Speeding up Inference of Homologous Recombination in Bacteria

Felipe J Medina-Aguayo[1], Xavier Didelot[2,3], and Richard G Everitt[3]

[1]Centro de Investigación en Matemáticas (CIMAT), Mexico
[2]School of Life Sciences, University of Warwick, United Kingdom
[3]Department of Statistics, University of Warwick, United Kingdom

Abstract

Bacteria reproduce clonally but most species recombine frequently, so that the ancestral process is best captured using an ancestral recombination graph. This graph model is often too complex to be used in an inferential setup, but it can be approximated for example by the ClonalOrigin model. Inference in the ClonalOrigin model is performed via a Reversible-Jump Markov Chain Monte Carlo algorithm, which attempts to jointly explore: the recombination rate, the number of recombination events, the departure and arrival points on the clonal genealogy for each recombination event, and the range of genomic sites affected by each recombination event. However, the Reversible-Jump algorithm usually performs poorly due to the complexity of the target distribution since it needs to explore spaces of different dimensions. Recent developments in Bayesian computation methodology have provided ways to improve existing methods and code, but are not well-known outside the statistics community. We show how exploiting one of these new computational methods can lead to faster inference under the ClonalOrigin model.

Introduction

Recombination is a critical process in evolution, particularly when analysing within-species variation. Bacteria reproduce clonally, but recombination exists in most species, where a donor cell contributes a small segment of its DNA to a recipient cell. This process is analogous to gene conversion in eukaryotes and is typically modelled using an Ancestral Recombination Graph (ARG) model (Hudson, 1990; Griffiths, 1996; Wiuf and Hein, 2000). The simulation of genomic data under this model is relatively easy (Didelot et al., 2009; Brown et al., 2016) but inference of the ancestral process given genomic data is much harder. This inference problem is important even when recombination is not directly of interest, because ignoring recombination leads to inaccurate reconstruction of the clonal part of the ancestry (Schierup and Hein, 2000; Hedge and Wilson, 2014).

The ClonalOrigin model (Didelot et al., 2010) can be regarded as a good approximation of the aforementioned ARG process, in which recombination events are modelled independently given the clonal genealogy, denoted throughout as $\mathcal{T}$. For completeness, we note the existence of related approximations to the ARG such as the sequential Markov coalescent for eukaryotes (McVean and Cardin, 2005; Marjoram and Wall, 2006) and the bacterial sequential Markov coalescent (De Maio and Wilson, 2017). There are also simpler approximations such as the ClonalFrame model (Didelot and Falush, 2007; Didelot and Wilson, 2015) in which the origin of each recombination event is not modelled.

The major challenge when implementing the ClonalOrigin model is to efficiently explore the joint posterior distribution of the recombination rate (ρ), the number of recombination events (R), their departures ($a_1, ..., a_R$) and arrivals (b_1,...,b_R) on the clonal genealogy and the sites delimiting the start ($x_1, ..., x_R$) and end ($y_1, ..., y_R$) points of each recombination event on the genome. In order to explore such a complex distribution using Markov Chain Monte Carlo (MCMC), one must resort to the Reversible-Jump MCMC

(RJMCMC) algorithm (Green, 1995) which in plain words aims at visiting spaces of different dimensions. This is the approach that was taken previously in both the original standalone implementation of the ClonalOrigin model (Didelot et al., 2010) and a recent reimplementation (Vaughan et al., 2017) within the BEAST2 framework (Bouckaert et al., 2019).

Unfortunately, as known by computational statisticians, the RJMCMC algorithm usually performs poorly due to the difficulty of proposing "good" trans-dimensional jumps. Because of this, the number of iterations (and consequently the running time) required by the algorithm for obtaining a reasonable approximation of the posterior distribution may be impractically large. Recent developments in Bayesian computation methodology provide ways of improving existing methods and code, but are not well-known outside the statistics community.

Such state-of-the-art methods belong to the framework of Sequential Monte Carlo (SMC) methods (see e.g. Doucet et al., 2001; Del Moral et al., 2006) where inference is performed using an appropriate sequence of intermediate target distributions leading to the desired one. Some of these methods have been successfully implemented to some extent when inferring coalescent trees as data arrives (see e.g. Dinh et al., 2018; Everitt et al., 2019). By performing inference in a sequential manner the complexity of the problem is reduced and the the desired target distribution may be explored more efficiently. The ideas presented here are based on the aforementioned SMC methodology that can lead to faster inference in the ClonalOrigin model; nonetheless, these ideas could be in principle applied to other evolutionary models.

Methods

Bayesian inference under the ClonalOrigin model

In this section we briefly describe the elements of the ClonalOrigin model, namely the parameters of interest, prior distributions on these parameters and the likelihood function for the data. For more details on assumptions and derivations of formulae please refer to cited references and the supplementary material.

- We use a coalescent tree to represent the clonal genealogy of n samples ($\mathcal{D} = D_{1:n}$) and denote such a tree by $\mathcal{T} = (\tau, \mathbf{t})$, which is composed of a topology τ and a vector $\mathbf{t} = (t_2, \ldots, t_n)$ of branch lengths. A Kingman's coalescent prior (Kingman, 1982) is assumed for $\mathcal{T}$.
- Let R denote the number of recombination events affecting the DNA sequences, which we assume a priori to be distributed according to a Poisson random variable with mean $\rho\mathcal{L}/2$, where ρ denotes the global recombination rate and $\mathcal{L} = \sum_{i=2}^{n} it_i$ is the total branch length of the tree $\mathcal{T}$, see e.g. Wiuf and Hein (1999) for more details. We also let $\mathcal{L}(s,t)$ denote the sum of branch lengths from time s to time t on the tree.
- Each recombination event is formed by the following variables: departure and arrival points on the genealogy, and start and end sites on the genome. These four variables, when referring to the i-th recombination edge, are denoted by a_i, b_i, x_i and y_i, respectively.
- Both variables a_i and b_i are fully determined by a time and lineage on the tree (denoted respectively by $a_i[t]$ and $a_i[l]$, respectively), and the chosen priors are those from the construction of an ARG as a point process along the genomes (Wiuf and Hein, 1999).
- The priors for x_i and y_i are constructed assuming a uniform distribution on the sequence for x_i and a geometric distribution of mean $\delta > 0$ for the difference $y_i - x_i \mid x_i$. When the sequence is made of B blocks comprising a total length of L the priors need to be modified accordingly as in Didelot et al. (2010).

- The entire set of variables describing the recombination events is denoted by

$$\mathcal{R} = (R, a_{1:R}, b_{1:R}, x_{1:R}, y_{1:R}) .$$

- Mutation events occur at rate $\theta/2$ across the genealogy and on existing recombination edges; for simplicity we assume that all substitutions are equally likely, as in the evolutionary model JC69 (Jukes and Cantor, 1969).

Under the above the assumptions, the full prior for the set of parameters of interest is given by (see the supplementary material for the derivation)

$$\begin{aligned} p_0(\rho, \delta, \theta, \mathcal{T}, \mathcal{R}) &= p_0(\mathcal{R} \mid \rho, \delta, \mathcal{T})\, p_0(\mathcal{T})\, p_{0,\rho}(\rho)\, p_{0,\delta}(\delta)\, p_{0,\theta}(\theta) \\ &= \left[\exp\left\{ -\frac{\rho T}{2} \right\} \left(\frac{\rho}{2}\right)^R \prod_{i=1}^{R} \exp\left\{ -\mathcal{L}\left(b_i[t], a_i[t]\right) \right\} p_{0,x}(x_i|\delta)\, p_{0,y}(y_i|x_i, \delta) \right] \\ &\quad \times \exp\left\{ -\sum_{i=2}^{n} \binom{i}{2} t_i \right\} p_{0,\rho}(\rho)\, p_{0,\delta}(\delta)\, p_{0,\theta}(\theta) , \end{aligned}$$

where $p_{0,\rho}, p_{0,\delta}, p_{0,\theta}$ are arbitrary priors for the recombination rate, mean-tract length and mutation rate. For the likelihood computation, first recall that mutation events occur at rate $\theta/2$ across the genealogy and on existing recombination edges; we then assume for simplicity that all substitutions are equally likely (Jukes and Cantor, 1969). The computation of the likelihood function $\mathfrak{L}(\mathcal{T}, \mathcal{R}, \theta; \mathcal{D})$, for any tree $\mathcal{T} = (\tau, \mathbf{t})$, set of recombination events $\mathcal{R}$, and mutation rate θ, is done using Felsenstein's pruning algorithm (Felsenstein, 1973, 1981) (see also the supplementary material for more details).

The posterior on the full set of parameters is obtained through Bayes' Theorem

$$\pi(\rho, \delta, \theta, \mathcal{T}, \mathcal{R} \mid \mathcal{D}) \propto p_0(\rho, \delta, \theta, \mathcal{T}, \mathcal{R})\, \mathfrak{L}(\mathcal{T}, \mathcal{R}, \theta; \mathcal{D}) . \tag{1}$$

However, inferring the whole set of parameters represents a big challenge. One could instead fix one or more parameters and work with the resulting conditional distributions, for example finding a point estimate of $\mathcal{T}$ and then fix it to infer the rest of the parameters. We will consider this incomplete approach for some of the examples presented later where we aim to explore

$$\pi(\mathcal{R} \mid \rho, \delta, \theta, \mathcal{T}, \mathcal{D}) \propto \tilde{\pi}(\mathcal{R}) = p_0(\mathcal{R} \mid \rho, \delta, \mathcal{T})\, \mathfrak{L}(\mathcal{T}, \mathcal{R}, \theta; \mathcal{D}) . \tag{2}$$

The algorithm

Markov chain Monte Carlo (MCMC) is commonly the method of choice for exploring posterior distributions. If we had access to the the marginal posterior for the number of recombination events $\pi(R \mid \rho, \delta, \theta, \mathcal{T}, \mathcal{D})$, the inference of the remaining parameters could be carried out independently across different values of R. Since such a marginal is not available we need to infer R jointly with the rest of the parameters leading to a posterior that has no fixed dimension. The celebrated Reversible-Jump MCMC (RJMCMC) algorithm (Green, 1995) provides an elegant solution and corresponds to the generalisation of the Metropolis-Hastings (MH) algorithm that allows "jumps" across different dimensions. In our context, these jumps will correspond only to going up or down by one dimension, i.e. given $R \geq 1$ recombination events they can go up to $R + 1$ or down to $R - 1$. In addition to these trans-dimensional moves, we still need to perform intra-dimensional moves that allow the full exploration of the desired posterior. Therefore, for fixed R we could also perform moves on ρ, δ, θ, $\mathcal{T}$ and $\mathcal{R} \mid R = \{a_{1:R}, b_{1:R}, x_{1:R}, y_{1:R}\}$.

Unfortunately, RJMCMC typically suffers from bad mixing in the sense that the resulting chain converges slowly to the desired posterior; this will in turn require a prohibitively large number of iterations for obtaining accurate answers. Due to recent developments in methodology (Karagiannis and Andrieu, 2013; Andrieu et al., 2018), the acceptance ratio for trans-dimensional moves within the RJMCMC can be understood as an importance sampling estimate of the ratio of two marginal densities, i.e. for an upwards move the ratio corresponds to $\pi(R + 1 \mid \mathcal{D}) / \pi(R \mid \mathcal{D})$, and using a single importance point (more

details on this can be found in the supplementary material). One could then ask whether improving the aforementioned estimate could result in a chain with better mixing. The answer turns out to be positive but with some restrictions. Two straightforward approaches to achieve variance reduction of the estimator are annealed importance sampling (Neal, 2001) and simply using more than one importance point.

In plain words, the annealing procedure creates a smooth bridge between distributions of different dimension; so in a sense instead of attempting one big jump from R to $R+1$ we attempt many, say $T > 1$, smaller jumps. Clearly, the downside of this approach is the extra cost of performing T steps that could result in a rejection in the MCMC algorithm. Whether annealing provides an advantage or not will entirely depend on the quality of the smaller jumps. In the ClonalOrigin context, when proposing to add a new recombination event that is in a bad region of the posterior, the idea is to use the smaller jumps to correct its position before deciding to accept or not.

On the other hand, considering $N \geq 2$ importance points when jumping from R to $R+1$ requires the creation of several proposed recombination events which are used for estimating the aforementioned ratio of marginals. However, if the upwards ends up being accepted one needs to select which of the proposed recombination events will be retained for the next iteration of the algorithm. To do this, a categorical distribution is used that selects which recombination event will survive according to its weight or contribution to the estimate of the acceptance ratio. Therefore, a recombination event that is more plausible than the rest will have a larger weight and, if the upwards move is accepted, it will have a greater chance of being retained ready for the next iteration.

