given state of the chain (see the discussion at the end of Section 44.1). To confirm the consistency of the results, the algorithm was run four times independently.

Figure 7a shows the trace plots of unnormalised log posterior densities accompanied with the trace plots for the number of recombinations. Each of the chains appears to spend most of the time sampling ARGs with the minimum number of recombinations, but occasional visits to up to 10 recombinations can be seen.

Figure 7b shows the maximum a posteriori (MAP) tree sequences for the four runs. The coalescent times of these MAP trees are the sample averages over the sequences with matching tree structure. Each of these sequences corresponds to an ARG. One can see that although the MAP ARGs are similar, they are not identical. This leads us to believe that in the posterior, there are ARGs with slightly different structure but approximately same posterior probability, hence the MAP ARGs in different runs do

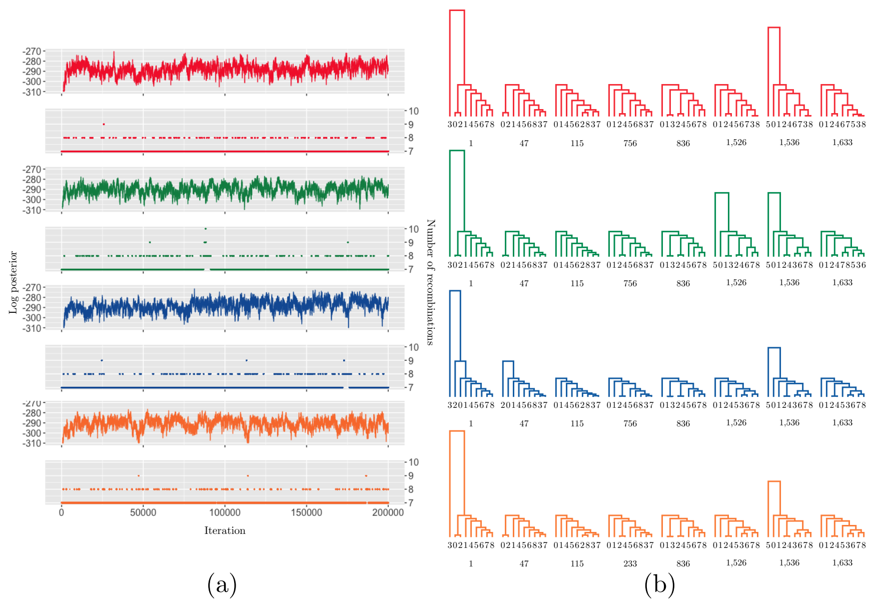


Figure 7: (a) The unnormalised log posterior and the number of recombinations trace plots for four independent runs. (b) The maximum a posteriori tree sequences of four independent runs. Coalescence times are calculated as the averages of coalescence times over all tree sequences with matching structure. The site of the first occurrence of a given tree is shown below each tree.

not have to be precisely the same (sampling variation might also have an effect). Each of the MAP ARGs is displaying 7 recombinations which is consistent with the number of recombinations trace plots in Figure 7.

The algorithm was implemented in C and is available at `https://github.com/heinekmp/Arbores`. The running time of the algorithm is random. For the numerical experiments reported here, the running time was approximately 10 hours on an off-the-shelf MacBook Pro (2.9 GHz Intel Core i7).

5.2 Comparison with ARGweaver

In our second experiment, we run both Arbores and ARGweaver on the same simulated data. The data were first generated with MaCS software of [3] after which it was preprocessed by removing sites with minor allele count less than two; the data ware also shifted so that first segregating site was given index one. After preprocessing, the data consisted of eight sequences across 10,000 sites of which 24 were segregating. The datasets and the exact calls of the algorithms are available at `https://github.com/heinekmp/Arbores/tree/master/test_runs`.

We compare the outputs of the algorithms against each other rather than against the true ARG that was used for generating the data. This is done because with Bayesian MCMC methods the actual target is the posterior distribution which may or may not be an accurate representation of the generating ARG.

