Time-sensitive reconstruction of joint evolutionary trees in a host-parasite model with host switches

Dissertation
for the award of the degree

"Doctor of Philosophy "
Division of Mathematics and Natural Sciences

of the Georg-August-Universität Göttingen
within the doctoral degree program
International Max Planck Research School for Genome Science
of the Georg-August-University School of Science (GAUSS)

Submitted by

Zsuzsanna Bősze
from Bonyhád, Hungary

Göttingen, 2024

Thesis Advisory Committee:

Prof. Dr. Anja Sturm
Institute for Mathematical Stochastics, Georg-August-Universität Göttingen

Prof. Dr. Burkhard Morgenstern
Department of Bioinformatics, Institute for Microbiology and Genetics, Georg-August-Universität Göttingen

Prof. Dr. Axel Munk
Institute for Mathematical Stochastics, Georg-August-Universität Göttingen

Members of the Examination Board:

Referee: Prof. Dr. Anja Sturm
Institute for Mathematical Stochastics, Georg-August-Universität Göttingen

2nd Referee: Prof. Dr. Christoph Bleidorn
Department for Animal Evolution and Biodiversity, Johann-Friedrich-Blumenbach Institute for Zoology and Anthropology, Georg-August-Universität Göttingen

Further Members of the Examination Board:

Prof. Dr. Burkhard Morgenstern
Department of Bioinformatics, Institute for Microbiology and Genetics, Georg-August-Universität Göttingen

Prof. Dr. Axel Munk
Institute for Mathematical Stochastics, Georg-August-Universität Göttingen

Dr. Johannes Söding
Computational Biology, Max Planck Institute for Multidisciplinary Sciences

Prof. Dr. Dominic Schuhmacher
Institute for Mathematical Stochastics, Georg-August-Universität Göttingen

Date of oral examination: October 22, 2024

Acknowledgements

I would like to express my gratitude to all those who supported me throughout my PhD journey. Without their support and encouragement, this thesis would not have been possible.

First and foremost, I am immensely grateful to my supervisor, Prof. Dr. Anja Sturm, not only for giving me the possibility to work on this interesting project, but also for her invaluable guidance and patience. Her expertise and encouragement were instrumental in shaping this research. I am especially thankful for her flexibility in accomodating the working conditions my personal circumstances required.

I would like to thank Prof. Dr. Burkhard Morgenstern and Prof. Dr. Axel Munk for being part of my thesis advisory committee. Their suggestions and insights have improved the quality of this work.

I would like to express my gratitude to Prof. Dr. Christoph Bleidorn for his detailed explanation of some biological concepts that were essential for writing this thesis. My thanks also go to him and Thilo Schulze for providing datasets for my newly developed algorithm.

Additionally, I am grateful to Prof. Dr. Christoph Bleidorn for agreeing to be the second reviewer of this thesis. I would also like to thank Dr. Johannes Söding and Prof. Dr. Dominic Schuhmacher for serving as members of my examination committee.

I would like to thank all of my colleagues at the Institute for Mathematical Stochastics for the pleasant working atmosphere and for the countless interesting discussions. I cherish the memory of all the retreats, pub quizzes, Christmas and summer parties we shared together.

I acknowledge the financial support provided by the International Max Planck Research School for Genome Science. In particular, I would like to thank the programme coordinators, Dr. Henriette Irmer and Frauke Bergmann, for their immediate help in any given situation. I am grateful for the Research Training Program 2088 of the DFG for sponsoring me in my last year of my PhD.

I am deeply grateful to my family, particularly to my parents, Ágnes and Tibor, and my sister, Zsófi, for their unwavering support, love and encouragement during this challenging journey. To my friends, especially Eszti, Zsófi and Ádám, thank you for always being there for me.

I would like to express my deepest gratitude to my husband, Miguel, for his support, patience and love from the beginning to the very end of this journey. His belief in me, even when I doubted myself, has been a constant source of strength and motivation. Thank you *mi amor* for being my partner in every sense of the word.

Last but not least, I owe this thesis to my son Oliver, who was born during my PhD. Your arrival brought immense joy and perspective to my life, reminding me of what truly matters. Balancing motherhood and research was challenging, but your smiles, laughter and endless energy were the best motivation I could have ever asked for. You have been my greatest inspiration.

Summary

This thesis focuses on mathematical models that describe the joint evolution of host-parasite systems in the presence of host switches, or in other words, horizontal transfers. This is a phenomenon where a parasite individual can be passed to a host individual in another way than passing it from parent to offspring. The presence of such host switches can lead to significant incongruences between the separate phylogenetic trees of the hosts and parasites. In our models these separate phylogenetic trees are assumed to be rooted, binary and ultrametric trees, where branch lengths represent the passage of time.

The primary objective of this thesis is to reconstruct the joint phylogeny of the hosts and parasites using information coming from their separate phylogenetic trees. To achieve this, we introduce a novel probabilistic tree reconciliation algorithm, which takes the two fully dated trees as input. To the best of our knowledge, it is the only existing tree reconciliation algorithm that works with fully dated trees. This algorithm identifies the horizontal transfers by analyzing differences in the coalescence times and topologies between the host and parasite tree. When a transfer is not fully identifiable, the algorithm makes an informed random choice out of the feasible possibilities, where the different outcomes are weighted by the probability of their occurrence. These probabilities depend on the horizontal transfer rate, denoted by τ, which requires an a priori estimate of τ based on the separate trees. In this thesis we also provide a novel estimator for this horizontal transfer rate τ and analyze some of its mathematical properties such as existence, consistency and asymptotic normality.

We evaluate the performance of the algorithm using both simulated and biological datasets, with the help of two distance measures on the space of joint phylogenetic trees. The first is an already existing measure that only considers the topologies of the two joint phylogenetic trees to be compared, while the second is a new distance measure that also accounts for time differences between events, that we propose in this thesis. We also compare the outputs of our reconciliation algorithm with the output of the tree reconciliation software `eMPRess`.

We observe that our reconstruction algorithm performs very well for small rates of horizontal transfer, especially on simulated data. For higher transfer rates the reconstruction becomes more difficult.

Contents

1 Introduction

Evolution is defined as the change in the genetic material and characteristics of biological populations over generations. The concept of organismal development over time can be traced back to the 18th century, more precisely to Albrecht von Haller(1708-1777), who used evolution to describe the development of the individual in the egg in 1744 [HH08]. However, we consider Darwin's *On the Origin of Species* [Dar59] as the beginning of modern evolutionary sciences. In this publication Darwin provided evidence for evolution and explained how natural selection (which was also discovered by Alfred Russel Wallace independently) drives the evolution of species over time. Since then it has been proven that besides natural selection, there are other important processes that drive evolution, such as mutation, genetic drift or gene flow. The main paradigm of evolutionary sciences is that evolutionary relationships between different species can be represented as a tree, called the tree of life (TOL), whose root represents the common ancestor of all life on Earth. Understanding evolution and obtaining the tree of life has been a major goal of mathematical biology. Thanks to today's advanced sequencing technologies and computational methods, bigger and bigger parts of the tree are being discovered, a version published in 2016 [HBA+16] is depicted in Figure 1.

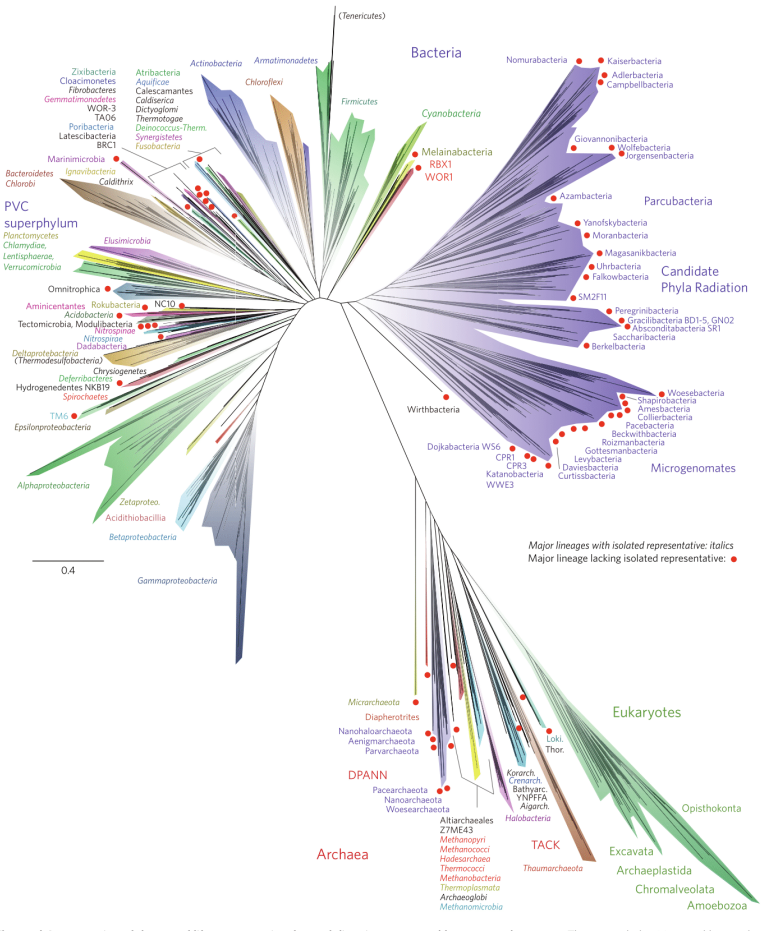


Figure 1: A version of the tree of life from 2016

However, there are a lot of difficulties on the way to obtain the tree of life. Generally, genetic information is only available from currently existing species as it is a difficult task to get samples from extinct species: fossils are hard to find and most likely there are species we will never discover. As we (mostly) have information about present species, we need to infer the tree, for which a model of evolution is needed. Due to the lack of information, mostly stochastic models are used for this purpose, and these models have been used very successfully. A very simple model for describing the evolution of species is the Moran model (precisely given in Definition 2.1). Here a neutral setting is assumed, where every species has equal chances in competing for resources. Under this conditions, at a certain rate a species dies out, and at the same time a new species comes into its place which is a result of a speciation (splitting) of a previously existing species. As mentioned before, in phylogenetic analyses one does not have access to the full evolutionary history, we only can sample from species found at the present. The sample most often consists of DNA sequences or amino acid sequences. It is also possible to work with features derived from such sequences, such as absence/presence of a certain gene, or a type of a certain allele. If we are to sample some of the species and reconstruct their ancestry, we get a rooted binary tree, where the root is the most recent common ancestor (MRCA) of all sampled species. Under the Moran model, the ancestry is well studied and is known to have a particular behavior when the number of species gets large; the limit object itself is called Kingman's coalescent [Kin82b]. In Figure 2 we can see a graphical representation of this model: each line represents a host species, and the arrows indicate the reproduction/extinction events: the species at the tip of the arrows dies, and is replaced by a copy of the species that is at the source of the arrow. The arrows that are shown dashed relate to speciation events that do not play a role in the ancestry of the species found at present. Figure 2 also shows the ancestry of the species sampled at present. Note the different order of the lines in the two pictures, it has been changed in the second picture for the sake of better visualization.

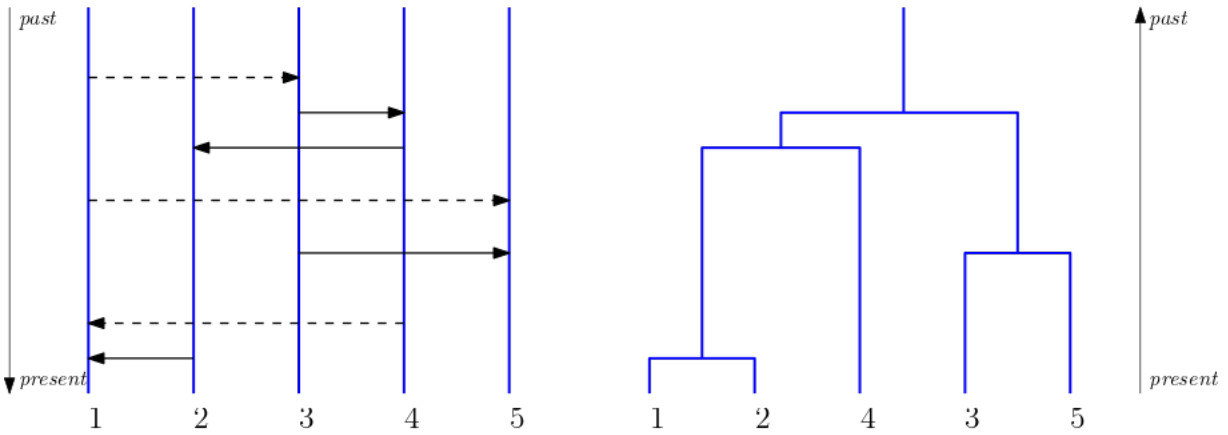


Figure 2: On the left hand side is the graphical representation of the Moran model with 5 species, on the right hand side is the ancestry of the individuals alive at present.

However, this neutral model does not work very well when we consider the different processes shaping evolution (such as natural selection and mutation), the spatial locations of different species, and we cannot ignore the interactions happening between different species either. These

all result in the complete joint evolutionary history of all species not being a tree, but rather a network. A special kind of interaction between species is when one species lives within another species, meaning that individuals of one species are physically contained in individuals of the other species. These are called endosymbiotic relationships, where the individuals that contain the individuals of the other species are called *hosts*, and the ones who are contained are called *symbionts*. As the symbionts are contained in the hosts, they influence each other's evolution and consequently needs to be considered together.

In mathematical biology, these are called host-parasite systems, even though the nature of symbiotic relationship is not always parasitic - it can also be mutualistic or commensalistic. Nevertheless, we use the host-parasite terminology in this thesis. Host-parasite models are of general interest in mathematical biology - they are used not only in evolutionary studies, but also, for example, modeling the behavior of infectious pathogens, see e.g. [PG19], [PW20], [AG16], [OW20]. Moreover, due to the hierarchy appearing in the models they can also be adapted to model the behavior of different hierarchical systems, e.g. genes in species [DR09], [RK12], individuals in populations [Not90], [Tak89], [LS06]. These hierarchical models are also called two-level models. We discuss these models in detail in Chapter 2.

This thesis focuses on host-parasite models with host switches, or in other words, horizontal transfers. This is a phenomenon where parasite individuals can be passed to host individuals in another way than passing it from parent to offspring (during reproduction). If no horizontal transfers happen, hosts and parasites co-evolve with each other and speciation in the hosts trigger a speciation in the parasites. This would result in similar phylogenetic trees of the two levels. However, horizontal transfers can cause significant differences between the phylogenies of the hosts and the parasites, which makes it difficult to recover the joint history of the two organisms. The main objective of this thesis is to introduce a stochastic model that describes the evolution of host-parasite systems with horizontal transfers, and then to develop a new algorithm that recovers the joint history of them, based on the separate phylogenetic trees of the hosts and the parasites.

Recovering the joint history is not only a theoretically interesting question, but also has a practical significance. We give an example here that involves a family of bacteria called Wolbachia and their hosts. Wolbachia are maternally inherited, intracellular alphaproteobacteria, estimated to infect around 40% of terrestrial arthropod species [ZH12]. They are able to manipulate their hosts with inducing different mechanisms in them, such as inducing cytoplasmic incompatibility (CI), male killing or parthenogenesis, see e.g. [WBC08]. It has been established that the phylogeny of the Wolbachia strains and those of their hosts are incongruent, see e.g. [BAH$^+$08]. An example of that is shown in Figure 3. This indicates frequent horizontal transfers of the different strains between the hosts. Even though presence of many horizontal transfers are supposed, there is not a lot of knowledge on the actual timing of transfers [GRB13], [GB16], or in other words, the complete joint evolutionary history of the two levels has not been recovered yet. Knowing the joint evolutionary history could help us understand how these bacteria became so successful. We could then use this knowledge to target certain harmful host species with Wolbachia: it has been shown that Wolbahcia can be used in controling populations of

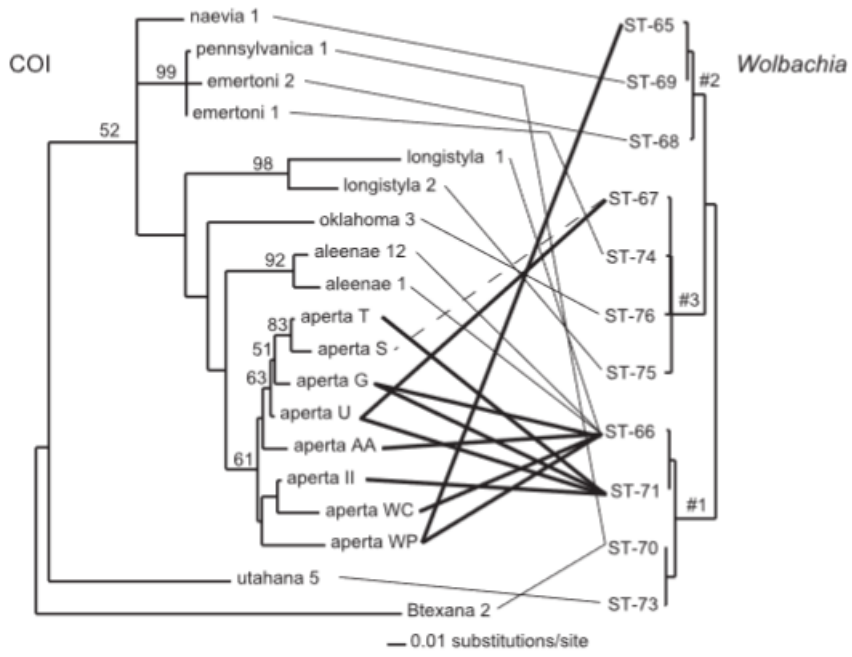


Figure 3: The phylogeny of eight Agelenopsis species compared to that of the 11 host-associated Wolbachia strains, from [BAH$^+$08].

mosquitoes such as Aedes egypti that can transmit serious tropical diseases such as dengue, malaria or chikungunya, see e.g. [MIOJ$^+$09], [BAGFS12].

Now let us describe a general stochastic model for the joint evolution of host-parasite systems with horizontal transfers. As mentioned before, the two levels can also be interpreted in many different contexts, for example individuals in different populations, genes within species.

We assume that there is a fixed number of hosts (host species). At any time, a host can be infected with at most one parasite (parasite species). At a constant rate 1 per of pairs of hosts, a reproduction event happens, when one member of the pair, along with its parasite (if it exists) dies, and is replaced by a copy of the other member of the pair and its parasite (if exists). Moreover, at a constant rate τ per pair of hosts, a horizontal transfer event occurs, where one member of the pair is selected as the donor and the other as the recipient, then a copy of the donor's parasite is transferred to the recipient. The transferred parasite replaces any parasite that was present previously. If the donor happens to have no parasite, then of course no transfer takes place.

We can represent this model graphically. Hosts are indicated by blue vertical lines, while their parasites are indicated by red vertical lines. At reproduction events, a black arrow is drawn. At this event, the host (and its parasite) at the tip of the arrow dies, and is replaced by a copy of the host (and its parasite) at the source of the arrow. At horizontal transfer events a red arrow is drawn, with the donor being the host at the source of the arrow, and the recipient at the tip of the arrow. Again, arrows that do not affect the joint history of the hosts and

parasites alive at the present are shown dashed.

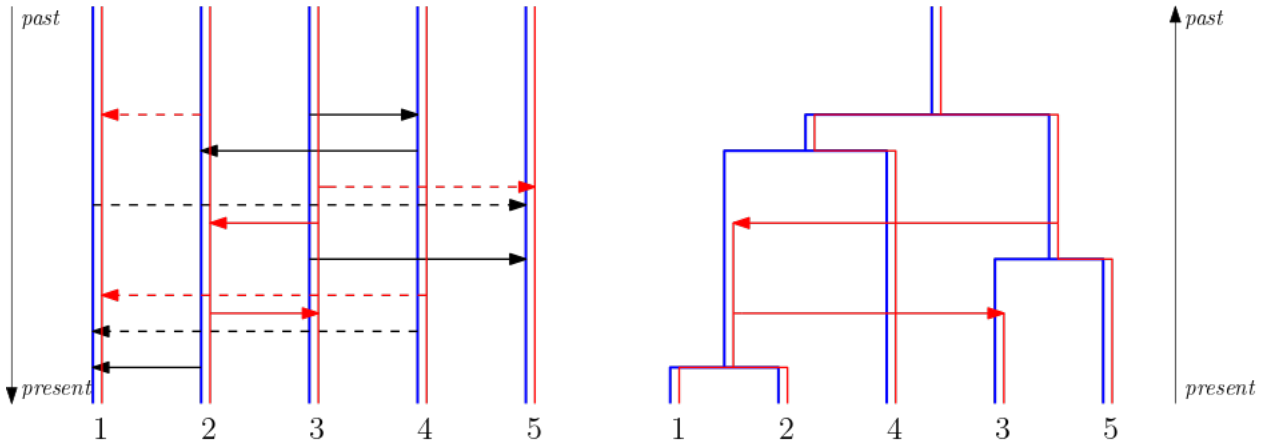


Figure 4: Left: Graphical representation of the model with 5 hosts. Right: the joint ancestry of the hosts and parasites alive at present.

Figure 4 displays an example of the graphical representation of the general model. If we know the complete scenario of what had happened in the forward time model, we can easily obtain the evolutionary history of the sampled species by turning time backwards and following the lines and the arrows that affect the corresponding hosts and/or parasites until there is only one line left, which would mean that all the sampled hosts and parasites have found their most recent common ancestor. When following backwards the history of the hosts, we can only use the black arrows, for the history of the parasites we need to use both the black and the red arrows. The complete joint history is also depicted in Figure 4.

Under this model, the ancestry of the sampled hosts and also of the sampled parasites will be a given by rooted binary trees. There are several extensively used methods to reconstruct the phylogeny of the host species or the parasite species separately, see e.g. Chapter 8 in [Ble17]. Our main goal in this thesis is to reconstruct the joint history with the information available from the separate phylogenetic trees, which means we would like to know which host was infected by which parasite at any time. This is called a reconciliation of the separate trees. Libeskind-Hadas has a recent overview [LH22] of tree reconciliation methods used in host-symbiont cophylogenetic analyses, such as `eMPRess` [SYL+21], `Jane` [CFOLH10] or `CoRE-PA` [MMW10]. We discuss them in more detail in Chapter 2.

Almost all of the commonly used reconciliation method use undated trees as input, i.e. only the topology of the trees are considered. This comes with a limitation regarding time consistency of the reconciliation. Namely, as there is no information on how much time have passed between two speciation events, it can happen that the softwares assign horizontal transfers between branches that actually have never existed at the same time. Our reconciliation method introduced in Chapter 4 takes a new approach and takes dated trees as both the species tree and parasite tree as well as a matching of the leaves, and based on the differences of the merging times of corresponding host and parasite lineages in the two trees, it decides where and when the horizontal transfers had happened. When it is not possible to tell with certainty, it makes

a random choice out of the feasible possibilities. This choice is not uniformly at random, but rather an informed random choice where the choices are weighted by the probabilities with which the different outcomes happen. We must note here that these probabilities depend on the transfer rate τ itself. Generally one does not have information about τ, that is why we need to estimate it from the information available from the separate trees. This is done in Chapter 3 in detail. This way, the reconciled scenario is guaranteed to be strongly time consistent, however it comes with some limitations that are discussed in Chapters 6 and 7.

The rest of the thesis is organized as follows. In Chapter 2 we define the models we will be working with in a mathematically precise way. We also introduce the most important definitions and concepts that we will need. In Chapter 3 we introduce an estimator for the rate τ of horizontal transfers, and prove some of its mathematical properties. First we show that it is well-defined with high probability for large enough trees. We subsequently prove that it is always consistent and that it is asymptotically normal under certain assumptions. Chapter 4 is devoted to the description of our new stochastic tree reconciliation algorithm. In Chapter 5 we analyze the performance of this algorithm on simulated data. In Chapter 6 we apply the algorithm to several biological datasets and we compare the results of our algorithm with the results given by the widely used tree reconciliation software `eMPRess`. Finally, in Chapter 7 we summarize and discuss the most important findings of this thesis, and we mention some related open questions.

2 Mathematical background and related results

In this chapter we introduce a mathematically rigorous description of the different models and related results. We also summarize the biological motivation and background behind these models. We also briefly describe the most common models of DNA sequence evolution and how phylogenetic trees are obtained from DNA sequence data based on these models. But first let us review the related literature on this topic.

2.1 Preliminaries

All of the models we are going to mention are stochastic models, and use Markov processes as the main ingredient. For general knowledge on Markov processes we refer the reader to e.g.[EK86], [Bre10]. There is a vast literature on modeling two-level biological systems, which can be interpreted in several different contexts: parasites infecting hosts, genes evolving in different species, individuals reproducing in different populations. We will give an overview on the most important and relevant such models, both with and without incorporating horizontal transfers.

Most of these models based on some well-known and well-studied (one-level) mathematical models and processes. In the population genetical context, the Cannings model [Can74], [Can75] is widely used to describe reproduction of individuals or evolution of species. The Cannings model assumes a homogeneous setting with a fixed number of individuals that are reproducing neutrally. Time can be measured in discrete generations or continuously. At each generation, (or in case of continuous time, after a random amount of time has passed which is exponentially distributed with a given parameter; or shorter, at a certain rate) each individual produces a random number of offspring (that will make the next generation) in such a way that the number of individuals in the new generation must stay the same, with an additional condition that the joint distribution of the number of offspring for the individuals needs to be exchangeable, i.e. renaming the individuals will not change the joint distribution of the number of offsprings of all the individuals. The model can be used to describe speciation dynamics as well: those species that have 0 offspring go extinct, those that have exactly 1 offspring continue as they are, those who have more than 1 offspring undergo speciation. A special case for Cannings model is the Moran model [Mor62], which we are going to define in the continuous time setting.

Definition 2.1 *(Continuous time Moran model) Let there be a fixed number N of individuals, and let $\sigma > 0$ be a fixed real number. Reproduction events happen according to the following dynamics: For each pair of individuals independently, there is a reproduction event occurring at rate σ, meaning that the time that has passed since the last reproduction event of the pair is exponentially distributed with rate σ. At such an event, one randomly chosen member of the pair dies, while at the same time the other reproduces which results in exactly two offspring.*

As we are focusing on describing and recovering ancestries (in other words, genealogies) in this

thesis, we need some mathematical definitions and notation for them. We will use the terms ancestor and descendant as they are naturally meant.

Definition 2.2 *(Most recent common ancestor) For a set of individuals, their most recent common ancestor (MRCA) is the most recent individual that is an ancestor for all of the members of the considered set.*

Whenever we consider genealogies, we assume that time goes *backwards*, meaning that $t = 0$ represents the present time, and any time $t > 0$ corresponds to the past, exactly t (units of) time back. First we introduce simple genealogies.

Definition 2.3 *(Simple genealogy) Suppose we have a sample of size N, labeled with the elements of $\{1, 2, ...N\}$. The genealogy of the elements in $\{1, 2, .., N\}$ is given as a partition-valued process $(\Pi(t))_{t\geqslant 0}$ with state space $\mathcal{P}_N$, the set of all partitions of $\{1, 2, ..., N\}$, with the following rule: for any two elements $i, j \in \{1, 2, .., N\}$ and any time $t \geqslant 0$, it must hold that*

$$i \sim_{\Pi(t)} j \iff i \text{ and } j \text{ had a common ancestor at time } t$$

with $i \sim_{\Pi(t)} j$ meaning that i and j are in the same block of $\Pi(t)$.

Genealogies and trees are very closely related. If every two elements find their most recent common ancestor in finite time, then genealogies can be represented as trees. To do so, first we recall the definition of trees and their basic properties.

Definition 2.4 *(Rooted binary tree) A graph $T = (V, E)$ with vertex set V and edge set E is called a rooted binary tree with leave set L and root ϱ if and only if the following hold:*

(T1) it is connected and acyclic (tree property)

(T2) if $\deg(v) = 2$ for some $v \in V$ then $v = \varrho$ must hold (only the root has degree 2)

(T3) it holds that $\deg(v) = 3$ for all $v \in V \setminus (L \cup \{\varrho\})$ (every inner vertex has degree 3, except the root)

Edges that connect a leaf to a non-leaf vertex are called external edges*, and edges that connect two non-leaf vertices are called* internal edges*.*

Note that in Definition 2.4, a tree is defined only by its topology, edge lengths are not considered. However, in both biological and mathematical studies of evolution, not only the topology, but also the edge lengths are important as they carry crucial information. In many applications, and also in the settings of interest here, the edge length corresponds to the time that has passed between the two endpoints of the edge. In forward models, time is always directed away from the root, while at backward models, time starts at the tips of the tree and is directed towards the root.

From now on we suppose that each rooted binary tree $T = (V, E)$ is equipped with edge lengths $(l_e)_{e\in E}$ with $l_e > 0$ for all external egdes, and $l_e \geqslant 0$ for all internal edges.

Definition 2.5 *(Distance of vertices) Suppose that $T = (V, E)$ is a rooted binary tree with root ϱ, leaf set L and edge lengths $(l_e)_{e\in E}$. Let v_1 and v_2 be two vertices of the tree, then the distance of v_1 and v_2 is defined as*

$$d(v_1, v_2) = \sum_{e\in P_{v_1 v_2}} l_e$$

with $P_{v_1 v_2}$ being the (unique) path connecting v_1 and v_2 in the tree.

With the help of the distance defined above, we can now define ancestral relationships between the vertices of rooted trees.

Definition 2.6 *(Ancestral relationships in trees) Suppose that $T = (V, E)$ is a rooted tree with root ρ and edge lengths $(l_e)_{e\in E}$. For vertices $u, v \in V$ we say that vertex u is an ancestor of v if and only if u lies on the unique path from v to the root and $d(v, \rho) > d(u, \rho)$ holds. In this case we can also say that v is a descendant of u.*

A special subclass of rooted trees are called ultrametric trees, which are exactly the ones which would emerge when representing genealogies as trees.

Definition 2.7 *(Ultrametric rooted binary tree) A rooted binary tree $T = (V, E)$ with root ϱ, leaf set L and edge lengths $(l_e)_{e\in E}$ is called ultrametric iff every leaf has the same distance to the root, i.e. $\forall \ell_1, \ell_2 \in L$ it holds that $d(\ell_1, \varrho) = d(\ell_2, \varrho)$.*

From now on, each tree that we are considering will be assumed to be an ultrametric, rooted binary tree, and the edge lengths will correspond to the amount of time passed. In this backward time context we can encode ultrametric rooted binary trees mathematically in the following way:

Definition 2.8 *(Encoding of ultrametric rooted binary trees) Each rooted, ultrametric binary tree with N leaves, labeled with $1, 2, ..., N$ can be encoded with the following two mathematical sequences:*

- *merging times: $0 < T_1 \leqslant T_2 \leqslant ... \leqslant T_M < \infty$. It is a sequence of all timepoints when two branches of the tree merge. We also call them coalescence times. If at each such time exactly two branches merge, then $M = N - 1$, otherwise $M < N - 1$.*
- *partitions of the leaf set $\{1, 2, .., N\}$: $\{\{1\}, \{2\}, ..., \{N\}\} = \mathcal{P}_0 \prec \mathcal{P}_1 \prec ... \prec \mathcal{P}_M = \{\{1, 2, ..., N\}\}$ with $\mathcal{P}_i$ being the partition of the leaf set such that two leaves are in the same block if and only if they have found their most recent common ancestor by time T_i. (Here, for any partitions Q_1 and Q_2, the relation $Q_1 \prec Q_2$ means that each block of the partition Q_1 is a subset of a block of Q_2.)*

Note that such encoding of ultrametric rooted binary trees are unique: for each such tree one can construct an encoding, and different encodings refer to different trees. Figure 5 shows an example for 5 leaves:

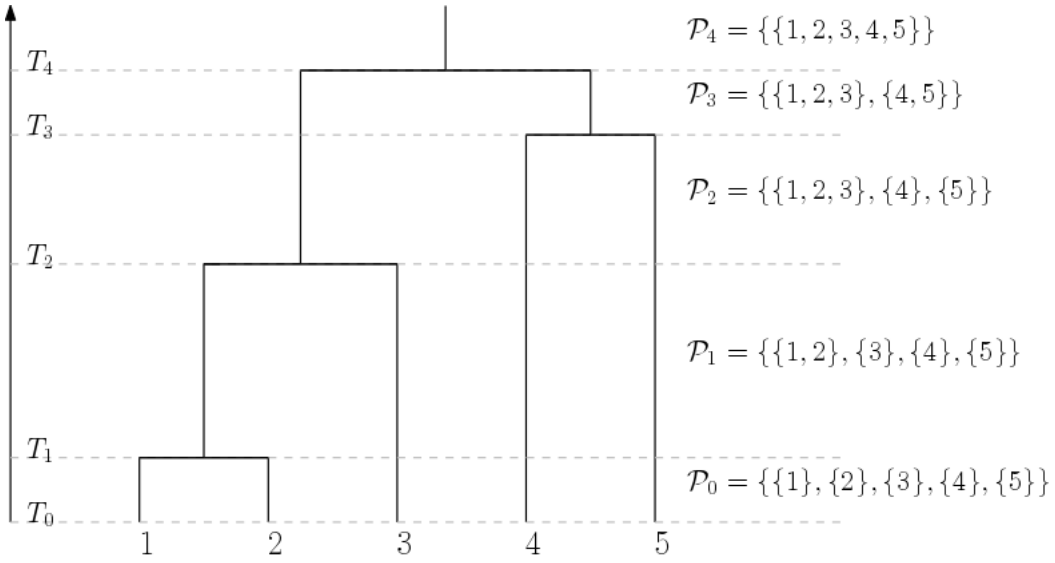


Figure 5: Encoding of a rooted binary tree with $N = 5$ leaves

Notice that the encoding of rooted binary trees are directly related to the genealogies: the coalescence times are the jump times of the partition-valued process $(\Pi(t))_{t\geqslant 0}$ and the partitions $\mathcal{P}_0, \mathcal{P}_1, ..., \mathcal{P}_{N-1}$ are exactly the partitions appearing in the genealogy at times $T_0, T_1, ..., T_{N-1}$.

Now let us go back to the Moran model. If we look at the genealogy of the individuals at the present time, we get a special kind of ultrametric binary tree that is called N-Kingman tree, introduced by Kingman [Kin82a], [Kin82b]. Here we also give a definition for it.

