Non-Rooted Start of Evolution in GenPhyloData

Edvin von Platen

Bachelor’s Thesis in Computer Science
at Stockholm University, Sweden, 2020

Non-Rooted Start of Evolution in GenPhyloData

Edvin von Platen

Bachelor’s Thesis in Computer Science (15 ECTS credits)
Bachelor’s Programme in Computer Science
Stockholm University year 2020
Supervisor at the Department of Mathematics was Lars Arvestad
Examiner was Chun-Biu Li

Department of Mathematics
Stockholm University
SE-106 91 Stockholm, Sweden

Abstract

Phylogenetics is the study of evolutionary relationships and the history of organisms. A phylogenetic tree is a representation of hypothesized evolutionary relationships. By embedding a phylogenetic gene tree into a species tree one creates a reconciliation, to explain the evolution of the genes using the known species tree. Most software for simulating gene trees, given a species tree, begins the simulation at the root of the species tree. However, in nature it is not always so. We implement non-rooted simulation start in GenPhyloData and evaluate how capable the JPrIME-DLRS software is at inferring and reconciling gene trees generated using non-rooted start. Features implemented include the explicit selection of vertex for evolution start, uniform selection of a vertex over the timespan of the species tree, specification of a timespan over the species tree to select uniformly from, and the possibility to exclude the root of the species tree from the possible vertices. We found that our non-rooted start implementation in GenPhyloData is sound and that JPrIME-DLRS, in general, performs poorly on non-rooted start gene trees compared with trees generated using a rooted start. JPrIME-DLRS also appears incapable of inferring and reconciling the non-rooted start gene trees. However our implementation can be used as a reference and for the evaluation of software incorporating lateral gene transfer events, contributing to more realistic phylogenetic simulations.

Sammanfattning

Icke-rotad evolutionsstart i GenPhyloData

Fylogeni är läran om organismers evolutionära släktband och historia. Ett fylogenetiskt träd representerar organismers hypotestiska evolutionära släktband. Genom att bädda in ett fylogenetiskt träd i ett artträd skapas en rekonsiliering, vars syfte är att förklara geners evolution med hjälp av det kända artträdet. Det har observerats att dagens mjukvara för simulering av genträd, givet ett artträd, börjar simuleringen vid artträdets rot, men det har visat sig att detta antagande inte alltid gäller i naturen. Vi har implementerat icke-rotad simulering-start i GenPhyloData och utvärderat hur programmet JPrIME-DLRS klarar av att inferera och rekonsiliera genträd simulerade med icke-rotad start. Implementerade funktioner inkluderar: explicit val av nod för evolutionsstart, likformigt val av startnod över artträdets tidsspann, specifiering av ett tidsspann som startnod väljs ifrån och ett val att exkludera artträdets rotnod från möjliga startnoder. Våra resultat visar att implementationen av icke-rotad start i GenPhyloData fungerar väl och att JPrIME-DLRS inte klarar av att inferera eller rekonsiliera genträd simulerade med icke-rotad start. Vår implementation av icke-rotad start kan användas som referens och för att evaluera mjukvara som implementerar lateral gen överföring, vilket kan bidra till mer realistisk fylogenetisk simulering.

Contents

1 Background, Problem and Purpose

Phylogenetics is the study of evolutionary relationships and the history of organisms. Phylogenetic trees are used to represent the hypothesized evolutionary relationships between organisms. For example, a species tree represents the evolutionary relationships between species, while a gene tree represents the evolutionary relationships between genes.

While the phylogenetics of species are known in general, the evolution and relationships of the species genes are not. To explain the evolution of genes using the known species evolution, reconciled evolution is used. Reconciled evolution is the embedding of a phylogenetic gene tree into a species tree [1]. Figure 1 visualizes one such embedding, we say that the lineages (edges) of the guest tree evolves over the lineages of the species tree.

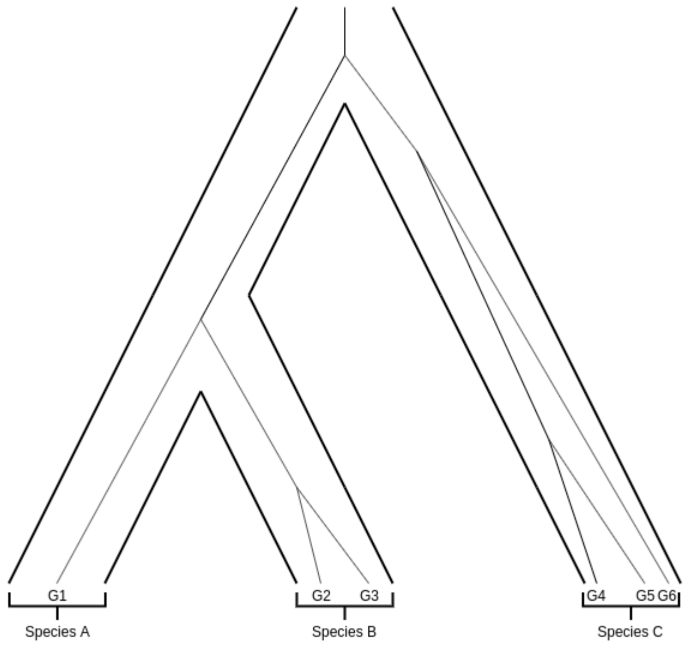


Figure 1: The embedding of a gene tree (guest tree, thin tree) of six genes within a species tree (host tree, thick tree) of three species. We also say that the gene tree has evolved over the species tree.

Software is used for the simulation of reconciled evolution. However, it has been observed that the software always starts the simulation at the root of the species tree, ignoring that in nature it is possible for a gene to appear later in the tree and by lateral gene transfer spread to other branches.

GenPhyloData [2] is a set of the three tools – HostTreeGen, GuestTreeGen, and BranchRelaxer, used for the simulation of realistic phylogenetic trees and is a part of the JPrIME [3] software suite for phylogenetics. HostTreeGen is used for simulating species trees which can then be used as input for the GuestTreeGen tool. GuestTreeGen takes a species tree as input and generates a gene tree evolving over the species tree.

We present a non-rooted start of evolution implementation for GuestTreeGen. Users of GuestTreeGen can explicitly select evolution start vertex, randomize the vertex, or select a time interval of the species tree over which the start will be randomized. With non-rooted start of evolution in GuestTreeGen users can simulate more realistic gene trees.