Figure 1 illustrates both the annealing and the multiple importance points schemes when $T = 5$ and $N = 4$. The trees in black correspond to the current value of $\mathcal{T}$ on which recombination events are appended. Colours correspond to different stages of the proposed recombination event when perturbed; the idea of annealing is that the proposed event in blue at $t = 1$ will be perturbed in such way that the the one in red at $t = T$ has a better chance of being accepted. This process can be repeated N times, possibly in parallel, noting that perturbations could be drastic (as in $n = N$ in the figure) or almost nonexistent (as in $n = 3$); it all depends on the quality of the the proposed event at $t = 1$ and the perturbation mechanism. Only one event from those at $t = T$ is selected for the final accept-reject step in the MCMC, but events leading to a higher value in the posterior will have a better chance of being selected and possibly added later on, as discussed previously.

Algorithm 1 describes one step of the full process which we have termed Reversible-multiple-Jump MCMC (RmJMCMC) in line withB Andrieu et al. (2018), where it was firstly introduced and studied, and belonging to the wider class of MH with Asymmetric Acceptance Ratio (MHAAR) algorithms. Here we have adapted RmJMCMC to the ClonalOrigin context. We refer the reader to the appendix, where we describe the UAM and DAM algorithms used in the RmJMCMC. The implementation in `C++` of the RmJMCMC algorithm is freely available at: `https://github.com/fmedina7/ClonOr_cpp`

Notice that the upwards move agrees with the description in the previous paragraph; however, the downwards move appears to be more intricate. In words, we propose to delete a recombination event but in order to decide whether to accept or not we must test if this deletion is convenient. This is done by generating $N-1$ recombination events and computing the resulting acceptance ratios as if we were trying an upwards move. Low values for the aforementioned ratios would imply that the proposed deletion is possibly a good decision, whereas if many ratios result in high values it might indicate that the deletion is a poor choice. The final decision rule results in a valid algorithm as shown in Andrieu et al. (2018), otherwise the algorithm would not be exact in the sense that it does not target the desired posterior distribution. Non-exact or noisy methods have been explored in the past (see e.g. Alquier et al., 2016), however we do not discuss them any further as the bias that is introduced is typically difficult to quantify.

One further observation is the obvious increased computational cost of RmJMCMC as opposed to performing only RJMCMC (equivalent to RmJMCMC when $T = N = 1$) or running multiple RJMCMC chains in parallel. Every iteration of RmJMCMC is at least TN times more expensive than one RJMCMC iteration. However, as opposed to running RJMCMC for longer or multiple independent RJMCMC chains, RmJMCMC has provable better convergence properties (Andrieu et al., 2018) that can reduce burn-in times and improve convergence towards a region of high posterior probability. This is discussed in more

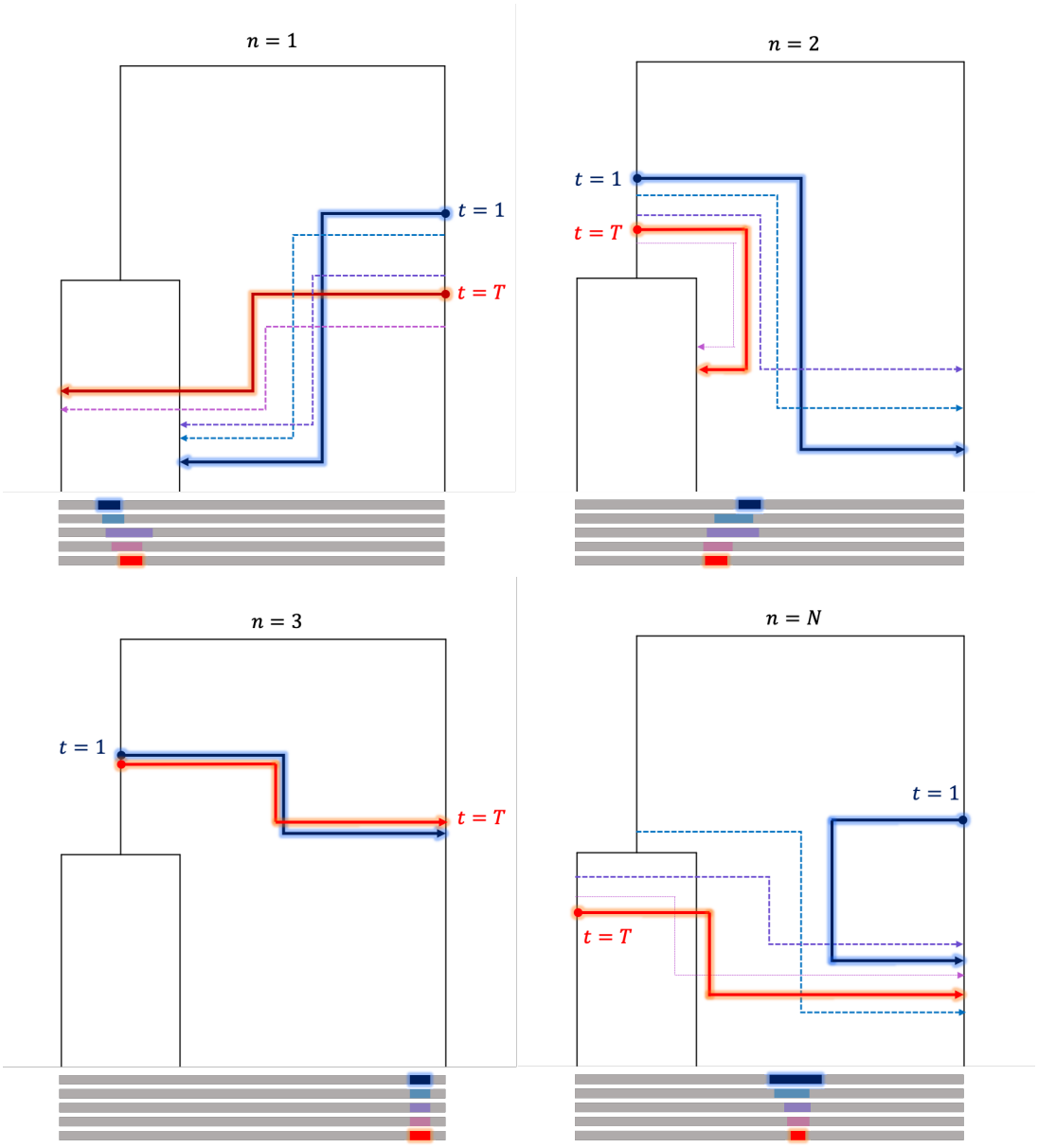

Figure 1: Illustration of annealing and multiple importance points for $T = 5$ and $N = 4$.

depth in the Results section where we look at some examples.

Flavours of RmJMCMC

In order to implement Algorithm 1, we must specify the sampling auxiliary distributions $\overrightarrow{\varphi}(\cdot \mid \mathcal{R})$ and $\overleftarrow{\varphi}(\cdot \mid \mathcal{R})$ (which could both depend on other parameters, e.g. $\rho, \delta, \theta, \mathcal{T}$). In the following section we present results using simple choices, the joint prior on (a, b, x, y) for the distribution $\overrightarrow{\varphi}$ and a discrete uniform on the set $\{1, \ldots, R\}$ for $\overleftarrow{\varphi}$ assuming there are R active recombination events. In mathematical terms, the associated densities are

$$\overrightarrow{\varphi}(a, b, x, y) = \exp\left\{-\mathcal{L}\left(b\left[t\right], a\left[t\right]\right)\right\} p_{0,x}\left(x|\delta\right) p_{0,y}\left(y|x, \delta\right),$$

and $\overrightarrow{\varphi}(j \mid R) = R^{-1}$ for $j \in \{1, \ldots, R\}$. The previous choices greatly simplify the computation of the output ratios in Algorithms 2 and 3, which involve mainly ratios of likelihood functions.

Algorithm 1 Reversible-multiple-Jump MCMC (RmJMCMC)

NOTATION: Let $E_r = (a_r, b_r, x_r, y_r)$ denote the r-th recombination event, and $E_{1:r} = (a_{1:r}, b_{1:r}, x_{1:r}, y_{1:r})$.
REQUIRES: Sequence $\{\gamma_t\}_{t=0}^{T}$ such that $0 = \gamma_0 < \gamma_1 < \cdots < \gamma_T = 1$.
INPUT: Current values for ρ, δ, θ, $\mathcal{T}$, and $\mathcal{R} = (R, E_{1:R})$ for $R \geq 0$.
OUTPUT: New value for $\mathcal{R}$.

- With probability $1/2$ try an upwards move:
  1. Generate N recombination events $\left\{E_*^{(n)} = \left(a_*^{(n)}, b_*^{(n)}, x_*^{(n)}, y_*^{(n)}\right)\right\}_{n=1}^{N}$ using the common distribution $\overrightarrow{\varphi}(\cdot \mid \mathcal{R})$.
  2. For each $n \in \{1, \ldots, N\}$ perform an UAM as in Algorithm 2 using as input $\left(\mathcal{R}, E_*^{(n)}\right)$, obtaining $r_T^{(n)}$ and the set of variables $\left(\bar{\mathcal{R}}^{(n)}, J_*^{(n)}\right)$.
  3. Compute $\bar{r}_T = N^{-1} \sum_{n=1}^{N} r_T^{(n)}$ and draw $U \sim Unif\,(0, 1)$.
  4. If $U \leq \bar{r}_T$
     - Draw $K \sim Mult\left(N, \left(r_T^{(1)}, \ldots, r_T^{(N)}\right)\right)$.
     - Return $\bar{\mathcal{R}}^{(K)}$.

     Otherwise return $\mathcal{R}$.
- Else if $\mathcal{R} \neq (0, \emptyset)$ rename $E_{1:R}$ and $\mathcal{R} = (R, E_{1:R})$ by $\bar{E}_{1:R}$ and $\bar{\mathcal{R}} = \left(R, \bar{E}_{1:R}\right)$, respectively. Try a downwards move:
  1. Sample an index $J_*^{(1)} \in \{1, \ldots, R\}$ from the distribution $\overleftarrow{\varphi}(\cdot \mid \mathcal{R})$, which indicates the recombination index to be deleted.
  2. Delete the recombination event with index $J_*^{(1)}$ from $\bar{E}_{1:R}$ to obtain $E_{1:R-1}$, denoting by $E_*^{(1)} = \bar{E}_{J_*^{(1)}}$.
  3. Perform a DAM as in Algorithm 3 using as input $\left(\bar{\mathcal{R}}, J_*^{(1)}\right)$, retaining only $r_T^{(1)}$ and discarding the other output variables.
  4. Define $\mathcal{R} = (R - 1, E_{1:R-1})$ and generate $N - 1$ recombination events $\left\{E_*^{(n)} = \left(a_*^{(n)}, b_*^{(n)}, x_*^{(n)}, y_*^{(n)}\right)\right\}_{n=2}^{N}$ using the common distribution $\overrightarrow{\varphi}(\cdot \mid \mathcal{R})$.
  5. For each $n \in \{2, \ldots, N\}$ perform an UAM as in Algorithm 2 using as input $\left(\mathcal{R}, E_*^{(n)}\right)$, retaining only $r_T^{(n)}$ and discarding the other output variables.
  6. Compute $\bar{r}_T = N^{-1}\left(\left(r_T^{(1)}\right)^{-1} + \sum_{n=2}^{N} r_T^{(n)}\right)$ and draw $U \sim Unif\,(0, 1)$.
  7. If $\bar{r}_T \leq 1/U$ return $\mathcal{R}$, otherwise return $\bar{\mathcal{R}}$.
- Otherwise return $\mathcal{R} = (0, \emptyset)$.

Additionally, we must define the way to perturb recombination events at every small step in the annealing process. This is done using an MCMC algorithm as explained in Algorithms 2 and 3; hence the perturbation is fully defined once we select a proposal distribution for $\mathcal{R}$ fixing the value of R, i.e. we need to perturb at least one recombination event within the existing R events. For the examples in the following section we choose to perturb (using the prior as proposal) only the newly created recombination event when going upwards, or the recombination event to be deleted when going downwards, for both cases this event is denoted by E_* in Algorithms 2 and 3. Doing this provides a very simple expression for the acceptance ratio in the MCMC steps within the annealing that involves only ratios of likelihood functions.