Definition 2.9 *(N-Kingman tree) An N-Kingman tree with coalescence rate σ is a rooted, ultrametric binary tree with N leaves and the following encoding:*

- *The coalescence times $0 < T_1 < T_2 < ... < T_{N-1} < \infty$ satisfy that $T_1 - T_0, T_2 - T_1, ..., T_{N-1} - T_{N-2}$ are all independent from each other with $T_i - T_{i-1}$ having an exponential distribution with parameter $\sigma\binom{N+1-i}{2}$ for all $i = 1, 2, ..., N$.*
- *The partitions $\{\{1\},\{2\},...,\{N\}\} = \mathcal{P}_0 \prec \mathcal{P}_1 \prec ... \prec \mathcal{P}_{N-1} = \{\{1,2,...N\}\}$ satisfy that for each i, we can get $\mathcal{P}_i$ from $\mathcal{P}_{i-1}$ by merging exactly two of its blocks into one.*

There is a simple way to construct N-Kingman trees:

Proposition 2.10 *Let us start with N individuals at the present. As time goes backwards, each pair of branches merge at rate σ, independently from each other, until there is only a single branch left. The tree obtained in such way is exactly an N-Kingman tree.*

Proof: It is straightforward to check that all the coalescence times and partitions satisfy the conditions in Definition 2.9 with probability 1. □

In population genetics, the limit of genealogies as $N \to \infty$ (with proper scaling) are objects of interest as they approximate genealogies of large populations. The $N \to \infty$ limit of the N-Kingman tree is called Kingman's coalescent [Kin82a], [Kin82b], and has an analogous behavior: each pair of branches merge at rate σ independently from each other, until there is only a single branch left. Due to the independence of the merging times, no two merging times are the same with probability 1. The special property of Kingman's coalescent is that it arises as the scaling limit of large population genealogies of a class of Canning models [MS01]. Cannings models have a lot of extensions for example with mutation, selection, spatial locations and migrations. For a general overview on population genetics and coalescent theory we refer the reader to Berestycki [Ber09], as well as the classical books of Durrett [Dur08] and Wakeley [Wak08].

One of the main assumption in Cannings models is the fixed population size, which is not always realistic. We can relax this assumption by using branching processes, both in discrete and continuous time and space. There, each individual has a random number of offspring independently from each other, without any constraints on the population size. These processes are well known and studied mathematical objects, see e.g. Atreya and Ney [AN72], and their use in biology is demonstrated for example by Kimmel and Axelrod [KA15]. There is a frequently used continuous time branching process called birth-and-death model, described in Bailey [Bai64]. Here individuals are allowed to have either 0 or 2 offspring, which correspond to death and birth, respectively. This is the branching process that is the most similar to the Moran model, without assuming a constant population size.

When modeling the evolution on two levels, usually the same (previously mentioned) mathematical objects are used, adapted for two levels. In some cases the same type of model is used for both of the levels, but it is also possible to choose different models for the two levels.

In the gene tree-species tree context, we need to mention the multispecies coalescent (MSC) [DR09], one of the first models aimed to explain and allow discordances between gene trees and species trees. It is based on the Wright-Fisher model for single populations, which is also a special subclass of Cannings models, where the number of offspring for each individual follows a symmetric multinomial distribution. The MSC extends the Wright-Fisher model to multiple species, allowing genes from two different species to merge later in the past than the divergence time of the two corresponding species (this is called incomplete lineages sorting, ILS). It provides a framework for inferring species trees from gene trees with several software implementations such as `ASTRAL` [ZRSM18] or `BEAST` [BVBS$^+$19]. Mathematically, joint ancestry of the two levels can be described using exchangeable nested partitions [Bla18], or in a more general setting, with two-level metric measure spaces [Mei18]. Blancas et al. [BDLSJ18] defined the two-level analogs of simple coalescent processes called simpled nested exchangeable coalescents and characterize their laws. A special subcase of simple nested exchangeable coalescent are the so-called nested Kingman coalescents (Kingman-in-Kigman) which are obtained when both the species tree and the gene tree are described by Kingman's coalescent. There are many extensions of the multispecies coalescent model. In [RK12] a third tree, a locus tree is added in order to account better for ILS events. Using this model, the authors developed the first reconciliation method that accurately infers gene duplications and losses in the presence of ILS. It is also possible to

extend the multispecies coalescent such that it can take reticulate evolutionary events (such as hybridization or horizontal gene transfer) into account [YDN12], [Kub09]. However, these methods assume that these events do not happen rarely, which contradicts some empirical findings [SLYA16]. To address this issue, Marin et al. [MACL20] developed a new approach called gene-based diversification models. This is a gene-based model that does not use any notion of ancestral relationships between species. Instead, they define species as lineages that can coexist without fusion in spite of gene flow (horizontal transfers). For two empirical datasets they show better support for their model than for the MSC model.

Now let us mention a few of the existing host-parasite models. Pokalyuk and Wakolbinger [PW20] introduce an individual-based model, where a fixed number of host individuals each contain a fixed number of parasites. Hosts evolve according to the Moran model. Parasites each have two types, and co-evolve with hosts, but with balancing selection to a certain equilibrium frequency. Reinfections of hosts are also possible, when a parasite can be copied and reinfected to another host. They study the limit of type frequencies in large host and parasite populations, that turns out to be a dynamical system with a globally stable equilibrium. The biological relevance of this model is discussed in [PG19]. Related hierarchical models have been mathematically studied by several authors [Wu93], [Daw18], [BT11], where they investigate the limit of frequency processes in case of mutations and/or selection. Models that also account for interaction (such as competition) between hosts and between parasites have been analysed for example in [MR13], [OW20]. Alsmeyer and Göttrup [AG16] propose a two-level branching model, where the host population grows like a Galton–Watson process, but the number of offspring produced by a host determines the reproduction of a parasite living in this cell and also the distribution of parasite offsprings to the daughter hosts. In this setting, they show conditions for almost sure extinction of parasites, and also limit theorems on the law of the number of parasites conditioned on survival.

2.2 Mathematical description of the model

In this section we will rigorously define our model and state some of its properties.

Definition 2.11 *(Two-level forward model) We assume that there is a fixed number N of host species. Each host is infected with at most one parasite. As time is going forward, there are two possible kinds of events that can occur:*

- *Reproduction: At a constant rate 1 per pair of hosts independently, one host of the pair dies together with its parasite (if it is infected) and is replaced by a copy of the other host together with its parasite.*
- *Horizontal transfer: At a constant rate τ per pair of host species independently, one of the hosts transfers its parasite (if it exists) to the other host, where the current parasite (if it exists) is replaced by the one that was received.*

2.12 Remark. If we only consider hosts, the model describing their evolution is exactly the continuous version of the Moran model, see Definition 2.1. Consequently, the evolutionary history of the host species is given by an N-Kingman tree with $\sigma = 1$. □

We would like to note that the genealogy of the sampled parasites will also be an N-Kingman tree, however with a coalescence rate $\sigma = 1 + \tau$, since each pair of parasite lineages coalesce either due to a host coalescence event that happen independently at rate 1 per pair, or due to a horizontal transfer event that happen independently of everything else at rate τ per pair.

Let us emphasize here that there is no guarantee that the sampled parasite lineages will always be associated with the sampled host lineages. In fact, after a time when the number of host lineages is significantly smaller than N, if a transfer event affects a host lineage carrying a parasite lineage, it will most likely involve another host species that is not part of the host tree. We do not have any information regarding such lineages as they either correspond to non-sampled host species or to species that went extinct. Moreover, if N is big, the number of such extinct and/or non-sampled lineages is also so big that it is impossible to keep track of them all.

This issue is addressed in [STLD13], where the authors propose a duplication-transfer-loss (DTL) model of evolution of genes in a big, but fixed number of species. They argue that as the number of sampled species is significantly smaller than the number of all existing species, the probability that gene lineages have been going along non-sampled or extinct lineages (either due to speciation or transfer event) can be high, and in fact the majority of transfer events involve a non-sampled or extinct lineage. Considering this, they calculate the probability of a gene tree given the species tree and the parameters of the model, which will serve as the base of the ALE (Amalgamated Likelihood Estimation) reconciliation method. However, the performed calculations and simplifications are not entirely mathematically rigorous. Foutel-Rodier et al. [FFRS20] also consider this phenomenon in a different context, but in a mathematically rigorous way. They introduce a fragmentation-coalescent process called Kingman's coalescent with erosion, that arises as the limit process of a partition-valued process based on the following model: They suppose they have a fixed number of species, which evolve according to the Moran model. Each species carries a fixed number of genes. Genes can be transferred to other species at a certain rate, where they replace the original copy. At present they sample one species and all of its genes, and follow backwards the ancestral lineages of the genes and the ancestral species which they belong to. From that, a partition-valued process can be obtained by putting to genes in the same partition at time t if and only if they belong to the same ancestral species t units of time before sampling. They also study its asymptotic behavior as the number of particles becomes large. From a practical point of view it is also important to account for evolution along non-sampled or extinct lineages as it can impact the relative timing of transfer events, see e.g. [BMMG24].

Here we consider two backward models that are inspired by our general forward dynamics. In the first we will completely ignore transfers leaving the host tree, and in the second model we will add a ghost line to represent all the non-sampled and extinct lineages, but we assume that

transfers leaving the host tree are infrequent and if they leave the host tree, they will likely come back at a time that is on the same timescale as the coalescence events in the host tree. Now we are going to formally define them as well.

Definition 2.13 *(Simplified Backward model I) Let us suppose we have N host species, each infected by a single parasite species. Suppose that each pair of host lineages coalesces at rate 1, independently from each other. At such coalescence events, if both of the merging hosts carry a parasite, the parasite lineages also coalesce. Moreover, independently of the host coalescence events, suppose that at a rate τ per pair of host lineages, a horizontal transfer happens, where the infection (if it exists) leaves one of the host lineages and is transferred to the other host lineage of the pair where it coalesces with the other parasite lineage (if it exists).*

A graphical representation of this backward model is depicted in Figure 6. Host lineages are indicated in blue, parasite lineages in red. The transfer events are depicted by red arrows. Note that under this model, it still remains true that the parasite tree is an N-Kingman tree, with coalescence rate $1+\tau$ per pair, while the host tree is an N-Kingman tree with coalescence rate 1.

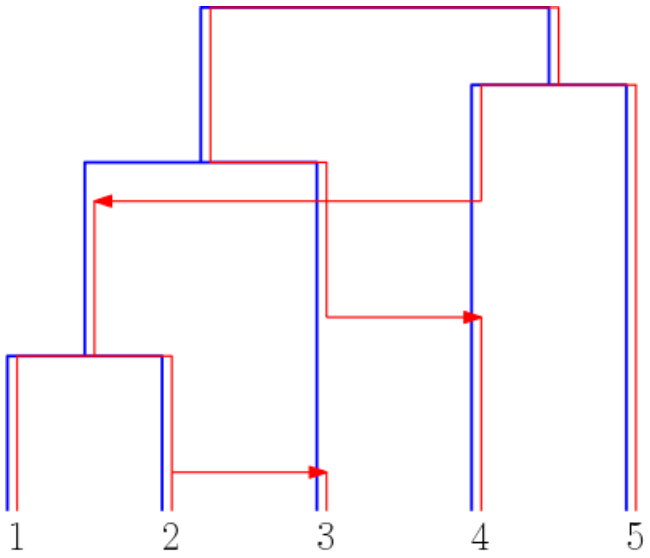


Figure 6: Illustration of Backward model I

As the main goal of this thesis is to recover the joint trees, we need a mathematical description for them. This description is based on the encoding of (single) trees introduced in Definition 2.8. Here we would like to point out the fact that with probability 1, each coalescence time and horizontal transfer time are different, which means that at coalescence events exactly two branches merge.

Definition 2.14 *(Encoding of joint trees under Backward model I) Let us suppose that the host tree is given by coalescence times $0 = T_{H,0} < T_{H,1} < ... < T_{H,N-1} < \infty$ and partitions $\{\{1\},\{2\},...,\{N\}\} = \mathcal{P}_0 \prec \mathcal{P}_1 \prec ... \prec \mathcal{P}_{N-1} = \{\{1,2,...N\}\}$, and the parasite tree is given by*

$0 = T_{P,0} < T_{P,1} < ... < T_{P,N-1} < \infty$ *and partitions* $\{\{1\},\{2\},...,\{N\}\} = \mathcal{Q}_0 \prec \mathcal{Q}_1 \prec ... \prec \mathcal{Q}_{N-1} = \{\{1,2,...N\}\}$. *Let us suppose that the union of all coalescence times and all horizontal transfer times is given by the sequence* $0 = T_0 < T_1 < ... < T_{k_N}$ *with* k_N *denoting the size of the union. For all intervals* $[T_{i-1}, T_i)$ $(i = 1, ..., k_N - 1)$ *there exist unique indices* i^H *and* i^P *such that* $[T_{i-1}, T_i) \subset [T_{H,i^H-1}, T_{H,i^H})$ *and* $[T_{i-1}, T_i) \subset [T_{P,i^P-1}, T_{P,i^P})$ *holds. On the time interval* $[T_{i-1}, T_i)$, *the alignment (or with order words, matching or pairing) of the two trees is given by the set of pairs*

$$M^i = \{(b^i_1, d^i_1), (b^i_2, , d^i_2), ..., (b^i_{N-i^H}, d^i_{N-i^H})\} \tag{2.1}$$

with b^i_j *being a block of the partition* $\mathcal{P}_{N-i^H}$, *and* d^i_j *being either a block of* $\mathcal{Q}_{N-i^P}$ *or being the empty set for all* $j = 1, 2, ..., i^H$. *We say that on the time interval* $[T_{i-1}, T_i)$, *the branch of the host tree corresponding to* b^i_j *is infected by the branch of the parasite tree corresponding to* d^i_j *if* $d^i_j \neq \emptyset$. *If* $d^i_j = \emptyset$ *then we say that the branch* b^i_j *is not infected on the time interval* $[T_{i-1}, T_i)$. *The entire matching is characterized by the sequences of pairs for all time intervals. i.e.*

$$\mathbb{M} = (M^i)_{i=1}^{k_N}. \tag{2.2}$$

Note that for the encoding we need to consider all horizontal transfer events, not just those that result in a coalescence event in the parasite tree. Figure 7 shows an example of an encoding of a joint tree with 5 leaves.

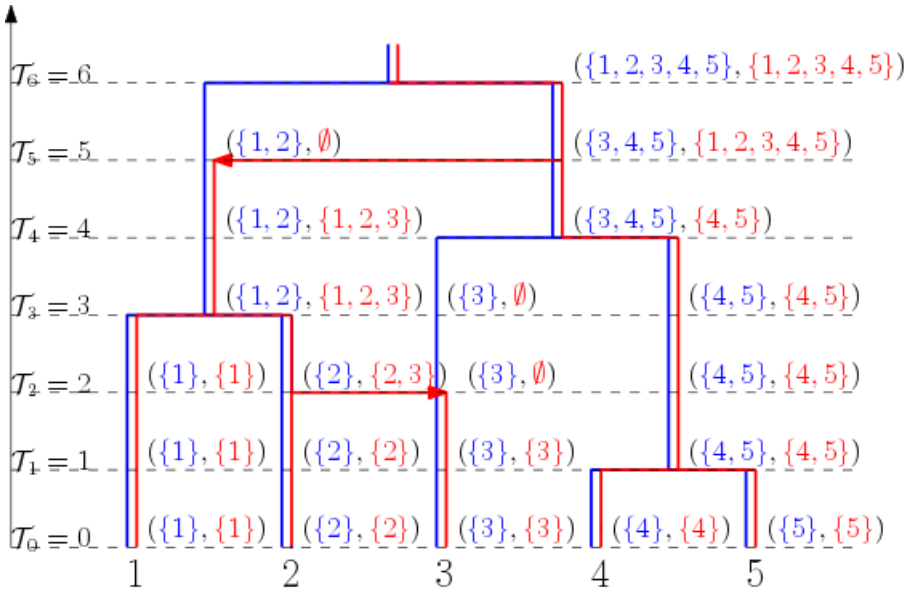


Figure 7: Example of encoding a joint tree with $N = 5$ leaves

Now we define our second backward model, which allows transfer of parasite lineages to the ghost line.

Definition 2.15 *(Simplified Backward model II) We will keep the dynamics from Backward model I, but we will also add an extra host lineage called the ghost line, which will represent all*

the extinct or non-sampled host species. Parasite lineages are assumed to exit the host tree (and enter the ghost line) at a rate $p > 0$ per branch, while lineages leave the ghost line and return to the host tree at a rate $q > 0$. At such an event the host lineage which will host the parasite lineage coming back from the ghost line, will be chosen randomly. If there are several parasite lineages associated with the ghost line, they coalesce at a rate $r > 0$ per pair independently.

2.16 Remark. It is not specified how we choose the rates p, q, r. Generally, we would like to have the host and parasite trees on the same timescale, hence it makes sense to assume that transfers to the ghost line are not so frequent ($p < \tau$), and once they are transferred, they come back to the host tree in a relatively short time ($c \approx 1$).

A graphical representation of this backward model is depicted in Figure 8. Similarly to Backward model I, host lineages are drawn in blue, parasite lineages are drawn in red. Red arrows indicate transfer events. The ghost line is depicted by a dashed blue line, infections along it are also drawn dashed, but red. Note that due to the different transfer rates to and from the ghost line, it is no longer true that the parasite tree is an N-Kingman tree.

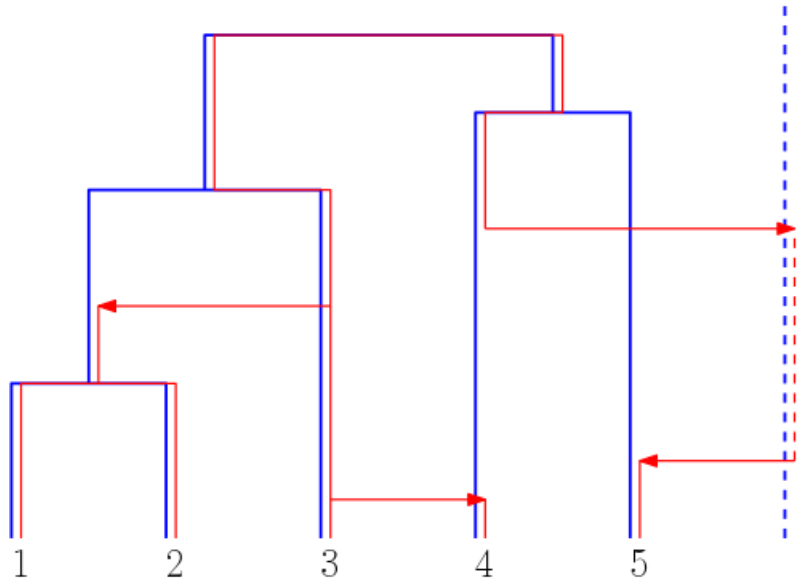


Figure 8: Illustration of Backward model II

Note that the ghost line is the only host lineage that can carry several parasites at the same time. If we do not allow such merging events on the ghost line, then under Backward model II the parasite tree would be a seedbank coalescent, see e.g. [KKL01], [BGCKWB16].

We will use a similar encoding of joint trees under Backward model II, i.e. in the presence of the ghost line.

Definition 2.17 *(Encoding of joint trees under Backward model II) Let us suppose that the host tree, the parasite tree and the sequence of all coalescence times and horizontal transfer times given as in Definition 2.14. On each time interval, the pairing along the branches of the*

host tree is characterized similarly as in Definition 2.14, but an extra pair is added. Its first entry is denoted by g indicating the ghost line, and the second entry of the pair is a set of blocks that are currently associated with the ghost line. The matching M^i on interval the $[T_{i-1}, T_i)$ will be given by

$$M^i = \{(b_1^i, d_1^i), (b_2^i, , d_2^i), ..., (b_j^i, d_j^i), (g, \{gd_1, gd_2, ..., gd_k\})\} \tag{2.3}$$

where j denotes the number of host lineages, and k denotes the number of blocks currently associated with the ghost line. If there are no blocks on the ghost line, the last pair is simply reduced to be $(g, \emptyset)$.

Figure 9 shows an example for an encoding of a joint tree with the ghost line.

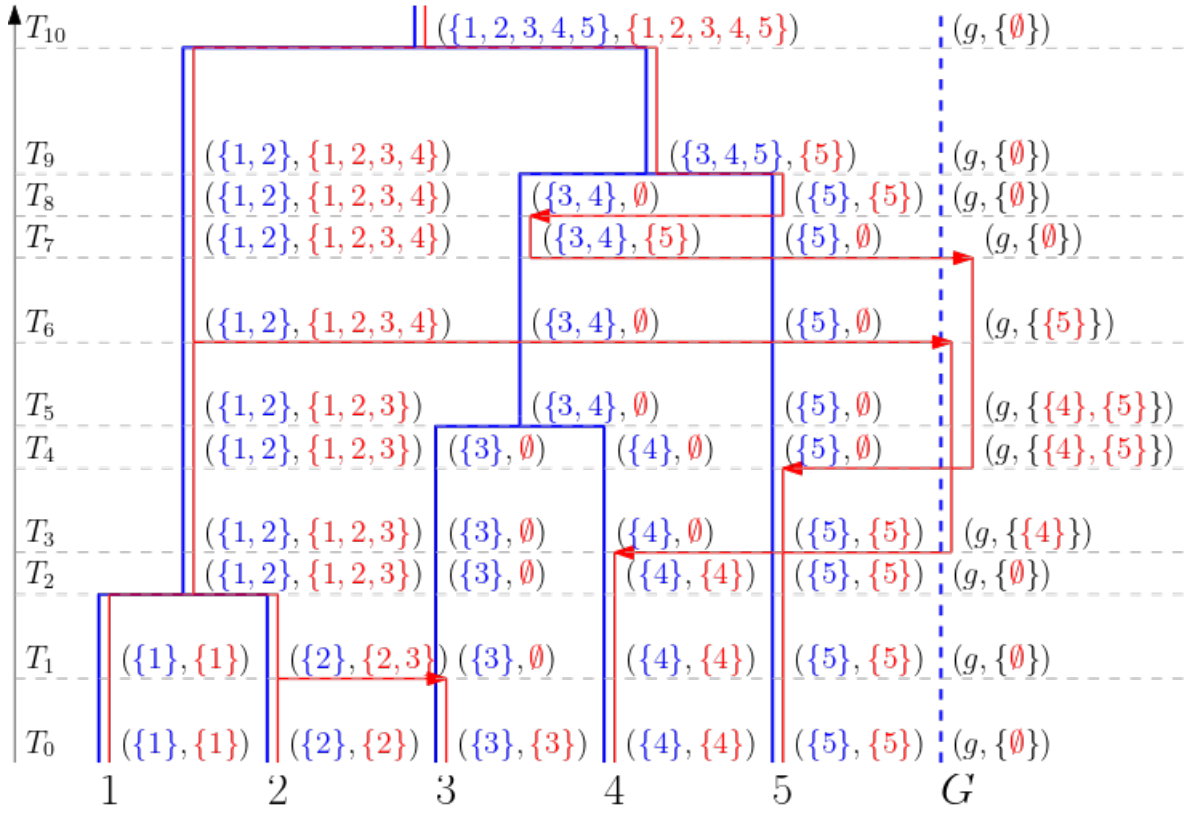


Figure 9: Example of encoding a joint tree with $N = 5$ leaves with the ghost line

It is the goal of our tree reconciliation algorithm described in Chapter 4 to recover the joint trees under both models, i.e. to obtain matchings as defined in Definition 2.14 and Definition 2.17.

2.3 Reconstructing phylogenetic trees from DNA sequences

As later we would like to use our algorithm on biological data, it is important to know how one gets to the separate trees from DNA sequences. In this chapter we summarize the most

important methods used to reconstruct phylogenetic trees from DNA sequence data and some differences between the different methods. For details we refer the reader to Chapter 8 of [Ble17].

2.3.1 Substitution models

When working with real-life data, in particular with DNA sequences, besides modeling the evolution of different species (speciations, extinctions), one needs to have an additional model that describes the evolution of the DNA sequences itself along the (implicit, to be reconstructed) evolutionary tree. For us, DNA sequences are "words" formed from the letters A, C, G and T, referring to the nucleotides building up the DNA: adenine, cytosine, guanine and thymine. Positions in the sequence are often referred to as sites. There are several methods to reconstruct the tree in question, each of them relies on explicit models of the evolution of the sequences. The most commonly used models are the so called substitution models, which are modeling the mutations along the branches of the underlying tree in a Markovian way.

The most crucial assumptions about the Markov chain is that it is a homogeneous, stationary Markov process for each site of the DNA sequence, and that mutations at different sites happen independently. Most of the early models assume the same overall mutation rate for all sites, which has been found to be unrealistic as there are parts in the genome that are well known to be more prone to mutations than others. There are several ways to account for this rate heterogeneity, i.e. the variation in the overall mutation rate between different sites. The most commonly used correction is that each site has an overall mutation rate that is given by i.i.d. copies of a $\Gamma(\alpha, 1/\alpha)$ distributed random variable for some $\alpha > 0$.

We will now introduce the so called GTR + Γ model in detail. It is one of the most commonly used (class of) substitution models for modeling DNA sequence evolution.

Definition 2.18 *(GTR + Γ model) Let $T = (V, E)$ be a (non necessarily binary) tree with N leaves and edge lengths $(l_e)_{e\in E}$. Let us suppose that we have a sample of N DNA sequences of length L. The GTR + Γ model is a collection of L independent, homogeneous, stationary, time-reversible Markovian mutation processes $(X_1(t))_{t\geqslant 0}, (X_2(t))_{t\geqslant 0}, ..., (X_L(t))_{t\geqslant 0}$ along the branches of T with state space $S = \{A, C, T, G\}$, and common stationary distribution $\boldsymbol{\pi} = (\pi_A, \pi_c, \pi_G, \pi_T)$. For each i, the substitution rate matrix of $(X_i(t))_{t\geqslant 0}$ that is given by $\mu_i \boldsymbol{Q}$ with*

$$\boldsymbol{Q} = \begin{pmatrix} -(a\pi_C + b\pi_G + c\pi_T) & a\pi_C & b\pi_G & c\pi_T \\ a\pi_A & -(a\pi_A + d\pi_G + e\pi_T) & d\pi_G & e\pi_T \\ b\pi_A & d\pi_C & -(b\pi_A + d\pi_C + f\pi_T) & f\pi_T \\ c\pi_A & e\pi_C & f\pi_G & -(c\pi_A + e\pi_C + f\pi_G) \end{pmatrix}$$

with a, b, c, d, e, f being the substitution rates: a is the rate of change from A to C and from C to A, and so on. We assume that the numbers μ_i denote the overall mutation rate in the

process X_i, and are i.i.d. copies of a $\Gamma(\alpha, 1/\alpha)$ distributed random variable. After a branching event in the tree, mutations are assumed to happen independently along the daughter branches.

The name GTR refers to 'general time reversible', as this is the most general substitution model that is given by a homogeneous, stationary and reversible Markov chain [Tav86]. With special assumptions we can decrease the number of free parameters. The most commonly used assumptions are that the invariant distribution of the process puts equal weights on each of the nucleotides. The other simplifying assumption only allows two kinds of rates for mutations: one is for transitions and the other is for transversions. (Transitions are mutations between nucleotides that have the same base (purine (A-G) or pyrimidine (C-T)), while transversions are interchanges between a purine and a pyrimidine based nucleotide.) The special subcases (which are considered models on their own) of the model are summarized in Figure 10.

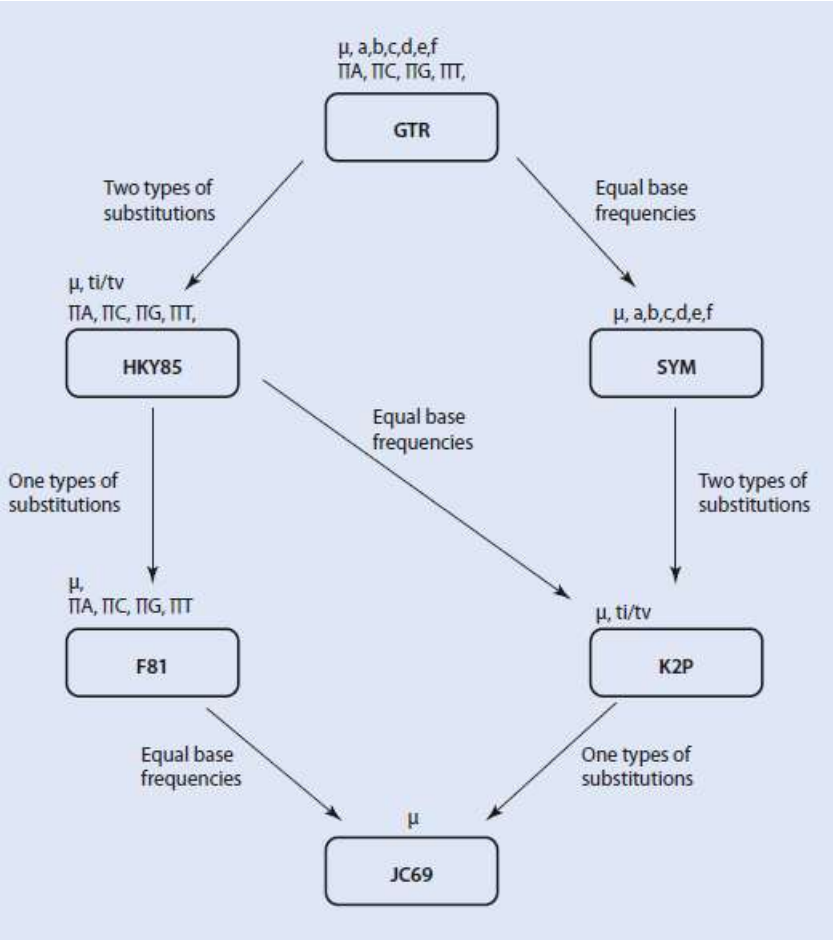


Figure 10: Special submodels of the GTR + Γ model, from [Ble17]

A very important property of the GTR+Γ model is its identifiability, that was proven by Allman et al. in 2008 [AAR08]:

Theorem 2.19 *The 4-state GTR +Γ model is identifiable from the joint distributions on subtrees with 3 leaves for all parameters on any tree with three or more leaves.*

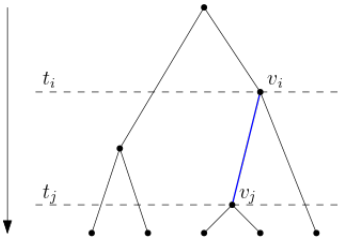
It means, that if we would have an infinite sample of infinitely long DNA sequences coming from each of the N species under this model, we could get all of the character distributions at the leaves of each 3-leaf subtree, and from those distributions theoretically we could get back all of the parameters of the model. This identifiability is crucial for all statistical inference, and the authors emphasize that it does not only hold for generic parameters, but also for all the special subcases depicted in Figure 10.

2.3.2 Model selection

When choosing which model to work with, we need to be careful as there is a trade-off: using more parameters can help us to represent the underlying data in a more realistic way, but there is the risk of overfitting. There are several model selection methods that try to fit the data as much as possible while also trying to avoid overfitting. We will introduce here the widely used Akaike and Bayesian information criterions (AIC and BIC), which can both compare simultaneously all of the models in question.

First, we need to select an initial tree, a 'quick guess' which we will use to calculate likelihoods under all of the different models. Such a tree can be produced for example by the widely used neighbor joining (NJ) algorithm of Saitou and Nei [SN87]. Then, given this tree, one can calculate the likelihood of the data for each possible model, while also performing maximum likelihood estimation for the free parameters. If we have the tree $T = (V, E)$ and a fixed substitution model with rate matrix Q, and a set of N (aligned) DNA sequences of length L each, denoted by $S = \{S_{ij}, 1 \leqslant i \leqslant N, 1 \leqslant j \leqslant L\}$, then the likelihood of S given T and Q is given by

$$\ell(S|T,Q) = \sum_{S'} \prod_{(v_i,v_j)\in E} \prod_{k=1}^{L} \left(\mathrm{e}^{(t_j-t_i)Q}\right)_{s'_{ik},s'_{jk}}$$



where the sum is taken over all possible ancestral sequences S' of S at the vertices of T, the first product is taken over all edges (v_i, v_j) in T with v_i corresponding to a merging in the tree at time t_i and v_j corresponding to a merging event in the tree at time t_j (meaning that the edge (v_i, v_j) has a length of $t_j - t_i$). Even though the number of possible sequences S' is growing exponentially as N (the number of leaves) grows, the likelihood can be computed efficiently by Felsenstein's pruning algorithm [Fel81]. Once the likelihood ℓ is calculated, the AIC and BIC are given by

$$\mathrm{AIC} = -2\log\ell + 2k \qquad \text{and} \qquad \mathrm{BIC} = -2\log\ell + k\log L,$$

where k is the number of free parameters. The model with the lowest corresponding AIC /BIC score will be selected. The model selection algorithm has many good implementations, e.g. `ModelTest-NG` [DPK+20], [FICD+14].

2.3.3 Tree reconstruction

Once we have selected a model, we can start to reconstruct the underlying tree. There are several methods, including distance-based, maximum-likelihood (ML), maximum parsimony (MP) and Bayesian methods. As the majority of the most commonly used phylogenetic softwares such as `RAxML` [Sta14] or `IQTREE` [NSvHM15] are all maximum-likelihood based, we will only introduce the main ideas behind these methods. For background on the other three methods, we again refer the reader to Chapter 8 of [Ble17].

Our aim is to find the tree with the highest possible likelihood. First we need to find the best topology and then we also need to optimize the branchlengths. As the number of possible tree topologies grows exponentially with the number of leaves growing, it is computationally impossible to consider all possible topologies and choose the overall best. Instead, the methods rely on heuristics such that only a small fraction of all possible topologies are investigated, while ensuring that the probability that the overall best tree is within those investigated ones is big enough.