The non-rooted start implementation is then used for evaluating the JPrIME-DLRS phylogenetic software. We compare how capable JPrIME-DLRS is at inferring and reconciling gene trees generated using non-rooted start and rooted start.

2 Theory

To evaluate our implementation we make full use of the GenPhyloData software suite, the sequence generation software Seq-Gen [4], the PrIME-DLRS tool, and the visual Markov chain Monte Carlo (VMCMC) [5] tool. Following is an overview of the tools used and the theory they are built upon.

2.1 GenPhyloData

The GenPhyloData software suite is available at Github [3]. The three tools in the GenPhyloData software suite are,

- HostTreeGen,
- GuestTreeGen,
- BranchRelaxer.

2.1.1 HostTreeGen

The HostTreeGen tool uses a birth-death process to generate a bifurcating host tree over a specified time interval [2].

The birth-death process is a continuous-time Markov chain that can transition from state n only to state $n-1$ or $n+1$ [6]. In a birth-death process, the time until a transition from state n to state $n+1$ is $Exp(\lambda_n)$ distributed, here λ_n is known as the *rate* of the transition. Correspondingly, the time until a transition from state n to state $n-1$ is $Exp(\mu_n)$ distributed. A transition from state n to $n+1$ known as a "birth" and to $n-1$ as a "death".

The host tree generated by HostTreeGen starts as a single lineage at the specified interval start. A birth event splits the lineage into two separate child lineages which evolve independently of each other and all other lineages [2], here each child linage is a new separate birth-death process. In the event of a death in the birth-death process for a lineage, the process is stopped, this means that for a lineage to survive until the interval end there must be no death events. HostTreeGen use the same rates λ and μ for births and deaths for all lineages, i.e. $\lambda_i = \lambda_j \ \forall i, j$ and $\mu_i = \mu_j \ \forall i, j$.

We can view HostTreeGen as a single birth-death process where being in state n means that there are n active lineages that die and gives births independently. This means that if there are n active lineages the rate of a birth event is $n \cdot \lambda$ and a death event $n \cdot \mu$, see Figure 2.

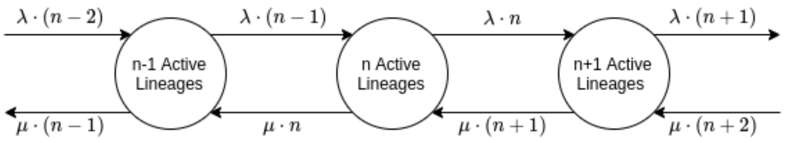


Figure 2: The number of active lineages during a HostTreeGen run can be seen as a single birth-death process. The transition rates grow and decline linearly as each lineage evolves independently of others.

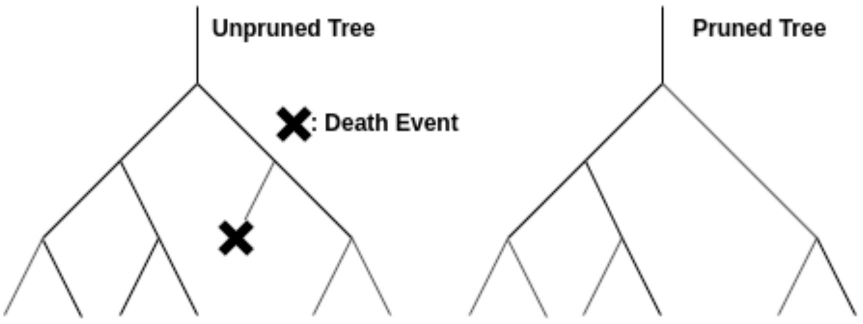


Figure 3: Example of an unpruned tree and its pruned version. HostTreeGen outputs both the unpruned tree and the pruned, the pruned version of the tree is created by removing all lineages (edges) with a death event.

Lineages that do not make it to the interval end are pruned away. Pruning an edge means that it is removed from the tree, see Figure 3 for an example of am unpruned tree and its pruned version. The output consists of the unpruned tree, the pruned tree, and auxiliary information files.

2.1.2 GuestTreeGen

The GuestTreeGen tool takes a bifurcating, dated, and clock-like host tree and generates a dated guest tree over the host tree, using either the duplication-loss, DL, model or the duplication-loss-transfer, DLT, model [2].

The DL model (or process) is similar to the birth-death process of HostTreeGen but also features deterministic duplication events. It can be described as follows, beginning at the stem of the host tree S, the DL process starts with a single guest lineage – a guest lineage is a lineage (edge) over one of the lineages of the host tree. The DL process takes two parameters, the duplication rate λ and the loss rate μ (just as HostTreeGen takes a speciation and death rate), starting at the stem the DL process then continues independently over the edges of S. A duplication event causes the guest lineage to split into two lineages, both evolving separately and independently over the host tree lineages. A loss event causes the process to stop. A lineage reaching a vertex x of the host tree causes the lineage to split in two, one evolving over the left child of x and the other over the right child. When a lineage reaches a leaf of S the process stops.

The DLT model is identical to the DL model except that it also takes a transfer rate τ, the rate of lateral gene transfer events. A lateral gene transfer event is when a gene transfers to another species by means other than reproduction. If a transfer event occurs, the guest lineage will split in two, one continuing its evolution over the current host edge, while the other is instantly transferred to another, uniformly selected, separate edge in S, where it continues independently.

The output consists of the unpruned and pruned reconciled guest tree, a file describing its reconciliation with the host tree, and other auxiliary information files.

2.1.3 BranchRelaxer

The BranchRelaxer tool takes a guest tree and applies a relaxed evolutionary clock model to it [2]. The guest trees branch lengths represent the number of substitutions on average until the next speciation. A substitution is a mutation that changes a base into another, for example $ACGT \rightarrow ACCT$ where G was changed into C.

The molecular clock model is a tool for finding the time of speciation of two species. For example, given a fixed mutation rate for nucleotides, one can observe the differences in the two species DNA and calculate the number of substitutions required to make them identical. The number of substitutions and their rate is then used to deduce the speciation time of the two species. The relaxed molecular clock model does not assume fixed mutation rates over lineages, instead, the rate can vary.