Figure 8 shows the trace plots for the number of recombinations for both algorithms. The initialisation of Arbores aims at starting with the minimum recombination

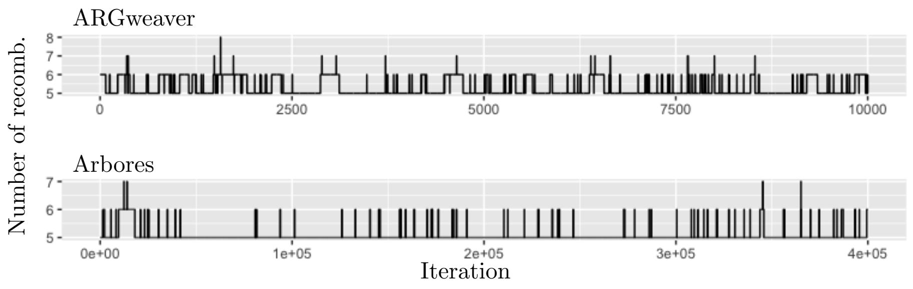


Figure 8: Trace plots of the number of recombinations for ARGweaver and Arbores. For Arbores the horizontal axis shows all iterations which amounts to all segments being sampled roughly 15,000 times.

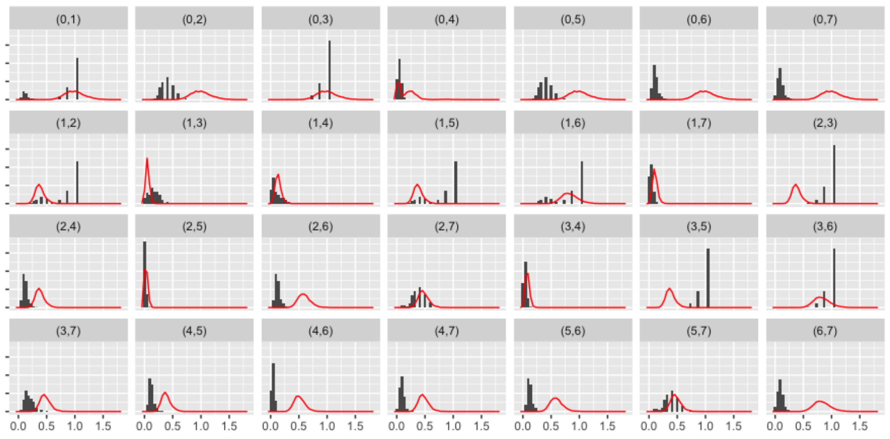


Figure 9: Histograms of TMRCAs for Arbores (red) and ARGweaver (black). See the text for the definition of TMRCA. The title on each histogram indicates the pair of observed sequences indexed by $\{0, \ldots, 7\}$.

ARG but this is not guaranteed. The results nevertheless seem to suggest that the minimal number or recombinations for this dataset is 5. In some simulations ARGweaver sampled ARGs with fewer than 5 recombinations but in all such cases the sampled ARG was not compatible with the data. It is worth pointing out that Arbores never returns an ARG that is not compatible with the data. From Figure 8 we see that ARGweaver mixes somewhat better between different numbers of recombinations although the trace plots are similar. This may be due to genuinely different mixing properties, or that the algorithms target different posteriors.

In order to compare the resulting approximate posterior structure of the ARG, Figure 9 shows the histograms of the times to most recent common ancestor (TMRCA) for both algorithms. Note that to better capture the structure of the ARG we use a slightly non-standard definition of TMRCA. TMRCA was calculated for each pair of observed sequences (indexed by $\{0, \ldots, 7\}$) by calculating the minimum time to the most recent common ancestor at each site and then by taking the minimum over all sites. TMRCAs were also scaled so that the greatest mean over all histograms for both algorithms is equal to one.