To begin the search, one needs a reasonable starting tree and its likelihood. The most commonly used starting trees are obtained by neighbor joining or carrying out MP analysis, or just simply by randomly selecting one tree. Then the tree gets modified using one of the allowed tree operations, and the likelihood is again calculated. If the likelihood of the new tree is higher than the likelihood of the previous tree, then we keep the new tree, otherwise we stick with the previous one. After all allowed tree operations are performed, we will get a tree whose likelihood is the highest, no matter how many more allowed operations we perform, and this will be the final tree that we choose for representing the evolutionary history of the sample. The allowed operations usually are nearest neighbor interchange (NNI), subtree pruning and regrafting (SPR) and tree bisection and reconnection (TBR), which are illustrated in Figure 11. They are used as their number (given a tree topology) is feasible and the likelihood for each new tree obtained by one of these operations can be calculated efficiently.

It is important to note that it is not guaranteed that we will find the best tree given as we do not search the whole space of possible topologies. That is why several iterations of the algorithms are recommended, with possibly different starting trees. This method can also result in finding a tree whose likelihood is locally, but not globally maximal. There exists several methods to avoid such a case, for example `IQTREE` uses random perturbation of trees to avoid that. This also means that different softwares may produce different trees for the same DNA sequence data. There are several attempts to represent phylogenetic trees in a suitable space which is embedded with a geometry such that it is possible to calculate distances and geodesics between each two trees, which allows to calculate 'averages' of trees (see e.g. [LGNH24]) which can be used to arrive to a consensus if we have different possible outputs of phylogenetic softwares.

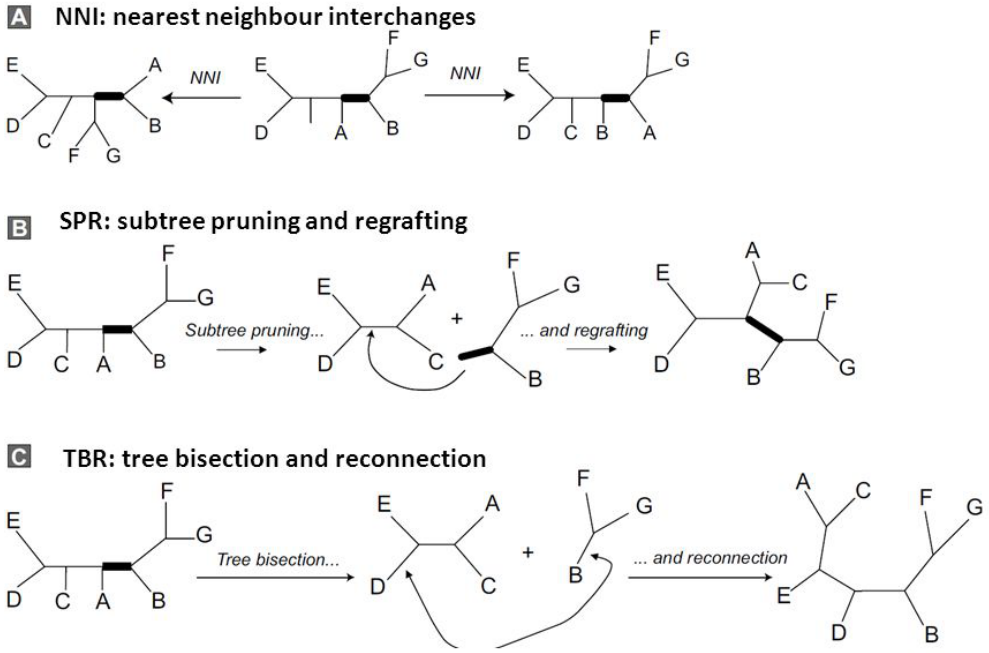


Figure 11: Allowed tree operations, from Slide 6 of the presentation Tree searching by K. Müller. Available at `https://www.slideserve.com/aquila/tree-searching`

2.4 Inferring horizontal transfers

In this chapter we again consider two-level systems with horizontal transfers. Detecting horizontal transfers is key when trying to find the joint evolutionary history of the two levels (hosts and parasites). Here we give a brief overview over the various existing methods. We also introduce a notion of identifiability of horizontal transfers in our model and present a theoretical result regarding the probability of identifiability of horizontal transfers.

2.4.1 Existing methods

Even though a significant part of the existing methods focus on detecting horizontal transfers in the species tree-gene tree setting, most of them can also be interpreted and used in the host-parasite context just as well, and there are also some methods that are designed specially for host-parasite cophylogenetic analyses. For details we refer the reader to the overviews of Ravenhall et al. [RvLD15] and Libeskind-Hadas [LH22], on which this short review is founded. Almost all tools that we describe here are based on analyses of DNA sequence data. They can be divided into two main groups: parametric and phylogenetic methods. Parametric methods look for parts of the genome that significantly differ from the average, such as GC content or codon usage. Phylogenetic methods search for incongruencies in the two trees. There are two possible ways to do so: explicitly by reconciling the two trees, or implicitly by focusing on properties that correlate with the evolutionary history of the lower level objects (genes, parasites, etc), such as presence or absence of certain patterns, or significantly shorter or longer

evolutionary distances.

Explicit methods. First let us overview the explicit methods, i.e. the existing reconciliation methods and some softwares implementing them. Again we emphasize that a big part of the methods were developed for species tree-gene tree reconciliations, but are also applicable for host-parasite cophylogenetic analyses. We will again use the host-parasite terminology. A tree reconciliation essentially maps the nodes of the parasite tree onto the nodes and edges of the host tree with some restricting conditions. For that we need of course the host and the parasite tree, which are assumed to be binary and rooted. However, in most of the cases only their topology is considered, the branch lengths are ignored. Moreover, a mapping between the leaves of the two trees is needed as well, which tells us which parasites are associated with which hosts at the present (time of sampling). The formal definition is given as follows:

Definition 2.20 *(Reconciliation) Let us be given the host tree and the parasite tree with vertex and edge sets* $(V(H), E(H))$ *and* $(V(P), E(P))$*, respectively. Let their set of leaves be denoted by* $L(H)$ *and* $L(P)$*. Morever, let* $\phi : L(P) \to L(H)$ *be a mapping between the tips of the two trees telling us the associations of the hosts and parasites at the leaves. A map* $\Phi : V(P) \to V(H) \cup E(H)$ *is called a reconciliation of the two trees if it satisfies the following conditions:*

1. *It is consistent with the tip mapping, i.e. for all* $l \in L(P)$ *we have that* $\Phi(l) = \phi(l)$ *holds.*
2. *It is consistent with the ancestral relationships in the two trees, i.e. for all non-leaf nodes* p *in the parasite tree and their pair* $\Phi(p)$ *it must hold that for all descendants* q *of the node* p*,* $\Phi(q)$ *cannot be an ancestor of* $\Phi(p)$*. This property is called weak time consistency.*

As the second property of a reconciliation is called weak time consistency, it is natural that there exists a stronger concept called strong time consistency. While weak time consistency requires that ancestral relationships are preserved by the reconciliation considering each event separately, strong time consistency requires the same but considering every event at once. In order to formally define it, we first need to define time maps.

Definition 2.21 *(Time map) Let* $T = (V, E)$ *be a rooted binary tree. We say that the map* $\kappa_T : V \to [0, \infty)$ *is a time map for* T *if for all* $x, y \in V$ *with* $x \prec_T y$ *we have that* $\kappa_T(x) < \kappa_T(y)$*, where* $x \prec_T y$ *means that* y *is an ancestor of* x*.*

This means that we can assign a timepoint to each vertex of the tree such that for each node it holds that its assigned time is smaller than its ancestors. Now with the help of time maps we can define strong time consistency of reconciliations. We will keep the notation of Definition 2.20

Definition 2.22 *(Strongly time consistent reconciliation) A reconciliation map* $\Phi : V(P) \to V(H) \cup E(H)$ *is strongly time consistent if it satisfies the conditions from Definition 2.20 and in addition, there exist time maps* κ_H *for the host tree and* κ_P *for the parasite tree such that for each* $u \in V(P)$ *the following hold:*

1. *If* $\Phi(u) \in V(H)$, *then* $\kappa_P(u) = \kappa_H(\Phi(u))$

2. *If* $\Phi(u) = (x, y) \in E(H)$ *with* $x \prec_H y$, *then* $\kappa_H(x) < \kappa_P(u) < \kappa_H(y)$

The first condition requires that cospeciation events have the same time, and the second property requires that the time of any nodes in the parasite tree that are mapped to an edge of the host tree is between the times at the endpoint of the given edge.

Note that strong time consistency implies weak time consistency, but the converse is not true. A good example for that is found in Figure 5 in [LH22].

How the reconciliation function Φ maps the nodes of the parasite tree onto the host tree depends on the underlying model that is assumed, i.e. what kind of events does the model allow. Most of the commonly used reconciliation methods are based on the DTL model, where the letters correspond to different events: duplication(D), transfer(T) and loss(L). Duplication occurs when a node (speciation) of the parasite tree and both of its children nodes are mapped to the same edge of the host tree. When there is a speciation node in the parasite tree such that one of its children is mapped to the same edge (donor) as the parent but the other is mapped to another edge (recipient) of the host tree, a (horizontal) transfer has happened. Loss events cannot be defined in such a way as lost lineages do not appear in the reconciled scenario. They can only be assigned retrospectively. Finally, there is another event that is always allowed, which is cospeciation, where a non-leaf parasite node p is mapped to a node h of the host tree, and the children of p are mapped to the children (or on the edges from h to its children) of h.

How these events are assigned depends on the goal of the reconciliation method in question. The two main types are maximum parsimony (MP) and likelihood-based methods. Maximum parsimony methods first assign a cost for each of the events mentioned above, and then they try to find the scenario with the lowest possible cost that leads to the given separate trees and tip matching. Likelihood-based, or in other words, probabilistic methods assign rates at which the events are happening, and try to find the scenario with the highest possible likelihood given the separate trees, the tip matching and the rates. It is also possible to estimate the parameters while simultaneously reconciling the two trees.

Maximum parsimony is the more commonly used framework. Given the event costs, it is very easy to calculate the overall cost of an evolutionary scenario. However, MP has some limitations that one needs to keep in mind. First, different event costs may result in different optimal reconciliations, from which we may arrive to different conclusions regarding the evolutionary history. There is no standard method that would give us 'good' event costs, however it is possible to divide the cost space into sections where MP reconciliations are essentially equivalent [LHWBK14]. We would like to note here that event costs are directly related to event frequencies, as events of lower costs are preferred and hence occur more often. Another limitation is the time-consistency of the MP reconciliations. Namely, it has been shown (for undated trees) that finding weak (or strong) time-consistent reconciliations is NP-hard [OFCLH11], [THL11], [HT19]. Therefore one needs to use efficient polynomial-time dynamic programming algorithms. However, there is no guarantee that MP reconciliations are weakly or strongly time consistent.

Nevertheless, there are methods to test for this. As a third challenge we need to mention the (potentially) very large number of MP reconciliations. It has been proven that the number of such reconciliations can grow exponentially as the size of the trees grow [CDW12], and these reconciliations can be significantly different from each other. Wang et al. [WMSS20] proposed different ways of defining equivalence classes on the space of MP reconciliations and proved that these equivalence classes can be computed efficiently. However, despite dividing the large space of MP reconciliations into smaller classes, the number of reconciliations in these classes can still be enormous. Rather than dividing reconciliations into different equivalence classes, Nguyen et al [NRBS13] proposed a method to select a single best representative of the large space of optimal reconciliations, which is called the median reconciliation. It is based on the symmetric set distance of two reconciliations, i.e. the number of events that appear in exactly one of the two considered scenarios.

There are many implementations for searching for maximum parsimony reconciliations. The three most widely used packages for host-parasite cophylogenetic analyses are `TreeMap` [Pag95], `CoRE-PA` [MMW10] and `Jane` [CFOLH10], and we also need to mention the more recent softwares `Capybara` [WMSS20] and `eMPRess` [SYL+21] (which is essentially the successor of `Jane`). These methods differ from each other regarding handling event costs, runtime, and there are also differences regarding allowing dated trees or multiple-host parasites (when a single parasite species can infect multiple host species).
`TreeMap` identifies MP reconciliations that are Pareto-optimal with the help of a heuristic efficient algorithm. It guarantees a strongly time consistent solution, but this solution may not be globally optimal due to the NP-hardness of the problem and the use of heuristic algorithm(s). To avoid very large runtimes, it is possible to set an (upper) bound for the number of transfer events. This leads to less accurate results compared to other softwares.
`CoRE-PA` automatically selects event costs by optimizing a (mathematical) objective function but is not clear whether the costs are biologically feasible. The method also supports dating information on the trees, not with exact times, but the relative order of the different nodes. The solution is guaranteed to be of optimal cost due to using very efficient algorithms, but it is not guaranteed to be time consistent.
When using `Jane`, all of the costs need to be selected by the user. It also offers support for dating information of trees similarly to `CoRE-PA`. Similarly to `TreeMap`, the solution is guaranteed to be strongly time consistent but not guaranteed to be of globally minimal cost. It is the only method that allows multi-host parasites (i.e. the possibility that a single parasite species can infect multiple host species).
`Capybara` also allows the user to select all event costs, and identifies the equivalences classes of MP reconciliations based on Nguyen et al. [NRBS13]. It automatically makes a list which consists of reconciliations from each class, with each reconciliation guaranteed to be of optimal cost but not guaranteed to be time-consistent.
`eMPRess` fixes the cost of cospeciations to 0 while the other costs can be freely selected. The equivalence classes of the MP reconciliations are displayed graphically and allows the user to obtain a median reconciliation from each class. These reconciliations are guaranteed to be of

globally optimal cost but not guaranteed to be time consistent. Nonetheless, it is indicated whether the reconciliations are weakly or strongly time consistent.
As the DTL model is mathematically equivalent both in species tree-gene tree and host tree-parasite tree contexts, maximum parsimony reconciliations under the DTL model developed for inferring gene trees within species trees can also be used for cophylogenetic studies, such as `ecceTERA` [JCS+16], `RANGER-DTL` [BKKK18], `EUCALYPT` [DBS+15] or `Mowgli` [DSG+10].

The other class of tree reconciliation methods belong to the probabilistic methods, where instead of assigning costs to events and trying to minimize the cost of a scenario, a random model is assumed for the evolution, and we look for the reconciliation that maximizes the probability that at the present we see exactly what we have obtained from the samples. Calculating these probabilities is hard as it requires to consider all of the possible evolutionary scenarios whose number can be extremely large. The `ALE` (Amalgamated Likelihood Estimation) method [STLD13] is the most commonly used such method. A general overview of different methods and the challanges and limitations of inferring gene trees with species tree is found in [STDB14].

We would like to note, that there are only very few reconciliation methods that allow for transfers to extinct or non-sampled lineages (i.e. the ghost lineage). Only `ALE`, `ecceTERA`, and a renewed version of `RANGER-DTL` called `Ranger-DTLx` [WB21] allow them, as well as a very recent method called `Synesth` [DEM24] which considers events on sets of multiple genes (syntenies) instead of single genes.

Finally, here we name some softwares that are (or were) also used for detecting horizontal transfers rather than for providing optimal reconciliations: `LatTrans` [HL01], `HorizStory` [MCDB05], `T-REX` [BPM10], [BDM12].

Implicit methods. Implicit methods do not necessarily require tree reconstruction since they compare evolutionary distances or similarities in sequences. It makes them simpler and faster than explicit methods, but they come with some limitations as well, namely there can be conflicts between the true underlying tree and the recovered evolutionary distances. A simple implicit method is to look for sequence matches in distantly related species, usually with the help of BLAST (basic local alignment search tool) [AGM+90], [AMS+97], however, this method is limited to finding relatively recent horizontal transfers. If the distance (time until most recent common ancestor) of two genes is significantly smaller or larger than the distance of the two corresponding species, then it is possible that a horizontal transfer took place between the two species. A recent paper of Schaller et al. [SLS+21] uses this idea to construct a later-divergence-time (LDT) graph, which they characterize as a mathematical object, and they provide a polynomial-time algorithm that gives back a scenario that explains the obtained LDT graph.

2.4.2 Horizontal transfers in our models

Now let us go back to our models and demonstrate how we characterize horizontal transfers and what we mean by identifiability.

By the methods described previously, we know that the host tree and the parasite tree can be reconstructed from DNA sequence data under both backward models. What we are interested in is to describe the joint evolutionary history of the sampled host and parasite species. To do so, one would need to give a pairing at each timepoint between the branches of the two trees to identify the time of all horizontal transfers alongside the donor and recipient branches taking part in them. When referring to branches of the (separate) trees, we will use the partitions corresponding to them as defined in Definition 2.8.

Definition 2.23 *(Horizontal transfer event) A horizontal transfer (HT) event E is denoted and uniquely characterized by the triple $E = (T, b_d, b_r)$, where T is the time of the event, and b_d and b_r are the donor and recipient branches of the host tree, respectively. Moreover, we say that a horizontal transfer event is fully identifiable if and only if all the three elements of the triple can be uniquely identified based on only the information coming from the host tree and the parasite tree.*

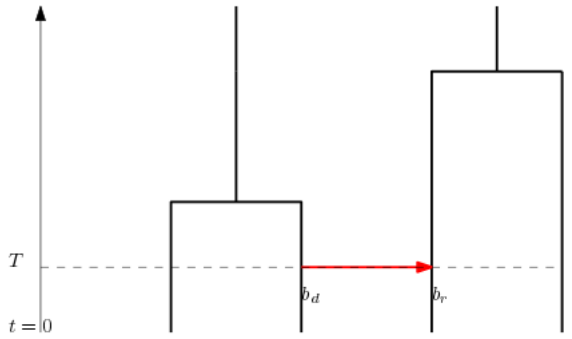


Figure 12: Characterization of a HT event

Note that even if a HT event is not fully identifiable, there are cases when one might actually have some information about the event, just not complete information. For example, we might know the time of the transfer and the two host branches participating in it, but we do not know the direction of the transfer. Figure 13 gives some examples where one can obtain only partial information about the HT event:

In Figure 13(a), we see that there is a horizontal transfer between branches 1 and 2, and right after the two branches coalesce with each other. In this case the direction of the transfer is not identifiable, since the transfer with the opposite direction yields the same host and parasite trees. In Figure 13(b) the direction of the first (closer to the leaves) transfer is again not identifiable, similarly to the previous case. This causes unidentifiability of the donor branch of the second transfer, as the donor branch of the second transfer should be the same branch that was the recipient of the first transfer. In Figure 38(c) we see again two transfers. The first one, drawn by blue is actually identifiable, since the time and the two participating branches are directly determined by checking the first merging event in the parasite tree, and the direction of it can be decided by looking at the other merging events in the parasite tree and checking which direction is compatible with them. Now let us look at the second transfer. After the first

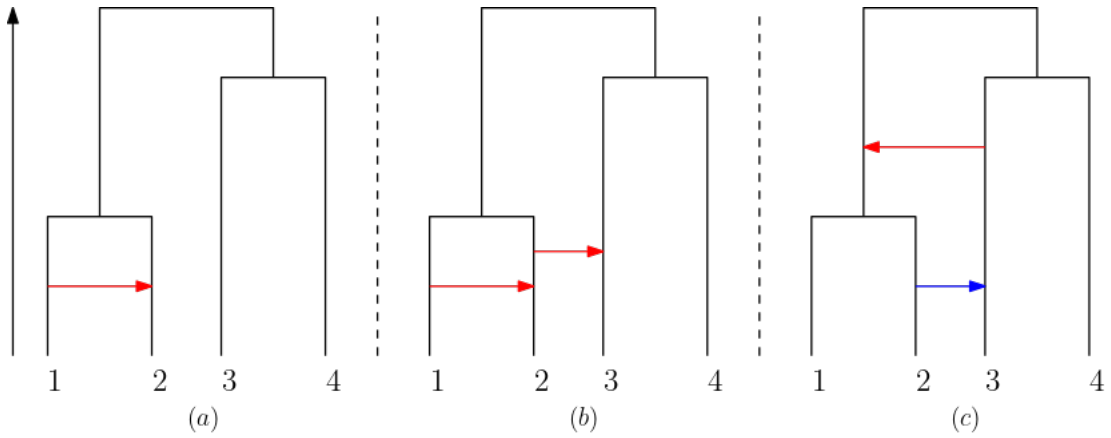


Figure 13: Examples for not fully identifiable HT events

transfer event, host branch 3 will no longer be associated with any branch of the parasite tree, so the time of the second transfer will not appear as a merging time neither in the parasite, nor in the host tree. However, as we do see a merging event in the parasite tree at the same time when host branches 3 and 4 merge, we know that branch 3 must have been a donor of a transfer event in order to have a parasite lineage associated with it by the time of the previously mentioned merging event. We also know that this transfer must have happened after host lineages 1 and 2 have coalesced, but it is not possible to determine the exact time of the transfer based on the host tree and parasite tree separately.

Now we will state a small result about identifiability of a single horizontal transfer event in a Kingman-tree.

Proposition 2.24 *Suppose that the host tree is an N-Kingman tree, and all the differences between the host and parasite tree are caused by one single horizontal transfer event. Then this horizontal transfer event is not fully identifiable with probability*

$$P_N = \frac{1}{N-1}\left(1 + \sum_{k=3}^{N} \frac{2}{3(k-1)}\right)$$

Proof. As the host tree is an N-Kingman tree, we know that each pair of braches coalesces independently of the other with a constant rate 1, and the time happening between the i-th and $(i-1)$-th coalescence is an exponentially distributed random variable with rate $\binom{N-i+1}{2}$ for all $i =, 1, ..., N-1$ (with the 0th coalescence event meaning the starting point at the leaves of the tree).

First we will determine the probability p_k that the transfer is not identifiable, given that it happens when there are currently k branches of the host tree for all $k = N, N-1, ..., 2$. Note that the transfer is not identifiable exactly in those cases when the two branches participating in the transfer do not coalesce with other host lineages before coalescing with each other. Since the transfer rate is equal for each pair of species, each pair of branches is equally likely to be

participating in the transfer event. Clearly for $k = 2$ we have that $p_2 = 1$. For $k = 3$ we have $p_3 = 1/3$ since the transfer is not identifiable only if the two branches participating in the transfer are those who will coalesce next. For $k \geqslant 4$ we have the following recursion for p_k:

$$p_k = \frac{1}{\binom{k}{2}} + \frac{\binom{k-2}{2}}{\binom{k}{2}} \cdot p_{k-1}.$$

Here the first term is the probability that the two branches participating in the transfer event are the next pair to coalesce in the host tree. Moreover, note that if only one of the two branches taking part in the transfer event would participate in the next coalescence event of the host tree, then we would be able to determine the direction of the transfer. If no one of the two branches involved in the transfer event participate in the next coalescence event (this event has a probability of $\binom{k-2}{2}/\binom{k}{2}$), then after this coalescence event we will only have $k-1$ lineages left, with the same two branches not coalescing with any other branches before coalescing with each other with probability p_{k-1}.

With induction it is straightforward to prove that for $k \geqslant 3$ we have $p_k = \frac{2}{3(k-1)}$. We stated earlier that it holds for $k = 3$. Let us assume that it hold for some k, and now for $k+1$:

$$\begin{aligned} p_{k+1} &= \frac{1}{\binom{k+1}{2}} + \frac{\binom{k-1}{2}}{\binom{k+1}{2}} \cdot p_k = \frac{2}{(k+1)k}\left(1 + \frac{(k-1)(k-2)}{2} \cdot \frac{2}{3(k-1)}\right) \\ &= \frac{2}{(k+1)k}\left(1 + \frac{k-2}{3}\right) = \frac{2}{(k+1)k} \cdot \frac{k+1}{3} = \frac{2}{3k} \end{aligned}$$

Now we will find the probability that the transfer actually happens when there are k branches in the tree, for all $k = 2, 3, ..., N$. This depends on the number of transfers happening in each such sections of the host tree, so we will now determine the distribution of them. If we have currently k branches, there are a total number of $\binom{k}{2}$ pairs of branches, which are coalescing independently of each other at rate 1, hence the time until the first coalescence event is exponential with rate $\binom{k}{2}$. Transfers happen also independently from each other and independently from the coalescences at rate τ per pair, so the time until a transfer happens is also exponential, but with rate $\tau\binom{k}{2}$. Using the independence and the memoryless property of the exponential distribution, the probability that there are exactly i transfers happening before the first coalescence event is $\left(\frac{\tau}{1+\tau}\right)^i \cdot \frac{1}{1+\tau}$. Note that this number does not depend on k, and it gives exactly the distribution of a (shifted) geometric distribution with parameter $\frac{1}{1+\tau}$. It is well known that it has expectation $\frac{1-\frac{1}{1+\tau}}{\frac{1}{1+\tau}} = \tau$. So what we have found out is that for each k, the number of transfers when there are k branches in the host tree is an i.i.d. (shifted) geometric distribution with parameter $\frac{1}{1+\tau}$. That means that for each $k = 2, 3, ...N$ it is equally probable that the one transfer happens when there are exactly k branches in the tree. Hence, the probability P_N of the event that the one single transfer is not identifiable is

$$P_N = \frac{1}{N-1}\sum_{k=2}^{n} p_k = \frac{1}{N-1}\left(1 + \sum_{k=3}^{N} \frac{2}{3k}\right). \qquad \square$$

Note that for large N, using the approximation of the harmonic numbers, we have the following approximation for P_N :

$$P_N \sim \frac{2}{3} \cdot \frac{\log N}{N} \to 0 \text{ as } N \to \infty,$$

from which we can conclude that there is always a positive probability that a transfer event is not identifiable, even though it converges to 0 as the tree gets large. But of course, under both backward models, the probability that there is only 1 transfer in the whole tree also converges to 0 as the tree gets large, as the expected number of transfers in the whole tree is $(N-1) \cdot \tau$.

Even if the single horizontal transfer event is not fully identifiable, we can always determine the time it has happened and the two branches participating in the transfer, since the most recent horizontal transfer always induces a merging event in the parasite tree.

Now we could ask the same question for two transfers: what is the probability that both of them are fully identifiable? Again, we can try to calculate the probability of the complementary event, namely that at least one of the transfers are not fully identifiable. Although the question is the same as we have answered before for a single transfer, it gets much more complicated as now the answer depends even more on the exact topology of the underlying host tree. Therefore, we will give an upper bound on this probability in question.

Let us assume that the two transfers are happening when there are exactly k and l branches of the host tree, with $k \geqslant l$. First let us suppose that $l = 2$, in that case it is obvious that the second transfer cannot be identifiable. This happens with probability $\frac{1}{N-1}$ since the other transfer can be at any part of the tree.

Now let us consider the case when $N \geqslant k = l > 2$ holds. As transfers are happening independently from each other, this happens with probability $\frac{1}{(N-1)^2}$. Now let us consider first the transfer that is closer to the leaves of the tree (i.e. the more recent one). From Proposition 2.24, we know that it is not identifiable with probability $p_k = \frac{2}{3(k-1)}$. Otherwise it is a transfer event that would be identifiable if it was the only transfer event in tree. (Note the wording here: it can happen that the second transfer makes it unidentifiable, however, we will ignore such cases as we are giving a lower bound for the probability that at least one of them is not identifiable). In this case, the second transfer will not be identifiable if it affects the empty line (since in this case there is no merging in the parasite tree), or if it is between two branches that will not coalesce with other branches before they coalesce with each other (like in the case of 2.24). The probability that the empty line is involved in the second transfer is $\frac{k-1}{\binom{k}{2}} = \frac{2}{k}$ as the transfers happen independently at the same rate for each pair. If it does not involve the empty line, then it is non-identifiable with probability p_k. Considering each possibility, we get the probability that at least one of them is not identifiable is

$$\frac{1}{(N-1)^2}\left(p_k + (1-p_k)\left(\frac{2}{k} + \frac{k-2}{k} \cdot p_k\right)\right) = \frac{1}{(N-1)^2} \cdot \frac{2(15k^2 - 38k + 25)}{9(k-1)^2 k}$$

Now let us consider the case of $N \geqslant k > l > 2$. This happens with probability $\frac{2}{(N-1)^2}$. Similarly to the previous case, the more recent transfer is not identifiable with probability $p_k = 2/3(k-1)$.

If it is identifiable, then we try to calculate the probability that the second transfer (happening when there are l branches of the tree) is not identifiable. The first case we will examine is when the empty branch resulting from the first transfer has coalesced with someone else, i.e. it is non-empty anymore when the second transfer is happening. In that case, the only possibility for the second transfer to be non-identifiable is the same as before: it needs to happen between two branches that do not coalesce with other branches before coalescing with each other. From Proposition 2.24 we know that this happens with probability $p_l = 2/3(l-1)$. If it is not the case, transfers to/from the empty lineage will also result in non-identifiable transfers. If there are currently l branches, the empty lineage participates in the transfer with probability $2/l$, in that case we automatically have a non-identifiable second transfer. If the second transfer does not involve the empty lineage, it will be non-identifiable with probability p_l from the previous case. So now we need to calculate the probability q_l^k that the empty lineage has not participated in any coalescence event that happened between the two transfers. During this time, there are $k-l$ coalescence events. The first one does not involve the empty branch with probability $\frac{\binom{k-1}{2}}{\binom{k}{2}}$. After that, there are $k-1$ branches are left, and the next coalescence does not involve the empty line with probability $\frac{\binom{k-2}{2}}{\binom{k-1}{2}}$. Continuing until the $(k-l)^{th}$ coalescence event, the empty line is not involved in any of them with probability

$$q_l^k = \frac{\binom{k-1}{2}}{\binom{k}{2}} \cdot \frac{\binom{k-2}{2}}{\binom{k-1}{2}} \cdot \ldots \cdot \frac{\binom{k-l}{2}}{\binom{k-l+1}{2}} = \frac{\binom{k-l}{2}}{\binom{k}{2}} = \frac{(k-l)(k-l-1)}{k(k-1)}$$

Again, considering each possibility, we have that at least one transfer is not identifiable has a probability of at least

$$\frac{2}{(N-1)^2}\left(p_k + (1-p_k)\left(q_l^k\left(\frac{2}{l} + \frac{l-2}{l}\cdot p_l\right) + (1-q_l^k)p_l\right)\right)$$

Now, we need to sum over all possible values of k and l, from which we get that the probability that at least one of the transfers is not identifiable is at least

$$\frac{1}{(N-1)^2} + \sum_{k=3}^{N} \frac{1}{(N-1)^2} \cdot \frac{2(15k^2-38k+25)}{9(k-1)^2k}$$
$$+ \sum_{2<l<k\leqslant N} \frac{2}{(N-1)^2}\left(p_k + (1-p_k)\left(q_l^k\left(\frac{2}{l} + \frac{l-2}{l}\cdot p_l\right) + (1-q_l^k)p_l\right)\right)$$

Note that this is only a lower bound for the probability that at least one of the two transfers are not fully identifiable, and even this bound has no closed form.

For a higher number of transfers this probability gets even more complicated and quasi impossible to compute it.

One could perform simulations to estimate these probabilities (which are actually conditional probabilities, conditioned on the number of transfers happening), but this is not the main focus of this thesis.

3 Estimation of the horizontal transfer rate

Although our main goal is to reconcile the two separate trees, before doing so it is good to have some prior information what the joint history might look like. One such indicator would be to estimate the horizontal transfer rate τ as it would already tell us how aligned the trees would be: low τ means that there were only a few events and the trees are closely aligned with each other, while a high transfer rate would mean that the two trees are not aligned with each other so well. Moreover, we can make use of a good estimator of τ in the reconciliation algorithm itself by making random choices: with the approximate knowledge of τ one can make better informed decisions even when the information available is not enough to determine the true scenario with certainty. This chapter is devoted to introducing such an estimator and investigating its mathematical properties.

There exist several methods in order to estimate the rate of horizontal gene transfer in different settings. Mishra et al. [MKV+21] develop a system of ODEs describing the dynamics of multiple genes in a given species tree, and the parameters are estimated using the lowest root mean square error. Ge, Wang and Kim [GWK05] also work in the framework of a fixed species tree and with different gene trees. They introduce a statistical test where they test whether the trees are different (according to some specified metric), and whether the difference can be significantly better explained via an HT event rather than only duplications and losses of genes. Several papers estimate the transfer rate using gene presence or absence patterns, see for example Zamani-Dahaj et al [ZDOKH16]. The method of Linz, Radtke and von Haeseler [LRvH07] assumes a setting very similar to ours. They suppose that they also know the species tree and the gene tree, and suppose that differences between the two trees are explained only by horizontal transfers. They work with a maximum likelihood estimator, where the likelihood function is approximated from a number of 100 000 simulation scenarios. The main difference in our case will be that our estimator can be calculated from one single realization of the two separate trees, and the large number of leaves will guarantee that the estimator is close to the true value.

3.1 Defining the estimator

Now, let us suppose that the host tree is given by coalescence times $0 = T_{H,0} < T_{H,1} < ... < T_{H,N-1} < \infty$ and partitions $\{\{1\},\{2\},...,\{N\}\} = \mathcal{P}_0 \prec \mathcal{P}_1 \prec ... \prec \mathcal{P}_{N-1} = \{\{1,2,...N\}\}$, and the parasite tree is given by $0 = T_{P,0} < T_{P,1} < ... < T_{P,N-1} < \infty$ and partitions $\{\{1\},\{2\},...,\{N\}\} = \mathcal{Q}_0 \prec \mathcal{Q}_1 \prec ... \prec \mathcal{Q}_{N-1} = \{\{1,2,...N\}\}$. Let us suppose that the union of all coalescence times is given by the sequence $0 = T_0 < T_1 < ... < T_{k_N}$ with $N-1 \leqslant k_N \leqslant 2(N-1)$ denoting the size of the union.

Let the (true) number of HT events in the interval $[T_{H,i-1}, T_{H,i})$ be denoted by H_i for $i = 1,2,...,N-1$. As seen in the proof of Proposition 2.24, the number of HT events on each time interval $[T_{H,i-1}, T_{H,i})$ for $i = 1,2...,n-1$ is given by independent (shifted) geometric random

N	comp. inf.	sep. trees
100	0.1515	0.1414
150	0.067	0.534
200	0.1105	0.0954
250	0.0843	0.0843
300	0.1103	0.0969
350	0.126	0.106
400	0.0927	0.0877
450	0.0935	0.089
500	0.114	0. 104

(a) $\tau = 0.1$

N	comp. inf.	sep. trees
100	0.5050	0.3030
150	0.55	0.32
200	0.522	0.33
250	0.522	0.349
300	0.468	0.341
350	0.544	0.349
400	0.55	0.366
450	0.496	0.327
500	0.475	0.306

(b) $\tau = 0.5$

N	comp. inf.	sep. trees
100	2.09	0.6868
150	2.14.	0.704
200	1.864	0.688
250	1.815	0.650
300	1.983	0.688
350	1.876	0.679
400	1.99	0.651
450	1.868	0.625
500	1.78	0.627

(c) $\tau = 2$

Table 1: Estimating τ from complete information versus from information coming from separate trees

variables with parameter $\rho = \frac{1}{1+\tau}$. This, together with the law of large numbers implies that

$$\frac{1}{N-1}\sum_{i=1}^{N-1} H_i \xrightarrow{\text{a.s.}} \tau \quad \text{as } N \to \infty, \tag{3.1}$$

since $\mathbb{E}(H_i) = \tau$ holds for all $i = 1, 2, ..., N-1$.