BranchRelaxer multiplies the edge times of the guest tree to simulate the relaxed molecular clock model [2]. We use an uncorrelated *iid* model, where each edge is given a substitution rate modelled after an gamma variable – That the model is *iid* means that the random variables modelling the substitution rates are *independent* and *identically distributed*; log-normal and exponential distributions are also supported.

2.2 Seq-Gen

The Seq-Gen tool uses models of substitution processes to simulate the evolution of nucleotide sequences along a phylogeny [4], for example a relaxed guest tree from BranchRelaxer. Seq-Gen traverses (evolves) over the lineages of the phylogenetic tree using a Markov chain with transition matrix P, a 4×4 matrix with substitution probabilities for the four nucleotides (A,C,G,T).

$$P = \begin{pmatrix} P_{A,A} & P_{A,C} & P_{A,G} & P_{A,T} \\ P_{C,A} & P_{C,C} & P_{C,G} & P_{C,T} \\ P_{G,A} & P_{G,C} & P_{G,G} & P_{G,T} \\ P_{T,A} & P_{T,C} & P_{T,G} & P_{T,T} \end{pmatrix}$$

At each substitution site with state i, P is used to simulate a substitution $P_{i,j}$ – a substitution from i to j.

We use the Hasegawa, Kishino, and Yano (HKY) model [7]. The HKY model assumes that evolution is identical and independent at each substitution site and along all lineages. HKY also allows the four nucleotides to have varying transition probabilities [4]. The output consists of the simulated multiple sequence alignment.

2.3 Markov Chain Monte Carlo and the Metropolis-Hastings Algorithm

For many problems and applications one wants to sample some probability distribution. However, it is often very difficult to sample directly from the distribution, especially in higher dimensions.

Markov chain Monte Carlo (MCMC) is a set of techniques for sampling from such difficult distributions using Markov chains [8].

The main idea of MCMC is to construct a Markov chain whose stationary distribution equals the one from which we want to sample. By simulating this Markov chain it can then be used to estimate the stationary distribution. The Metropolis-Hastings algorithm is one technique used for generating a Markov chain with the required characteristics.

The generated Markov chain, with transition probabilities $P_{i,j}$ needs to be *ergodic* and *time reversible* [6].

That a Markov chain is ergodic means that it is irreducible and aperiodic. In an irreducible Markov chain each state is reachable from every other state. A state is said to have period $d \geq 1$ if $P^n_{i,i} = 0$, $\forall n \neq d$, i.e. the probability to return to state i in n steps is positive only if n is divisible by d, if $d = 1$ then the state is said to be aperiodic. A Markov chain is said to be aperiodic if all states are aperiodic.

Given an ergodic Markov chain with transition probabilities $P_{i,j}$ and stationary probabilities π_i. The Markov chain is said to be time reversible if

$$\pi_i P_{i,j} = \pi_j P_{j,i}.$$

The goal of the Metropolis-Hastings algorithm is to generate a time reversible Markov chain with stationary probabilities,

$$\pi(j) = \frac{b(j)}{B},\ j = 1, 2, 3, ...$$

where $B = \sum_j b(j)$. It is the normalization denominator B that is often difficult to calculate directly. With the Metropolis-Hastings algorithm we avoid it by the following construction.

Let Q be any specified irreducible Markov chain transition probability matrix. Let $\{X_n, n \geq 0\}$ be a Markov chain such that when $X_n = i$ generate a random variable Y, known as the *candidate* state, with $P(Y = j|X_n = i) = Q_{i,j},\ j = 1, 2, ...$ The distribution from which the candidate state is generated from is known as the *proposal* distribution. If $Y = j$ set

$$X_{n+1} = \begin{cases} j \text{ with probability } \alpha(i,j), \\ i \text{ with probability } 1 - \alpha(i,j). \end{cases}$$

$\alpha(i,j)$ is known as the *acceptance* probability, as $\alpha(i,j)$ decides if we should accept the transition to state j from i. This is a Markov chain with transition probabilities $P_{i,j}$,

$$P_{i,j} = \begin{cases} Q_{i,j}\alpha(i,j),\ j \neq i, \\ Q_{i,i} + \sum_{k \neq i} Q_{i,k}(1 - \alpha(i,k)). \end{cases}$$

The generated Markov chain is time reversible with the wanted stationary probabilities $\pi(j)$ if $\pi(i)P_{i,j} = \pi(j)P_{j,i},\ j \neq i$. Insert $P_{i,j} = Q_{i,j}\alpha(i,j)$ such that,

$$\pi(i)Q_{i,j}\alpha(i,j) = \pi(j)Q_{j,i}\alpha(j,i). \tag{1}$$

We can now choose the acceptance probability $\alpha(i,j)$ as

$$\alpha(i,j) = \min\left(\frac{\pi(j)Q_{j,i}}{\pi(i)Q_{i,j}}, 1\right)$$

for which both values (1) holds and shows that the generated Markov chain is time reversible and that the stationary probabilities $\pi(j)$ exists. Remember that since $\pi(j) = b(j)/B$ we get that

$$\alpha(i,j) = \min\left(\frac{b(j)Q_{j,i}}{b(i)Q_{i,j}}, 1\right).$$

$B = \sum_j b(j)$ is therefore not needed to define the generated Markov chain. Also, note that if the proposal distribution Q is symmetric, i.e. $Q_{i,j} = Q_{j,i}$, then the acceptance probability becomes,

$$\alpha(i,j) = \min\left(\frac{b(j)}{b(i)}, 1\right).$$

This mean that if $b(j) \geq b(i)$ the chain will transition to state j. Algorithm 1 shows the psuedocode for the Metropolis-Hastings algorithm as presented by Chib and Greenberg (1995) [8].

Algorithm 1 The Metropolis-Hastings algorithm.

```
1: let $x_0$ be the starting state and $m$ the number of iterations.
2: for $i = 0, 1, ..., m-1$ do
3:     Generate candidate state $y$ from the proposal distribution $Q_{x_i,y}$
4:     $\alpha(x_i, y) = min\left(\frac{\pi(y)Q_{y,x_i}}{\pi(x_i)Q_{x_i,y}}, 1\right)$
5:     Generate random number $z$ uniformly over $[0,1]$
6:     if $\alpha(x_i, y) \geq z$ then
7:         $x_{i+1} = y$
8:     else
9:         $x_{i+1} = x_i$
10:    end if
11: end for
12: return $\{x_0, x_1, .., x_m\}$
```

How to choose the proposal distribution depends on the application and is often not trivial. One factor to consider is the tail of the proposal distribution, with a large tail a larger spread of states are more likely to be proposed and thus more of the state space is possible explored, however too large of a tail and many very unlikely (maybe even impossible) states may be proposed which are then rejected (due to very low acceptance probability) making the Markov chain stay in the same state and thus does not explore the state space.