Finally, we must decide the number of annealing steps T, the sequence of real numbers $\{\gamma_t\}_{t=0}^{T}$ and the

value of replicates N. In the examples that follow we use different values for T and N which involve different costs and running times. As mentioned earlier, the computational cost increased as T and N increases, however an appealing property of RmJMCMC is that loops involving N (either upwards or downwards) can be performed in parallel, this may lead to higher efficiency when computing running times. Due to this, the value of N is commonly determined by the number of cores available in the computer or server. Respecting the required sequence $\{\gamma_t\}_{t=0}^{T}$ we simply choose a linear interpolation $\gamma_t = t/T$.

We want to emphasise that the choice of $\overrightarrow{\varphi}$ and $\overleftarrow{\varphi}$ were made in accordance to the original ClonalOrigin implementation from Didelot et al. (2010); whereas the choices for the proposals within the MCMC and the elements of the sequence $\{\gamma_t\}_{t=0}^{T}$ were made for convenience. However, the way RmJMCMC was formulated permits the use of more complex approaches that could provide better results in terms of efficiency. Some of these improvements are briefly described in the Discussion and are devoted to future work.

Results

This section compares the RmJMCMC algorithm and the standard RJMCMC for the ClonalOrigin model. We start with some toy simulated experiments, moving later on to results using real, and previously studied, datasets. The code used for generating the forthcoming results is available at: https://github.com/fmedina7/ClonOr_cpp

Application to simulated data

We present several simulated examples in order to explore the scalability and usability of RmJMCMC. We compare the settings when $N = T = 1$ (corresponding to the RJMCMC algorithm), when $T = 4$ and $N = 4$, when $T = 2$ and $N = 8$, and when $T = 1$ and $N = 16$. The efficiency between these configurations is compared using an estimation of the effective sample size (ESS) for a quantity of interest, as is commonly done in MCMC (Robert and Casella, 2004). In words, the ESS indicates how many independent samples were obtained from the total number of iterations for which the algorithm was run, and if the chain exhibits large correlation then the ESS will typically be much smaller than the number of iterations.

Throughout this section we consider a "typical setting" where data was simulated using a mean tract length of $\delta = 236$, a mutation rate per site $\theta_s = 0.03$, and a recombination rate per site $\rho_s = 0.005$. These three parameters remain fixed, and consequently the inference is carried only for the number of recombination events and their locations on the genome and on the tree, i.e. the target distribution is given by (2).

Example 1. Typical setting: 50 sequences, length 50 kilobase pairs (Kb), mean tract length $\delta = 236$, global mutation rate $\theta = 1500$ ($\theta_s = 0.03$), global recombination rate $\rho = 250$ ($\rho_s = 0.005$).

Figure 2 contains traceplots for the number of recombination events (top row) and for the value of the log-likelihood (middle and bottom rows). Observe that the schemes for which $N > 1$ have a similar performance when comparing against iteration number (plots on the left); this is not surprising since the annealing moves (those involving T) are perturbed using an MCMC algorithm with a proposal equal to the prior, and also the multiple instances (those involving N) are generated using the prior. However, the computational burden is different in each setting since for a fixed value of NT the algorithm could be parallelised using N cores but still requires $T - 1$ serial iterations of the annealing process. Hence, for this case, the setting where $T = 1$ and $N = 16$ should perform best when taking into account the running time. The plots on the right incorporate this information and notice that all the schemes still seem to perform better than the standard RJMCMC chain (in blue) as they reach a region where the likelihood is high in a shorter amount of time, this can also be seen more clearly in the plots from the

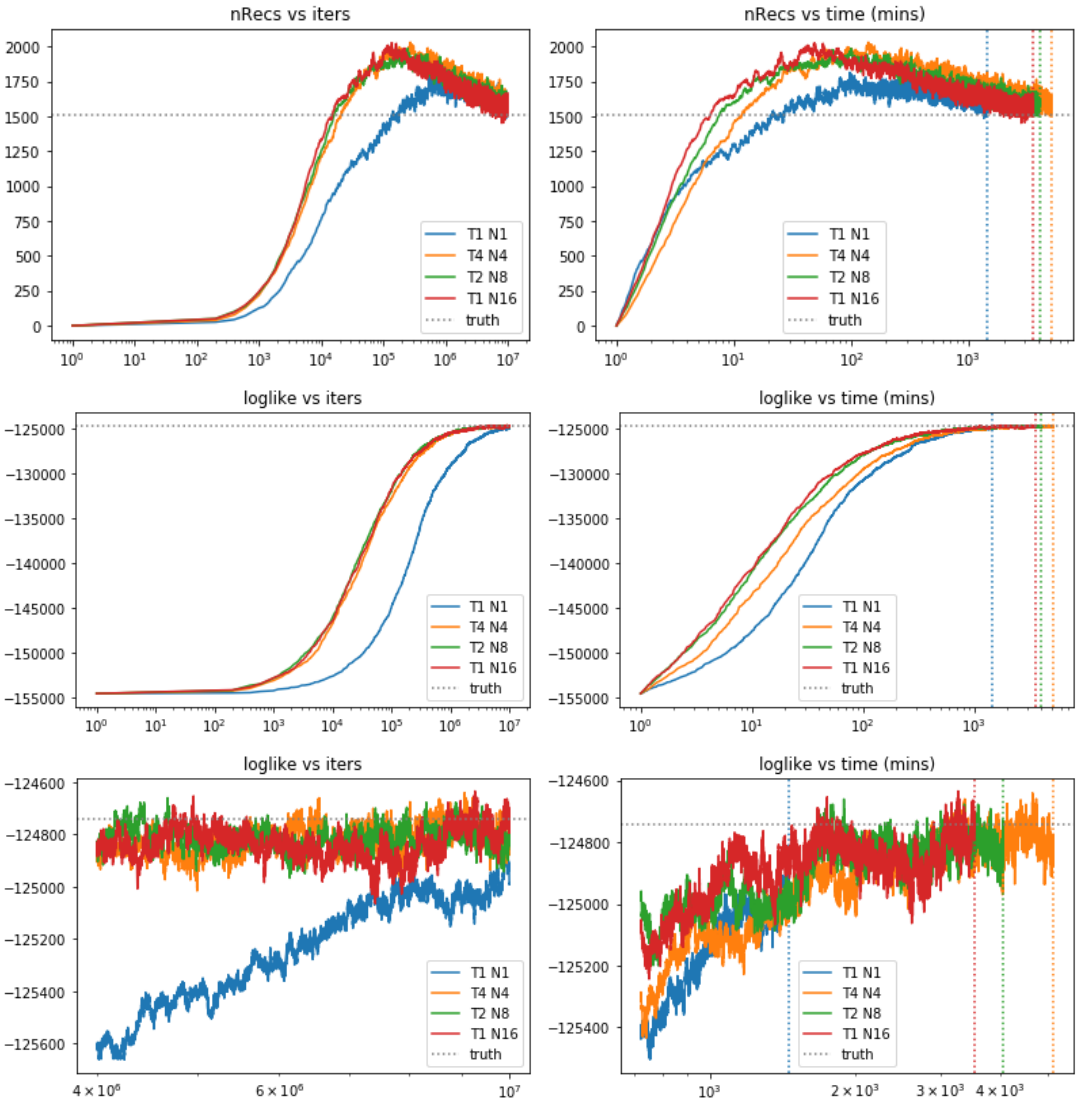

Figure 2: Trace plots for the total number of recombinations and for the log-likelihood for different combinations of T and N and using 10 million iterations. Plots on the left correspond to values vs iteration number, those on the right are vs running time. Grey dotted line corresponds to ground truth, coloured dotted lines indicate time when algorithm stopped. Bottom row: trace plots of the log-likelihood for the last 6 million iterations (left) and after 12 hours of running time (right).

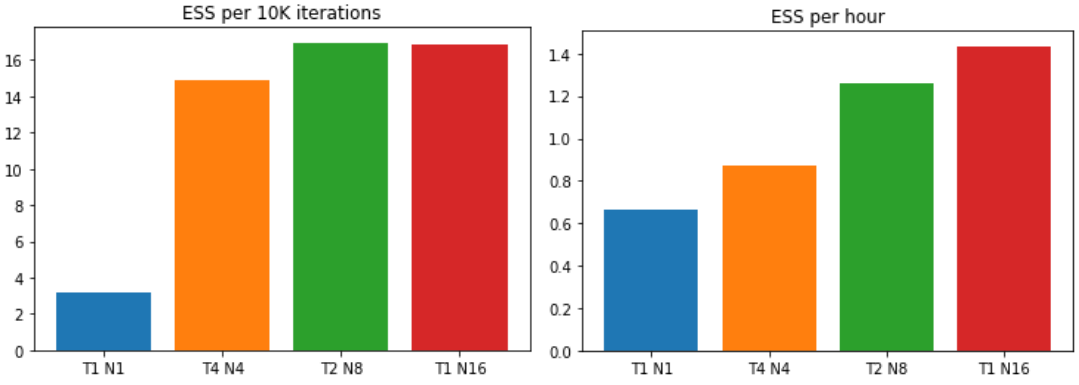

Figure 3: Effective sample sizes per 10 thousand iterations (left) and per hour of running time (right).

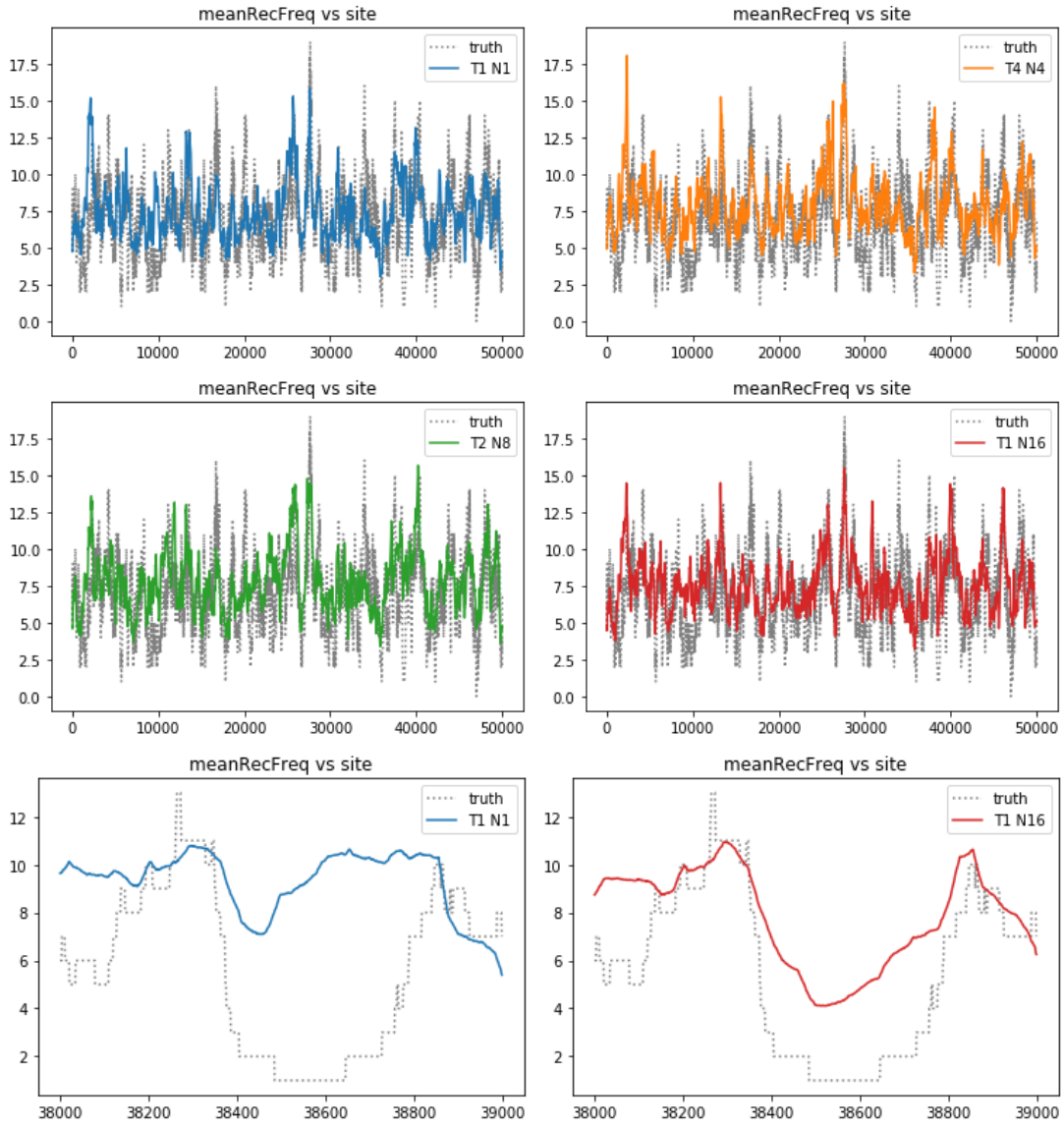

Figure 4: Average recombination frequency vs site number for different combinations of T and N. Grey dotted line corresponds to the true recombination frequency.

bottom row. It is worth noting that despite the RJMCMC chain being closer to the true number of recombination events, it is not a reliable indicator of whether the chain has converged since the value of the log-likelihood is still in a transient phase.