Of course, based on the information coming from the separate trees we cannot determine the quantities H_i, since some transfers cannot be inferred from the coalescence times alone. To be more precise, only those HT events are captured by the coalescence times in the parasite tree, where both the donor and the recipient have a branch of the parasite tree associated to them. If we only count those as horizontal transfers and take their average similarly as in (3.1), we will end up with an underestimation of τ. The extent of the underestimation depends on the value of τ: we can observe that as $\tau \to 0$, the second estimator becomes more accurate, which is also no surprise since $\tau \to 0$ would mean there are a very few transfers in the whole tree, which means that they are all detectable with high probability.

However, for $\tau > 1$ there will be a huge difference between the two estimators. This is due to the fact that at most we can detect $N-1$ horizontal transfers from the separate trees only, however, the expectation of the total number of horizontal transfers in the whole tree is $(N-1)\tau$.

These effects are demonstrated in Table 1 that is based on simulated data.

Thus, we need a different approach in order to have a good estimator for τ. We would like to emphasize that throughout this chapter we work under the assumptions of Backward model I (see Definition 2.13). Before we proceed, we will introduce some processes and quantities that will play a crucial role in the estimation.

For a fixed N, and for each $t \geqslant 0$, let the number of host lineages at time t be denoted by h_t^N and the number of parasite lineages denoted by p_t^N. Moreover, let $C_t^N = (h_t^N, p_t^N)$ denote the pair of number of lineages at time t.

Proposition 3.1 *The process $(C_t^N)_{t\geqslant 0} = (h_t^N, p_t^N)_{t\geqslant 0}$ is a (homogeneous) Markov process with state space $\mathcal{S}_{\mathcal{N}} = \{1, 2, ..., N\} \times \{1, 2, ..., N\}$ with transition rates $q_{ss'}$ from state $s = (k, l)$ to $s' = (k', l')$ as follows:*

$$q_{ss'} = \begin{cases} \tau \cdot \binom{l}{2} & \text{if } k' = k,\ l' = l-1 \text{ and } k \geqslant l, \\ \binom{k}{2} - \binom{l}{2} & \text{if } k' = k-1,\ l' = l \text{ and } k > l, \\ \binom{l}{2} & \text{if } k' = k-1,\ l' = l-1 \text{ and } k \geqslant l, \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

with the convention that $\binom{1}{2} = 0$.

Proof. The rate at which two parasite lineages merge due to horizontal transfer is $\tau \cdot \binom{l}{2}$ and the rate at which two host lineages merge (coalesce) is at rate $\binom{k}{2}$. If there is a merging in the host tree, the two merging host lineages are selected uniformly at random, meaning that both lineages have an infection with probability $\frac{\binom{l}{2}}{\binom{k}{2}}$ and thus we see a decrease in the number of the parasite lines as well. Otherwise, only the number of the host lineages will decrease. □

3.2 Remark. Note that the process starts from state (N, N) and arrives to state $(1, 1)$, i.e. $(h_0^N, p_0^N) = (N, N)$ and $(h_t^N, p_t^N) = (1, 1)$ for all $t \geqslant T_{k_N}$. Also note that the times at which there is a jump in the process $(C_t^N)_{t\geqslant 0}$ from one state to another are exactly the times at which we see a merging event in any of the two trees, i.e. the times $T_0, T_1, ..., T_{k_N}$.

Observe further, that conditional on the jump chain of the process $(C_t^N)_{t\geqslant 0}$, i.e. conditional on the values $(h_{T_i}^N, p_{T_i}^N)$ for all $i = 0, 1, 2, ..., k_N - 1$, the times between the jumps $T_{i+1} - T_i$ are all independent, exponentially distributed random variables. This is because the next merging event is a minimum of two independent exponentials, one for the time until the next detectable HT event, and one for the next host coalescence event, with $T_{i+1} - T_i$ having parameter $a_i^N + \tau \cdot b_i^N$ with

$$a_i^N := \binom{h_{T_i}^N}{2} \text{ and } b_i^N := \binom{p_{T_i}^N}{2}, \tag{3.3}$$

again with the convention $\binom{1}{2} = 0$. In other words, a_i^N gives the number of all possible pairs of host lineages during the time interval $[T_i, T_{i+1})$, and similarly, b_i^N gives the number of all possible pairs of parasite lineages during the time interval $[T_i, T_{i+1})$. We will only work with the time differences up to $T_{N-1} - T_{N-2}$ to since after T_{N-1} there might be only one parasite lineage and the parameter τ that we would like to estimate does not appear in the distribution of the further time differences. Of course, it would also be possible to consider these differences up to the latest time when $p_t^N > 1$ holds, as this would give us more information. Nevertheless we will stick with considering the first $N - 1$ time intervals between the jumps.

To summarize what we have so far, conditional on the jump chain, for each fixed N we have $N-1$ independent, exponentially distributed random variables $Z_i^N := T_{i+1} - T_i$ with parameter $(a_i^N + \tau \cdot b_i^N)$, and from this we want an estimator for τ. Let $X_i^N := b_i^N \cdot Z_i^N$, then X_i^N is exponentially distributed with parameter $c_i^N + \tau$ with $c_i^N := a_i^N / b_i^N$. This can be checked by calculating for example the survival function of X_i^N, that

$$\mathbb{P}(X_i^N > t) = \mathbb{P}(b_i^N Z_i^N > t) = \mathbb{P}(Z_i^N > t/b_i^N) = \mathrm{e}^{-(a_i^N + \tau \cdot b_i^N)\frac{t}{b_i^N}} = \mathrm{e}^{-(\frac{a_i^N}{b_i^N} + \tau)t} = \mathrm{e}^{-(c_i^N + \tau)t},$$

which is exactly the survival function of an exponential random variable with parameter $(c_i^N + \tau)$. We once again emphasize the dependency of all the random variables on N and on the pairs (a_i^N, b_i^N) which are directly related to the jump chain of $(C_t^N)_{t \geqslant 0}$, which is implicitly depending on τ as well. Another important thing to note is that when one goes from N to $N+1$, in contrast to what one usually obtains when trying to find an estimator, we do not simply add a new value to the sequence of realizations, but in fact the whole experiment changes and we will have a whole new realization of this new experiment.

Note that the time differences $Z_i^N = T_{i+1} - T_i$ and the pairs $(h_{T_i}^N, p_{T_i}^N)$ can be directly read off from the separate trees, and so given the separate trees one can directly obtain the realizations of the random variables X_i^N for $i = 0, 1, ..., N-2$. Independence of the X_i^N follow from the independence of the Z_i^N. Thus, we now consider the following abstract setting:
For all N we have a sample of size $N-1$ consisting of independent, but not identically distributed random variables X_i^N for $i = 0, 1, ..., N-2$ with $X_i^N \sim \mathrm{Exp}(c_i^N + \tau)$ with the constants c_i^N fixed. In the following we are going to calculate the maximum likelihood estimator (MLE) of τ for each N. Whenever talking about functions of the parameter to be estimated, the variable will be denoted by θ, while the true value of the parameter will remain denoted by τ. The likelihood function is given as

$$\mathcal{L}_N(\theta) := \mathcal{L}_N(\theta | x_0, x_1, .., x_{N-2}) := \prod_{i=0}^{N-2} (c_i^N + \theta) \mathrm{e}^{-(c_i^N + \theta)x_i}, \tag{3.4}$$

where x_i denotes the realization of X_i^N, and its domain is the half line $\theta > 0$ as we know that the horizontal transfer must be positive. The log-likelihood is consequently

$$l_N(\theta) := l_N(\theta | x_0, x_1, ..., x_{N-2}) := \log \mathcal{L}_N(\theta) = \sum_{i=0}^{N-2} \log(c_i^N + \theta) - \sum_{i=0}^{N-2} (c_i^N x_i + \theta x_i). \tag{3.5}$$

The first and second derivative (w.r.t θ) is given by

$$l_N'(\theta) = \sum_{i=0}^{N-2} \frac{1}{(c_i^N + \theta)} - \sum_{i=0}^{N-2} x_i \qquad \text{and} \qquad l_N''(\theta) = \sum_{i=0}^{N-2} -\frac{1}{(c_i^N + \theta)^2}. \tag{3.6}$$

As the second derivative is negative everywhere, any zero of the first derivative will actually correspond to a local maximum of the log-likelihood. It is easy to see that $l_N'(\theta) = 0$ if and only if

$$\sum_{i=0}^{N-2} \frac{1}{(c_i^N + \theta)} = \sum_{i=0}^{N-2} x_i. \tag{3.7}$$

Since $\theta > 0$ by definition and the LHS of the equation is strictly monotone decreasing in θ, there is at most one possible estimator for τ. If $\sum_{i=0}^{N-2} x_i \geqslant \sum_{i=0}^{N-2} 1/c_i^N$ then there is no solution, since the RHS is bigger than the biggest possible value of the LHS. Otherwise there is exactly one positive solution, since the LHS is continuous in θ. This unique solution $\hat{\tau}_N$ will be the MLE of τ. Note that usually it cannot be computed symbolically, it can only be approximated numerically.

Naturally, this estimator needs to be at least consistent for it to be a good estimator. Furthermore, asymptotic normality allows assessment of its efficiency. To check whether consistency and asymptotic normality holds, we will use the ideas from Leroy et al. [LDTB16]. However, the results and proofs have to be modified due to the difference in our and their setting. The main difference is that they have the "classical" setting for realizations, meaning that they repeat the same experiment every time when adding a new realization to the sample, while we have to do a completely new experiment to have a new realization and even discard our old realizations at each step. Some of their assumptions do not hold in our case so that additional arguments are needed. Furthermore, even though the constants c_i^N are assumed to be known, they are actually not deterministic but random variables that are functions of the stochastic process $(C_t^N)_{t\geqslant 0}$ (and implicitly also τ) defined in Proposition 3.1.

3.2 Existence

First of all, we will discuss the question whether a solution to (3.6) exists in our setting (for N big enough). In our setting where both the c_i^N and the x_i^N are (independent) realizations of the same underlying random model, we replace the (deterministic) x_i with the random X_i^N and also consider c_i^N to be random. Thus, we want that the following equation

$$\sum_{i=0}^{N-2} \frac{1}{(c_i^N + \theta)} = \sum_{i=0}^{N-2} X_i^N \tag{3.8}$$

has a unique solution with high probability for N big enough.

We start by proving some technical lemmas that we use throughout when proving the consistency and asymptotic normality of our estimator. As the X_i^N have a certain distribution given the constants $\boldsymbol{c}^N = (c_i^N)_{i=0,1,\dots,N-2}$, the probabilities and expectations that appear in the calculations will be conditional on the vector $\boldsymbol{c}^N$. Throughout this chapter, conditional probabilities and expectations conditional on $\boldsymbol{c}^N$ will be denoted by $\mathbb{P}_c$ and $\mathbb{E}_c$, respectively. For each N and θ, the density of X_i^N will be denoted by f_i^N, i.e. for all $x \geqslant 0$ we have

$$f_i^N(x, \theta) = (c_i^N + \theta)\mathrm{e}^{-(c_i^N+\theta)x}. \tag{3.9}$$

We will frequently work with the function $\log f_i^N(x, \theta)$ and its first three derivatives, so here

we calculate them for later references:

$$\log f_i^N(x,\theta) = \log(c_i^N + \theta) - (c_i^N + \theta)x, \tag{3.10}$$

$$\frac{\partial \log f_i^N(x,\theta)}{\partial \theta} = \frac{1}{c_i^N + \theta} - x, \tag{3.11}$$

$$\frac{\partial^2 \log f_i^N(x,\theta)}{\partial \theta^2} = -\frac{1}{(c_i^N + \theta)^2}, \tag{3.12}$$

$$\frac{\partial^3 \log f_i^N(x,\theta)}{\partial \theta^3} = \frac{2}{(c_i^N + \theta)^3}. \tag{3.13}$$

First we start with a lemma about the behavior of the constants c_i^N.

Lemma 3.3 *For each N and each $i = 0, 1, ..., N-2$ we have that*

$$1 \leqslant c_i^N \leqslant \frac{N(N-1)}{(N-i)(N-i-1)}$$

for each realization of the vector $\boldsymbol{c}^N$.

Proof. By definition, $c_i^N = a_i^N / b_i^N$ where a_i^N is the number of pairs of host lineages at time T_i, and b_i^N is the number of pairs of parasite lineages at time T_i. Since there cannot be more than one parasite infection on a host lineage, the number of parasite lineages cannot be bigger than the number of host lineages, therefore $1 \leqslant c_i^N$ holds for each N and each $i = 0, 1, ..., N-2$, for each realization of the vector $\boldsymbol{c}^N$. On the other hand, we obtain the largest possible value for c_i^N if the number of host lineages is unchanged, and the number of parasite lineages is the smallest possible, i.e. in the scenario when all the parasite lineages merge before the first coalescence event in the host tree happens. In this case T_i would exactly correspond the i-th merging event in the parasite tree, and at that time there are N host lineages and $N-i$ parasite lineages, so $c_i^N = \frac{N(N-1)}{(N-i)(N-i-1)}$ holds for all $i = 0, 1, .., N-2$. Otherwise c_i^N would be smaller, so

$$c_i^N \leqslant \frac{N(N-1)}{(N-i)(N-i-1)} \quad \text{for all } i = 0, 1, .., N-2$$

holds for all N and all realizations of $\boldsymbol{c}^N$. □

We will need another technical lemma about the behavior of the second and third derivatives of the log-likelihood function.

Lemma 3.4 *For each $\theta > 0$, there exist deterministic constants $\delta_1 = \delta_1(\theta), \delta_2 = \delta_2(\theta)$ such that*

$$\left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{\partial^2 \log f_i^N(X_i^N, \theta)}{\partial \theta^2} \right| < 1 \quad \textit{and} \quad \left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{\partial^2 \log f_i^N(X_i^N, \theta)}{\partial \theta^2} \right| > \delta_1$$

as well as

$$\left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{\partial^3 \log f_i^N(X_i^N, \theta)}{\partial \theta^3} \right| < 2 \quad \textit{and} \quad \left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{\partial^3 \log f_i^N(X_i^N, \theta)}{\partial \theta^3} \right| > \delta_2$$

hold for each $N \geqslant 6$ and for all realizations of $\boldsymbol{c}^N$.

Proof. We are going to demonstrate the proof only for the second derivatives. It can be done similarly for the third derivatives. First, we show boundedness from below. Using (3.12), we need that there exists a $\delta_1 > 0$ such that for each $N \geqslant 6$ we have

$$\left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{-1}{(c_i^N + \theta)^2} \right| > \delta_1.$$

Using the inequality between the arithmetic and quadratic mean, and by Lemma 3.3 we have that

$$\left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{-1}{(c_i^N + \theta)^2} \right| \geqslant \frac{1}{2(N-1)} \sum_{i=0}^{N-2} \frac{1}{(c_i^N)^2 + \theta^2} \geqslant \frac{1}{2(N-1)} \sum_{i=0}^{N-2} \frac{1}{\left(\frac{N(N-1)}{(N-i)(N-i-1)} \right)^2 + \theta^2}$$

Note that the expression $\frac{N(N-1)}{(N-i)(N-i-1)}$ is monotone increasing in i and for $i = \lfloor N/2 \rfloor$ we have $\frac{N(N-1)}{(N-i)(N-i-1)} = \frac{4j-2}{j-1}$ if $N = 2j$ for some $j \in \mathbb{N}$ and $\frac{N(N-1)}{(N-i)(N-i-1)} = \frac{4j+2}{j+1}$ if $N = 2j+1$ for some $j \in \mathbb{N}_0$. In both cases, the expression is bounded from above by 5 for all $N \geqslant 6$, and so by using this bound for the first half of the summands and ignoring the second half of the summands we get that

$$\frac{1}{2(N-1)} \sum_{i=0}^{N-2} \frac{1}{\left(\frac{N(N-1)}{(N-i)(N-i-1)} \right)^2 + \theta^2} \geqslant \frac{1}{2(N-1)} \cdot \left\lfloor \frac{N}{2} \right\rfloor \cdot \frac{1}{25 + \theta^2} \geqslant \frac{1}{4} \cdot \frac{1}{25 + \theta^2} =: \delta_1(\theta) > 0,$$

since the expression $\frac{1}{2(N-1)} \cdot \left\lfloor \frac{N}{2} \right\rfloor$ is always greater or equal to $1/4$. Hence we have showed that the expression is bounded from below. Now, for the upper bound, by Lemma 3.3 and by $\theta > 0$ we have

$$\left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{-1}{(c_i^N + \theta)^2} \right| \leqslant \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{1}{(1+\theta)^2} \leqslant \frac{1}{N-1} \sum_{i=0}^{N-2} 1 = 1.$$

As mentioned before, the same methods can be used to prove the bounds for the expression with the third derivatives, where the upper bound will be 2 instead of 1. □

The next lemma is a convergence result.

Lemma 3.5 *Fix a parameter $\theta > 0$. Then for all $\varepsilon > 0$ and for all realizations of the vectors $(\boldsymbol{c}^N)_{N \in \mathbb{N}}$ there exists a sequence $(q_N)_{N \in \mathbb{N}}$ with $q_N \to 0$ (as $N \to \infty$) that does not depend on the vectors $(\boldsymbol{c}^N)_{N \in \mathbb{N}}$ such that*

$$\mathbb{P}_c \left(\left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{\partial \log f_i^N(X_i^N, \theta)}{\partial \theta} \right| > \varepsilon \right) \leqslant q_N. \tag{3.14}$$

It also implies that

$$\frac{1}{N-1} \sum_{i=0}^{N-2} \frac{\partial \log f_i^N(X_i^N, \theta)}{\partial \theta} \xrightarrow{\mathbb{P}} 0 \quad \text{as } N \to \infty. \tag{3.15}$$

with respect to all randomness and uniformly over all vectors $(\boldsymbol{c}^N)_{N \in \mathbb{N}}$.

Proof. For all N, let $c_0^N, c_1^N, ..., c_{N-2}^N$ be a fixed realization and let $Y_i^N := \frac{1}{c_i^N+\theta} - X_i^N$ for all N and $i = 0, 1, ..., N-2$. Since the X_i^N are independent (given the c_i^N), so are the Y_i^N, and it holds that $\mathbb{E}_c(Y_i^N) = 0$ for all $i = 0, 1, ..., N-2$. Note that by (3.11), Y_i^N is exactly the first derivative appearing in (3.15). So what we need to prove is that

$$\overline{Y}_{N-1} := \frac{1}{N-1} \sum_{i=0}^{N-2} Y_i^N \xrightarrow{\mathbb{P}} 0 \quad \text{uniformly as } N \to \infty.$$

By the definition of the random variables Y_i^N and by Lemma 3.3 we have that

$$\mathbb{E}_c((Y_i^N)^2) = \mathrm{Var}_c(X_i^N) = \frac{1}{(c_i^N+\theta)^2} \leqslant 1 \quad \text{for all } N \text{ and for all } i = 0, 1, ..., N-2. \tag{3.16}$$

Note that this bound is uniform in the sequence $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$. Now let $\varepsilon > 0$ be fixed. By the Markov inequality for the average $\overline{Y}_{N-1}$ we get that

$$\mathbb{P}_c(\overline{Y}_{N-1} > \varepsilon) \leqslant \frac{1}{\varepsilon^2}\, \mathbb{E}_c(\bar{Y}_{N-1}^2) = \frac{1}{(N-1)^2\varepsilon^2}\, \mathbb{E}_c((Y_0^N + ... + Y_{N-2}^N)^2)$$

Using the conditional independence of the Y_i^N and (3.16), we can continue by

$$\mathbb{P}_c(\overline{Y}_{N-1} > \varepsilon) \leqslant \frac{1}{(N-1)^2\varepsilon^2} \sum_{i=0}^{N-2} \mathbb{E}_c((Y_i^N)^2) \leqslant \frac{1}{(N-1)\varepsilon^2} \cdot 1 =: q_N$$

for which $q_N \to 0$ obviously holds. This implies convergence in probability uniformly over all possible sequences $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$. □

Now we are ready to prove that equation (3.8) has a solution with high probability for all N large enough.

Theorem 3.6 *Fix a parameter $\theta > 0$. There exists a sequence $(q'_N)_{N\in\mathbb{N}}$ that does not depend on the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$ such that $q'_N \to 1$ and*

$$\mathbb{P}_c\left(\sum_{i=0}^{N-2} X_i^N < \sum_{i=0}^{N-2} \frac{1}{c_i^N}\right) \geqslant q'_N$$

holds. This implies that

$$\mathbb{P}\left(\sum_{i=0}^{N-2} X_i^N < \sum_{i=0}^{N-2} \frac{1}{c_i^N}\right) \to 1$$

as $N \to \infty$, uniformly over all realizations of $\boldsymbol{c}^N$.

Proof. Let $N \in \mathbb{N}$ and the constants $(c_i^N)_{i=0,1,...,N-2}$ be fixed. Since the X_i^N and the c_i^N are positive, we have for all $t > 0$ that

$$\mathbb{P}_c\left(\sum_{i=0}^{N-2} X_i^N \geqslant \sum_{i=0}^{N-2} \frac{1}{c_i^N}\right) = \mathbb{P}_c\left(\mathrm{e}^{t\sum_{i=0}^{N-2} X_i^N} \geqslant \mathrm{e}^{t\sum_{i=0}^{N-2} \frac{1}{c_i^N}}\right) \leqslant \frac{\mathbb{E}_c\left(\mathrm{e}^{t\sum_{i=0}^{N-2} X_i^N}\right)}{\mathrm{e}^{t\sum_{i=0}^{N-2} \frac{1}{c_i^N}}}$$

by the Markov inequality. In the numerator of the RHS we see exactly the moment generating function of the sum $\sum_{i=0}^{N-2} X_i^N$, which can be written as a product as the X_i^N are (conditionally) independent. Writing $\lambda_i^N = c_i^N + \theta$ as the parameter of X_i^N, we have, given $t < \theta < \lambda_i^N$ (uniformly for all i), that

$$\frac{\mathbb{E}_c\left(\mathrm{e}^{t\sum_{i=0}^{N-2} X_i^N}\right)}{\mathrm{e}^{t\sum_{i=0}^{N-2}\frac{1}{c_i^N}}} = \frac{\prod_{i=0}^{N-2}\frac{\lambda_i^N}{\lambda_i^N - t}}{\mathrm{e}^{t\sum_{i=0}^{N-2}\frac{1}{c_i^N}}} = \prod_{i=0}^{N-2}\frac{c_i^N+\theta}{c_i^N+\theta-t}\cdot \mathrm{e}^{-\frac{t}{c_i^N}} = \prod_{i=0}^{N-2}\left(1+\frac{t}{c_i^N+\theta-t}\right)\cdot \mathrm{e}^{-\frac{t}{c_i^N}}.$$

Since we chose $t < \theta$, it holds for all i that

$$\left(1+\frac{t}{c_i^N+\theta-t}\right)\cdot \mathrm{e}^{-\frac{t}{c_i^N}} < \left(1+\frac{t}{c_i^N}\right)\cdot \mathrm{e}^{-\frac{t}{c_i^N}} \leqslant 1,$$

as for all $x > 0$ it is true that $(1+x) \leqslant \mathrm{e}^x$. As seen in the proof of Lemma 3.4, we have that $c_i^N \leqslant 5$ for $i = 0, 1, .., N/2$ if $N \geqslant 6$, and consequently $t/c_i^N \geqslant t/5$ for all pairs (N, i) with $N \geqslant 6$ and $i = 0, 1, ..., N/2$. Using this, and the fact that $f(x) := (1+x)\mathrm{e}^{-x}$ is monotone decreasing for $x > 0$, we have that

$$\prod_{i=0}^{N-2}\left(1+\frac{t}{c_i^N+\theta-t}\right)\cdot \mathrm{e}^{-\frac{t}{c_i^N}} \leqslant \left(\left(1+\frac{t}{5}\right)\cdot \mathrm{e}^{-\frac{t}{5}}\right)^{\lfloor N/2\rfloor} \to 0$$

holds for all $0 < t < \theta$ as $q := \left(1+\frac{t}{5}\right)\cdot \mathrm{e}^{-\frac{t}{5}} < 1$. Note that the same constant q can be used for each N, and the upper bound does not depend on the constants c_i^N, which means that the statement of the lemma holds with $q'_N := 1 - q^{\lfloor N/2\rfloor}$. As q'_N does not depend on the constants c_i^N, the inequality holds also without conditioning. □

3.3 Consistency

Now we are ready to prove consistency of our estimator. Here we recall that the true value of the horizontal transfer rate is denoted by τ.

Theorem 3.7 *(Consistency of $\hat{\tau}_N$) The estimator $\hat{\tau}_N$ obtained by solving equation (3.7) is a consistent estimator for τ, meaning that for each $\varepsilon > 0$ there exists a sequence $(Q_N)_{N\in\mathbb{N}}$ that does not depend on the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$ such that*

$$\mathbb{P}_c(|\hat{\tau}_N - \tau| < \varepsilon) \geqslant Q_N \to 1 \quad \text{as } N \to \infty.$$

This implies that

$$\mathbb{P}(|\hat{\tau}_N - \tau| < \varepsilon) \geqslant Q_N \to 1 \quad \text{as } N \to \infty,$$

uniformly over all vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$.

Proof. As mentioned before, we will use the ideas from Leroy et al. [LDTB16]. First, note that by (3.10) to (3.12)the derivatives of the function $\log f_i^N(x_i, \theta)$ exist and are continuous (both

with respect to x and to θ), and so by the Taylor-Lagrange formula, for all $\theta > 0$, for all N and for all $i = 0, 1, ..., N-2$ we have that

$$\log f_i^N(x_i,\theta) = \log f_i^N(x_i,\tau) + (\theta-\tau)\left.\frac{\partial \log f_i^N(x_i,\theta)}{\partial\theta}\right|_{\theta=\tau} + \frac{1}{2}(\theta-\tau)^2\left.\frac{\partial^2 \log f_i^N(x_i,\theta)}{\partial\theta^2}\right|_{\theta=\tau} + \frac{1}{6}(\theta-\tau)^3\left.\frac{\partial^3 \log f_i^N(x_i,\theta)}{\partial\theta^3}\right|_{\theta=\theta_i'}$$

with θ_i' in the interior of the interval $(\tau-d, \tau+d)$ with $d = |\tau-\theta|$ for all $i = 0, 1, ..., N-2$. Using that $\sum_{i=0}^{N-2} \log f_i^N(x_i,\theta) = l_N(x_0, ..., x_{N-2}, \theta)$, and by the linearity of derivatives, we have for all $\theta > 0$ by the Taylor-Lagrange formula for $\frac{1}{N-1}l_N(\theta)$ that

$$\begin{aligned}\frac{1}{N-1}l_N(\theta) - \frac{1}{N-1}l_N(\tau) = (\theta-\tau)\frac{1}{N-1}\sum_{i=0}^{N-2}\left.\frac{\partial \log f_i^N(x_i,\theta)}{\partial\theta}\right|_{\theta=\tau} \\ + \frac{1}{2}(\theta-\tau)^2\frac{1}{N-1}\sum_{i=0}^{N-2}\left.\frac{\partial^2 \log f_i^N(x_i,\theta)}{\partial\theta^2}\right|_{\theta=\tau} \\ + \frac{1}{6}(\theta-\tau)^3\frac{1}{N-1}\sum_{i=0}^{N-2}\left.\frac{\partial^3 \log f_i^N(x_i,\theta)}{\partial\theta^3}\right|_{\theta=\theta'}\end{aligned}$$

again with $\theta' \in (\tau-d, \tau+d)$ (note that now θ' does not depend on i). Now let $\zeta > 0$ fixed. We are going to give bounds for the different terms on the right hand side separately.

By Lemma 3.5, we have for all $\theta > 0$ that

$$\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial \log f_i^N(X_i^N,\theta)}{\partial\theta} \xrightarrow{\mathbb{P}} 0$$

uniformly over all possible sequences $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$ and so for every $\varepsilon > 0$ there exists $N_0 = N_0(\varepsilon)$ such that for all $N \geqslant N_0$ it holds that

$$\mathbb{P}\left(\frac{1}{N-1}\sum_{i=0}^{N-2}\left.\frac{\partial \log f_i^N(X_i^N,\theta)}{\partial\theta}\right|_{\theta=\tau} < \zeta^2\right) > 1-\varepsilon. \tag{3.17}$$

For the sake of simplicity, we will denote the event appearing in (3.17) by E^N. Obviously, we have $\mathbb{P}(E^N) \to 1$ as $N \to \infty$ uniformly over all realizations of the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$. Now we are going to investigate the sign of $\frac{1}{N-1}l_N(\theta) - \frac{1}{N-1}l_N(\tau)$ under the event E^N and for all θ for which $|\theta-\tau| = \zeta$ holds. First, since we are under the event E^N, we have that

$$\left|(\theta-\tau)\frac{1}{N-1}\sum_{i=0}^{N-2}\left.\frac{\partial \log f_i^N(X_i^N,\theta)}{\partial\theta}\right|_{\theta=\tau}\right| < \zeta^3.$$

Moreover, by Lemma 3.4 and the fact that the second derivative is negative, we know that there exists $\delta = \delta(\tau) > 0$ such that

$$\frac{1}{2}(\theta-\tau)^2\frac{1}{N-1}\sum_{i=0}^{N-2}\left.\frac{\partial^2 \log f_i^N(X_i^N,\theta)}{\partial\theta^2}\right|_{\theta=\tau} < -\delta\zeta^2 < 0$$

uniformly for all possible vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$. Again by Lemma 3.4 we have that

$$\left|\frac{1}{6}(\theta-\tau)^3\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial^3 \log f_i^N(x_i,\theta)}{\partial\theta^3}\bigg|_{\theta=\theta'}\right| < \frac{2}{6}\zeta^3$$

again uniformly for all possible vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$. Putting these three inequalities together, we get under E^N and for all θ with $|\theta-\tau|=\zeta$ that

$$\frac{1}{N-1}l_N(\theta)-\frac{1}{N-1}l_N(\tau) < \zeta^3-\delta\zeta^2+\frac{1}{3}\zeta^3$$

uniformly for all possible vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$. Then, assuming that $\zeta < 3\delta/4$ we get under E^N that

$$\frac{1}{N-1}l_N(\theta)-\frac{1}{N-1}l_N(\tau) < 0 \quad \text{for all } \theta \text{ with } |\theta-\tau|=\zeta \tag{3.18}$$

again uniformly for all possible sequences (c_i^N). Denoting the event appearing in (3.18) by F^N we have that $\mathbb{P}(F^N) \geqslant \mathbb{P}(E^N) \to 1$ as $N\to\infty$. Since the function $l_N(\theta)$ is continuous, it must admit its maximum value in the closed interval $[\tau-\zeta,\tau+\zeta]$. As $\mathbb{P}(F^N)\to 1$, the probability that this maximum is admitted on the boundaries of the interval goes to zero, meaning that with high probability it is going to be admitted in the open interval $(\tau-\zeta,\tau+\zeta)$. Let the place where the maximum is admitted denoted by $\hat{\tau}_N$, which then needs to be the solution of the equation (3.7) (which exists with high probability if N is large enough by Theorem 3.6) in order to be a maximum. Since this value belongs to the open interval $(\tau-\zeta,\tau+\zeta)$, we have that $|\hat{\tau}_N-\tau|<\zeta$ must hold. So we have proven that for all $\zeta<3\delta/4$, we have that

$$\mathbb{P}(|\hat{\tau}_N-\tau|<\zeta) \geqslant \mathbb{P}(F^N)\to 1 \quad \text{as } N\to\infty,$$

uniformly for all possible vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$. □

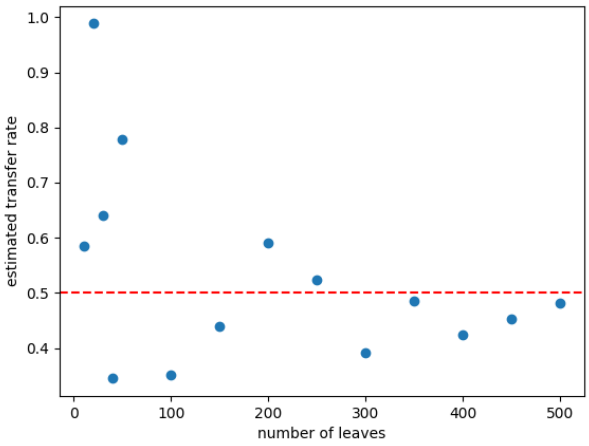
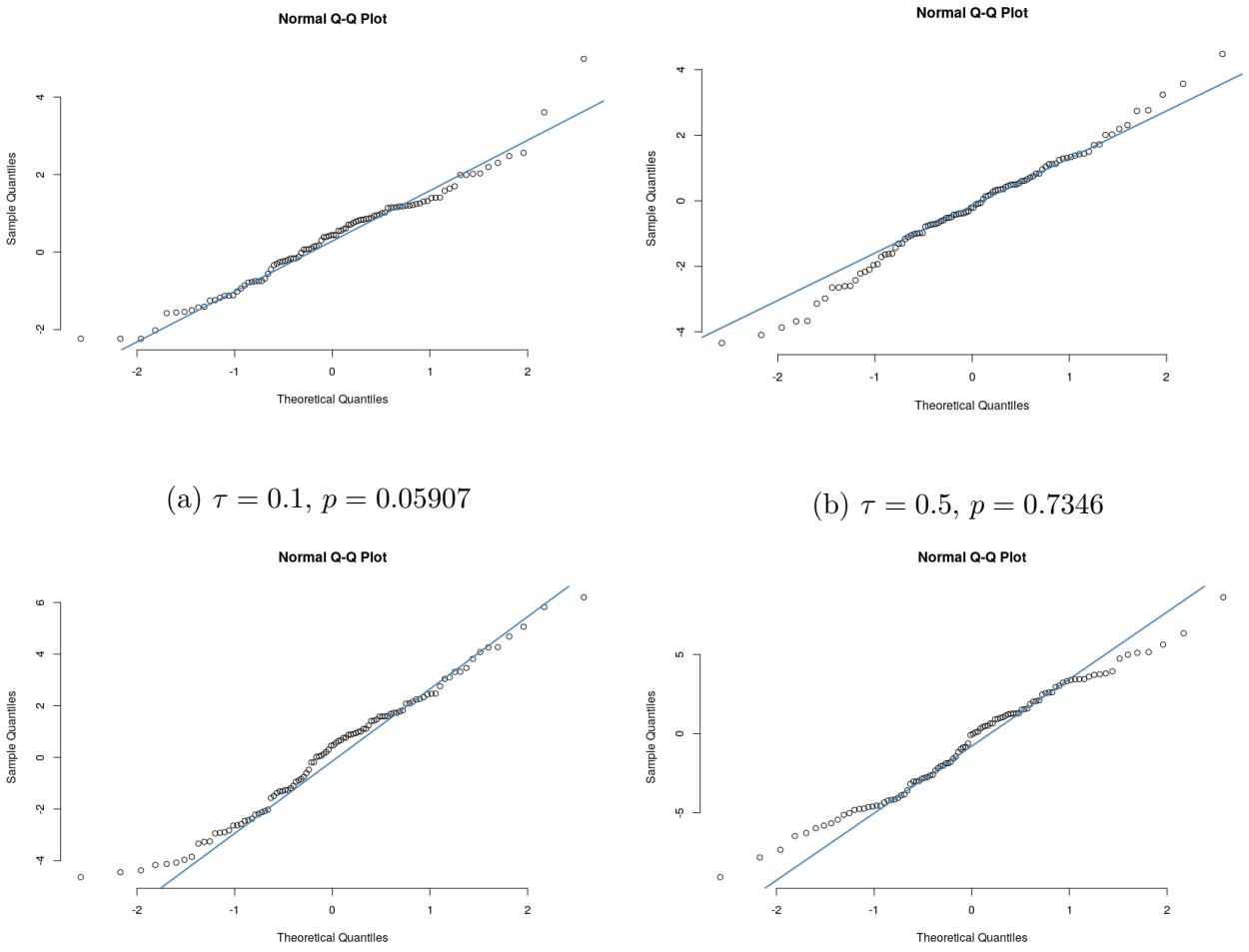


Figure 14: Consistency of the MLE

Figure 14 shows the MLE of τ for simulated trees of different sizes. When calculating the root of the likelihood equation, we used the simple bisection method with a margin error of 0.01. The dashed red line shows the true value of τ, which is 0.5 in our case.