In Algorithm 1 all visited states are returned, this is known as the MCMC *trace*. However, in practice, often only every $i'th$, $i \in Z^+$ visited state is returned. This is mainly due to two reasons: 1) if we run the algorithm 10^6 iterations the full trace gives 10^6 samples, this amount of data can make the post-processing of the trace very slow or infeasible. 2) In the full trace each state is dependent on the previous state due to the algorithm traversing a Markov chain, by not sampling two following states we may reduce the dependency of the included states in the trace.

2.4 Duplication, Loss, Rates and Sequence Evolution – DLRS

We evaluate how capable the JPrIME-DLRS [9] software is at inferring and reconciling gene trees generated using non-rooted start.

DLRS is a probabilistic model for duplication events, loss events, rate heterogeneity, and sequence evolution and takes a species tree S, multiple sequence alignment D, and a relaxed gene tree G as input.

Let $r(e)$, $\ell(e)$ and $t(e)$ be functions for edge specific rate, length and time for each edge e in G, and Θ the different model parameters, for example μ and λ in the DL model, note that $r(e) = \frac{\ell(e)}{t(e)}$. JPrIME-DLRS then uses a MCMC framework to approximate the posterior probability (2) of the gene tree G, ℓ, and Θ given D, S, and t,

$$\begin{aligned} p(G,\ell,\Theta|D,S,t) &= \frac{p(G,\ell,\Theta,D,S,t)}{P(D,S,t)} \\ &= \frac{p(D,G,\ell|\Theta,S,t)p(\Theta,S,t)}{P(D|S,t)P(S,t)} \\ &= \frac{P(D|G,\ell,\Theta,S,t)p(G,\ell|\Theta,S,t)p(\Theta|S,t)P(S,t)}{P(D|S,t)P(S,t)} \\ &= \frac{P(D|G,\ell)p(G,\ell|\Theta,S,t)p(\Theta)}{P(D|S,t)}. \end{aligned} \tag{2}$$

Here, the model parameters Θ are independent of S and t, and D is decided exactly given G and ℓ.

JPrIME-DLRS use the Metropolis-Hastings algorithm to generate a Markov chain with states consisting of guest trees G, edge lengths ℓ, and the model parameters Θ which are all assigned uniform prior distributions [10]. When a candidate state is to be proposed, only one of the state parameters $\{G, \ell, \Theta\}$ (and only one of the parameters in Θ) is updated, let $\{G', \ell', \Theta'\}$ denote the updated state parameters, all of the parameters have the same probability of being updated. This means that the proposal distribution is symmetric and the acceptance probability is thus decided by the ratio of the probability density for the candidate state $p(G', \ell', \Theta'|D, S, t)$ and the current $p(G, \ell, \Theta|D, S, t)$ [11].

JPrIME-DLRS also supports several different substitution models [9]. We use the Jukes and Cantor (JC69) DNA substitution model [12]. The JC69 model assumes uniform mutation rates and base frequencies.

Output is the MCMC trace from the posterior distribution (2).

2.5 VMCMC

The visual Markov chain Monte Carlo (VMCMC) software is used to analyse phylogenetic MCMC [5].

If the MCMC simulation of the posterior (2) is to be correct, the Markov process needs to converge to the stationary distribution π, which it will do in the limit. This means that before the Markov chain has converged the trace will have a different distribution. Removing the first fraction of samples from the trace is known as *burn-in* and can make the trace better approximate the posterior.

VMCMC can be used to visualize the trace, parameter estimation, and burn-in estimation. We use VMCMC to visually inspect the JPrIME-DLRS traces to see after how many samples it appears that the Markov chain has converged, apply burn-in to the trace, and to generate trace statistics in the JSON format.

2.6 Robinson-Foulds Distance

The Robinson-Foulds, RF, distance is a metric for labelled phylogenetic trees [13]. There are two different RF distances, one for rooted trees, and one for unrooted trees. While the trees in this study are rooted we will use the unrooted RF distance, this is due to the relatively small size of the trees. For smaller trees the rooted RF distance may be very large, but since the trees are small the actual distance between them is small. The unrooted RF distance will thus provide a more sensible metric for smaller trees.

We will now define the unrooted RF distance. The removal of any edge in a tree $G = (V, E)$ splits the vertex set V in two disjoint sets. We denote a split by,

$$v_1 v_2 v_3 ... v_k | v_{k+1} ... v_n \text{ where } n > k \text{ and } |V| = n.$$

Note that $v_1|v_2$ is seen as the same split as $v_2|v_1$. We say that a split is *trivial* if either of the two sets in the split only contains one leaf.

Let $T_1 = G(V_1, E_1)$ and $T_2 = G(V_2, E_2)$ be two labelled phylogenetic trees, and S_1, S_2 be the set of all possible splits in T_1 and T_2. The RF distance is then defined as the sum of unique splits for both trees,

$$RF = |S_1 \setminus S_2| + |S_2 \setminus S_1|.$$

Here $\setminus$ is the set difference operator. For two sets A and B we define the difference $A \setminus B$ as the set,

$$A \setminus B = \{a; \text{ such that } a \in A \text{ and } a \notin B\}.$$

I.e. all elements of A that are not in B. And $|\cdot|$ is the cardinality (number of elements) of a set.

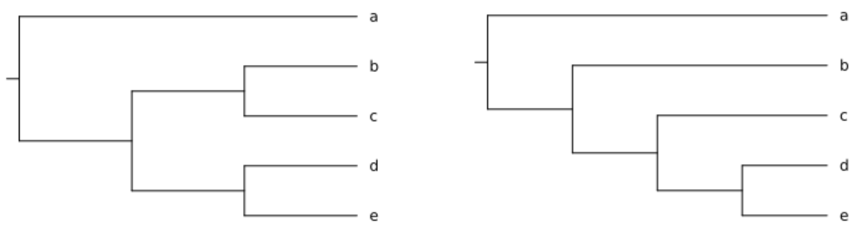


Figure 4: Two trees, T1 and T2, with an unrooted Robinson-Foulds distance of 2.