Figure 3 compares the ESS fusing the values of the log-likelihood for the different schemes. Notice that when the running time is not taken into account (left plot) the three settings of RmJMCMC (those with $NT = 16$) have very similar ESS, as expected from the left plots of Figure 3. The right plot is in line with the conjecture that the setting when $T = 1$ and $N = 16$ is the most efficient, observing that all three still outperform the standard RJMCMC chain.

Figures 4 and 5 compare other quantities of interest. Figure 4 shows the recombination frequencies across the sequence of length 50K depicting the active number of recombinations affecting each site. Despite the apparent similarity of the results for the different schemes (top and middle rows), when looking at a smaller scale there are important deviances from the truth (bottom plots) that are direct consequence of a bad mixing from the chain. Settings with large values of NT have greater chances of better exploring the complicated state space in the ClonalOrigin model.

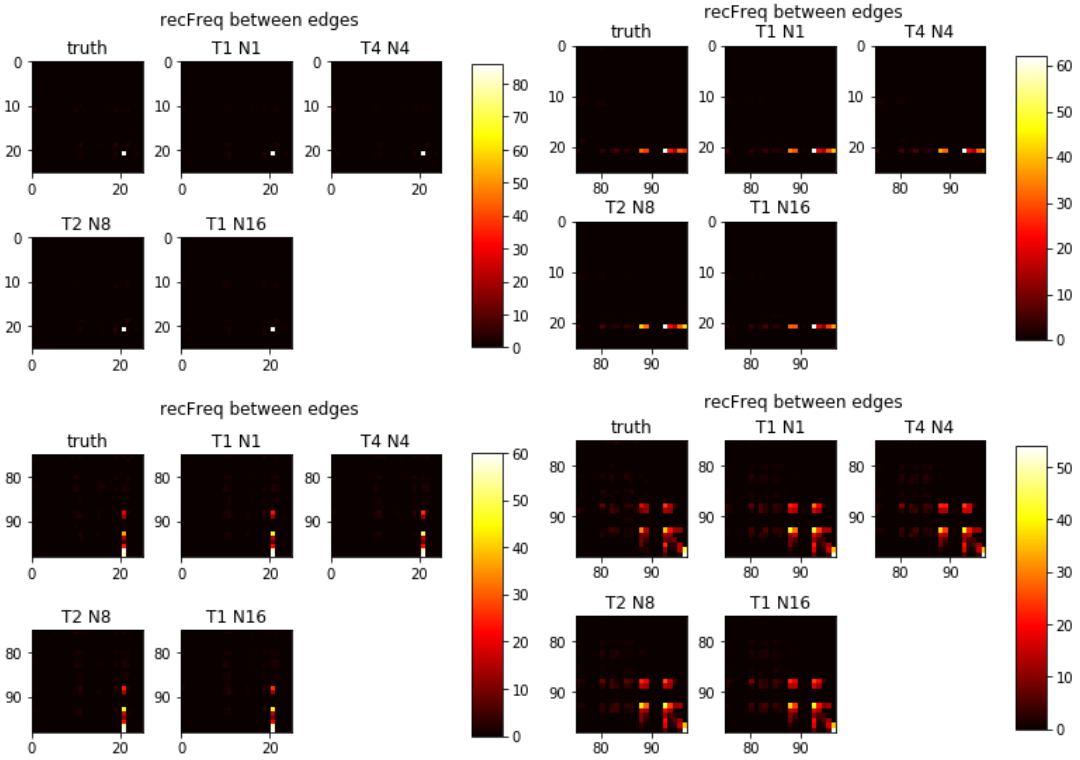

Figure 5: Average recombination frequency between edges, y-axis indicates the edge number where a recombination departed, x-axis denotes the edge number where a recombination landed. The plots correspond to the corners of the 98x99 matrices containing the recombination frequencies between edges.

Figure 5 also shows recombination frequencies, but between edges in the tree. Coloured squares indicate pair of edges on the tree joined by several recombination events, depending on the scale. In this case, the differences between the schemes and against the ground truth are negligible.

Example 2. Typical setting: 50 sequences, length 10Kb, mean tract length $\delta = 236$, global mutation rate $\theta = 300$ ($\theta_s = 0.03$), global recombination rate $\rho = 50$ ($\rho_s = 0.005$).

This example is similar to the previous one, except for the length of the sequence which is now 10Kb. The global mutation and recombination rates have been scaled appropriately leaving the per-site rates the same as before. Figure 6 contains some plots of interest, we observe once more an improvement of the ESS and faster convergence of the log-likelihood without considering running times. However, when time is taken into account (right plots) running RmJMCMC seems to be worth it only when $T = 1$ and $N = 16$. Having shorter sequences makes the inference problem less expensive, so the schemes where $T > 1$ do not seem to add any benefit.

Example 3. Typical setting: Comparison across schemes with different sequence lengths and different number of sequences. For all cases $\delta = 236$, $\theta_s = 0.03$, $\rho_s = 0.005$.

Here we compare how the ESS is affected as the length of the sequences becomes larger and as the number of sequences increases. Figure 7 presents the results for lengths of 10Kb, 20Kb and 50Kb. From the plots in the bottom row, the inverse of the ESS appears to be linear as the length of the sequence increases, more specifically, doubling the sequence length halves the ESS. Interestingly, the reduction of ESS for the RJMCMC chain (in blue) when taking into account the running time seems more drastic than the other schemes.

In contrast, increasing the sequence length seems to have a logarithmic effect on the inverse of the ESS as seen in Figure 8; however, when considering running times, the effect becomes linear. For this case,

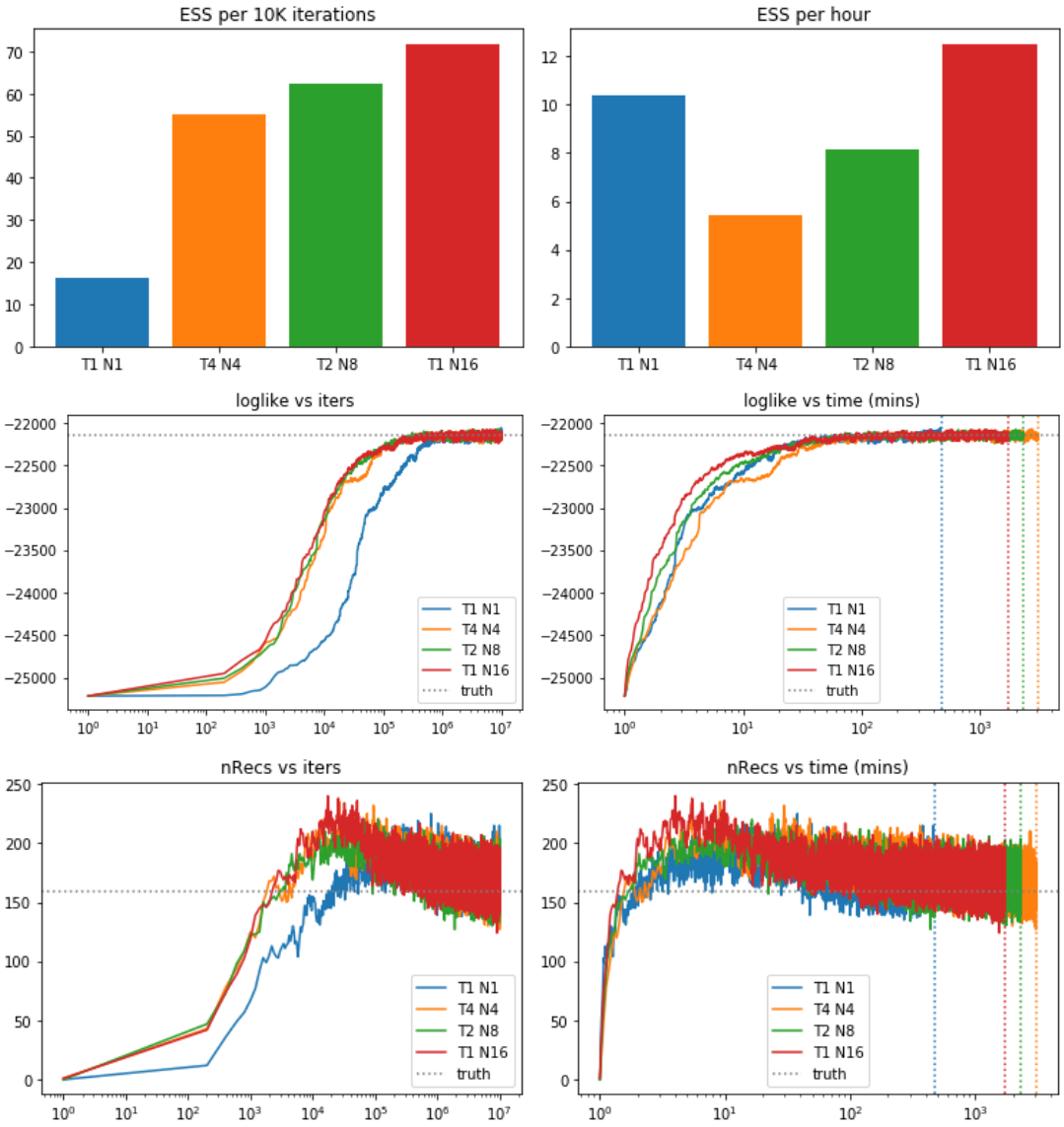

Figure 6: Top row: Effective sample sizes per 10 thousand iterations (left) and per hour of running time (right). Middle and bottom row: Trace plots for the total number of recombinations (bottom) and for the log-likelihood (middle) for different combinations of T and N and using 10 million iterations.

the method with less reduction of ESS is the RJMCMC chain. Thus, RmJMCMC seems to outperform RJMCMC when having larger sequences rather than when having lots of short sequences.

Application to *Escherichia coli* data

Here we present results for a dataset of 27 sequences from *Escherichia coli* on two regions with different recombination intensity; this dataset has been previously studied using ClonalOrigin (Didelot et al., 2012). Similarly as in the simulated examples, we have fixed the clonal genealogy $\mathcal{T}$ and also some of the global parameters. The specific details on how these fixed values were obtained can be found in Didelot et al. (2012).

Example 4. *E. coli* data on region of length 27.1Kb. The fixed parameters considered here are the clonal genealogy $\mathcal{T}$ and the mutation rate per site $\theta_s = 0.0125$. Therefore, the inference is performed on the set of recombination events $\mathcal{R}$ and the global parameters ρ and δ, i.e. the target distribution arises from (1) by fixing $\mathcal{T}$ and θ accordingly.