Our first observation from the results reported in Figure 9 is the impact of the time-discretisation adopted within ARGweaver. Time-discretisation is not required for the algorithm suggested in this paper, whereas for ARGweaver it is a necessity as the algorithm develops on a finite state-space for ARGs. For both algorithms the histograms are calculated similarly with 40 equally spaced bins, but only for Arbores the histograms appear to represent the densities of continuous distributions as desired. Particularly for the pairs with large mean, e.g. (3,6), ARGweaver returns truncated one tailed distributions which is not an accurate representation of the reality. Adjustment of the parameterisation, e.g. the maximum time or higher resolution of the time discretisation, are an obvious remedy to this issue, but due to the logarithmic scale of the time discretisation in ARGweaver, a large number of discrete times would be required to allow high resolution at large times, which will slow down the algorithm: doubling the number of time-discretisation points from 40 to 80, the computation time of 10,000 samples would increase from 1.5 hours to around 7.5 hours. Arbores took approximately 10 hours to generate 15,000 samples. The computing equipment used was as described in Section 55.1.

We also see that for some pairs of sequences, both algorithms identify a very recent common ancestor, see the pairs (1,3), (1,4), (1,7), (2,5) and (3,4). For some pairs both algorithms also agree on more distant common ancestor, see the pairs (0,1), (0,3), (1,6) and (3,6). Other histograms suggest that one of the algorithms was not able to explore all the modes, see e.g. pairs (0,4), (1,5) and (1,6). Further discrepancies between the histograms can be explained by the fact that ARGweaver had the additional degree of freedom to decide which of the two alleles appearing at a given segregating site was the mutated one while for Arbores the data determines the mutated alleles; data entries equal to one are mutated. While such flexibility may sometimes be desired, it also increases additional variation to the results in cases where the mutated alleles are known.

6 Concluding remarks

We have proposed an MCMC algorithm for simulating ARGs from the Bayesian posterior distribution given observed DNA polymorphism data. The algorithm is based on a novel bridging procedure which enables us to reduce the high dimensional problem into a set of substantially smaller scale problems. The main benefit of the algorithm is its suitability for parallel computing systems. This is due to the fact that to some extent the bridge segments can be processed in parallel independently of each other.

Further research is still needed to improve the scalability of the algorithm in the number of observed sequences. In particular, we believe that substantial improvements can be made by replacing the tree scanning step with more sophisticated methods that reduces the number of discarded paths in the time adjustment step, and thus avoid redundant computations. A potential approach is a node scoring approach whereby nodes are assigned a score between 0 and 1 according to the data at its leaf descendants. Score 1 (resp. 0) would correspond to all leaf descendants assuming value 1 (resp. 0). These scores might be indicative of the compatibility of the tree with the data and could be used for steering the tree paths in a manner which reduces the number of discarded paths. Other aspects of the algorithm warrant further investigation, e.g. non-uniform distributions for the choice of the proposed topology path could be tried. In general, Arbores provides a conceptually simple approach for sampling ARGs and as such, further potential improvements are seemingly easy to incorporate into its algorithmic framework.

Our comparison with ARGweaver shows that although its flexible parameterisation allows it to be used for more realistic problems than Arbores, the time-discretisation is a limitation which may be manifested already with modestly sized datasets. While a more thorough experimentation with ARGweaver might have lead to a parameterisation that efficiently mitigates the impact of time-discretisation, it can be argued that methods based on modelling continuous phenomena by discretisation, can reach the same accuracy as methods based on continuous models, such as Arbores, only asymptotically.

We have used SMC as an the Markovian approximation to ARG; our methodology can be applied without modification with the more accurate [25] SMC' approximation [14].

Ethics: The project did not involve research on humans or animals.

Data: The Kreitman data set is reported in [12] and can also be found in [7, page 144]. Other datasets and the exact calls of the algorithms are available at `https://github.com/heinekmp/Arbores/tree/master/test_runs`

Contributions: M.D.I, A.B., A.J., D.B. planned and designed the project, M.D.I, A.B., A.J., D.B., and K.H. carried out the research and wrote the manuscript. K.H. implemented the algorithm and carried out the numerical experiments.

Competing interests: We have no competing interests.

Funding: This work was supported by the EPSRC grant "Advanced Stochastic Computation for Inference from Tree, Graph, and Network Models" (ref: EP/K01501X/1).

Acknowledgements: A.B. also acknowledges support from a Leverhulme Trust Prize.