To illustrate we give an example RF calculation. We calculate the RF distance between the two trees in Figure 4. Both trees have the leaf set $V = \{a, b, c, d, e\}$ and possible non-trivial splits for the two trees are

$$\text{T1 splits: } S_1 = \{\ abc|de,\ ade|bc\}.$$
$$\text{T2 splits: } S_2 = \{\ abc|de,\ ab|cde\}.$$

We get a RF distance of,

$$|S_1 \setminus S_2| + |S_2 \setminus S_1| = |\{ade|bc\}| + |\{ab|cde\}| = 2.$$

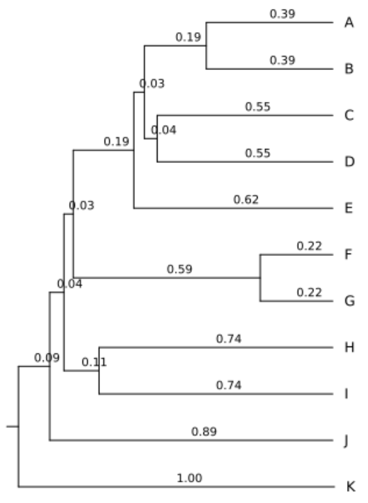
The maximum RF distance becomes larger the bigger trees are used. As we generate trees of different sizes but still want to compare the distance of different pairs of trees we can normalize the RF distance. The RF distance is normalized by dividing it by the maximum possible RF distance for the two trees. The normalized RF distance takes a value between 0 and 1. A normalized RF distance of 0 means that the two trees are isomorphic, and 1 that they are very dissimilar. We use the ETE 3 Toolkit [14] to calculate the unrooted RF distance between trees present in the JPrIME-DLRS traces.

3 Method

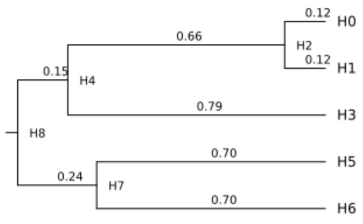
The JPrIME github home repository [3] was forked to [15] to avoid changes to the main branch.

An experiment using JPrIME-DLRS was performed to evaluate JPrIME-DLRS reconciliation capabilities for non-rooted start. We used the Snakemake workflow management system [16] to automate the steps below. All data used in the study, Snakemake files and, the custom scripts used to parse, plot and, evaluate JSON data are publicly available at GitHub [17].

A yeast species tree of 11 leaves and one simulated synthetic species tree generated by HostTreeGen of 5 leaves was used to test the non-rooted start implementation, see Figure 5.



(a) Host tree for real yeast genomes.



(b) Host tree generated with HostTreeGen.

Figure 5: Host trees used for generating guest trees with GuestTreeGen and in JPrIME-DLRS.

The experiment was performed through the following steps:

1. Use GuestTreeGen to generate guest tree G over the host tree S.
2. Use BranchRelaxer to relax the pruned guest tree G.
3. Use Seq-Gen to simulate sequence data D for the relaxed tree.
4. Use S, pruned G, and D as arguments to JPrIME-DLRS to generate a MCMC trace.

5. Use VMCMC to generate JPrIME-DLRS trace statistics in JSON format.
6. Use custom python scripts [17] to parse, plot and evaluate JSON data.

The following options were used for the tools required in the previous study implementation outline.

- HostTreeGen

  dup_rate=0.4. loss_rate=0.1.

- GuestTreeGen

  -nonrootrandomstart. -lowertime=0.4. -min 5. -max 15. dup_rate=0.4. loss_rate=0.1. trans_rate=0.0. For non-root start yeast trees.

  -min 5. -max 15. dup_rate=0.4. loss_rate=0.1. trans_rate=0.0. For rooted start yeast trees.

  -nonrootrandomstart. -min 5. -max 15. dup_rate=0.4. loss_rate=0.1. trans_rate=0.0. For non-root start synthetic trees.

  -min 5. -max 15. dup_rate=0.4. loss_rate=0.1. trans_rate=0.0. For rooted start synthetic trees.

- BranchRelaxer

  dist=IIDGamma. k=1.5. theta = 0.25.

- Seq-Gen

  model=HKY. seq_len=1000.

- DLRS

  -dmin=15. -dmax=200. -i=1 000 000. -t=100. -sm=JC69.

- VMCMC

  -b=2 000. -p. When generating tree statistics.

  -b=2 000. -s. When generating inferred duplication rates statistics.

3.1 Comparison of Rooted and Non-rooted Start for JPrIME-DLRS

For both the synthetic and yeast host tree we compare the JPrIME-DLRS traces for both the non-rooted and rooted start runs. Custom python scripts [17] are used to generate statistics and figures from JSON formatted files generated with VMCMC on JPrIME-DLRS traces.

The non-rooted and rooted start traces for the synthetic and yeast host tree are compared through visual inspection and a statistical test. The visual inspection consists of histograms of the following data:

- Maximum tree posterior probability for each trace.
- Mean duplication rates for each trace.
- Expected normalised Robinson-Foulds distance $E(RF_{norm}(T))$.

The expected normalised Robinson-Foulds distance is a way to evaluate the similarity of the guest trees appearing in a JPrIME-DLRS posterior to the input guest tree. It is calculated by,

$$E(RF_{norm}(T)) = \sum_{T \in Posterior} RF_{norm}(T)P(T),$$

where $P(T)$ is the probability of T in the posterior and $RF_{norm}(T)$ is the normalised RF distance between the tree $T \in Posterior$ and the input guest tree.

To supplement the visual inspection of the trace data a Mann-Whitney-U test is performed to test the Hypothesis (3). The Mann-Whitney-U test is a non-parametric statistical test used for testing if observations from two sample groups share the same distribution [18]. We use a non-parametric test because the data does not seem to follow a normal distribution.

$$\begin{aligned} H_0 &: \text{ Rooted and non-rooted start share the same distribution.} \\ H_1 &: \text{ Rooted and non-rooted start does not share distribution.} \end{aligned} \tag{3}$$

3.2 Study Limitations

The DL model was used in GuestTreeGen for all guest trees used in the JPrIME-DLRS evaluation. While it would have been more interesting, from a realistic and applied perspective, to use the

DLT model and evaluate the JPrIME-DLTRS [19] tool, a part of JPrIME which incorporates LGT events to JPrIME-DLRS, instead of JPrIME-DLRS. We were unable to get JPrIME-DLTRS runs to converge so JPrIME-DLRS was used instead.