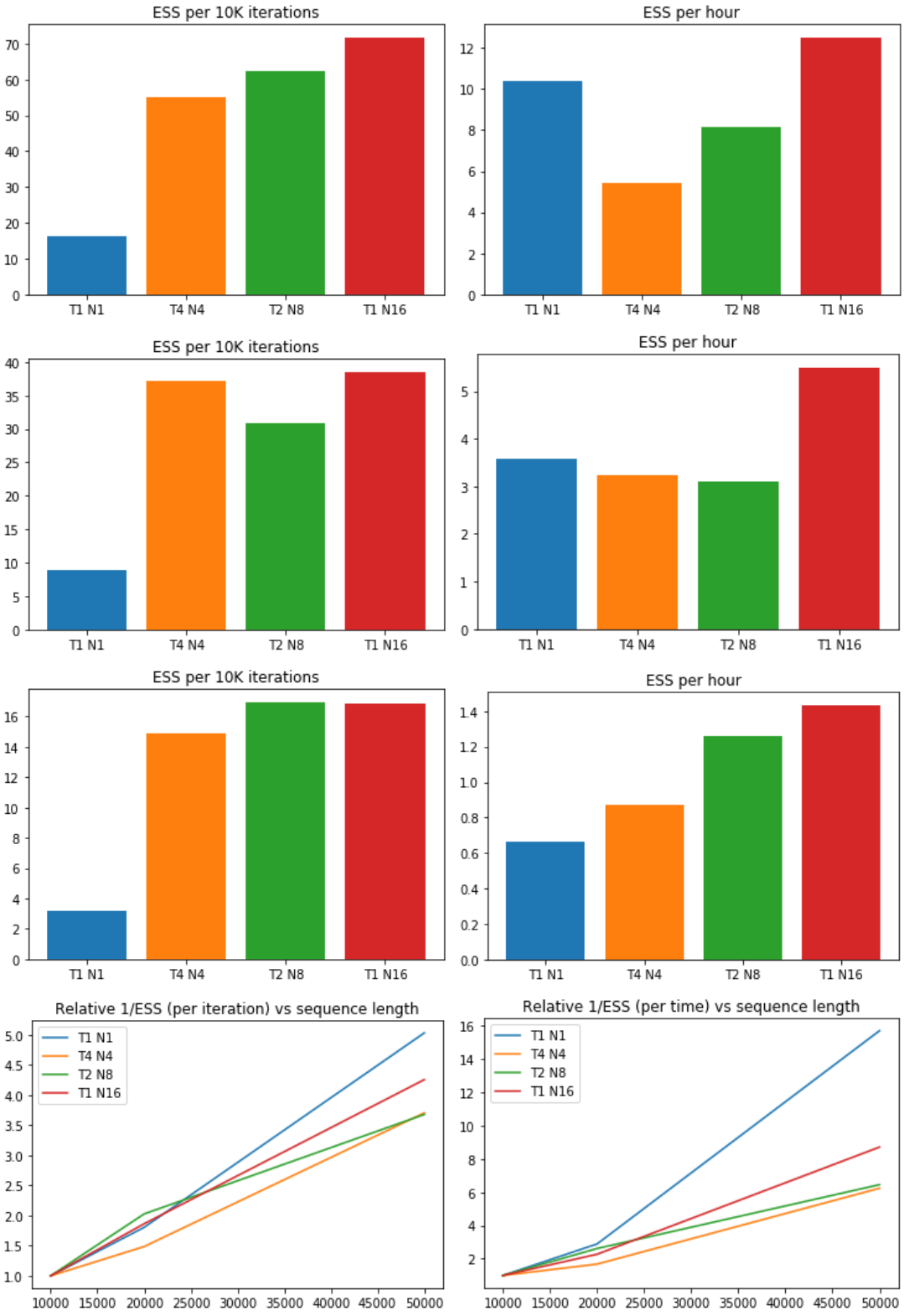

Figure 7: ESS comparison for 10Kb, 20Kb and 50Kb (plots in rows 1-3 , respectively) considering 50 sequences in all cases. Bottom plots: increase of the inverse of the ESS as a function of sequence length.

Figure 9 presents some preliminary results on a region with a moderate recombination frequency of ρ_s near 0.01. Observe that despite increasing N does not seem to improve the value of the ESS adjusted for running time, the log-likelihood trace plot appears to indicate there is some advantage of taking $N > 1$ during the burn-in phase. Interestingly, the trace plots in the bottom row show that when $N > 1$ the

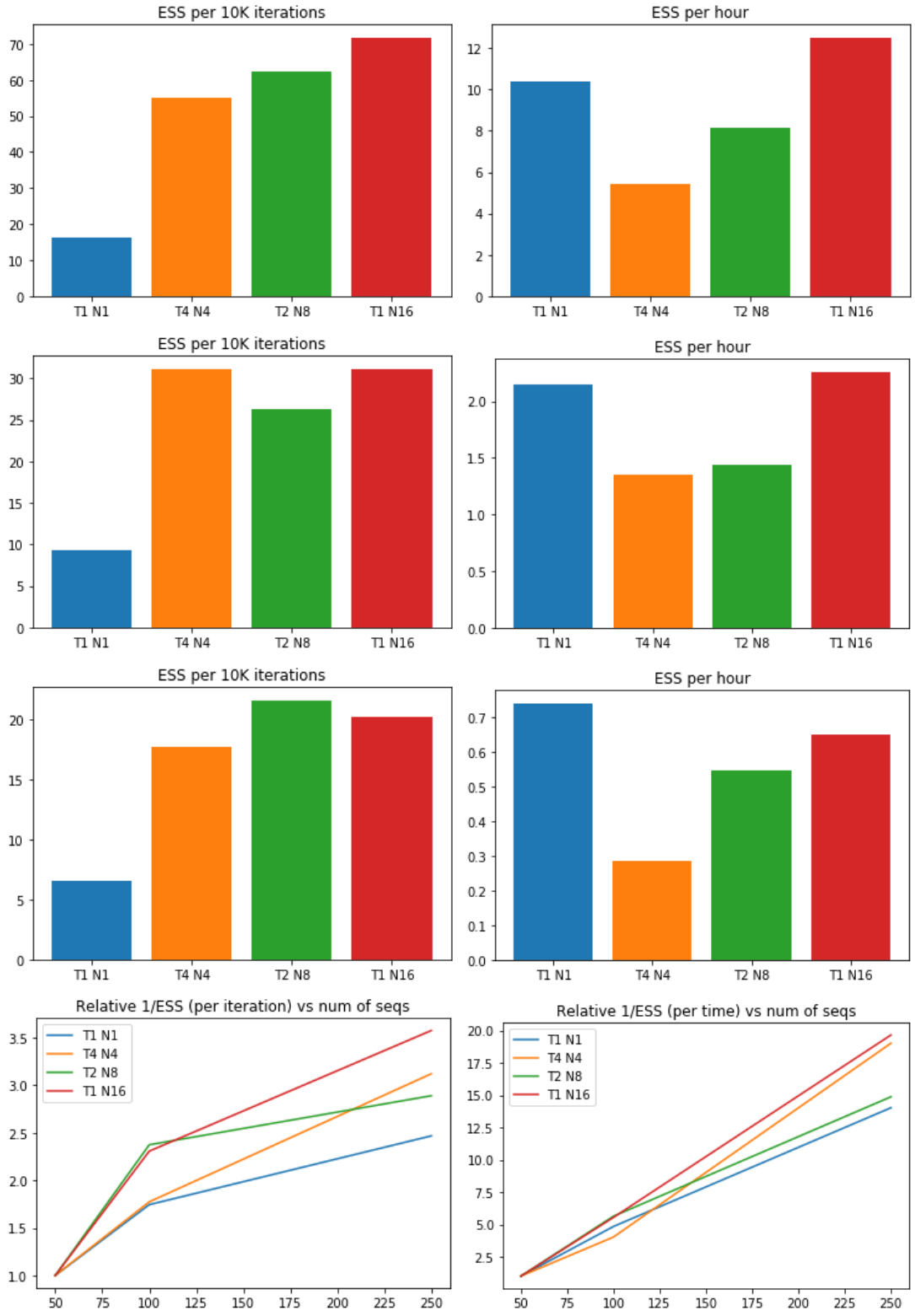

Figure 8: ESS comparison for 50, 100 and 250 sequences (plots in rows 1-3 , respectively) considering a length of 10K in all cases. Bottom plots: increase of the inverse of the ESS as a function of sequence length.

chain appears to favour large values for ρ during the burn-in stage, whereas for δ might take different routes towards the apparent high-posterior region. Clearly, further simulations are needed to rule out whether RmJMCMC offers a clear advantage.

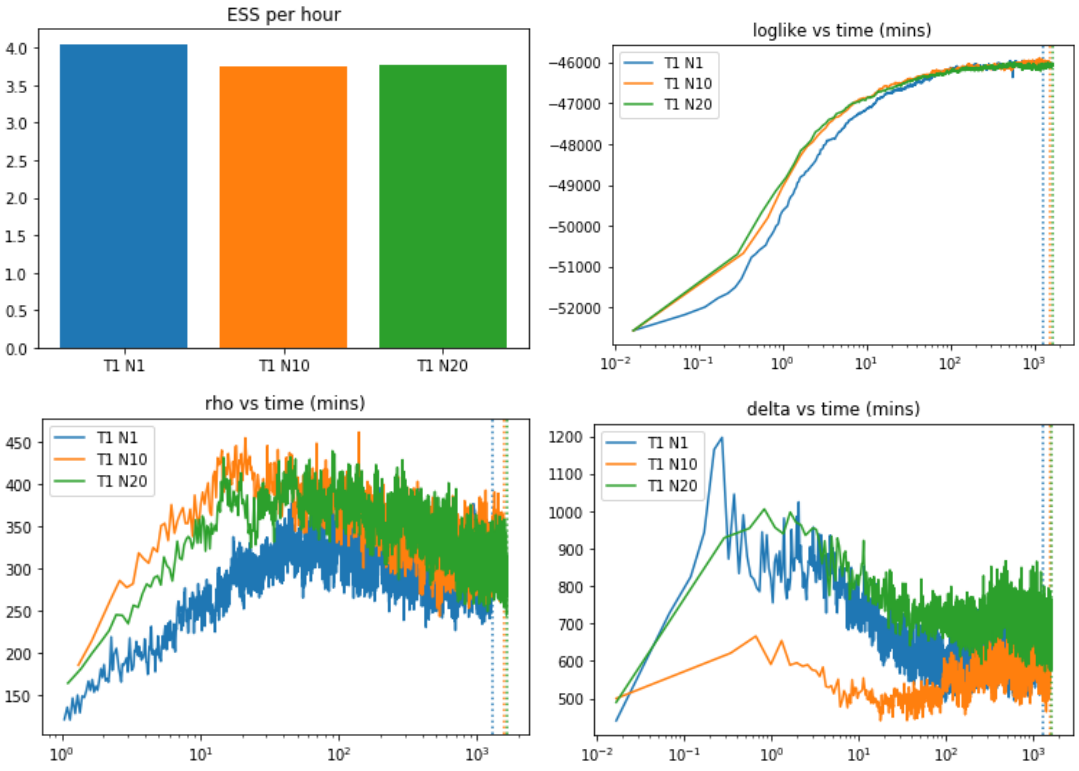

Figure 9: *E. coli* data on region of length 27.1Kb. Top-left: ESS comparison for three different RmJMCMC schemes where $T = 1$; top-right: trace plots for the log-likelihood; bottom-left: trace plots for ρ; bottom-right: trace plots for δ.

Example 5. *E. coli* data on region of length 3.4Kb. Here we consider a shorter region but for which we know the recombination frequency is higher. The fixed parameters considered once again are the clonal genealogy $\mathcal{T}$ and the mutation rate per site $\theta_s = 0.0125$, we have also fixed the tract length $\delta = 542$ due to the high levels of recombination. The inference is thus performed on the set of recombination events $\mathcal{R}$ and the global parameter ρ, i.e. the target distribution arises from (1) by fixing $\mathcal{T}$, θ and δ accordingly.

Observe from Figure 10 that this is a more challenging region for the *E. coli* data. In particular, observe that apparently the setting $N = 12$ does not provide an advantage in terms of reaching a high likelihood region in a shorter time; however, when looking at the trace plots for the number of recombination events and for ρ it does appear to have an improvement. In this case ESS plots have not been presented since clearly the chain has not reached stationarity, even after nearly 21 days of running the algorithms. Recall that accurate ESS values are based on the fact that the chains have converged. Finally, observe that the recombination rate in this region is much higher than the previous region, with ρ_s reaching values around 0.17. Due to the difficulty of this example, further simulations are needed in order to have a clearer picture of the benefits, if any, from running a RmJMCMC algorithm.

Discussion

The RmJMCMC algorithm has the potential of speeding up the convergence in the ClonalOrigin model towards a region of high likelihood, hence reducing the number of iterations and running time when compared to a standard RJMCMC. Here the RmJMCMC algorithm has been presented in the context of the ClonalOrigin model, but in principle it could be implemented to other models in population genetics where the dimension of the posterior is not fixed.