3.4 Asymptotic normality

Now we want to prove asymptotic normality (with proper scaling) of our estimator $\hat{\tau}_N$. First we performed simulations and checked whether asymptotic normality is a realistic expectation in our case. We simulated 100 joint scenarios with $N = 500$ leaves, calculated $\hat{\tau}_N$ and checked whether normality of $\sqrt{N}(\hat{\tau}_N - \tau)$ can be rejected or not with the Shapiro-Wilks normality test with significance level set to 0.05. For τ we took the values $0.1, 0.5, 1$ and 2. When calculating $\hat{\tau}_N$, we used the simple bisection method with a margin error of 0.01. The results are shown in Figure 15.



(a) $\tau = 0.1$, $p = 0.05907$

(b) $\tau = 0.5$, $p = 0.7346$

(c) $\tau = 1$, $p = 0.2176$

(d) $\tau = 2$, $p = 0.2757$

Figure 15: Results of Shapiro-Wilk normality tests with Q-Q plots for different values of τ

Before stating our result for asymptotic normality, we need a technical lemma that gives us bounds of the 1-Wasserstein distance d_W of a random sum and a normal distribution with zero mean. The definition and the most important properties of this distance is stated in Appendix A.2. For all $N \in \mathbb{N}$ and for $i = 0, 1, ..., N-2$, let $Y_i^N = \frac{1}{c_i^N + \tau} - X_i^N$. Recall that $\mathbb{E}_c(Y_i^N) = 0$

and $\mathrm{Var}_c(Y_i^N) = \mathrm{Var}_c((X_i^N)^2) = \frac{1}{(c_i^N+\tau)^2}$ hold. Let

$$\sigma_N^2 := \mathrm{Var}_c\left(\sum_{i=0}^{N-2} Y_i^N\right) = \sum_{i=0}^{N-2} \frac{1}{(c_i^N+\tau)^2} \tag{3.19}$$

and let

$$S_N := \frac{1}{\sigma_N} \sum_{i=0}^{N-2} Y_i^N \tag{3.20}$$

Note that σ_N only depends on the vector $\boldsymbol{c}^N$ and τ, and hence it is not random anymore when we are conditioning on the vector $\boldsymbol{c}^N$.

Lemma 3.8 *Let $C > 0$ be a constant that might depend on τ but not on N and the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$, and let η be a random variable with standard normal distribution. Then there exists a constant B not depending on N and the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$, and a sequence $(R_N)_{N\in\mathbb{N}}$ with $R_N \to 0$ as $N \to \infty$ uniformly over all possible vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$, such that*

$$d_W\left(\frac{1}{\sqrt{N-1}} \sum_{i=0}^{N-2} Y_i^N | \boldsymbol{c}^N, \sqrt{C}\eta\right) \leqslant B \cdot \left|\frac{\sigma_N}{\sqrt{N-1}} - \sqrt{C}\right| + R_N \tag{3.21}$$

holds.

3.9 Remark. Note that in (3.21) the Wasserstein distance of a conditional and a normal distribution is considered, which is a random quantity itself. (Wasserstein distance of random variables meant as the Wasserstein distance of their laws, for a precise definition please see Appendix A.2.) Each following equation and inequality involving such Wasserstein distances are meant that they are true almost surely for all ω in the σ-algebra generated by the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$.

Proof: By the triangle-inequality we have that

$$d_W\left(\frac{1}{\sqrt{N-1}} \sum_{i=0}^{N-2} Y_i^N | \boldsymbol{c}^N, \sqrt{C}\eta\right) \leqslant d_W\left(\frac{1}{\sqrt{N-1}} \sum_{i=0}^{N-2} Y_i^N | \boldsymbol{c}^N, \sqrt{C}S_N | \boldsymbol{c}^N\right) + d_W\left(\sqrt{C}S_N | \boldsymbol{c}^N, \sqrt{C}\eta | \boldsymbol{c}^N\right) \tag{3.22}$$

First we are going to give a bound on the first term on the RHS of (3.22). By the definition of S_N we have that

$$\frac{1}{\sqrt{N-1}} \sum_{i=0}^{N-2} Y_i^N = \frac{\sigma_N}{\sqrt{N-1}} S_N.$$

As the Wasserstein distance of two random variables is defined as the infimum of their expected Euclidean distance over all couplings of the two underlying distributions, we have that

$$d_W\left(\frac{\sigma_N}{\sqrt{N-1}} S_N | \boldsymbol{c}^N, \sqrt{C}S_N | \boldsymbol{c}^N\right) \leqslant \mathbb{E}_c\left(\left|\frac{\sigma_N}{\sqrt{N-1}} - \sqrt{C}\right| \cdot |S_N|\right).$$

Since σ_N is measurable w.r.t. the σ-algebra generated by $\boldsymbol{c}^N$, we have that

$$\mathbb{E}_c\left(\left|\frac{\sigma_N}{\sqrt{N-1}}-\sqrt{C}\right|\cdot|S_N|\right)=\left|\frac{\sigma_N}{\sqrt{N-1}}-\sqrt{C}\right|\cdot\mathbb{E}_c(|S_N|).$$

To bound $\mathbb{E}_c(|S_N|)$, we have by the Cauchy-Schwarz inequality that

$$\mathbb{E}_c(|S_N|)=\mathbb{E}_c\left(\left|\frac{1}{\sigma_N}\sum_{i=0}^{N-2}Y_i^N\right|\right)\leqslant\left(\mathbb{E}_c\left(\frac{1}{\sigma_N^2}\Big(\sum_{i=0}^{N-2}Y_i^N\Big)^2\right)\right)^{1/2}=\frac{1}{\sigma_N}\left(\sum_{i=0}^{N-2}\mathbb{E}_c\big((Y_i^N)^2\big)\right)^{1/2}$$

where we used again the measurability of σ_N, and also that for $i\neq j$ the random variables Y_i^N and Y_j^N are conditionally independent, both with 0 mean. Since $\mathbb{E}_c\big((Y_i^N)^2\big)=\frac{1}{(c_i^N+\tau)^2}\leqslant 1$ always holds and by Lemma 3.4 there exists a constant δ not depending on N and the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$ such that $\sigma_N\geqslant\sqrt{(N-1)\delta}$, we have that

$$\mathbb{E}_c(|S_N|)\leqslant\frac{1}{\sqrt{(N-1)\delta}}\sqrt{N-1}=\frac{1}{\sqrt{\delta}}.$$

So we have for each N conditional on $\boldsymbol{c}^N$ that

$$d_W\left(\frac{\sigma_N}{\sqrt{N-1}}S_N|\boldsymbol{c}^N,\sqrt{C}S_N|\boldsymbol{c}^N\right)\leqslant\frac{1}{\sqrt{\delta}}\left|\frac{\sigma_N}{\sqrt{N-1}}-\sqrt{C}\right|. \tag{3.23}$$

Now, considering the second term on the RHS of (3.22) we have that

$$d_W(\sqrt{C}S_N|\boldsymbol{c}^N,\sqrt{C}\eta\boldsymbol{c}^N)=\sqrt{C}d_W(S_N|\boldsymbol{c}^N,\eta)$$

due to the definition of the Wasserstein distance. Now we can directly apply a result of Ross [Ros11] that bounds the Wasserstein distance between S_N and the standard normal distribution using Stein's method:

By Theorem 3.6 in [Ros11], we have that

$$d_W(S_N|\boldsymbol{c}^N,\eta)\leqslant\frac{1}{\sigma_N^3}\sum_{i=0}^{N-2}\mathbb{E}_c\big(|Y_i^N|^3\big)+\frac{\sqrt{28}}{\sqrt{\pi}\sigma_N^2}\sqrt{\sum_{i=0}^{N-2}\mathbb{E}_c\big((Y_i^N)^4\big)}. \tag{3.24}$$

As Y_i^N is directly related to X_i^N which is exponentially distributed, we can calculate the moments of Y_i^N appearing in the RHS of (3.24):

$$\mathbb{E}_c\big(|Y_i^N|^3\big)=\frac{12-2\mathrm{e}}{\mathrm{e}(c_i^N+\tau)^3}\quad\text{and}\quad\mathbb{E}_c\big((Y_i^N)^4\big)=\frac{9}{(c_i^N+\tau)^4},$$

from which we can directly obtain the following upper bounds:

$$\mathbb{E}_c\big(|Y_i^N|^3\big)\leqslant 12\quad\text{and}\quad\mathbb{E}_c\big((Y_i^N)^4\big)\leqslant 9, \tag{3.25}$$

by using that $(c_i^N+\tau)\geqslant 1$. Moreover, by Lemma 3.4 there exists a constant $\delta=\delta(\tau)$ such that

$$\sigma_N^2\geqslant(N-1)\delta \tag{3.26}$$

holds for all N big enough. Putting everything together we get that
(3.27)

$$d_W(S_N|\boldsymbol{c}^N,\eta) \leqslant \frac{1}{\delta^{3/2}(N-1)^{3/2}}\sum_{i=0}^{N-2} 12 + \frac{\sqrt{28}}{\sqrt{\pi}(N-1)\delta}\sqrt{\sum_{i=0}^{N-2} 9} = \frac{12\delta^{-3/2}}{(N-1)^{1/2}} + \frac{3\sqrt{28}}{\sqrt{\pi}(N-1)^{1/2}\delta} \to 0$$

as $N \to \infty$, uniformly in the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$ as the bounds do not depend on these vectors. Putting everything together we get that

$$d_W\left(\frac{1}{\sqrt{N-1}}\sum_{i=0}^{N-2} Y_i^N|\boldsymbol{c}^N,\sqrt{C}\eta\right) \leqslant B\cdot\left|\frac{\sigma_N}{\sqrt{N-1}} - \sqrt{C}\right| + R_N \tag{3.28}$$

with

$$B := \frac{1}{\sqrt{\delta}} \quad \text{and} \quad R_N := \frac{12\delta^{-3/2}}{(N-1)^{1/2}} + \frac{3\sqrt{28}}{\sqrt{\pi}(N-1)^{1/2}\delta}.$$

□

Now we are ready to prove asmyptotic normality of our estimator.

Theorem 3.10 *(Asymptotic normality of $\hat{\tau}_N$) Let us suppose there exists a deterministic constant $C^* = C^*(\tau) > 0$ such that there exists a sequence $(R'_N)_{N\in\mathbb{N}}$ not depending on the constants $(\boldsymbol{c}^N)$ with $R'_N \to 0$ such that*

$$\left|\frac{\sigma_N}{\sqrt{N-1}} - \sqrt{C^*}\right| \leqslant R'_N. \tag{3.29}$$

uniformly over all possible vectors $\boldsymbol{c}^N$. Then it holds that

$$\sqrt{N}(\hat{\tau}_N - \tau) \xrightarrow{\mathcal{D}} \mathcal{N}\left(0, \frac{1}{C^*}\right) \tag{3.30}$$

with respect to every randomness appearing.

Proof: We are using again the ideas from Leroy et al. [LDTB16]. By the Taylor-Lagrange formula for the function $\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial \log f_i^N(X_i^N,\theta)}{\partial\theta}$ we get that for all $\tilde{\theta} > 0$ it holds that

$$\begin{aligned}\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial \log f_i^N(X_i^N,\theta)}{\partial\theta}\bigg|_{\theta=\tilde{\theta}} &= \frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial \log f_i^N(X_i^N,\theta)}{\partial\theta}\bigg|_{\theta=\tau} \\ &\quad + (\tilde{\theta}-\tau)\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial^2 \log f_i^N(X_i^N,\theta)}{\partial\theta^2}\bigg|_{\theta=\tau} \\ &\quad + \frac{1}{2}(\tilde{\theta}-\tau)^2\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial^3 \log f_i^N(X_i^N,\theta)}{\partial\theta^3}\bigg|_{\theta=\theta'}\end{aligned}$$

with θ' being in the interval $(\tau - \tilde{d}, \tau + \tilde{d})$ with $\tilde{d} = |\tau - \tilde{\theta}|$. Using that for $\tilde{\theta} = \hat{\tau}_N$ the LHS is zero (due to the definition of the maximum likelihood estimator), we have that

$$\begin{aligned}\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial \log f_i^N(X_i^N,\theta)}{\partial\theta}\bigg|_{\theta=\tau} &= -(\hat{\tau}_N-\tau)\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial^2 \log f_i^N(X_i^N,\theta)}{\partial\theta^2}\bigg|_{\theta=\tau} \\ &\quad - \frac{1}{2}(\hat{\tau}_N-\tau)^2\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial^3 \log f_i^N(X_i^N,\theta)}{\partial\theta^3}\bigg|_{\theta=\theta'}.\end{aligned}$$

Now, by factoring the RHS and multiplying both sides by $\sqrt{N-1}$ we get that

$$\frac{1}{\sqrt{N-1}}\sum_{i=0}^{N-2}\frac{\partial \log f_i^N(X_i^N,\theta)}{\partial\theta}\bigg|_{\theta=\tau} = \sqrt{N-1}(\hat{\tau}_N-\tau)\cdot H_N$$

with

(3.31)
$$H_N := \left(-\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial^2 \log f_i^N(X_i^N,\theta)}{\partial\theta^2}\bigg|_{\theta=\tau} - \frac{1}{2}(\hat{\tau}_N-\tau)\frac{1}{N-1}\sum_{i=0}^{N-2}\frac{\partial^3 \log f_i^N(X_i^N,\theta)}{\partial\theta^3}\bigg|_{\theta=\theta'}\right).$$

With $Y_i^N := \frac{1}{c_i^N+\tau} - X_i^N$ and by (3.11) we have that

$$\frac{1}{\sqrt{N-1}}\sum_{i=0}^{N-2} Y_i^N = \sqrt{N-1}(\hat{\tau}_N-\tau)\cdot H_N. \tag{3.32}$$

We will first show that the LHS has the desired asymptotic behavior. By Lemma 3.8 and Proposition A.5 we can bound the 1-Wasserstein distance of the LHS of (3.32) and an $\mathcal{N}(0, C^*)$ distribution:

$$\begin{aligned}d_W\left(\frac{1}{\sqrt{N-1}}\sum_{i=0}^{N-2} Y_i^N, \mathcal{N}(0,C^*)\right) &\leqslant \mathbb{E}\left(d_W\left(\frac{1}{\sqrt{N-1}}\sum_{i=0}^{N-2} Y_i^N | \boldsymbol{c}^N, \mathcal{N}(0,C^*)\right)\right) \\ &\leqslant \mathbb{E}\left(B\cdot\left|\frac{\sigma_N}{\sqrt{N-1}} - \sqrt{C^*}\right| + R_N\right) \\ &\leqslant \mathbb{E}(B\cdot R'_N + R_N) = B\cdot R'_N + R_N \to 0,\end{aligned}$$

by (3.29) since B, R'_N and R_N are constants. Note that in Proposition A.5 the conditioning is given with respect to a sub-σ algebra of $\mathcal{B}(\mathbb{R})$ (the Borel σ-algebra on $\mathbb{R}$). In our case we condition on the σ-algebra generated by the vector $\boldsymbol{c}^N$, which can be pushed to such a desired conditioning (with non-trivial calculations that we omit here). Since the random variables in question have finite expectation, convergence in Wasserstein distance implies convergence in distribution, so we get that

$$\frac{1}{\sqrt{N-1}}\sum_{i=0}^{N-2} Y_i^N \xrightarrow{\mathcal{D}} \mathcal{N}(0,C^*) \quad \text{as } N\to\infty \tag{3.33}$$

with respect to all randomness. This means that the RHS in (3.32) should converge in distribution to the $\mathcal{N}(0, C^*)$ distribution as well.

Now we are going to investigate how the term H_N behaves as $N \to \infty$. We already know from the assumption (3.29) that the first term of H_N is converging uniformly to the same deterministic constant C^*. Moreover, by Lemma 3.4 it holds that

$$\left| \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{\partial^3 \log f_i^N(X_i^N, \theta)}{\partial \theta^3} \bigg|_{\theta=\theta'} \right| \leqslant 2.$$

So by Theorem 3.7 we have that for all $\varepsilon > 0$ there exists a sequence (Q_N^*) such that

$$\mathbb{P}_c \left(\left| \frac{1}{2}(\hat{\tau}_N - \tau) \frac{1}{N-1} \sum_{i=0}^{N-2} \frac{\partial^3 \log f_i^N(X_i^N, \theta)}{\partial \theta^3} \bigg|_{\theta=\theta'} \right| \geqslant \varepsilon \right) \leqslant Q_N^* \to 0 \tag{3.34}$$

uniformly over all vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$. This means that it holds for all $\varepsilon > 0$ that there exists a sequence $(\tilde{Q}_N)_{N\in\mathbb{N}}$ not depending on the vectors $(\boldsymbol{c}^N)$ with $\tilde{Q}_N \to 0$ such that

$$\mathbb{P}_c \left(|H_N - C^*| \geqslant \varepsilon \right) \leqslant \tilde{Q}_N. \tag{3.35}$$

Since all convergences are uniform in the vectors $(\boldsymbol{c}^N)_{N\in\mathbb{N}}$, we can say that these convergences hold also without the conditioning. Using (3.33) and (3.35) we can apply Slutsky's Lemma (see Lemma A.4) for (3.32), from which we get that

$$\sqrt{N-1}(\hat{\tau}_N - \tau) \xrightarrow{\mathcal{D}} \mathcal{N}\left(0, \frac{1}{C^*}\right) \quad \text{as } N \to \infty$$

with respect to all randomness. As $\sqrt{N}/\sqrt{N-1} \to 1$ as $N \to \infty$, it also holds that $\sqrt{N}(\hat{\tau}_N - \tau) \xrightarrow{\mathcal{D}} \mathcal{N}\left(0, \frac{1}{C^*}\right)$. □

3.11 Remark. The assumption (3.29) was crucial in proving the theorem, since without it we cannot use Slutsky's Lemma. So far it has not been proven yet that this assumption always holds. However, simulations strongly suggest that the assumption is indeed fulfilled. These simulations also provide an intuition how C^* depends on τ.

Figure 16 shows values of the simulated averages $\frac{\sigma_N^2}{N-1}$ (whose square root appears in assumption (3.29))for different values of τ. The red lines always correspond to the value that was obtained from the biggest tree (for us with $N = 800$ leaves).

Figure 17 shows the dependency between τ and these limits that now correspond to $(C^*)^2$. The red points correspond to the average of the second moments for $N = 800$ leaves, suggesting and exponential decrease. The blue curve is the best fitted (least squares error) exponential function to the data with approximately $f(x) = 0.94\mathrm{e}^{-2.2x}$. From that we can guess that the true value of $(C^*)^2 = (C^*(\tau))^2$ is of the form $(C^*(\tau))^2 = c\mathrm{e}^{-\alpha\tau}$ for some $\alpha > 0$ (that is not far from 2.2) and for some $c > 0$, although $c = 1$ is most likely.

We will use $\hat{\tau}_N$ in the algorithm described in the following chapter.

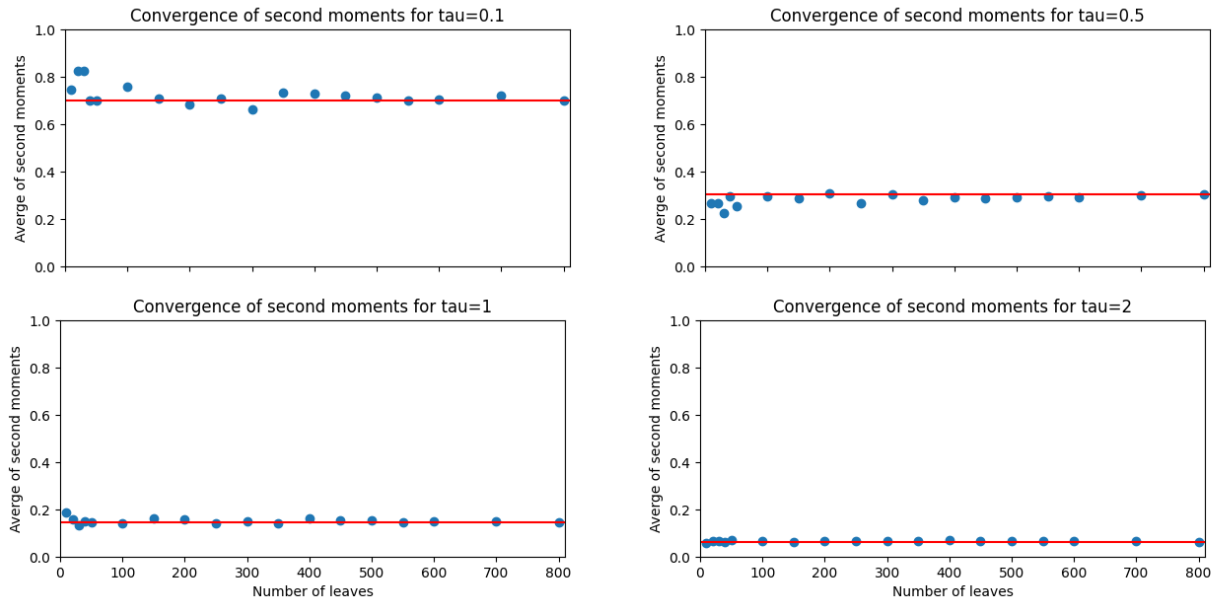


Figure 16: Convergence of second moments with different values of τ.

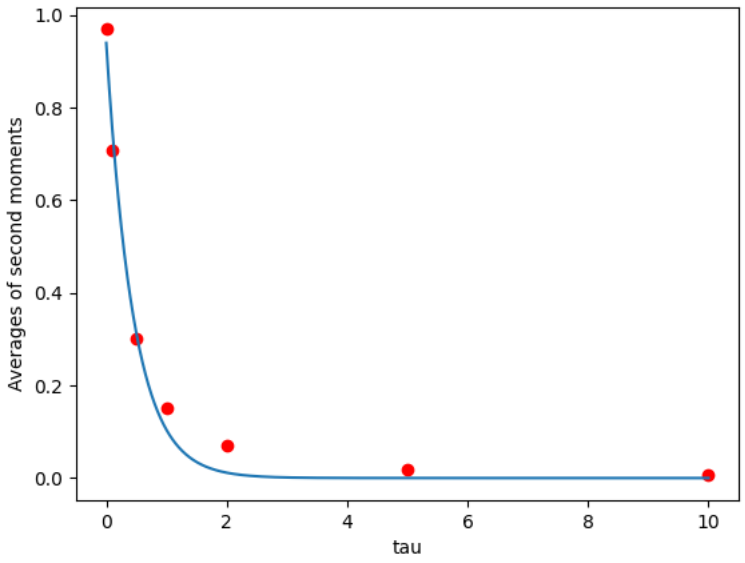


Figure 17: Dependency between τ and C^*

4 A stochastic algorithm for reconstructing joint trees

In this chapter we are describing and analyzing the algorithm for reconstructing the joint tree based on the information coming from the separate phylogenetic trees of the hosts and parasites, both with and without the possibility of using the ghost line.

As the aim of the algorithm is to recover the joint evolutionary history of host and parasites, let us recall the definition of encoding of joint scenarios under both backward models, precisely given in Definition 2.14 and Definition 2.17 .

Let us suppose the sequence of all coalescence times and horizontal transfer times in the scenario is given by $0 = T_0 < T_1 < ... < T_{k_N}$. For each the time interval $[T_{i-1}, T_i)$, the alignment (or with order words, matching or pairing) of the two trees is given by the set of pairs

$$M^i = \{(b_1^i, d_1^i), (b_2^i, , d_2^i), ..., (b_j^i, d_j^i)\}$$

with j denoting the number of currently existing host lineages, b_k^i is a block of the corresponding host partition, and d_k^i is either a block of the corresponding parasite partition or is the empty set for all $k = 1, 2, ..., j$. We say that on the time interval $[T_{i-1}, T_i)$, the branch of the host tree corresponding to b_k^i is infected by the branch of the parasite tree corresponding to d_k^i if $d_k^i \neq \emptyset$. If $d_k^i = \emptyset$ then we say that the branch b_k^i is not infected on the time interval $[T_{i-1}, T_i)$. If the ghost line is also considered, then the matching on the time interval $[T_{i-1}, T_i)$ is given by

$$M^i = \{(b_1^i, d_1^i), (b_2^i, , d_2^i), ..., (b_j^i, d_j^i), (g, \{gd_1, gd_2, ..., gd_k\})\}$$

where j denotes the number of host lineages, and k denotes the number of blocks currently associated with the ghost line. These blocks are denoted by $gd_1, ..., gd_k$ and are blocks of the corresponding parasite partition. If there are no blocks on the ghost line, the last pair is simply reduced to be $(g, \emptyset)$.

4.1 Setup of the algorithm

4.1.1 Input

Our algorithm will take the two separate trees, described by their set of coalescence times and partitions (as in Definition 2.8) as an input together with a matching at the leaves which tells us which host species was infected by which parasite species at the time of sampling. More precisely, the host tree will be encoded with its coalescence times `CoalTimesH` $= \{T_{H,0}, T_{H,1}, ..., T_{H,N-1}\}$ and partitions `PartitionH` $= \{\mathcal{P}_0, \mathcal{P}_1, ..., \mathcal{P}_{N-1}\}$, and the parasite tree with its coalescence times `CoalTimesP` $= \{T_{P,0}, T_{P,1}, ..., T_{P,N-1}\}$ and partitions `PartitionP` $= \{\mathcal{Q}_0, \mathcal{Q}_1, ..., \mathcal{Q}_{N-1}\}$. Moreover, at the beginning we need a matching that tells us which host lineage is infected by which parasite lineage at the time of sampling. When working with simulated data, the first sampled leaf is always denoted by 1, then 2, and so on, and we always have that host lineage i is infected by parasite lineage i at the time of sampling for all $i = 1, 2, ..., N$.

When working with real-life data, the names of the actually sampled species are given, and we relabel them with numbers $1, 2...., N$ in a way that host lineage i is infected by parasite lineage i at the time of sampling for all $i = 1, 2, ..., N$.

4.1.2 Output

The output of the algorithm is a pairing function `PairH`, that always takes a time and a block of the host branch existing at any given time point, and returns the corresponding parasite block with which it is infected, or the empty set in case of no infection. We need to define it at all possible coalescence times, so we create the set `AllTimes`$= \{T_0, T_1, T_2, ..., T_{k_N}\}$ as the union of `CoalTimesH` and `CoalTimesP`. Then `PairH`$(T_i, b) = d$ means that on time interval $[T_i, T_{i+1})$, host branch b is infected by branch d of the parasite tree. If $d = \emptyset$ then there is no current infection on branch b. In the course of the algorithm we use a similar function `PairP` which assigns a host branch to each parasite branch, i.e. `PairP`$(T_i, d) = b$ means that on time interval $[T_i, T_{i+1})$, the parasite lineage corresponding to d is infecting host branch b. Note that `PairP` cannot have the empty set as a value. As described in the previous section, the initial values of `PairH` and `PairP` are given by `PairH`$(0, \{i\}) = \{i\}$ and `PairP`$(0, \{i\}) = \{i\}$ for all $i = 1, 2, ..., N$. We only use `PairP` as an aid to quickly find current branches along which there are infections. All necessary information is already encoded in `PairH` and therefore the output only consists of `PairH`, plus a list of all recovered HT events called `HTlist`.

4.2 Main step of the algorithm

As we know the matching of hosts and parasites at the leaves of the tree, we will go bottom up from the leaves until the root trying to find a pair for each branch of the host tree. At each time $T_i, i = 1, 2, ..., k_N$ in `AllTimes` we check whether it is a merging event only in the parasite tree, only in the host tree or in both. Based on that, we have three different cases.

4.2.1 Merging only in the parasite tree

Let us suppose that we are at index i in `AllTimes` where we have that $T_i \in$ `CoalTimesP` but $T_i \notin$ `CoalTimesH`. Then at time T_i there must have been a HT event which backwards in time results in the merging of two parasite lineages, which we denote by d_1 and d_2. Since we know all the pairs at time T_{i-1}, we know which host lineages carry the two merging parasite lineage. We denote the host lineage carrying d_1 by b_1, and similarly the host lineage carrying d_2 by b_2. Now we know the time of the transfer event and the two branches participating in it, we just need to decide its direction, i.e. which branch is the donor and which is the recipient.

If T_i is the last merging event in the parasite tree (in backwards sense), then we do not have any more information to base our decision on (as there are no more merging events in the parasite tree), so in this case we make a random choice out of b_1 and b_2, and the picked one

will be the donor and the other one the recipient. Without loss of generality, assuming that we picked b_1, then we have $\texttt{PairH}(T_i, b_1) = d_1 \cup d_2$, $\texttt{PairH}(T_i, b_2) = \emptyset$ and $\texttt{PairP}(T_i, d_1 \cup d_2) = b_1$. All other host and parasite branches keep their pairs from the previous time interval.

However, **if T_i is not the last merging event** in the parasite tree, then based on the merging events still to come in both trees, we can make a better informed decision. For that we introduce some variables that we will use during the process. Let t denote the next time when the newly merged parasite block $d_1 \cup d_2$ is merging again with another parasite block, which we denote by d_3. Moreover, let $\tilde{t}$ be the next (smallest) time when either b_1 or b_2 is merging with another branch of the host tree. Lastly, let $t_{MRCA}(b_1, b_2)$ denote the time when b_1 and b_2 find their most recent common ancestor. Now we distinguish between the following cases:

Case 1: if $\tilde{t} = t_{MRCA}(b_1, b_2)$.
In this case the two host branches participating in the HT event in question are merging with each other before merging with other host lineage. In this case there is no information on which branch is the donor, as after they merge, only one of them remains which carries the infection along anyway. So in this case we make a random choice to assign the donor and the recipient of the transfer. Figure 18 illustrates this scenario. The blue tree represents the host tree, and the red lines along the branches of the host tree are representing the so far assigned branches of the parasite tree, while the dashed red lines are the branches that are yet to be assigned. The horizontal dashed line represent the time of the currently investigated transfer event. Any following illustrations should be interpreted in the same way.
From now on, in all the other cases we suppose that $\tilde{t} \neq t_{MRCA}(b_1, b_2)$.

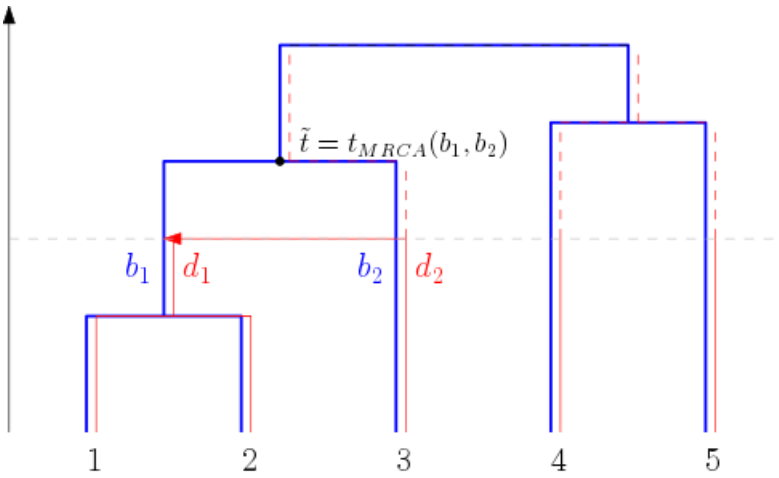


Figure 18: A scenario described in Case 1, where the direction of the transfer is chosen randomly

Case 2: if $\tilde{t} > t$.
If the next merging event of the newly merged block $d_1 \cup d_2$ is before than any of the two host branches carrying the parasite blocks d_1 and d_2, then there is another HT event happening. However, we know that at t, the new block is merging with d_3. Let the set ot descendants of d_3

at the current time T_i be denoted by $\{d_3^1, d_3^2, ..., d_3^j\}$ and the host lines which they are infecting $\{b_3^1, b_3^2, ..., b_3^j\}$. For each $k = 1, 2, .., j$, let us compare the times $t_{MRCA}(b_1, b_3^j)$ and $t_{MRCA}(b_2, b_3^j)$. Let h_1 be the number of times where $t_{MRCA}(b_1, b_3^j)$ is strictly bigger than $t_{MRCA}(b_2, b_3^j)$, and let h_2 be the number of times with $t_{MRCA}(b_2, b_3^j)$ being strictly bigger than $t_{MRCA}(b_1, b_3^j)$. In this step we divide the host tree into two subtrees, one that contains b_1 and the other that contains b_2, and compared which has more common ancestors with the lineages along which the descendants of d_3 evolve. If $h_1 > h_2$, then we choose the host lineage b_1 to be the donor of the transfer, and if $h_2 > h_1$ then we choose b_2 to be the donor. In the case of $h_1 = h_2$ we cannot decide, so the donor of the transfer is chosen randomly. Figure 19 illustrates a scenario that is in Case 2. In this case, d_3 has two descendants at time T_i, and both of the host lineages corresponding to these descendants merge with b_2 before than with b_1.