3.3 Implementation of Non-rooted Evolution start

The implementation of non-rooted evolution start was done by altering the Java source code of GenPhyloData, the new features are available at GitHub [15]. Several different options for the non-root start are accessible to users through the following command-line options:

-nonroot <integer>, allows the user to specify a JPrIME ID over which the host tree edge into a vertex the evolution should start. JPrIME ID is a label that can be assigned to a vertex in the extended Newick tree format used by the JPrIME software suite.

-randomstart, a starting edge is chosen by: (i) a time t is uniformly selected over the dated species tree timespan interval. (ii) The edge with start vertex closest to t is selected as the simulation start, the start vertex is the vertex of the edge with the lowest distance to the root of the tree. (iii) If two or more edges start vertices have the same distance to t, then the starting edge is selected at random using a uniform distribution over the edges.

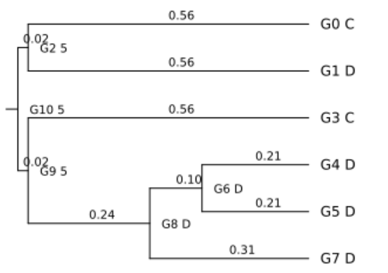
-nonrootrandomstart, identical to -randomstart except that the stem time of the species tree is excluded from the interval which t is selected from. The stem time is the time the evolution has elapsed before reaching the root node. This forces the guest tree to not start from the root of the host tree.

-uppertime <decimal> and **-lowertime <decimal>** allows the user to specify the uniform time interval from which t is selected from.

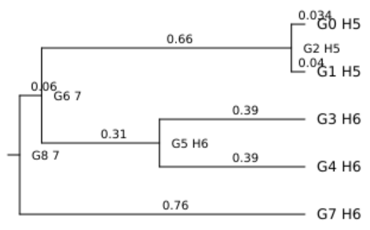
4 Results

We will first demonstrate the non-root start implementation in GenPhyloData and then present the results of the JPrIME-DLRS evaluation. Figure 5 shows the two host trees with their respective branch lengths used in tests and for the evaluation.

During the evaluation no bugs were encountered when using the non-root start implementation. Figure 6 shows two example guest trees generated by GuestTreeGen using the *-nonrootrandomstart* option over the two host trees in Figure 5.



(a) Guest tree generated over yeast genomes host tree.



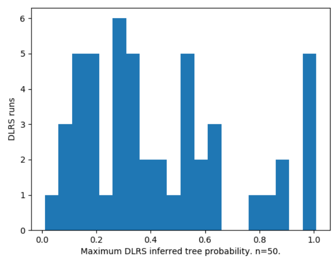
(b) Guest tree generated over five leaf host tree.

Figure 6: Two guest trees generated over the host trees in Figure 5 using the -nonrootrandomstart option for GuestTreeGen.

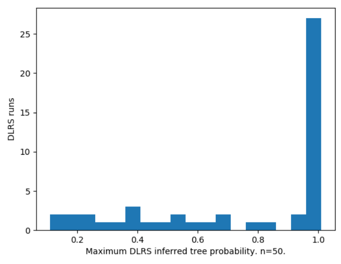
4.1 JPrIME-DLRS Evaluation

We have explored how capable JPrIME-DLRS is at inferring and reconciling guest trees generated using non-rooted start. For each of the two host trees in Figure 5 JPrIME-DLRS was run on
100 guest trees, where half of the guest trees were generated using non-rooted start. JPrIME-DLRS was run with 10^6 iterations with a sample taken every $100th$ iteration. When generating JSON files from the MCMC trace with VMCMC a burn-in of 20 % was used, i.e. the first 2000 samples were removed to allow the Markov chain to converge. In total, the statistics generated was based on 8000 samples from each trace.

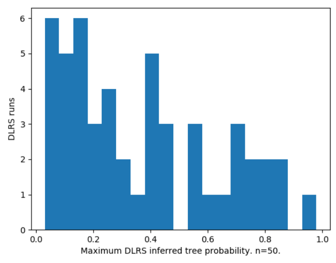
Figure 7 shows the maximum tree posterior probability achieved by JPrIME-DLRS in each run for both host trees and using rooted and non-rooted evolution start. JPrIME-DLRS can find a reconciliation of high probability for the rooted start in a majority of the runs. However, using the non-rooted start guest trees JPrIME-DLRS becomes significantly less capable of finding a high probability reconciliation.



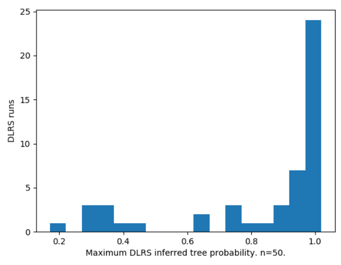
(a) Yeast host tree. Non-rooted start.



(b) Yeast host tree. Rooted start.



(c) Synthetic host tree. Non-rooted start.



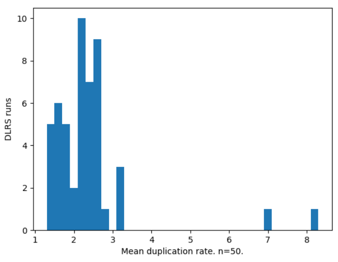
(d) Synthethic host tree. Rooted start.

Figure 7: Posterior probabilities of the trees with the maximum probability in the JPrIME-DLRS posterior for each run. For both rooted and non-root GuestTreeGen start. Using both host trees in Figure 5.

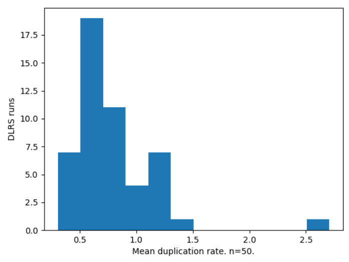
Figure 8 shows the inferred mean duplication rates from the JPrIME-DLRS trace for both host trees. For reference, the duplication rate used when generating all guest trees was 0.4. JPrIME-DLRS infers duplication rates close to the used rate for the guest trees using rooted start, especially for the yeast host tree which distribution is more concentrated around the known duplication rate. While for non-rooted guest trees the inferred duplication rates are not as precise.