The version used in the examples presented here is possibly the simplest one could implement, namely

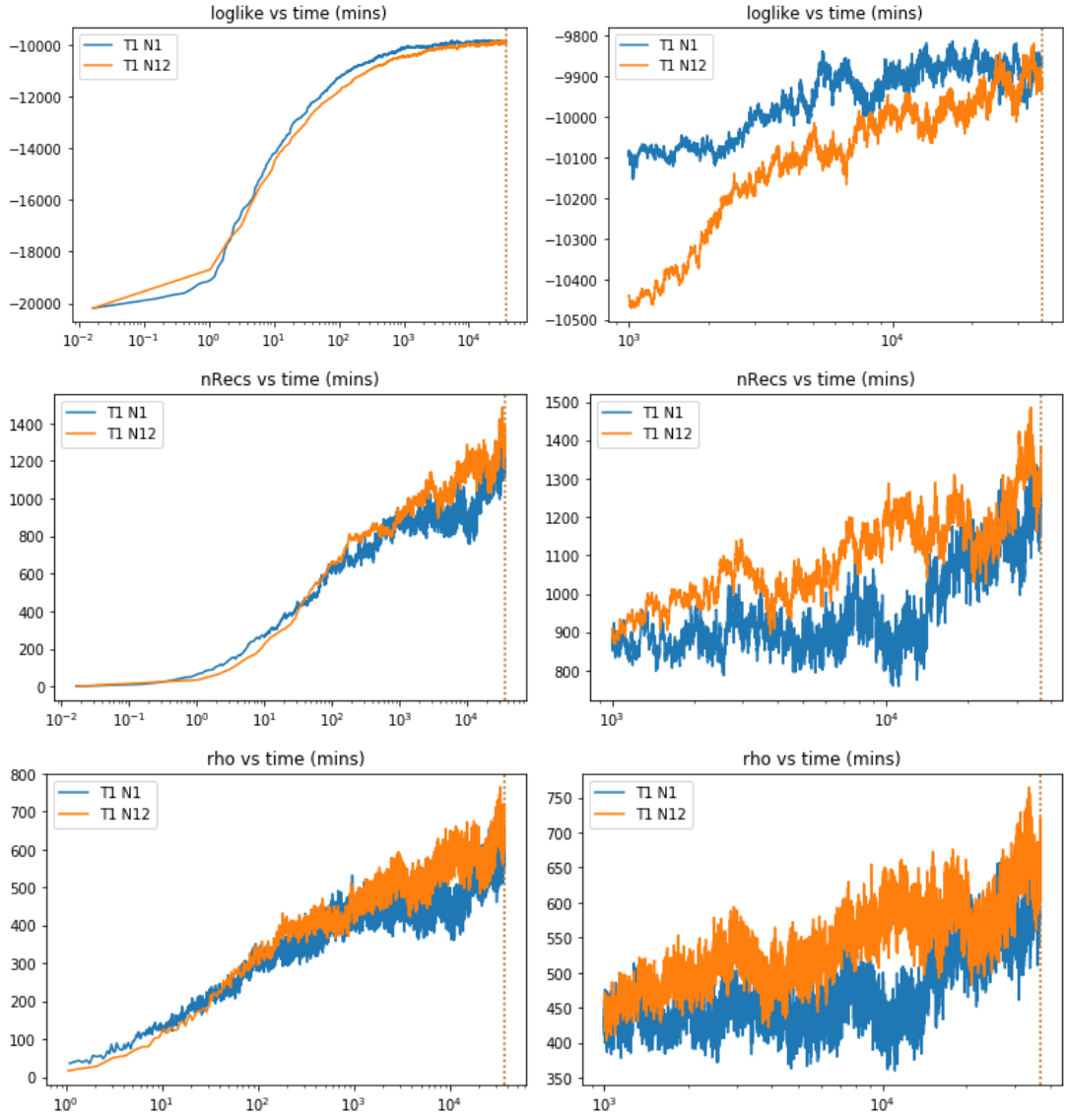

Figure 10: *E. coli* data on region of length 3.4Kb. Top-left: ESS comparison for three different RmJMCMC schemes where $T = 1$; top-right: trace plots for the log-likelihood; bottom-left: trace plots for ρ; bottom-right: trace plots for δ.

using the prior distribution for proposing and perturbing recombination events. However, other alternatives are possible that may lead to better results, for example splitting or merging existing recombination events, or modifying the prior accordingly if some regions in the tree or in the genome are more likely to recombine. Similarly, different perturbations in the annealing step may be considered since in the presented examples there is no clear advantage when taking $T > 1$. Alternatives include random-walk style proposals for some (or all) of the variables involved in the proposed recombination event (a_i, b_i, x_i, y_i). In this respect, within the annealing process, one should also consider perturbing all of the existing recombination events and not just the recently proposed one.

Finally, it is worth pointing out that the number of cores used in the presented examples was not particularly large; nonetheless we were able to observe significant improvements in some cases. It would be interesting to perform a much larger study where several hundred cores are at hand.

References

P. Alquier, N. Friel, R. Everitt, and A. Boland. Noisy Monte Carlo: convergence of Markov chains with approximate transition kernels. *Statistics and Computing*, 26(1-2):29–47, jan 2016. ISSN 0960-3174. doi: 10.1007/s11222-014-9521-x.

C. Andrieu, A. Doucet, S. Yıldırım, and N. Chopin. On the utility of Metropolis-Hastings with asymmetric acceptance ratio. mar 2018. URL http://arxiv.org/abs/1803.09527.

R. Bouckaert, T. G. Vaughan, M. Fourment, A. Gavryushkina, J. Heled, K. Denise, N. D. Maio, M. Matschiner, H. Ogilvie, L. Plessis, and A. Popinga. BEAST 2.5 : An Advanced Software Platform for Bayesian Evolutionary Analysis. *PLoS Comput. Biol.*, 15(4):e1006650, 2019.

T. Brown, X. Didelot, D. J. Wilson, and N. De Maio. SimBac: simulation of whole bacterial genomes with homologous recombination. *Microb. Genomics*, 2:10.1099/mgen.0.000044, 2016. ISSN 2057-5858. doi: 10.1099/mgen.0.000044.

N. De Maio and D. J. Wilson. The Bacterial Sequential Markov Coalescent. *Genetics*, 206(1):333–343, may 2017. ISSN 0016-6731. doi: 10.1534/genetics.116.198796.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, jun 2006. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2006.00553.x.

X. Didelot and D. Falush. Inference of bacterial microevolution using multilocus sequence data. *Genetics*, 175(3):1251–66, 2007. ISSN 0016-6731. doi: 10.1534/genetics.106.063305.

X. Didelot and D. J. Wilson. ClonalFrameML: Efficient Inference of Recombination in Whole Bacterial Genomes. *PLoS Comput. Biol.*, 11(2):e1004041, feb 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004041.

X. Didelot, D. J. Lawson, and D. Falush. SimMLST: simulation of multi-locus sequence typing data under a neutral model. *Bioinformatics*, 25(11):1442–4, 2009. ISSN 1367-4811. doi: 10.1093/bioinformatics/btp145.

X. Didelot, D. Lawson, A. Darling, and D. Falush. Inference of homologous recombination in bacteria using whole-genome sequences. *Genetics*, 186(4):1435–1449, 2010. ISSN 00166731. doi: 10.1534/genetics.110.120121.

X. Didelot, G. Méric, D. Falush, and A. E. Darling. Impact of homologous and non-homologous recombination in the genomic evolution of Escherichia coli. *BMC Genomics*, 13(1):256, 2012. ISSN 1471-2164. doi: 10.1186/1471-2164-13-256.

V. Dinh, A. E. Darling, and F. A. Matsen. Online Bayesian phylogenetic inference: Theoretical foundations via sequential Monte Carlo. *Systematic Biology*, 2018. ISSN 1076836X. doi: 10.1093/sysbio/syx087.

A. Doucet, N. Freitas, and N. Gordon. An Introduction to Sequential Monte Carlo Methods. In *Sequential Monte Carlo Methods in Practice*, pages 3–14. Springer New York, New York, NY, 2001. doi: 10.1007/978-1-4757-3437-9_1.

R. G. Everitt, R. Culliford, F. Medina-Aguayo, and D. J. Wilson. Sequential Monte Carlo with transformations. *Statistics and Computing*, 2019. ISSN 0960-3174. doi: 10.1007/s11222-019-09903-y.

J. Felsenstein. Maximum Likelihood and Minimum-Steps Methods for Estimating Evolutionary Trees from Data on Discrete Characters. *Systematic Biology*, 22(3):240–249, sep 1973. ISSN 1063-5157. doi: 10.1093/sysbio/22.3.240.

J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.*, 17(6):368–376, 1981. ISSN 00222844. doi: 10.1007/BF01734359.

P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995. ISSN 0006-3444. doi: 10.1093/biomet/82.4.711.

R. C. Griffiths. Ancestral inference from samples of DNA sequences with recombination. *Journal of Computational Biology*, 1996. ISSN 10665277. doi: 10.1089/cmb.1996.3.479.

J. Hedge and D. J. Wilson. Bacterial phylogenetic reconstruction from whole genomes is robust to recombination but demographic inference is not. *MBio*, 5(6):6–9, 2014. ISSN 21507511. doi: 10.1128/mBio.02158-14.

R. R. Hudson. Gene genealogies and the coalescent process. In *Oxford Surveys in Evolutionary Biology*. 1990.

T. H. Jukes and C. R. Cantor. Evolution of protein molecules BT - Mammalian protein metabolism. In *Mammalian protein metabolism*. 1969.

G. Karagiannis and C. Andrieu. Annealed Importance Sampling Reversible Jump MCMC Algorithms. *Journal of Computational and Graphical Statistics*, 22(3):623–648, jul 2013. ISSN 1061-8600. doi: 10.1080/10618600.2013.805651.

J. F. C. Kingman. The coalescent. *Stochastic Processes and their Applications*, 1982. ISSN 03044149. doi: 10.1016/0304-4149(82)90011-4.

P. Marjoram and J. D. Wall. Fast "coalescent" simulation. *BMC Genetics*, 2006. doi: https://doi.org/10.1186/1471-2156-7-16.

G. A. McVean and N. J. Cardin. Approximating the coalescent with recombination. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1459):1387–1393, jul 2005. ISSN 0962-8436. doi: 10.1098/rstb.2005.1673.

R. M. Neal. Annealed Importance Sampling. *Statistics and computing*, 11(2):125–139, 2001.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer New York, New York, NY, 2004. ISBN 978-1-4419-1939-7. doi: 10.1007/978-1-4757-4145-2.

M. H. Schierup and J. Hein. Consequences of recombination on traditional phylogenetic analysis. *Genetics*, 156(2):879–91, oct 2000. ISSN 0016-6731.

T. G. Vaughan, D. Welch, A. J. Drummond, P. J. Biggs, T. George, and N. P. French. Inferring Ancestral Recombination Graphs from Bacterial Genomic Data. *Genetics*, 205(2):857–870, feb 2017. ISSN 0016-6731. doi: 10.1534/genetics.116.193425.

C. Wiuf and J. Hein. Recombination as a point process along sequences. *Theoretical Population Biology*, 1999. ISSN 00405809. doi: 10.1006/tpbi.1998.1403.