References

[1] M. Arenas. The Importance and Application of the Ancestral Recombination Graph. *Frontiers in Genetics*, 4(206), 2013.

[2] Richard J Boys, Darren J Wilkinson, and Thomas BL Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18(2):125–135, 2008.

[3] G.K. Chen, P. Marjoram, and J.D. Wall. Fast and flexible simulation of DNA sequence data. *Genome Research*, 19:136–142, 2009.

[4] P. Fearnhead and P. Donnelly. Estimating recombination rates from population genetic data. *Genetics*, 159:1299–1318, 2001.

[5] R. C. Griffiths and P. Marjoram. Ancestral inference from samples of dna sequences with recombination. *Journal of Computational Biology*, 3(4):479–502, 1996.

[6] R. C. Griffiths and P. Marjoram. An ancestral recombination graph. In P. Donnelly and S. Tavare, editors, *Progress in Population Genetics and Human Evolution*, pages 257–270. Springer Verlag, 1997.

[7] D. Gusfield. *ReCombinatorics*. The MIT Press, 2014.

[8] J. Hein, M. H. Schierup, and C. Wiuf. *Gene Genealogies, Variation and Evolution*. Oxford University Press, 2005.

[9] R.R. Hudson. Gene genealogies and the coalescent process. *Oxford Surveys in Evolutionary Biology*, 7:1–44, 1991.

[10] J. F. C. Kingman. The coalescent. *Stochastic Processes and Their Applications*, 13:235–248, 1982.

[11] J. F. C. Kingman. On the genealogy of large populations. *Journal of Applied Probability*, 19:22–43s, 1982.

[12] M. Kreitman. Nucleotide polymorphism at the alcohol dehydrogenase locus of drosophila melanogaster. *Nature*, 304:412–417, 1983.

[13] M. K. Kuhner. Maximum likelihood estimation of recombination rates from population data. *Genetics*, 156(3):1393–1401, 2000.

[14] Paul Marjoram and Jeff D Wall. Fast "coalescent" simulation. *BMC genetics*, 7(1):16, 2006.

[15] G. A. T. McVean and N. J. Cardin. Approximating the coalescent with recombination. *Phil. Trans. R. Soc. B*, 360:1387–1393, 2005.

[16] R. Nielsen. Estimation of population parameters and recombination rates from single nucleotide polymorphisms. *Genetics*, 154(2):931–942, 2000.

[17] M. D. Rasmussen, M. J. Hubisz, I. Gronau, and A. Siepel. Genome-wide inference of ancestral recombination graphs. *PLOS Genetics*, 10, 2014.

[18] C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Science+Business Media Inc., 2nd edition, 2004.

[19] Gareth O Roberts and Osnat Stramer. On inference for partially observed nonlinear diffusion models using the metropolis–hastings algorithm. *Biometrika*, 88(3):603–621, 2001.

[20] Y. S. Song. On the combinatorics of rooted binary phylogenetic trees. *Annals of Combinatorics*, 7(3):365–379, 2003.

[21] Y. S. Song. Properties of subtree-prune-and-regraft operations on totally-ordered phylogenetic trees. *Annals of Combinatorics*, 10(1):147–163, 2006.

[22] Y. S. Song and J. Hein. Parsimonious reconstruction of sequence evolution and haplotype blocks: Finding the minimum number of recombination events. *Algorithms in Bioinformatics*, 2003.

[23] Y. S. Song and J. Hein. Constructing minimal ancestral recombination graphs. *Journal of Computational Biology*, 12:159–178, 2005.

[24] Y. S. Song, Y. W. Wu, and D. Gusfield. Efficient computation of close lower and upper bounds on the minimum number of recombinations in biological sequence evolution. *Bioinformatics*, 21(1):i413–i422, 2005.

[25] Peter R Wilton, Shai Carmi, and Asger Hobolth. The SMC' is a highly accurate approximation to the ancestral recombination graph. *Genetics*, pages genetics–114, 2015.

[26] Y.X. Zhang, K. Perry, V.A. Vinci, K. Powell, W.P. Stemmer, and S.B. del Cardayre. Genome shuffling leads to rapid phenotypic improvement in bacteria. *Nature*, 415, 2002.