Figure 9 shows the expected RF_{norm} distance for each JPrIME-DLRS posterior distribution. Both rooted start runs seem to have similar distributions, with a majority of the runs having an expected RF_{norm} distance close to 0. The two non-rooted start runs also have similar distributions, but with larger expected RF_{norm} values present compared to the rooted start posteriors.

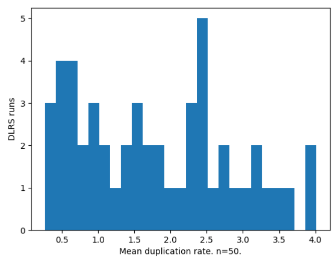
To confirm the visual inspection of the results in Figures 7, 8 and 9 we test the hypothesis (3). Rooted and non-rooted start sharing the same distribution means that JPrIME-DLRS is equally



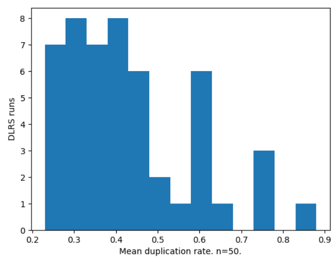
(a) Synthetic host tree. Non-rooted start.



(b) Synthetic host tree. Rooted start.



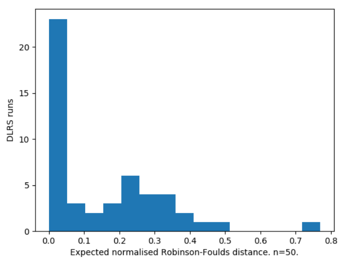
(c) Yeast host tree. Non-rooted start.



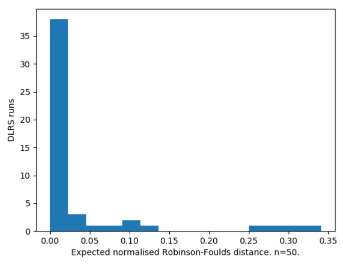
(d) Yeast host tree. Rooted start.

Figure 8: Mean duplication rates inferred by DLRS over n=50 runs for each subplot. For both rooted and non-root GuestTreeGen start. Using both host trees in Figure 5

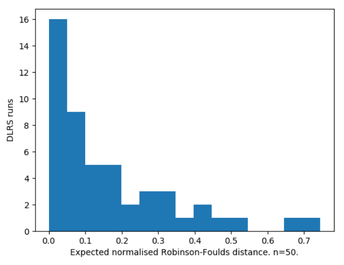
capable of inferring rooted and non-rooted start guest trees. We use the Mann-Whitney-U test to test the hypothesis. Table 1 show the calculated p-values and that we can dismiss H_0 with a strong statistically significant result for each of the six performed tests. The results of the tests are in line with the previous visual inspection.



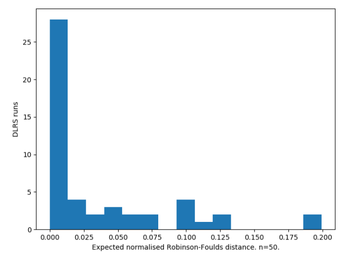
(a) Synthetic host tree. Non-rooted start.



(b) Synthetic host tree. Rooted start.



(c) Yeast host tree. Non-rooted start.



(d) Yeast host tree. Rooted start.

Figure 9: Expected RF_{norm} distance from each guest tree genereated by GuestTreeGen using both host trees in Figure 5, to each tree in the JPrIME-DLRS posterior in its corresponding run.

Table 1: Mann-Whitney-U test p-values for the hypothesis (3), rounded to three digits. The Mann-Whitney-U test was run on data inferred by JPrIME-DLRS. The tests was done for both the yeast and synthetic samples (rows), and for each of the three inspected statistics in Figures 7, 8 and 9 (columns). The two datasets used for each test was the corresponding rooted and non-rooted datasets.

	Mean Duplication Rate	**E(RF$_{\mathbf{norm}}$)**	**Posterior Probability**
Yeast	0.000	0.000	0.000
Synthetic	0.009	0.000	0.000

5 Discussion, Conclusions and Recommendations for Further Study

We start this chapter with a discussion of potential pitfalls when using the non-rooted start option in GenPhyloData and continue with a discussion of the JPrIME-DLRS evaluation results.

5.1 Non-Rooted Start Implementation

Even if the non-rooted evolution start implementation in GenPhyloData is sound there are some potential problems when using the feature. The biggest issue is a bias towards selecting a leaf-adjacent edge. As the generated guest trees are rooted and bifurcating they contain more leaves than internal nodes. Edges into leaf vertices can potentially cover a large time span of the species tree, causing a bias for the simulation to select an edge into a leaf as a starting point. To counteract the bias we recommend using the -lowertime $<$ *decimal* $>$ option to limit the number of possible leaves to select.

While it was not possible for this study, we recommend using the non-rooted start option together with lateral gene transfer. As a reason why say the evolution starts at an edge into a leaf of the species tree and the minimum amount of leaves is set to five for the guest tree. Since there are no speciation events during the evolution the simulation will rely solely on duplication events to generate the guest tree. There will thus be a clear bias towards lower leaf counts in the guest tree. However, with lateral gene transfer there can be LGT events, transferring a gene into another branch where there might be speciation events, forcing the branch to split and creating a larger tree.

5.2 JPrIME-DLRS Evaluation

As shown in the Results section, JPrIME-DLRS is in general not capable of converging to a single reconciled guest tree using non-rooted start. This can likely be explained by JPrIME-DLRS not supporting the DLT model causing us to generate the guest trees without lateral gene transfer events. As mentioned in the Method section, a more realistic and applied approach would have been to use reconciliation software supporting the DLT model, such as DLTRS. However, this does not make our results void, only less interesting from an applied perspective.