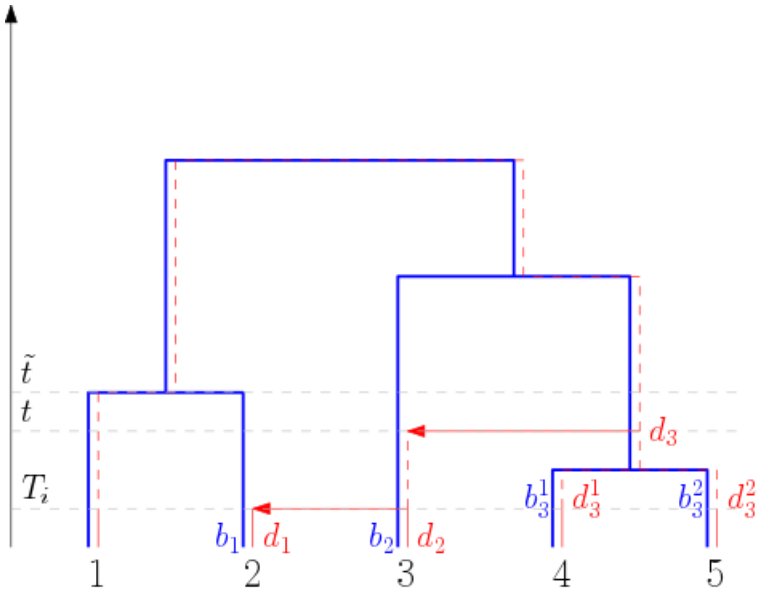


Figure 19: A scenario described in Case 2

Case 3: if $\tilde{t} = t$.

This is the case where we can determine the direction of the transfer with probability 1. Since two different merging events happen exactly at the same time with probability 0 in our model, we know that with probability 1, the merging event of the new parasite block $d_1 \cup d_2$ must be a cospeciation event at $\tilde{t}$. In this case we pick the donor of the transfer to be the one from b_1 and b_2 that is merging at $\tilde{t}$. Figure 20 illustrates this scenario.

Case 4: if $\tilde{t} < t$.

Here we again have two possibilities. Let $\tilde{b}$ denote the host lineage that is merging with either b_1 or b_2 at $\tilde{t}$. If $\tilde{b}$ has an active infection currently at time T_i but we do not see a merging event at time $\tilde{t}$ in the parasite tree, then it means that the HT must have transferred back the infection to the other lineage, i.e. if at $\tilde{t}$, the merging blocks are b_1 and $\tilde{b}$, then the donor of

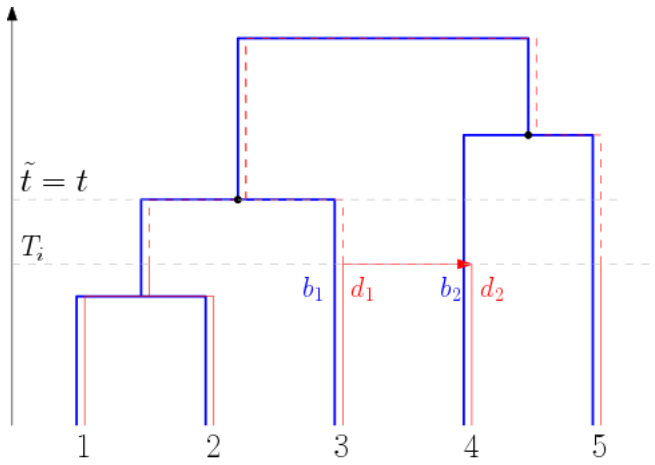


Figure 20: A scenario described in Case 3

the transfer will be b_2 so that we do not see any merging event at $\tilde{t}$ in the parasite tree, and vice versa. However, if $\tilde{b}$ has no infection at time T_i then we would not see a merging event anyway at time $\tilde{b}$ in the parasite tree, so we need to base our decision on something else. We are going to perform the same comparisons as in Case 2, and choose the donor of the transfer accordingly. Figure 21 illustrates this scenario when $\tilde{b}$ has an active infection, while Figure 22 illustrates the case when there is no infection along $\tilde{b}$. In this case d_3 has a single descendant at the current time T_i, which is associated with a host that merges with b_1 before merging with b_2, so the donor will be b_1.

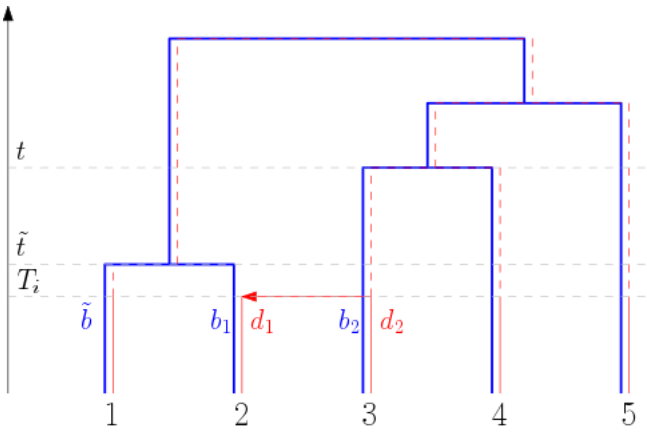


Figure 21: A scenario described in Case 4 with infection along $\tilde{b}$

Assigned pairs

The assigned pairs in all of the above cases are described as follows: Assuming we picked b_1 as the donor, then we have $\texttt{PairH}(T_i, b_1) = d_1 \cup d_2$, $\texttt{PairH}(T_i, b_2) = \emptyset$ and $\texttt{PairP}(T_i, d_1 \cup d_2) = b_1$. If we picked b_2 as the donor, then we have $\texttt{PairH}(T_i, b_2) = d_1 \cup d_2$, $\texttt{PairH}(T_i, b_1) = \emptyset$ and

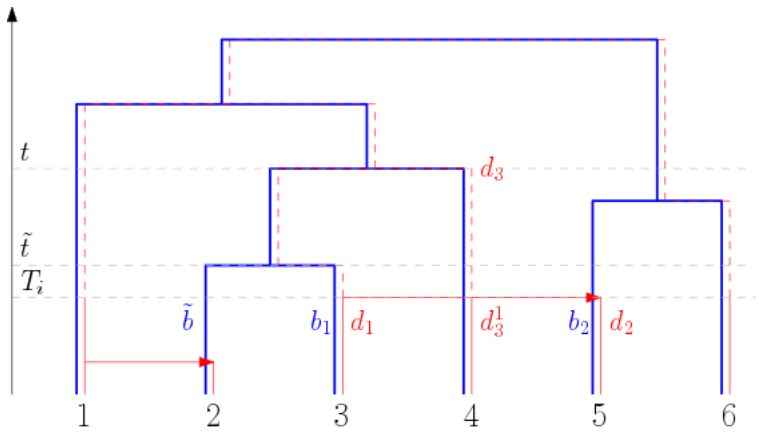


Figure 22: A scenario described in Case 4 without infection along $\tilde{b}$

`PairP`$(T_i, d_1 \cup d_2) = b_2$. All other host and parasite branches keep their pairs from the previous time interval. Moreover, we will add the HT event to `HTlist`.

Possible mistakes

It is important to note that, except in Case 3, it is still possible that we chose the wrong direction for our transfer, especially if we need to make a random choice. In Case 1 the choice we make is only visible until time $\tilde{t}$, after that it disappears with the merging of b_1 and b_2 - this is what we will call later a local mistake. In all the other cases, if we make a mistake it will not disappear and can cause conflicts later in the course of the algorithm - this is what we will call a global mistake. All events that may be a global mistake are stored in a list `GlobalM`. It means that except for Case 1, we always add the HT event at T_i to `GlobalM`. For precise definitions of the different types of mistakes and their influence on the whole joint tree please see Chapter 5.

4.2.2 Merging only in the host tree

In this case we see a merging event only in the host tree, meaning that we have that $T_i \in$ `CoalTimesH` but $T_i \notin$ `CoalTimesP`. Let the two host lineages that are merging at time T_i be denoted by b_1 and b_2, and let their current parasite pairs be denoted by d_1 and d_2.
If at least one of d_1 and d_2 are the empty set, then the matching is straightforward: the pair of the newly merged host branch $b_1 \cup b_2$ will be $d_1 \cup d_2$. If both d_1 and d_2 are empty, then the new host branch will also be empty. In this case, the new pairs will be `PairH`$(T_i, b_1 \cup b_2) = d_1 \cup d_2$ and, if $d_1 \cup d_2 \neq \emptyset$, `PairP`$(T_i, d_1 \cup d_2) = b_1 \cup b_2$, while all other host and parasite lineages keep their pairs from the previous time interval. Figure 23 shows an example of this scenario.

However, **if both $d_1 \neq \emptyset$ and $d_2 \neq \emptyset$** hold, then we face a problem, namely the pairs that we have already assigned and the data are conflicting. There are two possible solutions to this problem. The first possibility is that either d_1 or d_2 gets transferred to another host lineage,

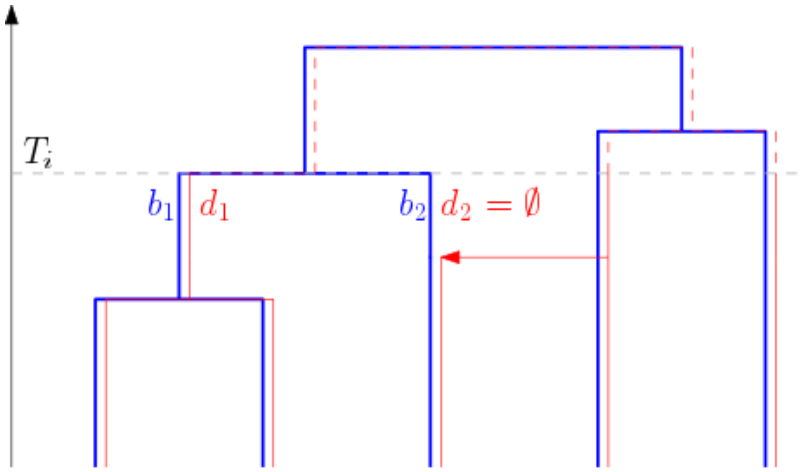


Figure 23: A scenario with no conflicts when merging only in host tree

which has currently no infection. Since at such events there are no merging in the parasite tree, it cannot be seen in the data explicitly. For details on how to choose the empty line to which one infection gets transferred to, please see Chapter 4.3. The other possibility is that previously we have made a mistake - in this case we need to go back to this mistake and correct it, and run the algorithm again from that timepoint. For details on the mistake correcting method please see Chapter 4.5. To choose which approach we are going to take, we first calculate the probability that a transfer actually happens to an empty host lineage in the time interval $[T_{i-1}, T_i)$. If there are currently k host lines that do not have an infection, then the rate at which d_1 gets transferred back to any of them is $k \cdot \tau/2$. Note that here we have a rate $k \cdot \tau/2$ instead of $k \cdot \tau$, since the transfers bringing away d_1 need to have the empty lineage as donor and b_1 as recipient. With the notation $t_i = T_i - T_{i-1}$, we have that the probability that d_1 does not get transferred to any of the empty host lines in the time interval $[T_{i-1}, T_i)$ is $p_i = \mathrm{e}^{-t_i \cdot k \cdot \tau/2}$. Since transfers happen independently for each pair of host lineages, we have that the probability that d_2 does not get transferred to an empty line is also p_i. This means that the probability that at least one of them gets transferred to an empty line is $q_i = 1 - p_i^2 = 1 - \mathrm{e}^{t_i \cdot k \cdot \tau}$. Now with this probability calculated, we can generate a Bernoulli random variable ξ with parameter q_i. If $\xi = 1$ then we will choose either d_1 or d_2 to be transferred to an empty line, and if that does not work, we try to correct a previous mistake. If $\xi = 0$, then first we try to correct a previous mistake, and if it does not solve the problem, then we try transferring d_1 or d_2 to an empty host line. Transfer to an empty host lineage is depicted in Figure 24, where the additional transfer event is drawn by green. If we decide to transfer one parasite block to an empty host line, we still need a time for that transfer. W.l.o.g assume that we want to transfer d_1 to an currently empty host line e. Let t_e denote the time of the next merging event of e in the host tree, and let $t_{min} = \min(T_i, t_e)$. Then we pick the time T of the new transfer according to the uniform distribution on the interval $[T_{i-1}, t_{min})$.

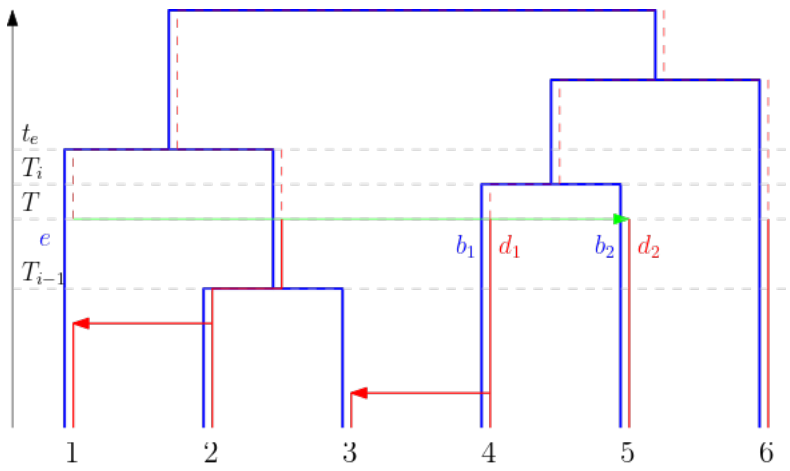


Figure 24: A scenario with an additional transfer involving an empty host lineage

Possible mistakes

Here it is also possible to make mistakes, but differently than in the previous section, where the only uncertainty was the direction of the transfer. Here we can make the mistake that we transfer d_1 to an empty lineage when we should have transferred d_2, and vice versa. This means that neither the donor nor the recipient is known, and of course also the time cannot be determined exactly. So if we decide to transfer one infection to an empty host line, we need to add this event to `GlobalM`.

4.2.3 Merging in both trees

In this case we have that $T_i \in$ `CoalTimesH` and $T_i \in$ `CoalTimesP`. This means that with probability 1 there is a co-speciation event at time T_i. Let the two merging host branches be denoted by b_1 and b_2, and the two merging parasite branches be denoted by d_1 and d_2. Moreover, let the current infections (given by our algorithm) along b_1 and b_2 be denoted by pb_1 and pb_2, i.e. $pb_1 =$ `PairH`(T_{i-1}, b_1) and $pb_2 =$ `PairH`(T_{i-1}, b_2).

If $\{d_1, d_2\} = \{pb_1, pb_2\}$, then this means that our algorithm has reconstructed the correct pairs so far and there is no conflict, at least not on this branch of the tree. In this case the new host branch $b_1 \cup b_2$ will carry the infection $d_1 \cup d_2$. The assigned pairs are `PairH`$(T_i, b_1 \cup b_2) = d_1 \cup d_2$ and `PairP`$(T_i, d_1 \cup d_2) = b_1 \cup b_2$. All other host and parasite lineages keep their pairs (pairings) from the previous time interval.

If $\{d_1, d_2\} \neq \{pb_1, pb_2\}$, then we need to again divide the analysis into several cases.

Case 1: $pb_1 \neq \emptyset$ **and** $pb_2 \neq \emptyset$**.**

This case arises when both merging host lines have an infection and they do not coincide with the merging parasite blocks. Here we have again several possibilities. First we check whether pb_1 or pb_2 coincides with either d_1 or d_2. Based on that, we divide the analysis again into several cases.

Case 1a): exactly one from pb_1 and pb_2 coincides with d_1 or d_2. Without loss of generality we will assume that $pb_1 = d_1$ holds, the other cases can be handled similarly. Let $pd_2 = \texttt{PairP}(T_{i-1}, d_2)$, i.e. the host line which is infected currently by d_2. In this case, we transfer the infection pb_2 to an empty line (if such a line exists) and then transfer d_2 to the newly empty line b_2 from its original host, all in the time interval (T_{i-1}, T_i). To do that, for the time of the first transfer we generate a random variable T that is uniformly distributed on the interval (T_{i-1}, T_i) and add (T, b, b_2) to `HTlist` where b is a currently empty line, chosen by the method described in Chapter 4.3. The new pairs at time T will be given by $\texttt{PairP}(T, d_1) = b_1$, $\texttt{PairP}(T, d_2) = pd_2$ and $\texttt{PairP}(T, pb_2) = b$ as well as $\texttt{PairH}(T, b_1) = d_1$, $\texttt{PairH}(T, b_2) = \emptyset$, $\texttt{PairH}(T, pd_2) = d_2$, and $\texttt{PairH}(T, b) = pb_2$ while all other lineages keep their pairs from the previous time T_{i-1}. Then, we generate another random variable T' that is uniformly distributed on the interval (T, T_i), and we add (T', b_2, pd_2) to `HTlist` where pd_2 is the host which is infected by d_2 previously. The new pairs at time T' will be given by $\texttt{PairP}(T', d_1) = b_1$, $\texttt{PairP}(T', d_2) = b_2$ and $\texttt{PairP}(T', pb_2) = b$ as well as $\texttt{PairH}(T', b_1) = d_1$, $\texttt{PairH}(T', b_2) = d_2$, $\texttt{PairH}(T', pd_2) = \emptyset$, and $\texttt{PairH}(T', b) = pb_2$ while all other lineages keep their pairs from the previous time T. Now the merging parasite blocks and the pairs of merging host lineages coincide and they can be merged at time T_i: $\texttt{PairH}(T_i, b_1 \cup b_2) = d_1 \cup d_2$ and $\texttt{PairP}(T_i, d_1 \cup d_2) = b_1 \cup b_2$, with all other lineages keeping their pairs from the previous time T'. Figure 25 illustrates this scenario, the additional horizontal transfer events are depicted with green arrows.

Of course, it is also possible that we have made a mistake previously and we need to correct it - this is always the case if there are currently no empty lines. To decide which option we are going to try first, we apply the following method: first we check whether there are currently empty host lineages at all. If yes, we generate a Bernoulli random variable ξ with parameter p, which is exactly chosen to be the probability that the wished transfers happen. If $\xi = 1$, we will add the two transfers and go to the next timepoint. If $\xi = 0$, we assume that we have made a mistake previously and will try to correct for this. To obtain p, suppose we have currently k empty lines at time T_{i-1}. Then the first transfer to an empty line that would bring away pb_2 happens at rate $k \cdot \tau/2$, the time until the first such transfer η_1 is exponentially distributed with the same parameter. If one of these happen before time T_i, then the rate of transfers that are sending back d_2 to b_2 from pd_2 is $\tau/2$, so the first such transfer ζ_1 is exponentially distributed with the same rate. Due to the memoryless property of the exponential distribution, the probability that both of them happen on the time interval $[T_{i-1}, T_i)$ is given by

$$\begin{aligned}
\mathbb{P}(\eta_1 + \zeta_1 < T_i - T_{i-1}) &= \int_0^{T_i - T_{i-1}} f_{\eta_1+\zeta_1}(t)\mathrm{d}t \\
&= \int_0^{T_i - T_{i-1}} \frac{k \cdot \frac{\tau}{2}}{k-1} \left(\mathrm{e}^{-\frac{\tau}{2}t} - \mathrm{e}^{-k\frac{\tau}{2}t}\right) \mathrm{d}t \\
&= \frac{-k\mathrm{e}^{-\frac{\tau}{2}(T_i - T_{i-1})} + \mathrm{e}^{-k\frac{\tau}{2}(T_i - T_{i-1})} + k - 1}{k-1},
\end{aligned}$$

where we used the convolution formula for calculating the density of η_1 and ζ_1 due to them being independent random variables.

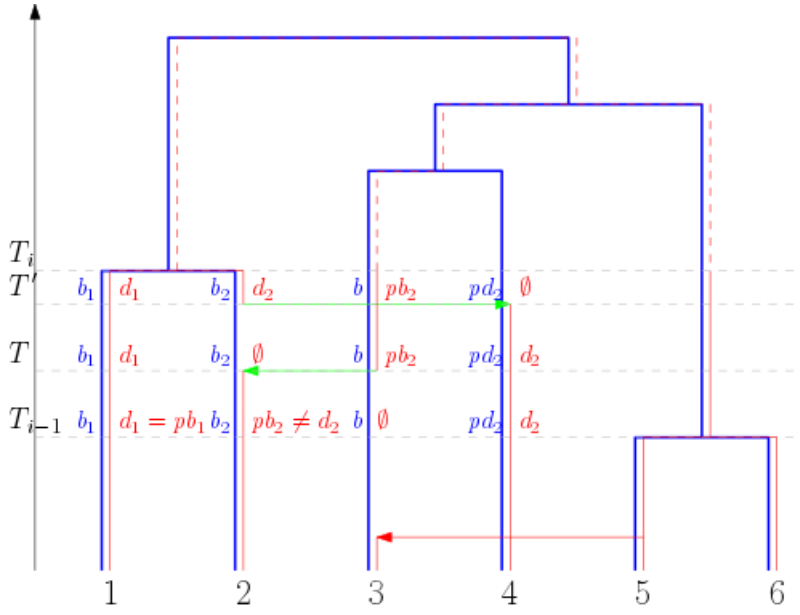


Figure 25: A scenario from Case 1 with two extra transfers

Case 1b): neither pb_1 or pb_2 coincides with d_1 or d_2. Here we use the same approach as in Case 1a): if we suppose we did not make a mistake then we need to transfer away both pb_1 and pb_2 to empty lines, and then transfer d_1 and d_2 to their original places. Otherwise we need to correct a previously committed mistake. We will again choose which one to try first by generating a Bernoulli random variable with parameter q which exactly will be the probability of all the required transfers happening in the correct order and all before T_i. But usually q will be so small such that in the majority of times we will start with trying to correct a previous mistake.

Case 2: $pb_1 \neq \emptyset$ and $pb_2 = \emptyset$ or vice versa.
Without loss of generality we assume here that $pb_1 \neq \emptyset$ and $pb_2 = \emptyset$, the opposite case can be handled in exactly the same way.
If $pb_1 = d_1$ or $pb_1 = d_2$ hold, then the empty host line b_2 must have gotten the infection by a transfer, we were just not able to detect it earlier. Without loss of generality we can assume that $pb_2 = d_2$, and b_1 gets the infection d_1 by transfer. To account for this, we generate a uniform random variable T on the interval (T_{i-1}, T_i) which represents the time of this 'invisible' HT event. The donor will be of course b_1, and the recipient will be the host line that is currently infected by d_1. In this case we add T to `AllTimes` and give a pair to each host and parasite lineage at this time as well. Let the host lineage currently infected by d_1 be denoted by b. Then the pairs at time T are given by $\texttt{PairP}(T, d_2) = b_2$, $\texttt{PairP}(T, d_1) = b_1$, $\texttt{PairH}(T, b_1) = d_1$, $\texttt{PairH}(T, b_2) = d_2$ and $\texttt{PairH}(T, b) = \emptyset$. All other host and parasite lineages keep their pairs

from time T_{i-1}. After creating the new HT event at time T and assigning the pairs accordingly, at time T_i we are in the (ideal) situation where the merging parasite blocks and the infections along the merging host lineages coincide and pairs can be assigned directly to the newly merged lineages: $\texttt{PairH}(T_i, b_1 \cup b_2) = d_1 \cup d_2$ and $\texttt{PairP}(T_i, d_1 \cup d_2) = b_1 \cup b_2$. Moreover, we will add (T, b_2, b) to `HTlist`. This scenario is illustrated in Figure 26, the additional transfer is depicted in green.

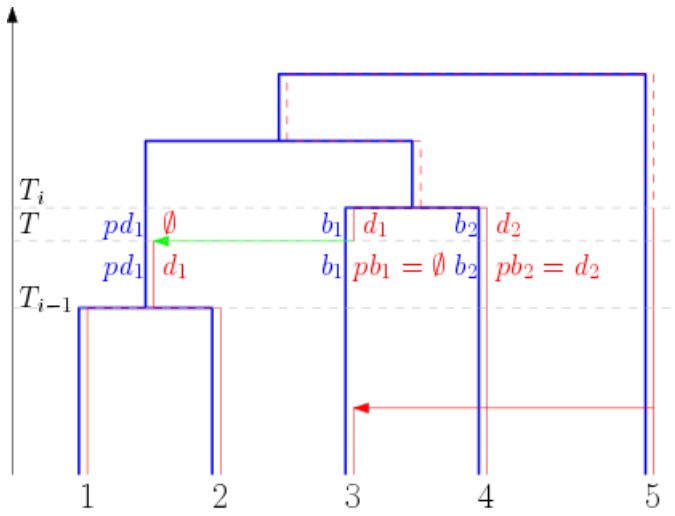


Figure 26: A scenario from Case 2 with $pb_1 \neq d_1$ and $pb_2 = d_2$

If $pb_1 \neq d_1$ and $pb_1 \neq d_2$,, we can use again the method from Case 1: either we transfer pb_1 to an empty line and transfer d_1 to its place while also transferring d_2 from its original host, or we have arrived again to a conflict, which we will resolve by correcting a previously made mistake, as described in Chapter 4.5. The decision which to do first is again decided by generating a random Bernoulli distributed number with parameter p that is exactly the probability that such transfers are happening.

Case 3: $pb_1 = \emptyset$ and $pb_2 = \emptyset$.
In this case, both b_1 and b_2 currently have no infections, so they must have received them by HT events that have so far not been detected. Since we have no more information about the time of these HT events, they will be represented by two independent random variables T and T' that are uniformly distributed on the interval (T_{i-1}, T_i). We will insert them into `AllTimes` and assign pairs to each lineages. Let the host lineages carrying d_1 and d_2 be denoted by b and b'. Without loss of generality we assume that $T < T'$ holds. In this case the pairs at time T are given by $\texttt{PairP}(T, d_1) = b_1$, $\texttt{PairP}(T, d_2) = b'$, $\texttt{PairH}(T, b_1) = d_1$, $\texttt{PairH}(T, b_2) = \emptyset$, $\texttt{PairH}(T, b) = \emptyset$ and $\texttt{PairH}(T, b') = d_2$, while all other lineages keep their pairs from time T_{i-1}. Then moving to time T', the pairs are given by $\texttt{PairP}(T', d_1) = b_1$, $\texttt{PairP}(T, d_2) = b_2$, $\texttt{PairH}(T', b_1) = d_1$, $\texttt{PairH}(T', b_2) = d_2$, $\texttt{PairH}(T', b) = \emptyset$ and $\texttt{PairH}(T', b') = \emptyset$, while all other lineages keep their pairs from time T. Now at time T_i we are again in the (ideal) situation

where the merging parasite blocks and the infections along the merging host blocks coincide and pairs can be assigned easily as $\texttt{PairP}(T_i, d_1 \cup d_2) = b_1 \cup b_2$ and $\texttt{PairH}(T_i, b_1 \cup b_2) = d_1 \cup d_2$, and all other lineages keep their pairs from time T'. Finally, we add (T, b_1, b) and (T, b_2, b') to `HTlist`.

Possible mistakes
In this case we can only make the so-called local mistakes which (will) disappear after the merging event at time T_i. The mistakes come from the fact that we cannot determine the exact time of the newly added HT events, we just know a lower and an upper bound for them. But these times will not be considered when we want to correct a previous mistake in the course of the algorithm.

4.3 Choosing the best empty line

If we have a scenario described in Chapter 4.2.2, when there are only a host merging event at a time but the algorithm has assigned an infection to both of the merging lineages, then we might need to include a new HT event which results in one of the infection being transferred to a previously empty host lineage. We keep the notation of Chapter 4.2.2 throughout this chapter as well. One can of course always randomly pick whether d_1 or d_2 get transferred, and one can randomly pick an empty host lineage as well, but this may result in similar conflicts, only at a different time point. Instead, we propose a more well-informed decision when picking which empty host line to transfer back. The same method is used as well when were are at a scenario described in Case 1a) or Case 1b) in Chapter 4.2.3, but we demonstrate it here using the notation of Chapter 4.2.2.

First we make a random choice out of b_1 and b_2 - without loss of generality we can assume that we picked b_1. We are going to try to transfer the infection d_1 along b_1 to an empty host lineage. First we again check the next merging event of d_1 in the parasite tree - let the time of this event be denoted by t and the other block participating in the merging event with d_3. Moreover, let the set of currently empty host lineages be $\{l_1, l_2, ..., l_k\}$. For each $j = 1, 2, ..., k$, let t_{l_j} be the time of the next merging event of l_j, with block b_{l_j}.
If there exists a $j \in \{1, 2, ..., k\}$ such that $t = t_{l_j}$, then we found which host lineages we need to choose as a merging event in the host tree and a merging event in the parasite tree can happen at the same time (with probability 1) if and only if it is a case of cospeciation. Let $t = \min(T_i, t_{l_j})$, and then we generate the time of the transfer to the empty lineage as a uniformly distributed random variable on the interval $[T_{i-1}, t)$. (We use t as the endpoint of the interval since the time cannot be bigger either than T_i or t_{l_j}.) Then time T is added to `AllTimes` and the pairs are assigned as follows: $\texttt{PairP}(T, d_1) = l_j$, $\texttt{PairP}(T, d_2) = b_2$, $\texttt{PairH}(T, b_1) = \emptyset$, $\texttt{PairH}(T, b_2) = d_2$ and $\texttt{PairH}(T, l_j) = d_1$ while all the other lineages keep their pairs from time T_{i-1}. Then at time T_i we only see d_2 along b_2 as d_1 is transferred away from b_1, and hence the pairs at time T_i are given as $\texttt{PairP}(T_i, d_2) = b_1 \cup b_2$, $\texttt{PairH}(T_i, b_1 \cup b_2) = d_2$, while all other lineages keep their pairs from time T. The HT event (T, l_j, b_1) is added to `HTlist`.

If $t \notin \{t_{l_1}, t_{l_2}, ..., t_{l_k}\}$, we need a different approach. We discard those empty lines l_j for which $t_{l_j} < t$ (since if we transferred d_1 to them it might result in a merging before t). First, we check the descendants of d_3 at the current time T_{i-1}, let them be denoted by $\{d_3^1, d_3^2, ..., d_3^m\}$, and for the non-discarded empty host lines we also check the descendants of the host lineages b_{l_j} (the line l_j is merging with next) for each $j = 1, 2, ..., k$, and we check if they have infections currently. Let the infections along the descendants of l_j be denoted by $\{d_{j,1}, d_{j,2}, ..., d_{j,m_j}\}$ for all $j = 1, 2, ..., k$. For all j we compare the sets $\{d_3^1, d_3^2, ..., d_3^m\}$ and $\{d_{j,1}, d_{j,2}, ..., d_{j,m_j}\}$, and let the index j_{max} be the one for which the intersection $\{d_3^1, d_3^2, ..., d_3^m\} \cap \{d_{j,1}, d_{j,2}, ..., d_{j,m_j}\}$ has the most elements. The idea behind this is to transfer the block d to an empty branch which is in a part of the host tree that carries most of the descendants of d_3 with which d is merging next. If this maximal intersection is non-empty, we will choose $l_{j_{max}}$ to be the empty host line to which we will transfer d_1 to. If there are several indices with the maximal intersection, e.g. j_{max}^1 and j_{max}^2 are both indices with maximal intersections, then we compare $t_{l_{j_{max}^1}}$ and $t_{l_{j_{max}^2}}$ and we choose the index corresponding to the smaller one. The pairs at time T are given by $\texttt{PairP}(T, d_1) = l_{j_{max}}$, $\texttt{PairP}(T, d_2) = b_2$, $\texttt{PairH}(T, b_1) = \emptyset$, $\texttt{PairH}(T, b_2) = d_2$ and $\texttt{PairH}(T, l_{j_{max}}) = d_1$, while all other lineages keep their pairs from T_{i-1}. The pairs at time T_i are given by $\texttt{PairP}(T_i, d_2) = b_1 \cup b_2$, $\texttt{PairH}(T_i, b_1 \cup b_2) = d_2$, while all other lineages keep their pairs from time T. The HT event $(T, l_{j_{max}}, b_1)$ is added to `HTlist`.
If the maximal intersection is empty (i.e. no common infections with any of the descendants of d_3), then we decide not to transfer the infection d_1 away. In this case we try to transfer infection d_2 away with the same procedure.

4.4 Involving the ghost line

If we allow parasite lineages exiting the host phylogeny via transfers, we need to include the ghost line in the algorithm as well. As mentioned before, it will represent the host lineages that are not sampled or have gone extinct. As the ghost line initially has no leave block associated with it, it will be denoted by the algorithm as `out` (as an abbreviation for outclass). As for each host lines, the algorithm will define $\texttt{Pair}(T_i, \texttt{out})$ for all T_i. But how do parasite lineages end up on the ghost line? As the rate at which parasite lineages get transferred to the ghost line is assumed to be small, we check if there is any other solution before actually deciding on using the ghost line. First we always try to resolve conflicts by transferring to empty host lines, or by correcting previous mistakes. But when these methods do not work, we transfer a parasite lineage to the ghost line.

As the ghost line can theoretically carry several infections, its 'pair' is given by a set of blocks that are running on the ghost line, or the empty set if there are no current infections.