Appendix A SPR operation search heuristic

In this section we prove the claim made earlier that the subtree search heuristic introduced in Section 4.3 does not introduce any error under the assumption of at most one recombination between consecutive segregating sites.

In addition to the black, grey, white colouring of the nodes described earlier, we introduce another classification of nodes depending on whether a node is the root of a maximal subtree of their respective colour or not. A subtree is said to be black (resp. white) if all its nodes are black (resp. white). A black (resp. white) subtree is said to be maximal if any strictly larger subtree containing it is not black (resp. white). A grey node cannot be a root of a subtree consisting only of grey nodes, so all grey nodes are classified as branch nodes by convention.

With this classification together with the node colouring we can construct equivalence classes of SPR operations which we denote by (x, y, z, w), where $x, z \in \{\mathrm{B}, \mathrm{W}, \mathrm{G}\}$ denoting the colours (black, white, grey) of the pruning and regrafting nodes respectively, and $y, w \in \{\mathrm{r}, \mathrm{b}\}$ denoting the classifications (root of a subtree, branch) of the pruning and regrafting node, respectively. Also we use $*$ as a wildcard to denote any possible value of a given entry. We have the following theorem:

Theorem 1. *Let the colours black, white and grey be assigned to the nodes of T_i as described above, and assume that T_i contains more than one maximal black subtree. The SPR operation that results in a tree containing at most one maximal black subtree (i.e. makes T_i compatible with the data), if such operation exists, belongs to* $(\mathrm{B}, \mathrm{r}, \mathrm{B}, *)$*,* $(\mathrm{W}, \mathrm{r}, \mathrm{W}, *)$*,* $(\mathrm{W}, \mathrm{r}, \mathrm{G}, *)$ *or* $(\mathrm{W}, \mathrm{r}, \mathrm{B}, \mathrm{r})$.

Proof. All possible SPR operations can be expressed as a set of equivalence classes:

$$(\mathrm{B}, \mathrm{r}, \mathrm{B}, *),\ (\mathrm{B}, \mathrm{r}, \mathrm{W}, *),\ (\mathrm{B}, \mathrm{r}, \mathrm{G}, *),\ (\mathrm{B}, \mathrm{b}, *, *),\ (\mathrm{W}, \mathrm{b}, *, *),$$
$$(\mathrm{W}, \mathrm{r}, \mathrm{W}, *),\ (\mathrm{W}, \mathrm{r}, \mathrm{G}, *),\ (\mathrm{W}, \mathrm{r}, \mathrm{B}, \mathrm{r}),\ (\mathrm{W}, \mathrm{r}, \mathrm{B}, \mathrm{b}),\ (\mathrm{G}, *, *, *).$$

Each of the Lemmata 1 – 5 below, shows that the SPR operations within a specific equivalence class cannot yield a compatible tree. This leaves us with the operations mentioned in the statement of the theorem. □

Lemma 1. *Operations in* $(\mathrm{B}, \mathrm{r}, \mathrm{W}, *)$ *or* $(\mathrm{B}, \mathrm{r}, \mathrm{G}, *)$ *do not reduce the number of maximal black subtrees.*

Proof. The classification of a node as a subtree root can only change if either 1° the colours of its descendants change or 2° the colour of its sibling changes. Suppose we have two black subtree root nodes, b_1 and b_2. Let us first consider how the pruning of b_1 affects the subtree root status of b_2.

The pruning of any node u can only affect the colours of the ancestors of u. Due to being a black subtree root, b_1 cannot be a descendant of b_2 and therefore pruning

b_1 cannot affect the colours of the descendants of b_2. Therefore the subtree root classification of b_2 can only change if the pruning of b_1 turns the sibling of b_2 black. By considering all possible scenarios we see that after pruning b_1 the ancestors of b_1 either remain grey or turn white and hence the sibling of b_2 cannot turn black due to the pruning of b_1. In conclusion, pruning b_1 will not change the status of b_2 as a black subtree root.