In the non-rooted start case, JPrIME-DLRS does, see Figure 7, not seem to perform better at converging to a single tree for the smaller synthetic host tree compared to the larger yeast tree. This is slightly surprising as the gene trees generated over the yeast tree were in general larger than the ones generated over the synthetic tree, partly to the usage of the *-lowertime* option. Larger trees mean a larger number of possible trees for reconciliation. One possible explanation is that JPrIME-DLRS performs worse the closer to the leaves the evolution starts. As *-lowertime* was used for the yeast tree, less of its guest trees started at its leaves. While there possibly was a bias towards starting into the leaves for the synthetic guest trees.

Figure 8 showed that JPrIME-DLRS infers the duplication rate best for the yeast host tree with rooted start. The inferred duplication rates are quite poor in other cases, especially for the non-rooted starts. That the results for the non-rooted start yeast tree are better than for the synthetic tree is likely due to the larger size of yeast tree compared with the synthetic tree. Due to the larger size, the guest trees are generated over a larger number of edges. Since the evolution over an edge is independent of the evolution over the other edges there is more data to infer the duplication rate from, which gives a more accurate estimation. This could also explain the, in comparison, poor inferred duplication rates for the synthetic rooted start trees. As the trees are smaller there are fewer edges for the guest tree to evolve over, resulting in less data to infer from.

As for the non-rooted start trees, the inferred duplication rates are in general far from the real value. This is not surprising, one reason being the one discussed above, as the evolution does not start in the root node the evolution will occur over a smaller tree and timespan. Another reason is that JPrIME-DLRS infers the duplication rate using the given host tree, but as the evolution did not start in the root node there is an incongruence between the trees.

When it comes to the expected RF_{norm} distance in Figure 9, both the non-rooted and rooted start posteriors have a large number of values close to 0. This is likely due to the size of the trees, as small trees are by default quite similar. However, as shown by the Mann-Whitney-U test, there is a statistically significant difference between the rooted and non-rooted distributions.

From the results of the visual data inspection and the Mann-Whitney-U tests it is clear that JPrIME-DLRS is more capable of reconciling guest trees generated using rooted start than non-rooted start.

It is not surprising that JPrIME-DLRS performs poorly on non-rooted start generated guest trees given that JPrIME-DLRS was designed for reconciliation of rooted start guest trees generated using the DL model.

5.3 Recommendations for Further Study

For further studies we recommend evaluating reconciliation software which supports the DLT model using our implementation of non-rooted evolution start. It would also be interesting to perform the study with increased computational capabilities as our study was limited in sample scope due to computational constraints.

References

[1] M. Goodman, J. Czelusniak, G. W. Moore, A. E. Romero-Herrera, and G. Matsuda, "Fitting the Gene Lineage into its Species Lineage, a Parsimony Strategy Illustrated by Cladograms Constructed from Globin Sequences," *Systematic Biology*, vol. 28, pp. 132–163, 06 1979.

[2] J. Sjöstrand, L. Arvestad, J. Lagergren, and B. Sennblad, "GenPhyloData: realistic simulation of gene family evolution," *BMC Bioinformatics*, vol. 14, p. 209, Jun 2013.

[3] J. Sjöstrand, J. Lagergren, L. Arvestad, and B. Sennblad, "JPrIME: Phylogenetics library in Java (version 0.3.7)." `https://github.com/arvestad/jprime/`. [Online; accessed 08-04-2019].

[4] A. Rambaut and N. C. Grass, "Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees," *Bioinformatics*, vol. 13, pp. 235–238, 06 1997.

[5] R. H. Ali, M. Bark, J. Miró, S. A. Muhammad, J. Sjöstrand, S. M. Zubair, R. M. Abbas, and L. Arvestad, "VMCMC: a graphical and statistical analysis tool for Markov chain Monte Carlo traces," *BMC Bioinformatics*, vol. 18, p. 97, Feb 2017.

[6] S. M. Ross, *Introduction to Probability Models*, vol. 10. Academic Press Inc, 2014.

[7] M. Hasegawa, H. Kishino, and T. Yano, "Dating of the Human-Ape Splitting by a Molecular Clock of Mitochondrial DNA," *Journal of Molecular Evolution*, vol. 22, pp. 160–174, 10 1985.

[8] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings Algorithm," *The American Statistician*, vol. 49:4, pp. 327–335, 11 1995.

[9] J. Sjöstrand, B. Sennblad, L. Arvestad, and J. Lagergren, "DLRS: gene tree evolution in light of a species tree," *Bioinformatics*, vol. 28, pp. 2994–2995, 09 2012.

[10] Å. Örjan, B. Sennblad, L. Arvestad, and J. Lagergren, "Simultaneous Bayesian gene tree reconstruction and reconciliation analysis," *National Academy of Sciences*, Mar 2009.

[11] O. Mahmudi, J. Sjöstrand, B. Sennblad, and J. Lagergren, "Genome-wide probabilistic reconciliation analysis across vertebrates," *BMC Bioinformatics*, vol. 14, 2013.

[12] T. Jukes and C. Cantor, "Evolution of protein molecules, Mammalian Protein Metabolism," vol. III, pp. 21–123, 1969.

[13] D. Robinson and L. Foulds, "Comparison of phylogenetic trees," *Mathematical Biosciences*, vol. 53, no. 1, pp. 131 – 147, 1981.

[14] J. HuertaCepas, F. Serra, and P. Bork, "ETE 3: Reconstruction, analysis and visualization of phylogenomic data," *Mol Biol Evol*, 2016.

[15] J. Sjöstrand, J. Lagergren, L. Arvestad, and B. Sennblad, "Jprime: Phylogenetics library in java (version 0.3.7)." `https://github.com/edvinvp/jprime/`. [Online; accessed 16-04-2019].

[16] J. Köster and S. Rahmann, "Snakemake - a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, pp. 2520–2522, 08 2012.

[17] von Platen, Edvin, "Phylogenetic Scripts." `https://github.com/edvinvp/Phylogenetic-Scripts`. [Online; Accessed 27-05-2019].

[18] S. E. Alm and T. Britton, *STOKASTIK Sannolikhetsteori och statistikteori med tillämpningar*, vol. 1. Liber AB, 2008.

[19] J. Sjöstrand, A. Tofigh, V. Daubin, L. Arvestad, B. Sennblad, and J. Lagergren, "A Bayesian Method for Analyzing Lateral Gene Transfer," *Systematic Biology*, vol. 63, pp. 409–420, 02 2014.