C. Wiuf and J. Hein. The coalescent with gene conversion. *Genetics*, 155(1):451–62, may 2000. ISSN 0016-6731.

Supplementary Material

Details on priors

We use a coalescent tree to represent the clonal genealogy of n samples and denote a tree by $\mathcal{T} = (\tau, \mathbf{t})$, which is composed of a topology τ and a vector $\mathbf{t} = (t_2, \ldots, t_n)$ of branch lengths. A Kingman's coalescent prior (Kingman, 1982) is assumed for $\mathcal{T}$, meaning

$$\begin{aligned} p_0(\mathcal{T}) = p_0(\mathbf{t} \mid \tau)\, p_0(\tau) &= \prod_{i=2}^{n} \binom{i}{2} \exp\left\{-\binom{i}{2} t_i\right\} \prod_{j=0}^{n-2} \binom{n-j}{2}^{-1} \\ &= \exp\left\{-\sum_{i=2}^{n} \binom{i}{2} t_i\right\}. \end{aligned}$$

The number of recombination events R is assumed, a priori, to be a Poisson random variable with mean $\rho\mathcal{L}/2$, where ρ denotes the global recombination rate and $\mathcal{L} = \sum_{i=2}^{n} i t_i$ is the total branch length of the tree $\mathcal{T}$. Hence

$$p_0(R \mid \rho, \mathcal{L}) = \exp\left\{-\frac{\rho\mathcal{L}}{2}\right\} \left(\frac{\rho\mathcal{L}}{2}\right)^R / R!.$$

Each recombination event is formed by the following variables: departure and arrival points on the genealogy, and start and end sites on the genome. These four variables, when referring to the i-th recombination edge, are denoted by a_i, b_i, x_i and y_i, respectively. Both variables a_i and b_i are fully determined by a time and lineage on the tree (denoted respectively by $a_i[t]$ and $a_i[l]$, respectively), and the chosen priors are those from the construction of an ARG (Wiuf and Hein, 1999). These are a uniform distribution on the tree for b_i, and from this arrival point the edge reconnects at a_i on the tree at a rate proportional to the number of active lineages. We then have

$$\mathbb{P}[b_i = \beta \mid \mathcal{T}] = \frac{1}{\mathcal{L}},$$

and, assuming that at time $b_i[t]$ there are $m \in \{2, \ldots, n\}$ active lineages and letting $s_k = \sum_{i=k}^{n} t_i$, if $u \in (b_i[t], s_m)$

$$\mathbb{P}[a_i[t] \geq u \mid b_i, \mathcal{T}] = \exp(-m(u - b_i[t])) = \exp(-\mathcal{L}(b_i[t], u)),$$

where $\mathcal{L}(s, t)$ denotes the sum of branch lengths from time s to time t on the tree. If $u \in (s_k, s_{k-1})$ for any $k \in \{2, \ldots, m\}$, considering $s_1 = \infty$,

$$\begin{aligned} \mathbb{P}[a_i[t] \geq u \mid b_i, \mathcal{T}] &= \mathbb{P}[a_i[t] \geq u, a_i[t] \geq s_k \mid b_i, \mathcal{T}] = \mathbb{P}[a_i[t] \geq u \mid a_i[t] \geq s_k, b_i, \mathcal{T}]\, \mathbb{P}[a_i[t] \geq s_k \mid b_i, \mathcal{T}] \\ &= \exp\{-(k-1)(u - s_k)\} \exp\left\{-m(s_m - b_i[t]) - \sum_{h=k}^{m-1} h(s_h - s_{h+1})\right\} \\ &= \exp\{-(k-1)(u - s_k)\} \exp\{-\mathcal{L}(b_i[t], s_m)\} = \exp\{-\mathcal{L}(b_i[t], u)\}. \end{aligned}$$

Therefore,

$$\mathbb{P}[a_i[t] = u \mid b_i, \mathcal{T}] = \begin{cases} m \exp\{-\mathcal{L}(b_i[t], u)\}, & \text{if } u \in (b_i[t], s_m) \\ (k-1) \exp\{-\mathcal{L}(b_i[t], u)\}, & \text{if } u \in (s_k, s_{k-1}) \text{ for } k \in \{2, \ldots, m\}. \end{cases}$$

Once $a_i[t]$ has been found, the lineage $a_i[t]$ on which the recombination event departs is chosen uniformly across the active lineages. Hence

$$\mathbb{P}[a_i[l] \mid a_i[t] = u, b_i, \mathcal{T}] = \begin{cases} 1/m, & \text{if } u \in (b_i[t], s_m) \\ 1/(k-1), & \text{if } u \in (s_k, s_{k-1}) \text{ for } k \in \{2, \ldots, m\}. \end{cases}$$

The previous two probabilities lead to

$$\mathbb{P}\left[a_i = \alpha \mid b_i, \mathcal{T}\right] = \mathbb{P}\left[a_i\left[l\right] = \alpha\left[l\right] \mid a_i\left[t\right] = \alpha\left[t\right], b_i, \mathcal{T}\right] \mathbb{P}\left[a_i\left[t\right] = \alpha\left[t\right] \mid b_i, \mathcal{T}\right] = \exp\left\{-\mathcal{L}\left(b_i\left[t\right], \alpha\left[t\right]\right)\right\}.$$

The priors for x_i and y_i are constructed assuming a uniform distribution on the sequence for x_i and a geometric distribution of mean $\delta > 0$ for the difference $y_i - x_i \mid x_i$. When the sequence is made of B blocks comprising a total length of $L = \sum_{b=1}^{B} L_b$, the priors need to be modified accordingly as in (Didelot et al., 2010). Let s_i^b denote the i-th site in block the b-th block, where $i \in \{1, \ldots, L_b\}$ and $s_{L_b}^b + 1 = s_1^{b+1}$, then if u is not the first site of a block $\mathbb{P}\left[x_i = u \mid \delta\right] = \kappa$ (a constant); whereas if u is at the beginning of a block (say $u = s_1^b$) we can use the uncensored variables $\bar{x}_i$ and $\bar{y}_i$ (which denote the true sites where the recombination begins and ends) to derive

$$\begin{aligned}
\mathbb{P}\left[x_i = s_1^b \mid \delta\right] &= \sum_{s=s_1^b-d}^{s_1^b} \mathbb{P}\left[\bar{x}_i = s, \bar{y}_i \geq s_1^b + 1 \mid \delta\right] \\
&= \sum_{s=s_1^b-d}^{s_1^b} \mathbb{P}\left[\bar{y}_i - s \geq s_1^b - s + 1 \mid \bar{x}_i = s, \delta\right] \mathbb{P}\left[\bar{x}_i = s \mid \delta\right] \\
&= \sum_{s=s_1^b-d}^{s_1^b} \left(1 - \delta^{-1}\right)^{s_1^b - s} \kappa = \kappa\delta\left(1 - \left(1 - \delta^{-1}\right)^{d+1}\right) \\
&\underset{d\to\infty}{\approx} \kappa\delta,
\end{aligned}$$

where d denotes the number of sites separating the $(b-1)$-th and b-th blocks. Summing over all possible values for u, i.e. B sites at the beginning of blocks and $L - B$ in other regions, we get $(L - B)\kappa + B\kappa\delta = 1$; this implies $\kappa = 1/\left(\delta B + L - B\right)$, leading to

$$p_{0,x}\left(u \mid \delta\right) = \frac{\delta}{\delta B + L - B}\mathbb{1}\left(u \in \left\{s_1^1, \ldots, s_1^B\right\}\right) + \frac{1}{\delta B + L - B}\mathbb{1}\left(u \notin \left\{s_1^1, \ldots, s_1^B\right\}\right).$$

In order to find the density of y_i we use the fact that $\bar{y}_i - \bar{x}_i \mid \bar{x}_i \sim Geom\left(\delta\right)$ on the positive integers. We then have for $v \geq u$ and $u \in \left\{s_2^b, \ldots, s_{L_b}^b\right\}$

$$\begin{aligned}
\mathbb{P}\left[y_i = v \mid x_i = u, \delta\right] &= \begin{cases} \mathbb{P}\left[\bar{y}_i = v \mid \bar{x}_i = u, \delta\right], & \text{if } u < v \leq s_{L_b}^b \\ \mathbb{P}\left[\bar{y}_i \geq v \mid \bar{x}_i = u, \delta\right], & \text{if } v = s_{L_b} + 1 \end{cases} \\
&\approx \begin{cases} \delta^{-1}\left(1 - \delta^{-1}\right)^{v-u-1}, & \text{if } u < v \leq s_{L_b}^b \\ \left(1 - \delta^{-1}\right)^{s_{L_b}^b - u}, & \text{if } v = s_{L_b}^b + 1, \end{cases}
\end{aligned}$$

and similarly if $u = s_1^b$

$$\begin{aligned}
\mathbb{P}\left[y_i = v \mid x_i = s_1^b, \delta\right] &= \begin{cases} \sum_{s=s_1^b-d}^{s_1^b} \frac{\mathbb{P}\left[\bar{y}_i = v, \bar{x}_i = s, x_i = s_1^b \mid \delta\right]}{\mathbb{P}\left[x_i = s_1^b \mid \delta\right]}, & \text{if } s_1^b < v \leq s_{L_b}^b \\ \sum_{s=s_1^b-d}^{s_1^b} \frac{\mathbb{P}\left[\bar{y}_i \geq v, \bar{x}_i = s, x_i = s_1^b \mid \delta\right]}{\mathbb{P}\left[x_i = s_1^b \mid \delta\right]}, & \text{if } v = s_{L_b}^b + 1 \end{cases} \\
&= \begin{cases} \sum_{s=s_1^b-d}^{s_1^b} \frac{\mathbb{P}\left[\bar{y}_i = v, x_i = s_1^b \mid \bar{x}_i = s, \delta\right] \mathbb{P}\left[\bar{x}_i = s \mid \delta\right]}{\kappa\delta\left(1 - (1 - \delta^{-1})^{d+1}\right)} & \text{if } s_1^b < v \leq s_{L_b}^b \\ \sum_{s=s_1^b-d}^{s_1^b} \frac{\mathbb{P}\left[\bar{y}_i \geq v \mid \bar{x}_i = s, \delta\right] \mathbb{P}\left[\bar{x}_i = s \mid \delta\right]}{\kappa\delta\left(1 - (1 - \delta^{-1})^{d+1}\right)}, & \text{if } v = s_{L_b}^b + 1 \end{cases} \\
&\approx \begin{cases} \delta^{-1}\left(1 - \delta^{-1}\right)^{v - s_1^b - 1}, & \text{if } s_1^b < v \leq s_{L_b}^b \\ \left(1 - \delta^{-1}\right)^{s_{L_b}^b - s_1^b}, & \text{if } v = s_{L_b}^b + 1 \end{cases}
\end{aligned}$$

Hence,

$$p_{0,y}\left(v \mid x = s_i^b, \delta\right) = \left(1 - \delta^{-1}\right)^{v - s_i^b - 1}\mathbb{1}\left(v = s_{L_b}^b + 1\right) + \delta^{-1}\left(1 - \delta^{-1}\right)^{v - s_i^b - 1}\mathbb{1}\left(v \in \left\{s_{i+1}^b, \ldots, s_{L_b}^b\right\}\right).$$

The previous equations fully define the prior to be used for the set of recombination events, denoted by $\mathcal{R} = (R, a_{1:R}, b_{1:R}, x_{1:R}, y_{1:R})$, i.e.

$$p_0\left(\mathcal{R}\,|\rho,\delta,\mathcal{T}\right) = R!p_0\left(R\mid\rho,\mathcal{T}\right)p_0\left(a_{1:R},b_{1:R},x_{1:R},y_{1:R}|R,\rho,\delta,\mathcal{T}\right),$$

where the factorial comes from ordering the recombination events (e.g. ordering according to one of the continuous variables $\mathbf{x}$ or $\mathbf{y}$). Assuming recombination events occur independently given the genealogy $\mathcal{T}$ and introducing priors for ρ and δ, we obtain

$$\begin{aligned}
p_0\left(\mathcal{R}\,|\rho,\delta,\mathcal{T}\right) &= \exp\left\{-\frac{\rho\mathcal{L}}{2}\right\}\left(\frac{\rho\mathcal{L}}{2}\right)^R\prod_{i=1}^R p_0\left(a_i,b_i,x_i,y_i|\rho,\delta,\mathcal{T}\right)\\
&= \exp\left\{-\frac{\rho\mathcal{L}}{2}\right\}\left(\frac{\rho\mathcal{L}}{2}\right)^R\prod_{i=1}^R p_0\left(a_i|b_i,\mathcal{T}\right)p_0\left(b_i|\mathcal{T}\right)p_0\left(x_i|\delta\right)p_0\left(y_i|x_i,\delta\right)\\
&= \exp\left\{-\frac{\rho\mathcal{L}}{2}\right\}\left(\frac{\rho}{2}\right)^R\prod_{i=1}^R \exp\left\{-\mathcal{L}\left(b_i\left[t\right],\alpha\left[t\right]\right)\right\}p_{0,x}\left(x_i|\delta\right)p_{0,y}\left(y_i|x_i,\delta\right).
\end{aligned}$$

Likelihood computation

Recall that the vector $\mathcal{D} = D_{1:n}$ corresponds to the n observed DNA sequences each composed of L sites, we denote the i-th site in the k-th sequence by $D_k^{(i)}$. For computing the likelihood function at the i-th site, the local tree $\mathcal{T}^{(i)} = \left(\tau^{(i)},\mathbf{t}^{(i)}\right)$ is constructed as a function of the clonal genealogy $\mathcal{T}$ and the set of recombination events $\mathcal{R}$. From this local tree with n leaves, let $z_{1:2n-1}^{(i)}$ denote the whole set of ordered nodes where of $z_{1:n}^{(i)}$ denotes the leaves of the tree. For $k \geq n+1$, let $v_k^{(i)}$ and $w_k^{(i)}$ be the left and right children nodes, respectively, of the node $z_k^{(i)}$. The likelihood of the mutation rate θ and the local tree $\mathcal{T}^{(i)}$ is given by the recursive formula

$$\mathfrak{L}_i\left(\theta,\mathcal{T}^{(i)};z_{1:n}^{(i)}\right) = \sum_{z_{2n-1}^{(i)}\in\{A,C,G,T\}}\pi\left(z_{2n-1}^{(i)}\right)C\left(z_{2n-1}^{(i)}\right),$$

$$\text{where}\quad C\left(z_k^{(i)}\right) = \begin{cases}\sum_{v_k^{(i)}}C\left(v_k^{(i)}\right)p_{z_k^{(i)},v_k^{(i)}}\left(t_{v_k^{(i)}}^{(i)}\right)\sum_{w_k^{(i)}}C\left(w_k^{(i)}\right)p_{z_k^{(i)},w_k^{(i)}}\left(t_{w_k^{(i)}}^{(i)}\right), & \text{if } k\geq n+1;\\ \mathbb{1}\left(z_k^{(i)}=D_k^{(i)}\right), & \text{if } k\leq n;\end{cases}$$

and $p_{z,w}\left(t\right)$ denotes the transition probability from the basis z to the basis w in t units of time, and π corresponds to the limiting probability of such transitions.