Let us then consider the effects of regrafting the subtree rooted at b_1. Regrafting can only affect the colours of the ancestors of the regrafting node. Since b_1 is regrafted either to a white or a grey node, it cannot be regrafted to a descendant of b_2 and therefore the status of b_2 can only change if the sibling of b_2 is an ancestor of the regrafting node and if it is black after regrafting. Regrafting a black node to a white or a grey node results in all the ancestors of the regrafting node being grey, so the status of b_2 as a black root is unchanged. Moreover, b_1 remains as a black subtree root, so the resulting tree has exactly the same number of maximal black subtrees as the original tree. □

Lemma 2. *Operations in* $(\mathrm{B}, \mathrm{b}, \mathrm{W}, *)$, $(\mathrm{B}, \mathrm{b}, \mathrm{B}, *)$ *or* $(\mathrm{B}, \mathrm{b}, \mathrm{G}, *)$ *when applied to a tree with more than one maximal black subtree cannot produce a compatible tree.*

Proof. We only need to consider regrafting, because after the pruning, the number of maximal black subtrees remains unchanged. From the proof of Lemma 1 we know that regrafting a black node to a white or a grey node cannot change the status of the existing black subtree roots but it introduces a new maximal black subtree. So the resulting tree cannot be compatible.

Regrafting a black node to a black branch node means inserting a black subtree into a black subtree whose root node remains unchanged. Regrafting a black node to a black subtree root, say b, implies that the new node introduced by the regraft operation becomes a new black subtree root and the classification of b changes from subtree root to branch. In any case, the number of black subtree root nodes remains unchanged. □

Lemma 3. *Operations in* $(\mathrm{W}, \mathrm{r}, \mathrm{B}, \mathrm{b})$ *cannot produce a compatible tree.*

Proof. After the regrafting, the black branch node, say b, to which the white node was regrafted becomes a black subtree root. Also the sibling of b becomes a black subtree root. Hence the resulting tree has at least two maximal black subtrees and cannot be compatible. □

Lemma 4. *Operations in* $(\mathrm{W}, \mathrm{b}, \mathrm{B}, *)$, $(\mathrm{W}, \mathrm{b}, \mathrm{W}, *)$ *or* $(\mathrm{W}, \mathrm{b}, \mathrm{G}, *)$ *when applied to a tree with more than one maximal black subtrees cannot produce a compatible tree.*

Proof. Pruning a white branch node does not affect the number of maximal black subtrees so we only need to consider regrafting. As in Lemma 3, regrafting to a black branch node will cause the regrafting node and its sibling to become black subtree roots and thus the tree will not be compatible.

When regrafting to a black subtree root, the black subtree root remains unchanged and all its ancestors become grey. Hence the classification of any black subtree root remains unchanged as a change would require its sibling to turn black which cannot be the case.

Regrafting a white node to a white branch node will have no consequences outside the white subtree containing the regrafting node. Regrafting to a white subtree root causes the new node introduced by the regraft operation to become a new white subtree root instead of the regrafting node, but the rest of the tree will remain unchanged.

When regrafting to a grey node, only the ancestors of the regrafting node will be affected, but because the ancestors of a grey node are grey, they will remain unchanged. □

Lemma 5. *Operations in* $(\mathrm{G}, *, *, *)$ *when applied to a tree with more than one maximal black subtree cannot produce a compatible tree.*

Proof. Regrafting a grey node to a black node (subtree root or a branch) results in a tree containing at least two maximal black subtrees: one rooted at the regrafting node and another one must be contained by definition in the subtree rooted at the pruned grey node.

Regrafting to a white or a grey node will cause all the ancestors of the regrafting node to become grey. This means that the classification of each black subtree root in the tree must remain unchanged, but since the subtree being regrafted contains, by definition, at least one maximal black subtree, the resulting tree must have at least one more maximal black subtree than the tree after pruning. If the tree after pruning contained at least one maximal black subtree, then the resulting tree would be incompatible and if the tree after pruning did not contain any maximal black subtrees, then the pruned subtree must contain at least two maximal black subtrees, since the original tree was assumed to have at least two maximal black subtrees. In any case, the resulting tree will be incompatible. □