When we do need to transfer to the ghost line, we do it in the following way. First, note that a transfer to the ghost line can only happen if we encounter a conflict during a host coalescence time - either when it is only a coalescence time in the host tree (see Chapter 4.2.2) or a coalescence time in both trees (see Chapter 4.2.3). If we are in a situation where at time T_i there is a merging of lineages b_1 and b_2 in the host tree only, but their pairs from the previous

time d_1 and d_2 are both non-empty, and neither transferring one of them to an empty lineage, nor correcting a previous mistake can resolve the conflict, then we pick one randomly from d_1 and d_2 and transfer it to the ghost line. The time of this transfer is chosen from the interval (T_{i-1}, T_i) uniformly at random. When T_i is the merging time of both host lineages b_1 and b_2 and parasite lineages d_1 and d_2 but the current pairs of the host lineages ($\texttt{Pair}(T_{i-1}, b_1)$ and $\texttt{Pair}(T_{i-1}, b_2)$) do not coincide with d_1 and d_2, and neither transferring to empty host lines nor correcting a previous mistake can resolve this conflict, then we will also decide to use the ghost line for d_1 or d_2, occasionally for both of them.

In case we allow the ghost line to be used, at each merging time T_i we need to consider and treat the cases separately when it is involved.
For instance, if T_i is a merging event only in the parasite tree where the merging blocks are d_1 and d_2, and their pairs from the previous time are b_1 and b_2, we need to consider the following cases separately: if neither b_1 nor b_2 are the ghost line, then we proceed exactly as described in Chapter 4.2.1. Else, if exactly one from b_1 and b_2 is the ghost line, then we decide that the transfer was originally from the non-ghost to the ghost, so that backwards in time an infection is transferred from the ghost to an existing branch of the host tree. Finally, if both b_1 and b_2 are the ghost line, then we merge them directly on the ghost line. In this case, we get the new pair of the ghost line by taking away both d_1 and d_2 from the previous pair and adding $d_1 \cup d_2$.
If T_i is a merging event only in the host tree, the only way that involves the ghost line is when we need to transfer one infection to it, as described previously.
If T_i is a merging event in both trees, besides considering the possibilities to transfer pb_1 or pb_2 to the ghost line, we also need to consider the cases when the original pairs of the merging infections pd_1 or pd_2 are the ghost line themselves and update the pairs accordingly.

4.5 Correcting a previous mistake

There are several possibilities where we might run into a conflict: this means that the merging events given by the data of the more distant past and the pairs that our algorithm has given so far are not compatible. In this chapter we describe how we can correct the previously assigned pairings. We always discover mistakes at host merging events - either when there is only a merging in the host tree, but we find infections along both merging host lines that do not merge in the parasite tree at the given time, or at cospeciation events, where the merging parasite blocks do not coincide with the infections along the merging host lineages. Let us suppose that we are at time T_i where we discovered such a mistake. Suppose moreover that at time T_i, the merging host blocks are b_1 and b_2, and their current infections are d_1 and d_2. As conflicts result from mistakes that affect the descendants of current infections, we only consider those mistakes that affect such infections. Recall that the list of HT events that can cause such a conflict is denoted by `GlobalM`. As we go bottom up the trees, the times of events in `GlobalM` are increasing. When we want to correct a mistake, we always try to correct the most recent one so that we do not need to discard so many of the assigned pairs. So we go along backwards in `GlobalM` and see whether the infection(s) along the branches participating in the

investigated HT event are part of the descendants of d_1 or d_2. If yes, we correct this mistake, and go back in time to this HT event, deleting every assigned pair that happened after this time, as well as deleting all HT events that happened after this time. From here, we run again the algorithm, but with the mistake corrected. So let us suppose that we have found the most recent HT event (T, b, b') in `GlobalM` where the transferred infection is actually a descendant of d_1 or d_2.
If this event is a HT event described in Chapter 4.2.1, then we just simply change the HT to (T, b', b) and update the pairs accordingly. After that we will delete all pairs after time T, and we remove those HT events that happened after T.
If the chosen HT event is a HT event to an empty host lineage, described in Chapter 4.2.2, then we try transferring the other infection along the other lineage. To determine exactly which empty line we choose, we apply the method of Chapter 4.3. We need to regenerate the time of the HT event as well, and update the pairs. Again, we delete all assigned pairs that happened after this re-generated HT event, and also those HT events that happen after this.
If the chosen HT event is a HT event that transfers a block of the parasite tree to the ghost line, then we try transferring the other possible block to the ghost line. As described previously, we need to regenerate the time of transfer to the ghost line as and update the pairs. Again, we delete all assigned pairs that happened after this newly assigned transfer, and also those HT events that happen after this.

To prevent the algorithm running into an infinite cycle of trying to correct mistakes, we always keep track of the global mistake with the smallest time among those which the algorithm has tried to correct so far. With this smallest time fixed, we will only allow two corrections for each mistake. If we arrive to a mistake whose time is smaller than all of the previously investigated mistakes, we allow again two possible corrections for each mistake. This way, if there are k possible global mistakes, there will be at most 2^k mistake corrections. However, this is just an upper bound as in no scenario will all the possible global mistakes be corrected two times.

4.6 Time complexity

In this chapter we investigate the time complexity of the algorithm described previously.

The algorithm is implemented in Python 3, so first we would like to recall the complexity of some well-known built-in functions that we often use in the course of the algorithm. When working with lists of length n, checking if an element is contained in the list, searching for the index of a given element, inserting an element to a given index, removing a certain element all have a complexity of $\mathrm{O}(n)$. Sorting a list of length n is of complexity $\mathrm{O}(n \log n)$. When working with sets, checking if a given element is contained in the set is of complexity $\mathrm{O}(1)$. When comparing two sets of lengths m and n, the complexity of finding their intersection is of $\mathrm{O}(\min(m, n))$, and finding the set difference of them can be done in also in linear time. When converting variables from one type to another (e.g. from list to tuple, set, etc.), it is also of complexity $\mathrm{O}(n)$. Copying and deleting lists (tuples, sets, etc.) of length n is also of linear complexity.

With this in mind, we first investigate the time complexity of some auxiliary functions used by the algorithm, with N denoting again the number of leaves.
The first thing that the algorithm checks at each timepoint T_i is which two blocks are currently merging. To do so, we need to find the partition corresponding to the time interval $[T_i, T_{i+1})$, and check the set difference of the blocks present in the partition corresponding to the previous time interval and the current set of blocks. Identifying the correct partition is equivalent to find the index of the currently investigated time in the list of coalescence times, which is of order $\mathrm{O}(N)$, and obtaining the set difference is also of order $\mathrm{O}(N)$, which makes this whole function of complexity $\mathrm{O}(N)$.
The next function in question is to find the next merging event of a given block of the host or parasite tree. For that, we first find the index j of the currently investigated time in the list of coalescence times. Then we increase j as long as the current block is present in the corresponding partition, and we stop when it is no longer there, because once this block is not present in the partition anymore means that it had coalesced with another block. The time of the next merging event then is the element of the list of coalescence times with the index where we stopped. This can be done in $\mathrm{O}(N^2)$ steps. After that it is easy to obtain the other block participating in the merging event, again by taking the set difference of the blocks of the partitions at the stopping time and at the previous time. So finding the next merging event can be done in $\mathrm{O}(N^2)$ steps.
Now we find the complexity of finding (the time of) the most recent common ancestor of two blocks, b_1 and b_2 of a tree. First we create the block $b_1 \cup b_2$, and then we need to find the time interval where $b_1 \cup b_2$ will be a subset of a block in the corresponding partition. This requires $\mathrm{O}(N^2)$ steps.
Next we consider the function that gives us the set of descendants at a certain time t of a currently (at time T_i) conidered block b. To do so, first we need to find the partition corresponding to the time where we want to obtain the descendant of the given block, i.e. we need an index j such that $T_j \leqslant t < T_{j+1}$. Then, for each block of that partition, we need to check whether they are a subset of the given block b, and if yes, add them to the set of descendants of b. As checking if a set is subset of another requires $\mathrm{O}(N)$ steps, the overall complexity of this method is $\mathrm{O}(N^2)$.
When trying to transfer parasite blocks to empty lines, the algorithm needs to obtain the set of all currently empty host lineages. This is done by simply checking the pair of each current host lineage, which takes $\mathrm{O}(N)$ time.
Now let us move on to the function which decides which empty line to choose in case of such a transfer. As described in Chapter 4.3, we need to consider the next merging events of all currently empty host lineages and the next merging event of the parasite block to be transferred - this already takes $\mathrm{O}(k \cdot N^2)$ steps, where k is the number of empty lines. To find whether the time of the next merging event of d coincides with the time of the next merging event of an empty line, takes k comparisons. If it does, then we already have our empty line. If not, then for all the k empty lines, we need to obtain the descendants of the parasite blocks along the host branches they are merging with next, which again requires $\mathrm{O}(k \cdot N^2)$ steps. Subsequently we need to check which one of these sets has the maximal intersection with the set of descendants

of the block d is merging with next, which again requires $O(k \cdot N^2)$ steps. All in all, considering that k lies between 0 and N, we get that the complexity of this whole method is $O(N^3)$.
Lastly, we consider the mistake correcting method described in Chapter 4.5. During the course of this method, we need to investigate all possible global mistakes, starting from the most recent one. As conflicts are always discovered at host merging events, we take the parasite lineages d_1 and d_2 associated with the two merging host branches in question, and compute their sets of descendants at the time of the currently investigated possible global mistake. Then we check which parasite lineages are associated with the host branches taking part in the transfer event at the time of the currently investigated global mistake. If at least one of them is in the set of descendant of d_1 or d_2, then we decide to correct this mistake. So far we have made $O(N^2)$ steps. Then we have three options. If during this global mistake we only assigned the wrong direction to the transfer, we simply change the donor and recipient branches, and update the pairs accordingly. This is done in $O(N)$ steps. If it was a mistake when transferring the wrong parasite lineage to an empty line, then we need to transfer the other possible parasite lineage to an empty line and then update the pairs, which is done in $O(N^3)$ steps. If we transferred a wrong parasite block to the ghost line, we correct this by transferring the other possible parasite block to the ghost line and update the pairs accordingly, it is also done in $O(N)$ steps. All in all, it takes a maximum of $O(N^3)$ steps for a global mistake to be corrected. If the first k possible global mistakes were not corrected (due to them not playing a role in the current conflict or they have been corrected two times), this mistake correcting method requires $O(k \cdot N^2 + N^3) = O(N^3)$ steps as k can maximally be of $O(N)$.

Finally, we are ready to derive the time complexity of our algorithm. We need to consider the complexity of each of the cases described in Chapters 4.2.1, 4.2.2 and 4.2.3.
In the case of Chapter 4.2.1, there is a merging event only in the parasite tree. In the worst case scenario we need to compute the set of descendants of d_3, consider the host branches associated with them, and then for each such host branch we need to compare whether they merge first with b_1 or with b_2. This requires $O(N^3)$ steps.
In the case of Chapter 4.2.2, in the worst case scenario we need to transfer a parasite lineage to an empty lineage or correct a previous mistake, both of them requiring $O(N^3)$ steps.
Similarly, in the case of Chapter 4.2.3, we either need to transfer parasite lineage(s) to empty host lineage(s) or need to correct a previous mistake, which again requires $O(N^3)$ steps.
Note that when correcting a previous mistake, we go back in time to this mistake, correct it and delete each assigned pair after this timepoint, and basically restart the whole algorithm. If the number of times when a mistake was corrected is not high, i.e. of $O(1)$, then the whole algorithm will have a complexity of $O(N^4)$, as the number of all possible time intervals where we need to assign pairs is of $O(N)$. This is due to there being exactly N host merging event and N parasite merging events, and on each time interval between merging events there are maximum two new HT events assigned which do not result in merging events. In very complicated scenarios (with high transfer rate) it can happen that the number of times when a mistake was corrected is of $O(N)$ (or even higher), which then would result in a complexity of $O(N^5)$ or higher. But in most of the cases the time complexity of the algorithm will be of $O(N^4)$. We would also

like to remark that as the number of host and parasite lineages decrease during the course of the algorithm, the number of performed steps in each method will also decrease. This makes the methods of originally orders $\mathrm{O}(N), \mathrm{O}(N^2), \mathrm{O}(N^3)$ overestimates. In reality those orders are $\mathrm{O}(k), \mathrm{O}(k^2), \mathrm{O}(k^3)$ for some $k < N$.

5 Performance of the algorithm on simulated data

This chapter is devoted to analyzing the algorithm described in Chapter 4. We first show results given by the algorithm, on input data that is generated by simulations, both under Backward model I and II. In order to quantitatively asses the quality of the reconstructed joint phylogeny, we introduce a distance measure on the space of the joint trees to quantify the distance between the output given by the algorithm and the simulated scenarios. We also describe how to compute this distance measure.

To get a first impression on how well the algorithm performs and the mistakes it can make, let us take a look at Figure 27, which is a simulated and reconciled scenario under Backward model I with $\tau = 1$. The blue tree indicates the host tree and the red tree indicated the parasite tree. Red arrows indicate horizontal transfers for which the algorithm recognized, that at that time a horizontal transfer happened. Green arrows indicate those horizontal transfer events in the simulation that were not recognized by the algorithm.

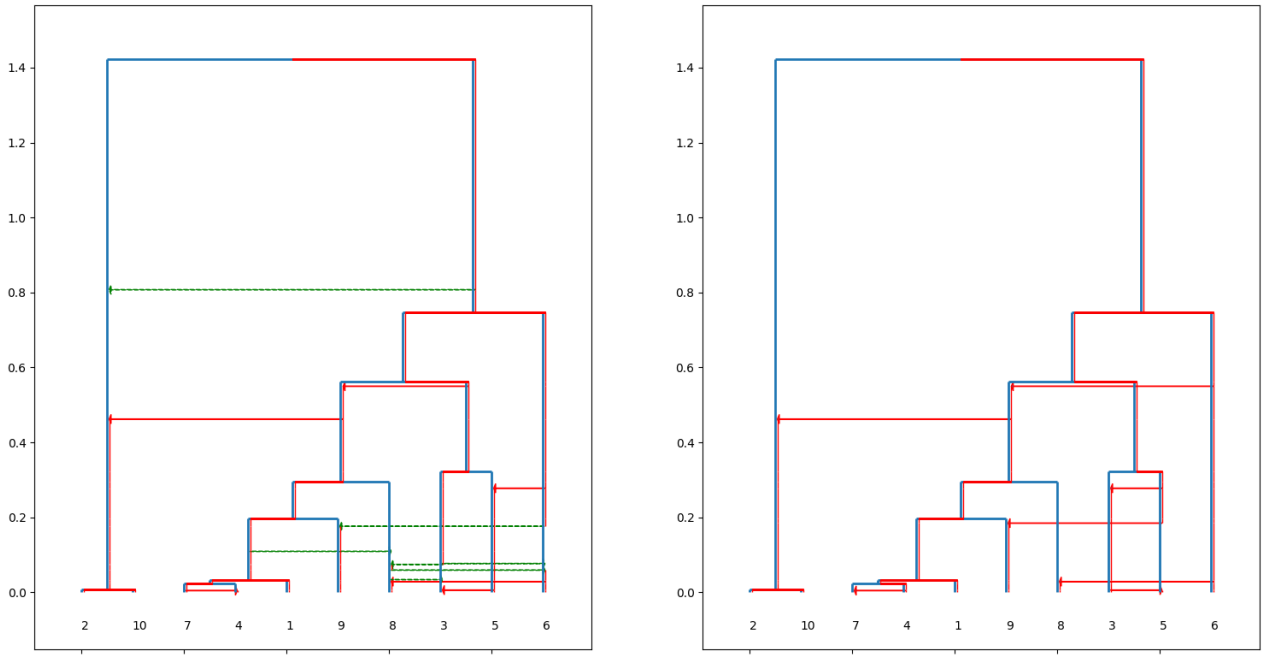


Figure 27: A first look on the algorithm: a simulated versus reconciled scenario for $\tau = 1$. On the left is the true (simulated) joint tree, on the right is the joint tree recovered by the algorithm.

At a first visual inspection, we can see that the reconstructed scenario is very similar to the true scenario. We would like to point out that some of the transfers in green (that were not recovered by the algorithm) are impossible to infer, in particularly those where the recipient (lineage at the tip of the arrow) has currently no parasite lineage associated to them, since the presence or absence of such transfers do not make any difference neither in the host, nor in the parasite trees.

Apart from not recognizing such transfers, the algorithm can make two kind of mistakes, which will be called local and global, respectively. They were already used in Chapter 4, but here we give a precise definition for them.

5.1 Mistakes and conflicts

In order to precisely define them, we need to introduce some notation and some quantities that we use frequently. From now on we assume that we have two joint trees: the "true" joint tree (to be recovered by the algorithm) with set of all coalescence and horizontal transfer times $0 = T_0^{(0)} < T_1^{(0)} < T_2^{(0)} < ... < T_{k_N}^{(0)} < \infty$ and matching $\mathbb{M}^{(0)} = (M_i^{(0)})_{i=0}^{k_N}$ with $M_i^{(0)} = \{(b_{i,1}^{(0)}, d_{i,1}^{(0)}), (b_{i,2}^{(0)}, , d_{i,2}^{(0)}), ..., (b_{i,j_i^0}^{(0)}, d_{i,j_i^0}^{(0)})\}$ being the matching on interval $[T_{i-1}^{(0)}, T_i^{(0)}]$, where the host tree has j_i^0 branches. The second tree will be the reconciled tree (output of the algorithm) with set of all coalescence and horizontal transfer times (that might differ from the true scenario) $0 = T_0^{(1)} < T_1^{(1)} < T_2^{(1)} < ... < T_{l_N}^{(1)} < \infty$ and matching $\mathbb{M}^{(1)} = (M_i^{(1)})_{i=0}^{l_N}$ with $M_i^{(1)} = \{(b_{i,1}^{(1)}, d_{i,1}^{(1)}), (b_{i,2}^{(1)}, , d_{i,2}^{(1)}), ..., (b_{i,j_i^1}^{(1)}, d_{i,j_i^1}^{(1)})\}$ being the matching on interval $[T_{i-1}^{(1)}, T_i^{(1)}]$, where the host tree has j_i^1 branches. We assume that the separate trees are the same in both joint trees.

When talking about only one joint tree, we omit the (0) or (1) superscripts for simplicity. First, we define the next merging event of a block, as it plays an important role in classifying the different mistakes made by the algorithm.

Definition 5.1 *(Next merging event) The next merging event (NME) of block b of the host tree seen at time t is a pair (T_b, B_b) where T_b is the time when b first merges with another block, and B_b denotes the block which b is merging with at time T_b. We use the notation $NME^H(b,t) = (T_b, B_b)$. Similarly, the next merging event of block d of the parasite tree seen at time t is a pair (T_d, B_d) where T_d is the time when d first merges with another block, and B_d is denoting the block which b is merging with at time T_d. We use the notation $NME^P(d,t) = (T_d, B_d)$.*

However, to compare the two joint trees, in particular the matchings on the branches, we need them to be defined on common time intervals. We can do this by taking the union of all coalescence times in both trees, and by refining the matching with respect to this new set of times.

Definition 5.2 *(Refinement of matching) Let the two joint trees with coalescence and horizontal transfer times $0 = T_0^{(0)} < T_1^{(0)} < T_2^{(0)} < ... < T_{k_N}^{(0)} < \infty$ resp. $0 = T_0^{(1)} < T_1^{(1)} < T_2^{(1)} < ... < T_{l_N}^{(1)} < \infty$, and with matchings $\mathbb{M}^{(0)}$ resp. $\mathbb{M}^{(1)}$ be given. The refinement of $\mathbb{M}^{(0)}$ with respect to the times $0 = T_0^{(1)} < T_1^{(1)} < T_2^{(1)} < ... < T_{l_N}^{(1)} < \infty$ is defined as follows: Let $0 = \tilde{T}_0 < \tilde{T}_1 < ... < \tilde{T}_{m_N}$ be the ordering of the set $\{T_0^{(0)}, T_1^{(0)}, ..., T_{k_N}^{(0)}\} \cup \{T_0^{(1)}, T_1^{(1)}, ..., T_{l_N}^{(1)}\}$. We start from time 0 and update $\mathbb{M}^{(0)}$ on each time interval $[\tilde{T}_i, \tilde{T}_{i+1})$, going from the leaves up to the root of the tree. If there exists an index j such that $\tilde{T}_i = T_j^{(0)}$, then the matching on the time interval $[\tilde{T}_i, \tilde{T}_{i+1})$ will be the same as on the time interval $[T_j^{(0)}, T_{j+1}^{(0)})$. Otherwise, if there*

exists an index j such that $T_j^{(0)} < \tilde{T}_i < T_{j+1}^{(0)}$ holds, then the matching on the interval $[\tilde{T}_i, \tilde{T}_{i+1})$ will be the same as on the time interval $[T_j^{(0)}, T_{j+1}^{(0)})$. The refinement of $\mathbb{M}^{(1)}$ with respect to the times $0 = T_0^{(0)} < T_1^{(0)} < T_2^{(0)} < ... < T_{k_N}^{(0)} < \infty$ is defined analogously.

Now, after refining both matchings, we can compare on each time interval whether the pairs are the same in the two scenarios or not. The refined matchings will be denoted as before the refinement. This will lead us to the definition of mistakes and conflicts.

Definition 5.3 *(Mistake) We say that there is a mistake on time interval $[\tilde{T}_i, \tilde{T}_{i+1})$ if there exists an index k such that $(b_{i,k}^{(1)}, d_{i,k}^{(1)}) \in M_i^{(1)}$ but $(b_{i,k}^{(1)}, d_{i,k}^{(1)}) \notin M_i^{(0)}$, i.e. if there exists a branch of the host tree with two different pairs in the two scenarios, or equivalently, there exists a branch of the parasite tree that is associated with different branches of the host tree in the two scenarios.*

This can be either a new mistake that was not present before, or it can arise as a result of a previous mistake that the algorithm has committed.

Definition 5.4 *(Conflict between matchings) We say that there is a conflict between the two matchings at time $\tilde{T}_i$ if there exists a block d in the parasite tree in the time interval $[\tilde{T}_i, \tilde{T}_{i+1})$ such that the next merging event of the host branch associated with d in the two scenarios happens at different times.*

Note that a mistake does not necessarily cause a conflict between the two matchings. Mistakes that do not cause conflicts are not so serious and we will call them local mistakes as they generally not change the global structure of the joint tree. However, when a mistake causes a conflict, it should be avoided (if possible) as it would mean that the order in which branches merge with each other would change globally. These mistakes will be called global mistakes. Before giving a precise definition for them, Figure 28 shows an example for a local and a global mistake. As always, the blue trees indicate host trees, and red trees correspond to parasite trees. Red arrow mean HT events (forward in time).

Now, we can give a formal definition for the types of mistakes.

Definition 5.5 *(Local and global mistakes) Let two joint trees be given with a mistake on time interval $[\tilde{T}_i, \tilde{T}_{i+1})$ with $(b_{i,j}^{(0)}, d_{i,j}) \in M_i^{(0)}$ and $(b_{i,j}^{(1)}, d_{i,j}) \in M_i^{(1)}$ such that $b_{i,j}^{(0)} \neq b_{i,j}^{(1)}$. Let $NME^H(b_{i,j}^{(0)}, \tilde{T}_i) = (t_0, b_0)$ and $NME^H(b_{i,j}^{(1)}, \tilde{T}_i) = (t_1, b_1)$. We say that the mistake is local if $t_0 = t_1$, otherwise we say that the mistake is global.*

As with probability one no two coalescence events happen at the same time, then the merging events at time $t_0 = t_1$ must be the same for local mistakes, which means that local mistakes are exactly the ones that will disappear after the next merging event of the given branch.

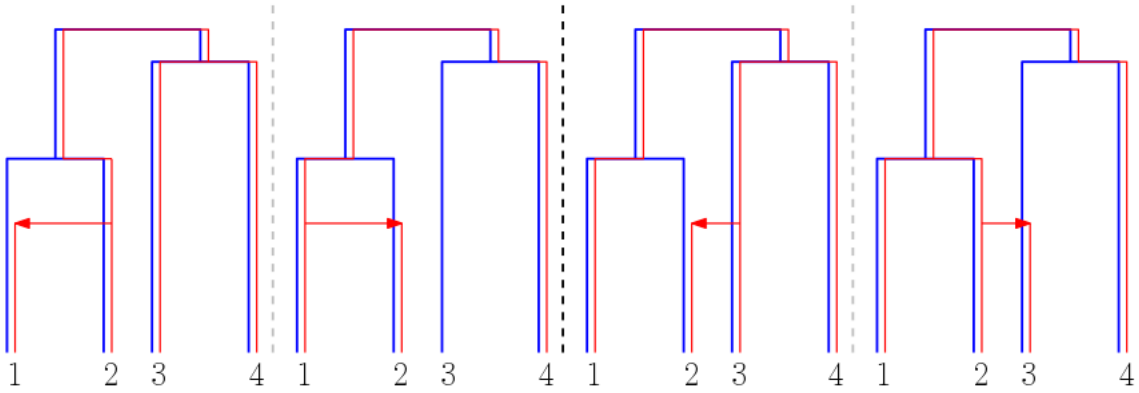


Figure 28: Example of local versus global mistakes. On the left hand side, reversing the direction of the transfer leads to a local mistake, while on the right hand side, reversing the direction of the transfer leads to a global mistake.

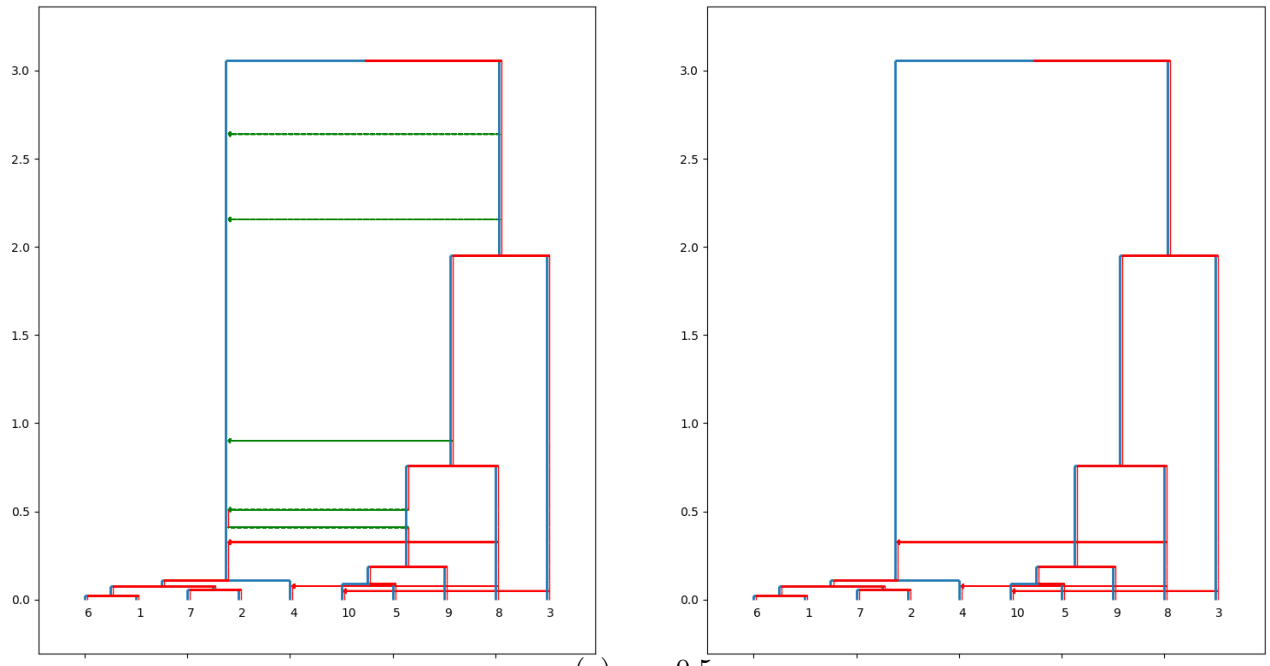
5.2 Performance under backward model I

In this chapter, all of the simulations are generated under Backward model I. First we show some simulated scenarios and how well the algorithm can reconstruct them for different values of τ. Figure 29 shows two such cases, one for $\tau = 0.5$ and one for $\tau = 2$.

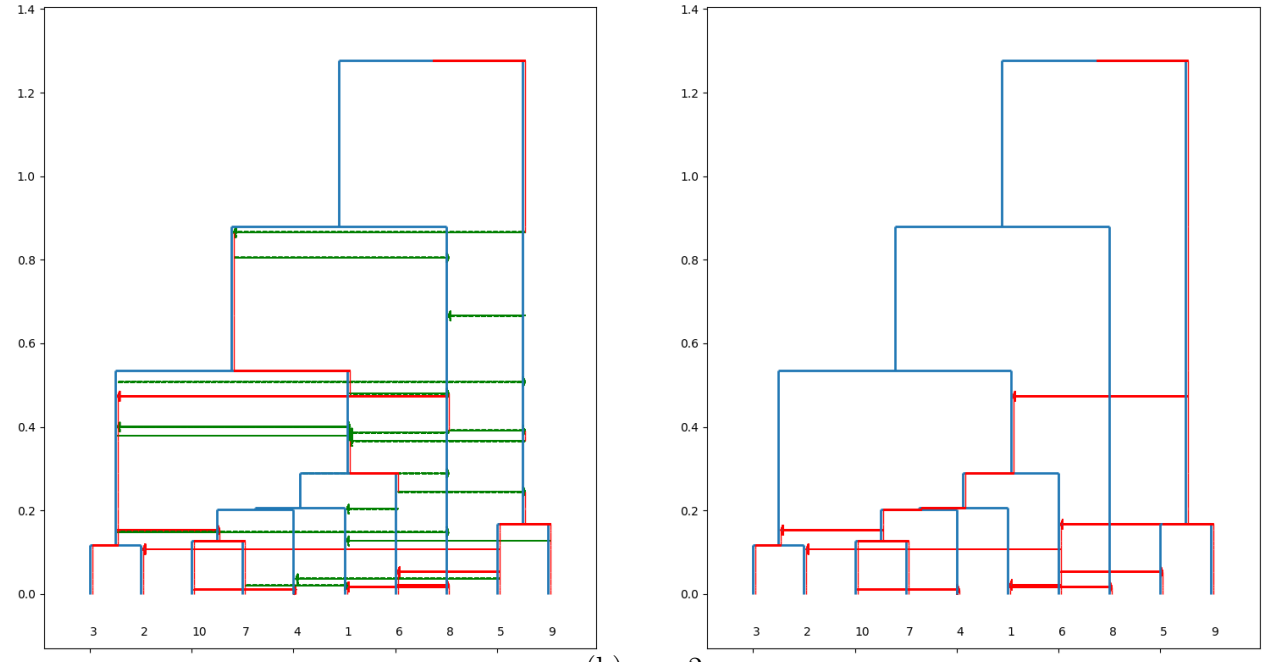
We can instantly see that there are more mistakes for bigger values of τ. We can also observe that in the case of $\tau = 0.5$, the reconstruction is very good, except for not recognizing some transfers. Bute these transfers could not have been detected in any case, since they are transfers to/from empty (non-infected) branches which do not appear in the description of the separate trees. These are exactly the local mistakes that cannot be avoided.

On the other hand, for $\tau = 2$, the reconstruction is a lot more challenging, as is the purely visual assessment of its goodness. One can see that even apart from not reconstructing the (relatively frequent) unrecoverable horizontal transfers, not all other events are captured correctly. This means that the algorithm have made both local and global mistakes.

Our aim is to introduce a distance measure which can be used to evaluate the performance of the algorithm, i.e. it describes how close reconstruction is to the original scenario. As a first idea, one can think to count all of the local and global mistakes at each time interval. However, that would not give us a real distance measure. Instead, we introduce a distance measure which is based on the idea of the edit distance of partitions.
We consider joint trees with the same number of leaves, and we assume that the host trees are the same in both of the scenarios, while the parasite trees do not need to be necessarily the same. We introduce two variants of the measure: one for when there is no ghost line (i.e. when we are under the assumptions of Backward model I) and the second when there is a ghost line (i.e. under the assumptions of Backward model II, see Chapter 5.3). Let the set of all coalescence times in both trees and horizontal transfer times be denoted by $0 = T_0 < T_1 < ... < T_{m_N} < \infty$ and with matchings $\mathbb{M}^{(0)}$ and $\mathbb{M}^{(1)}$ that are already refined with respect to each other (see Definition 5.2).



(a) $\tau = 0.5$



(b) $\tau = 2$

Figure 29: Simulated vs reconciled joint trees for different values of τ under Backward model I

Under Backward model I, on each time interval $[T_{i-1}, T_i)$ the matchings are given by the pairs $M_i^{(0)} = \{(b_{i,1}, d_{i,1}^{(0)}), (b_{i,2}, d_{i,2}^{(0)}), ..., (b_{i,H_i}, d_{i,H_i}^{(0)})\}$ and $M_i^{(1)} = \{(b_{i,1}, d_{i,1}^{(1)}), (b_{i,2}, d_{i,2}^{(1)}), ..., (b_{i,H_i}, d_{i,H_i}^{(1)})\}$, where H_i denotes the number of branches of the host tree on time interval $[T_{i-1}, T_i)$. Note that the first members of the pairs are the same in the two matchings as we assumed that the host tree is the same in both scenarios.

Definition 5.6 *(Distance of joint trees under Backward model I) Suppose we have two joint trees described as above. For each $i = 1, 2, ..., m_N$, let N_i be the minimum number of steps needed to transform the ordered partition $\{d_{i,1}^{(0)}, d_{i,2}^{(0)}, ..., d_{i,H_i}^{(0)}\}$ into $\{d_{i,1}^{(1)}, d_{i,2}^{(1)}, ..., d_{i,H_i}^{(1)}\}$ with two types of steps allowed:*

1. *Step type 1: taking a single element from one block and placing it in another block.*
2. *Step type 2: or swapping two blocks with each other.*

The distance of the matchings $\mathbb{M}^{(0)}$ and $\mathbb{M}^{(1)}$ is the defined as

$$d(\mathbb{M}^{(0)}, \mathbb{M}^{(1)}) = \frac{1}{T_{m_N}} \sum_{i=1}^{m_N} (T_i - T_{i-1}) N_i. \tag{5.1}$$

Before exploring the properties of this distance measure, let us give a short example for each possible step and an example of calculating the distance of two joint scenarios.