For the JC69 model (Jukes and Cantor, 1969), we have $\pi(z) = 0.25$ for each $z \in \{A,C,G,T\}$ and

$$p_{z,w}\left(t\right) = \begin{cases}0.25+0.75e^{-\theta t}, & \text{if } z=w;\\ 0.25-0.25e^{-\theta t}, & \text{if } z\neq w.\end{cases}$$

Hence, assuming independence across the L different sites in all the sequences we obtain

$$\mathfrak{L}\left(\theta,\mathcal{T},\mathcal{R};\mathcal{D}\right) = \prod_{i=1}^L\mathfrak{L}_i\left(\theta,\mathcal{T}^{(i)}\left(\mathcal{T},\mathcal{R}\right);D_{1:n}^{(i)}\right).$$

RJMCMC as an importance estimator within MCMC

To keep notation simple we remove the variables $\rho,\delta,\theta,\mathcal{T}$ and the data $\mathcal{D}$ from the conditional distributions, i.e. we write $\pi\left(R\right) \equiv \pi\left(R\mid\rho,\delta,\theta,\mathcal{T},\mathcal{D}\right)$. In order to jump up in dimension, i.e. to go from R recombination events to $R+1$, we must create four variables a_R, b_R, x_R, y_R. This is done using

an auxiliary distribution, for example the prior for such variables, which we denote by $\varphi_{R\to R+1}$ and that in principle could also depend on the data or other parameters. Similarly, in order to perform a jump down in dimension we need a way to merge or remove recombination events, this is done using the distribution $\varphi_{R+1\to R}$. Letting $\xi_{1:R} = \{a_{1:R}, b_{1:R}, x_{1:R}, y_{1:R}\}$, the auxiliary $\varphi' s$ allow us to transform $\xi_{1:R}$ into $\xi'_{1:R+1}$ and vice-versa, where $\xi_{1:R}$ is not necessarily the same as $\xi'_{1:R}$. For simplicity, suppose that the resulting one-to-one transformation (which we denote by $T_{R\to R+1}$) has a Jacobian with determinant equal to 1, then

$$\begin{aligned}
\frac{\pi(R+1)}{\pi(R)} &= \int\int \frac{\pi\left(d\xi'_{1:R+1}, R+1\right)\varphi_{R+1\to R}(du')}{\pi(R)} \\
&= \int\int \frac{\pi\left(\xi'_{1:R+1}, R+1\right)\varphi_{R+1\to R}(u')}{\pi(R)\,\pi(\xi_{1:R} \mid R)\,\varphi_{R\to R+1}(u)} \times \pi(d\xi_{1:R} \mid R)\,\varphi_{R\to R+1}(du) \\
&= \mathbb{E}\left[\frac{\pi\left(\xi'_{1:R+1}, R+1\right)\varphi_{R+1\to R}(u')}{\pi(\xi_{1:R}, R)\,\varphi_{R\to R+1}(u)}\right],
\end{aligned}$$

the above expectation is taken on the variables $\xi_{1:R} \sim \pi(d\xi_{1:R} \mid R-1)$, $u \sim \varphi_{R\to R+1}$, and where $\left(\xi'_{1:R+1}, u'\right) = T_{R\to R+1}(\xi_{1:R}, u)$. The ratio inside the previous expectation is exactly the ratio used in the RJMCMC algorithm, which serves as an estimate for the ratio of the "ideal" unfeasible scenario in which we could explore the posterior of R without the need of $\xi_{1:R}$.

Annealing Moves

The full algorithm requires the introduction of the upwards and downwards annealed moves. For this, we require the introduction of auxiliary distributions, the first $\overrightarrow{\varphi}(\cdot \mid \mathcal{R})$ is for generating a new recombination event, i.e. for moving upwards; whereas the second $\overleftarrow{\varphi}(\cdot \mid \mathcal{R})$ is for selecting an index when deleting a recombination event. Moreover, these two distributions might also depend on $\rho, \delta, \theta, \mathcal{T}$ and/or $\mathcal{D}$. Simple choices are the prior of a single recombination event (a, b, x, y) for $\overrightarrow{\varphi}$ (described above), and a uniform distribution on $\{1, \ldots, R\}$ for $\overleftarrow{\varphi}$ assuming there are R active recombination events. We now present the upwards and downwards moves.

Despite the previous algorithms being very similar, they have important differences. For instance, the upwards move requires as input a newly created recombination event, whereas for the downwards this requirement is replaced with the need for an index. Also, even though the sequence $\{\gamma_t\}_{t=0}^{T}$ is required in both cases, the construction of η_t is different depending on the type of move. We want to stress that this construction is essential for obtaining a valid algorithm, otherwise the procedure could target something different to the desired posterior. A simple choice for the sequence if $\gamma_t = t/T$, which implies $\gamma_{T-t} = 1 - \gamma_t$.

Finally, we emphasise that in both algorithms, the variables $(\mathcal{R}, E_*)$ can be obtained from $\left(\bar{\mathcal{R}}, J_*\right)$ and viceversa; this means that ratios of the form η_{t+1}/η_t can be solely expressed in terms of $(\mathcal{R}, E_*)$ or $\left(\bar{\mathcal{R}}, J_*\right)$. Similarly, when perturbing $(\mathcal{R}, E_*)$ using a MCMC process we are implicitly perturbing the corresponding $\left(\bar{\mathcal{R}}, J_*\right)$. In addition to this, the outputs are not that different since one set of variables can be obtained from the other. In both algorithms we chose to work with $(\mathcal{R}, E_*)$ and only compute the corresponding pair $\left(\bar{\mathcal{R}}, J_*\right)$ when needed.

Algorithm 2 Upwards Annealed Move

NOTATION: Let $E_r = (a_r, b_r, x_r, y_r)$ denote the r-th recombination event, and $E_{1:r} = (a_{1:r}, b_{1:r}, x_{1:r}, y_{1:r})$.
REQUIRES: Sequence $\{\gamma_t\}_{t=0}^{T}$ such that $0 = \gamma_0 < \gamma_1 < \cdots < \gamma_T = 1$.
INPUT:

- Full set of recombination events $\mathcal{R} = (R, E_{1:R})$ for $R \geq 0$.
- Recombination event to be added $E_* = (a_*, b_*, x_*, y_*)$ proposed from $\overrightarrow{\varphi}(\cdot \mid \mathcal{R})$.

OUTPUT: r_T and $\left(\bar{\mathcal{R}}, J_*\right)$.

1. Combine $E_{1:R}$ and E_* in order to obtain $\bar{E}_{1:R+1}$, and keep track of the index $J_* \in \{1, \ldots, R+1\}$ denoting the position of E_* inside $\bar{E}_{1:R+1}$.
2. Letting $\bar{\mathcal{R}} = \left(R+1, \bar{E}_{1:R+1}\right)$, notice that $\left(\bar{\mathcal{R}}, J_*\right)$ has a one-to-one correspondence with $(\mathcal{R}, E_*)$.
3. Compute:

$$r_1 = \frac{\eta_1(\mathcal{R}, E_*)}{\eta_0(\mathcal{R}, E_*)},$$

   where, for each $t \in \{0, \ldots, T\}$,

$$\eta_t(\mathcal{R}, E_*) = \left[\tilde{\pi}\left(\bar{\mathcal{R}}\right) \overleftarrow{\varphi}\left(J_* \mid \bar{\mathcal{R}}\right)\right]^{\gamma_t} \left[\tilde{\pi}(\mathcal{R}) \overrightarrow{\varphi}(E_* \mid \mathcal{R})\right]^{1-\gamma_t}.$$

4. For $t = 1, \ldots, T-1$:
   (a) Leaving R fixed, perturb $(\mathcal{R}, E_*)$ performing one (or several) MCMC step(s) targeting the density η_t. This corresponds also to a perturbation of $\left(\bar{\mathcal{R}}, J_*\right)$ (with R fixed) due to the one-to-one relationship.
   (b) Compute, using the perturbed versions of $\left(\bar{\mathcal{R}}, J_*\right)$ and $(\mathcal{R}, E_*)$,

$$r_{t+1} = r_t \times \frac{\eta_{t+1}(\mathcal{R}, E_*)}{\eta_t(\mathcal{R}, E_*)}.$$

5. Return r_T, and the most recent value of $\left(\bar{\mathcal{R}}, J_*\right)$.

Algorithm 3 Downwards Annealed Move

NOTATION: Let $E_r = (a_r, b_r, x_r, y_r)$ denote the r-th recombination event, and $E_{1:r} = (a_{1:r}, b_{1:r}, x_{1:r}, y_{1:r})$.
REQUIRES: Sequence $\{\gamma_t\}_{t=0}^{T}$ such that $0 = \gamma_0 < \gamma_1 < \cdots < \gamma_T = 1$.
INPUT:

- Full set of recombination events $\bar{\mathcal{R}} = \left(R, \bar{E}_{1:R}\right)$ for $R \geq 1$.
- Index $J_* \in \{1, \ldots, R\}$, proposed from $\overleftarrow{\varphi}\left(\cdot \mid \bar{\mathcal{R}}\right)$, denoting the recombination event to be deleted.

OUTPUT: r_T and $(\mathcal{R}, E_*)$.

1. Delete the recombination event with index J_* from $\bar{E}_{1:R}$ to obtain $E_{1:R-1}$, denoting by $E_* = \bar{E}_{J_*}$.
2. Letting $\mathcal{R} = \left(R-1, \bar{E}_{1:R-1}\right)$, notice that $(\mathcal{R}, E_*)$ has a one-to-one correspondence with $\left(\bar{\mathcal{R}}, J_*\right)$.
3. Compute:

$$r_1 = \frac{\eta_1\left(\mathcal{R}, E_*\right)}{\eta_0\left(\mathcal{R}, E_*\right)},$$

   where, for each $t \in \{0, \ldots, T\}$,

$$\eta_t\left(\mathcal{R}, E_*\right) = \left[\tilde{\pi}\left(\bar{\mathcal{R}}\right) \overleftarrow{\varphi}\left(J_* \mid \bar{\mathcal{R}}\right)\right]^{\gamma_{T-t}} \left[\tilde{\pi}\left(\mathcal{R}\right) \overrightarrow{\varphi}\left(E_* \mid \mathcal{R}\right)\right]^{1-\gamma_{T-t}}.$$

4. For $t = 1, \ldots, T-1$:
   (a) Leaving R fixed, perturb $(\mathcal{R}, E_*)$ performing one (or several) MCMC step(s) targeting the density η_t. This corresponds also to a perturbation of $\left(\bar{\mathcal{R}}, J_*\right)$ (with R fixed) due to the one-to-one relationship.
   (b) Compute, using the perturbed versions of $\left(\bar{\mathcal{R}}, J_*\right)$ and $(\mathcal{R}, E_*)$,

$$r_{t+1} = r_t \times \frac{\eta_{t+1}\left(\mathcal{R}, E_*\right)}{\eta_t\left(\mathcal{R}, E_*\right)}.$$

5. Return r_T, and the most recent value of $(\mathcal{R}, E_*)$.