1. Step type 1: Taking a single element from a block and placing it in a different block. Example: from $(\{1, 2, 3\}, \{4\})$ we can get to $(\{1, 2\}, \{3, 4\})$ by taking out element 3 from the first block and placing it into the second block.
2. Step type 2: Swapping two blocks with each other. Example: from $(\{1, 2\}, \{3\}, \{4\})$ we can get to $(\{1, 2\}, \{4\}, \{3\})$ by swapping blocks $\{3\}$ and $\{4\}$.

Example 5.7 *(Calculating joint distance) As an example, we calculate the distance of the joint trees depicted in Figure 30.*

- On interval $[T_0, T_1)$ the two matchings are exactly the same, so $N_1 = 0$.
- On interval $[T_1, T_2)$, the ordered partitions corresponding to the parasite blocks are $(\emptyset, \{1, 2\}, \{3\}, \{4\})$ and $(\{1, 2\}, \emptyset, \{3\}, \{4\})$. We can get to the second from the first by swapping blocks $\{1, 2\}$ and $\emptyset$, so $N_2 = 1$.
- On interval $[T_2, T_3)$ the two matchings are again the same, so $N_3 = 0$.
- On interval $[T_3, T_4)$ the ordered partitions corresponding to the parasite blocks are $(\{1, 2, 3\}, \emptyset, \{4\})$ and $(\{1, 2, 4\}, \{3\}, \emptyset)$. We can get to the second from the first by moving element 3 to the third block and moving element 4 to the first block, so $N_4 = 2$.
- On interval $[T_4, T_5)$, the two ordered partitions corresponding to the parasite blocks are $(\{1, 2, 3\}, \{4\})$ and $(\{1, 2, 3, 4\}, \emptyset)$. We can get to the second from the first by moving element 4 to the first block, so $N_5 = 1$.
- By multiplying with the interval lengths (all of length 1 in our simplified example) and norming with the height of the trees, we get that the distance of these two joint trees is

$$d = \frac{1}{5} \left((1 - 0) \cdot 0 + (2 - 1) \cdot 1 + (3 - 2) \cdot 0 + (4 - 3) \cdot 2 + (5 - 4) \cdot 1 \right) = \frac{4}{5}$$

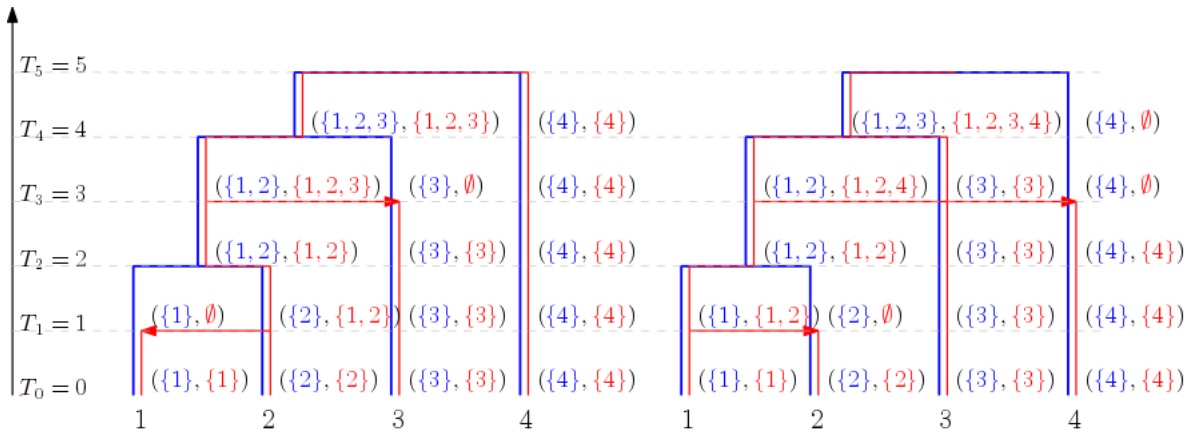


Figure 30: Two joint trees with 4 leaves. Their distance is calculated in Example

□

The numbers N_i give a discrete distance between (ordered) partitions. Here it is very important that the partitions have the same number of blocks, even though some of the blocks are the empty set (corresponding to the uninfected host branches). If we would not care about the order of the blocks (and thus there would be no need for swaps), N_i would exactly give the edit distance (in other words, partition distance, see [Gus02]) of the two partitions. There is no explicit formula of the edit distance of the two partitions, but it is well known that the problem of calculating it can be translated to an assignment problem [Gus02]. This is done in the following way: Let us suppose that the partitions are denoted by P and Q, with blocks $\{p_1, p_2, ..., p_k\}$ and $\{q_1, q_2, ..., q_k\}$, respectively. Finding the minimum number of elements that need to be changed is equivalent to finding the maximum number of elements that do not need to be changed, as one can be obtained by subtracting the other from N. The maximum number of elements do not need to be changed is given by

$$\max_{\sigma \in S_k} \sum_{i=1}^{k} |P_i \cap Q_{\sigma(i)}|$$

with S_k denoting the set of permutations of the set $\{1, 2, ..k\}$. This is exactly the assignment problem with profit matrix $R = (r_{ij})_{i,j=1,...,k}$ with entries $r_{ij} = |P_i \cap Q_j|$. The problem can be solved in polynomial time by the Hungarian algorithm. It was first developed by Kuhn in 1955 [Kuh55], and it was proven by Munkres in 1972 that it runs in polynomial time [Mun57]. The complexity of the original algorithm is $O(N^4)$, but it was later proven that it can be slighty changed in order to achieve a $O(N^3)$ running time [EK72]. Whenever using the algorithm, we used the Python 3 implementation available at `https://gist.github.com/eason1021/6b79784e52700d6da2bd89e982354d98`. After each element is in the correct block, we just need to rearrange the order of the blocks by swaps. It is well known that the selection sort algorithm sorts a list with the least number of swaps required with $O(N^2)$ time complexity. For details on the selection sort algorithm we refer the reader to Section 5.2.3 of [Knu97]. To sum up, N_i can be calculated in $O(N^3)$ steps for each interval $[T_{i-1}, T_i)$ by first arranging the elements with the help of the Hungarian algorithm so that the two non-ordered partitions are the same,

and then we arrange the order of the blocks by the selection sort algorithm. By calculating it for every interval, which have a number of order of N, the overall complexity of calculating the distance between two joint trees with N leaves is $\mathrm{O}(N^4)$ as the number of intervals we need to calculate the distance for is of $\mathrm{O}(N)$.

First, we are going to prove that d is a real distance measure indeed.

Proposition 5.8 *The distance measure d is a real distance measure between joint trees sharing the same host tree, meaning it is non-negative, symmetric and the triangle-inequality holds.*

Proof. It is trivial that $d(\mathbb{M}^{(0)}, \mathbb{M}^{(1)}) \geqslant 0$ always hold, and by definition it can only be 0 if and only if $N_i = 0$ for all i, which is exactly the case when the two matchings are the same.
We also have that $d(\mathbb{M}^{(0)}, \mathbb{M}^{(1)}) = d(\mathbb{M}^{(1)}, \mathbb{M}^{(0)})$ as the N_i are symmetric, since each step can be done in the opposite direction as well.
To check whether the triangle-inequality holds, let us take matchings $\mathbb{M}^{(0)}, \mathbb{M}^{(1)}$ and $\mathbb{M}^{(2)}$, and let us suppose that the set of all times appearing in any of the three matchings is given by $0 = T_0 < T_1 < .... < T_{m_N} < \infty$. Consider the refinement of each matching with respect to this time sequence. This further refinement of the refinements originally created for obtaining the distances $d(\mathbb{M}^{(0)}, \mathbb{M}^{(1)}), d(\mathbb{M}^{(0)}, \mathbb{M}^{(2)})$ and $d(\mathbb{M}^{(2)}, \mathbb{M}^{(1)})$ will not change these distances, since adding a new time in a matching that was previously not there divides a previous time interval into two without changing the pairings of the present branches. Then, on each time interval the number of steps needed to obtain the ordered blocks of parasite lineages from the partitions in $\mathbb{M}^{(0)}$ to those in $\mathbb{M}^{(1)}$ cannot be more than first obtaining the partitions in $\mathbb{M}^{(2)}$ and from them, obtain the partitions in $\mathbb{M}^{(1)}$, which directly implies the desired inequality. □

Now let us see in some simulated scenarios, how the distance measure d is changing as the different parameters of the model change.

First, we would like to point out the fact that if we run the algorithm several times with the same input (i.e. same host trees and same parasite trees), it might give different outputs due to the random choices it makes. This is illustrated in Figure 31 for $\tau = 2$. On the left hand side we can see the true simulated scenario, while in the middle and on the right hand side we can see two different reconciliations given by the algorithm for the same separate trees.

We can see that the main difference between the two outputs is the direction of the transfer between host branches $\{2, 4, 8\}$ and $\{1, 6\}$. The other difference between the two outputs is due to the local mistake when deciding the direction of the transfer between host branches $\{3\}$ and $\{7\}$. Neither of these two outputs captured the direction of the transfer between host branches $\{5, 6, 9, 10\}$ and $\{7\}$ correctly.

To assess the variability between the different outputs, we fixed the same joint tree and separate trees as in Figure 31 and $\tau = 2$. Then we ran the algorithm 100 times and calculated the distance of the outputs from the true scenario. The histogram of these distances are shown in Figure 32.

Of course, this distribution depends very much on the topology of the fixed joint tree and

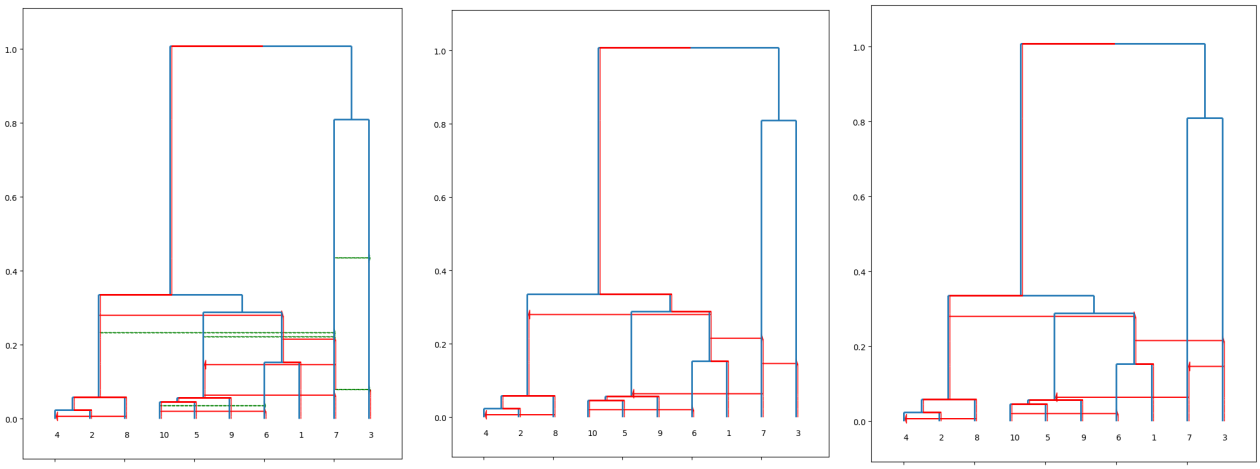


Figure 31: Different outcomes with the same separate trees for $\tau = 2$

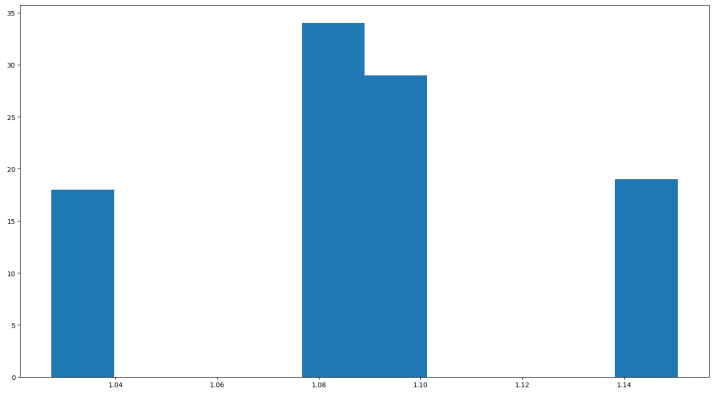


Figure 32: Histogram distances from the true scenario with fixed separate trees

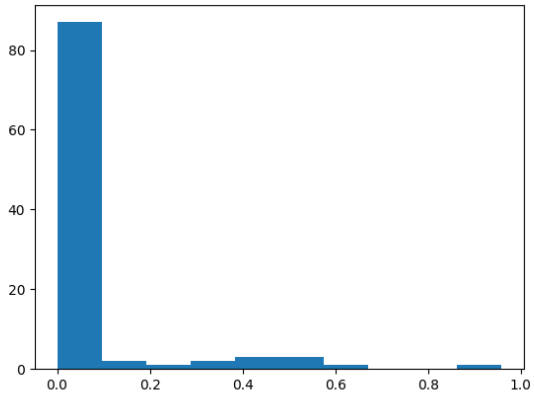
we cannot make general conclusions from it. Here we can see that the distance between each reconciliation and the true scenario can obtain 4 possible values. This is due to the two possibilities for mistakes in the scenario: one is when deciding the direction of transfer between host branches $\{2, 4, 8\}$ and $\{1, 6\}$, and the other is the direction of the transfer between host branches $\{3\}$ and $\{7\}$. Note that the transfer between branches $\{5, 6, 9, 10\}$ and $\{7\}$ will always be recovered in the incorrect direction due to the algorithm selecting the direction of the transfer based on which branch of the host tree is infected by the parasite block that will merge with the block obtained by the merging the corresponding parasite lineages of the host branches participating in the transfer next.

We would like to remark that if there would be horizontal transfers whose donor is an empty host branch, there would be an infinite number of possible distances between the two trees, since the time of such transfers are selected uniformly from an interval.

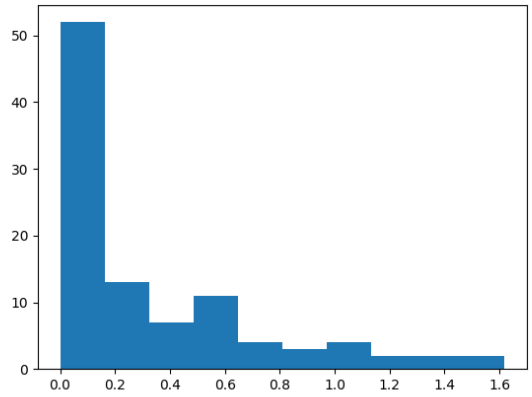
However, we will see in the following that for small values of τ (i.e. $\tau < 1$), the distances

between any recovered scenario and the true scenario is quite small and so all reconstructions are similar in this case.

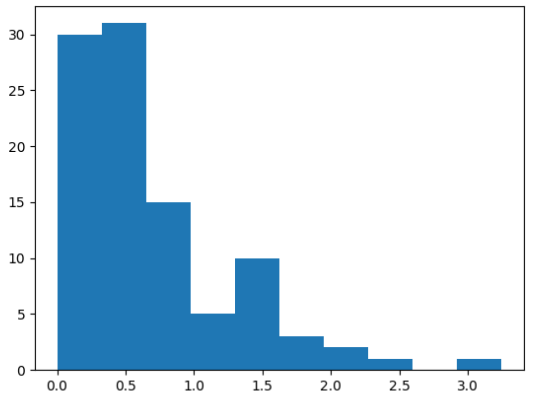
In order to obtain an overall view of the performance, we only fixed N and τ, and simulated 100 (different) joint scenarios with these parameters. We extracted the separate trees from the simulated scenarios and used them as the input of the algorithm to recover the joint scenario, and computed the distances between the simulated (true) scenarios and the recovered ones. For that we ran the algorithm only once for each pair of separate trees. The distribution (histogram) of the distances are shown for $\tau = 0.1, 0.5, 1, 2$ and for $N = 10$ (all histograms are based on 100 simulated scenarios) in Figure 33. Note that $\tau = 1$ is a special case since here the coalescence rate of host branches and the horizontal transfer rate between branches is the same (per pair).



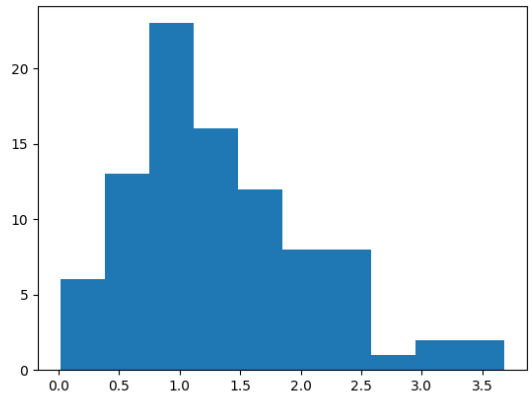
(a) $\tau = 0.1$



(b) $\tau = 0.5$



(c) $\tau = 1$



(d) $\tau = 2$

Figure 33: Distribution of distance from true scenario with $N = 10$ for different values of τ

We can conclude that the algorithm does better for smaller values for τ, as expected. For $\tau = 0.1$ and $\tau = 0.5$ we can see that the majority of the scenarios were recovered correctly, with an exponential-like tail for the bigger distances. For $\tau = 2$ the distances between the true and the recovered scenario are much higher. The case $\tau = 1$ seems to be a change point in the behavior of the distances, as at this point the rate at which pairs of host lineages coalesce is the same as the rate at which parasite lineages coalesce due to horizontal transfer. We would like to note here that for larger values of τ, i.e. when $\tau > 1$, the estimator introduced in Chapter 3 still works well (for big trees) and gives us a glimpse whether the trees are well aligned or not, and whether the algorithm will give a good reconstruction.

5.3 Performance under backward model II

Now we are moving on to introducing the second form of the distance measure, which is when the ghost line is involved. The basic idea remains the same, however, for the ghost line we need to introduce some special rules, namely, the order of the blocks on the ghost line will not matter.

Under Backward model II, where the ghost line is involved, we assume that two matchings are given (already refined with respect to each other) as in Definition 5.6. We only need to add the pairs corresponding to the ghost line to the matchings:

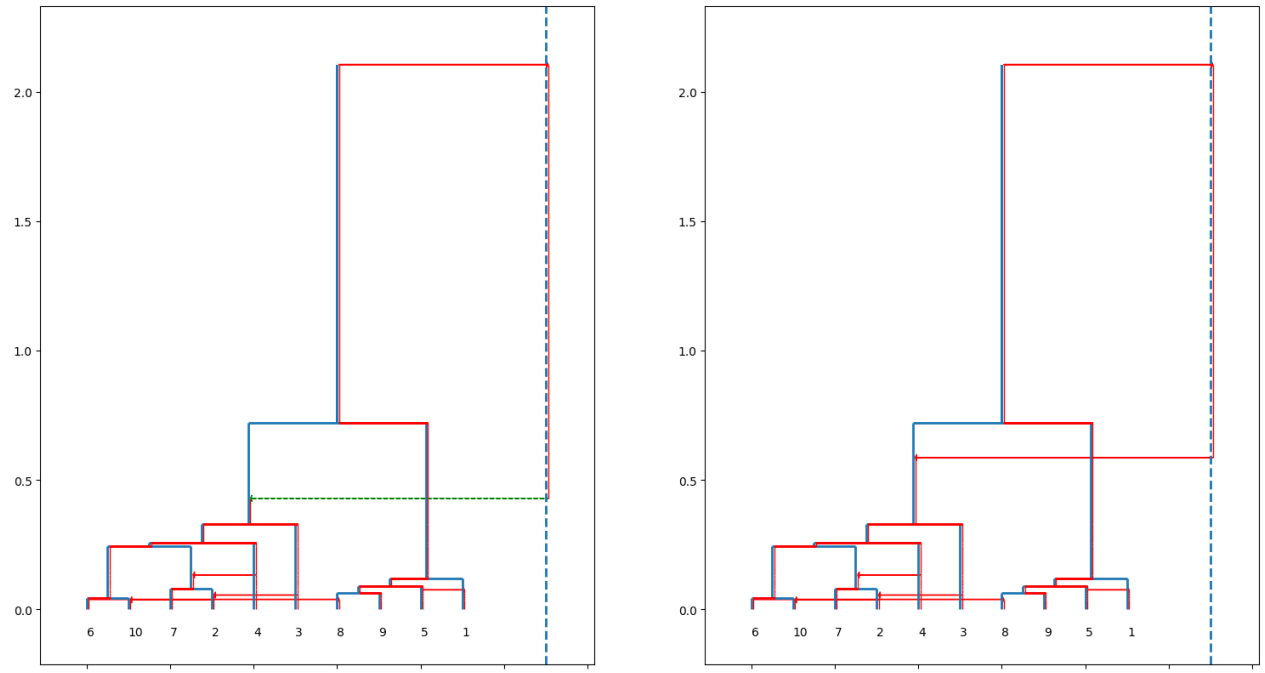
$$M_i^{(0)} = \{(b_{i,1}, d_{i,1}^{(0)}), ..., (b_{i,H_i}, d_{i,H_i}^{(0)}), (g, \{gd_1^{(0)}, ..., gd_{j_0}^{(0)}\})\}$$

and

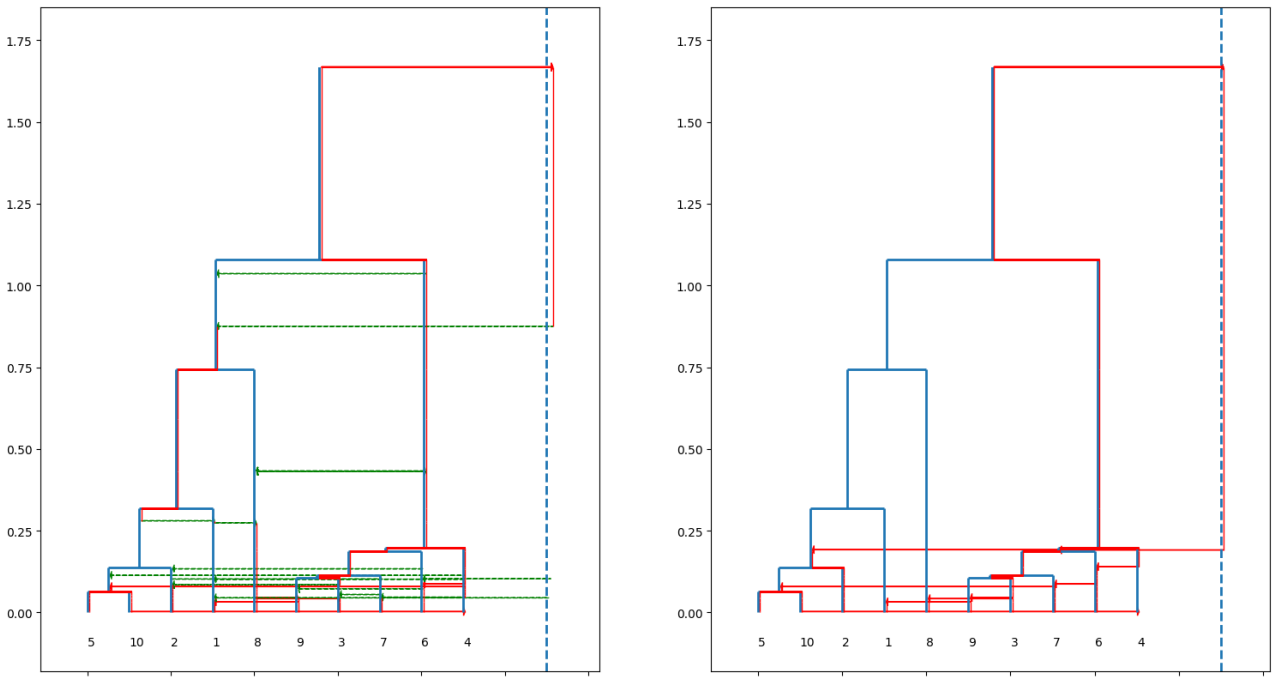
$$M_i^{(1)} = \{(b_{i,1}, d_{i,1}^{(1)}), ..., (b_{i,H_i}, d_{i,H_i}^{(1)}), (g, \{gd_1^{(1)}, ..., gd_{j_1}^{(1)}\})\}$$

Fist, let us show some simulated and recovered scenarios for different values of the parameters. Recall that parameter p corresponds to the rate at which parasite lineage are transferred to the ghost line (per branch of the host tree), and parameter c denotes the rate at which lineages are coming back from the ghost line. (The parameter τ still denotes the rate of horizontal transfers between pairs of branches of the host tree.) The parameter c os kept fixed at $c = 1$ so that it does not take long for the lineages return from the ghost line to the branches of the host tree. Otherwise the coalescence events would happen much faster compared with the length that parasite lineages would spend along the ghost line on average, so that the scenarios under the model would deviate significantly form the previous setting. This would not be realistic, as in the case of biological applications it is always assumed that the two trees are on the same timescale. (Most of the existing reconciliation methods do not allow for the possibility of using the ghost line at all.)

In Figure 34 illustrate the results with $p = 0.25$ fixed, meaning that we do not have a lot of transfers to the ghost line. In both simulated scenarios we only have one such transfer, and both of them are recognized by the algorithm, although not at the correct time. This is due to the fact that the algorithm only uses the ghost line when there is no other possibility. When we look at how much the change in the parameter τ influences the scenarios, we observe similar effects



(a) $\tau = 0.5$



(b) $\tau = 2$

Figure 34: Simulated versus reconciled joint trees for different values of τ under Backward model II with $p = 0.25$ and $c = 1$

as in Figure 29 under Backward model I: For $\tau = 0.5$ the transfers are well recognized, while for $\tau = 2$ the scenario is more complicated and besides the unrecoverable transfers, also some of the topologically important (i.e. those that lead to global mistakes if assigned incorrectly) transfers could not be recovered.

In the scenarios depicted in Figure 35 we allowed more frequent transfers to the ghost line, namely, we set $p = 1$. In both scenarios there are several occasions when the ghost line carries multiple parasite lineages. However, the algorithm recognizes only one such transfer in each scenario, due to it only using the ghost line when there are no other feasible possibilities. Again, similarly to Backward model I, for $\tau = 0.5$ the algorithm recognizes those transfers that are not involving empty host branches, while for $\tau = 2$ the situation is more complicated and even topologically important transfers cannot be recovered very well.

In order to assess the quality of the reconstruction, we introduce a new variant of our distance measure (see Definition 5.6) that also considers the ghost line.

Definition 5.9 *(Distance of joint trees under Backward model II) Suppose we have two joint trees with the ghost line described as before. For each $i = 1, 2, ..., m_N$, let $\tilde{N}_i$ be the minimum number of steps needed to transform the ordered partition $\{d_{i,1}^{(0)}, d_{i,2}^{(0)}, ..., d_{i,H_i}^{(0)}, \{gd_1^{(0)}, gd_2^{(0)}, ..., gd_{j_0}^{(0)}\}\}$ into $\{d_{i,1}^{(1)}, d_{i,2}^{(1)}, ..., d_{i,H_i}^{(1)}, \{gd_1^{(1)}, gd_2^{(1)}, ..., gd_{j_1}^{(1)}\}\}$ with two types of steps allowed:*

1. *Step type 1: Taking a single element from one block and placing it in another block*
2. *Step type 2: Swapping two blocks with each other*

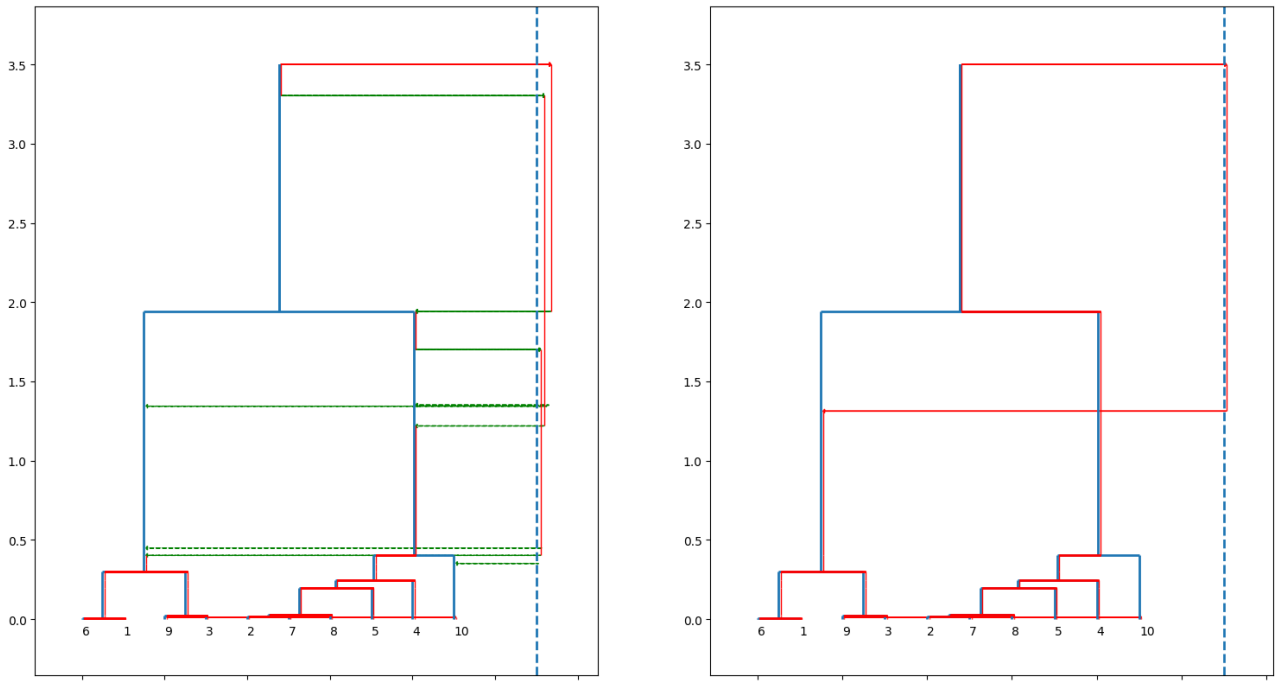
The distance of matchings $\mathbb{M}^{(0)}$ and $\mathbb{M}^{(1)}$ is the defined as

$$d(\mathbb{M}^{(0)}, \mathbb{M}^{(1)}) = \frac{1}{T_{m_N}} \sum_{i=1}^{m_N} (T_i - T_{i-1}) \tilde{N}_i \tag{5.2}$$

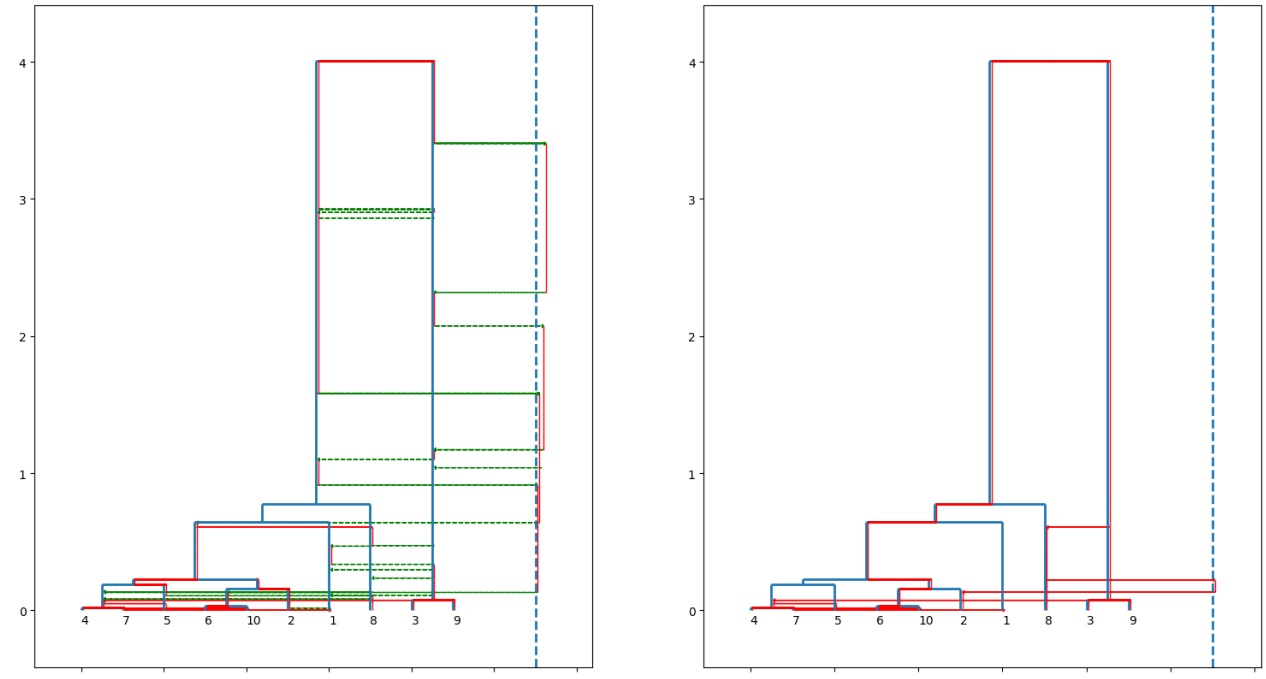
Here the blocks $gd_1^{(0)}, gd_2^{(0)}, ..., gd_{j_0}^{(0)}$ and $gd_1^{(1)}, gd_2^{(1)}, ..., gd_{j_1}^{(1)}$ are put in a set because for these blocks on the ghost line, the order does not matter but only that they are the correct blocks. This means that when calculating $\tilde{N}_i$, we can perform the Hungarian algorithm in order to place the elements into the correct blocks of the partition. But when reordering the blocks, we only need to place the blocks along the branches of the host tree in the correct order.

Example 5.10 *As an example we calculate the joint distance of the two joint scenarios depicted in Figure 36 is as follows:*

- On the interval $[T_0, T_1)$, the two matchings are exactly the same, so $\tilde{N}_1 = 0$.
- On the interval $[T_1, T_2)$ the only difference is in the order of the blocks on host branches $\{2\}$ and $\{3\}$, so with one swap we can get from the first to the second, $\tilde{N}_2 = 1$.



(a) $\tau = 0.5$



(b) $\tau = 2$

Figure 35: Simulated versus reconciled joint trees for different values of τ under Backward model II with $p = 1$ and $c = 1$

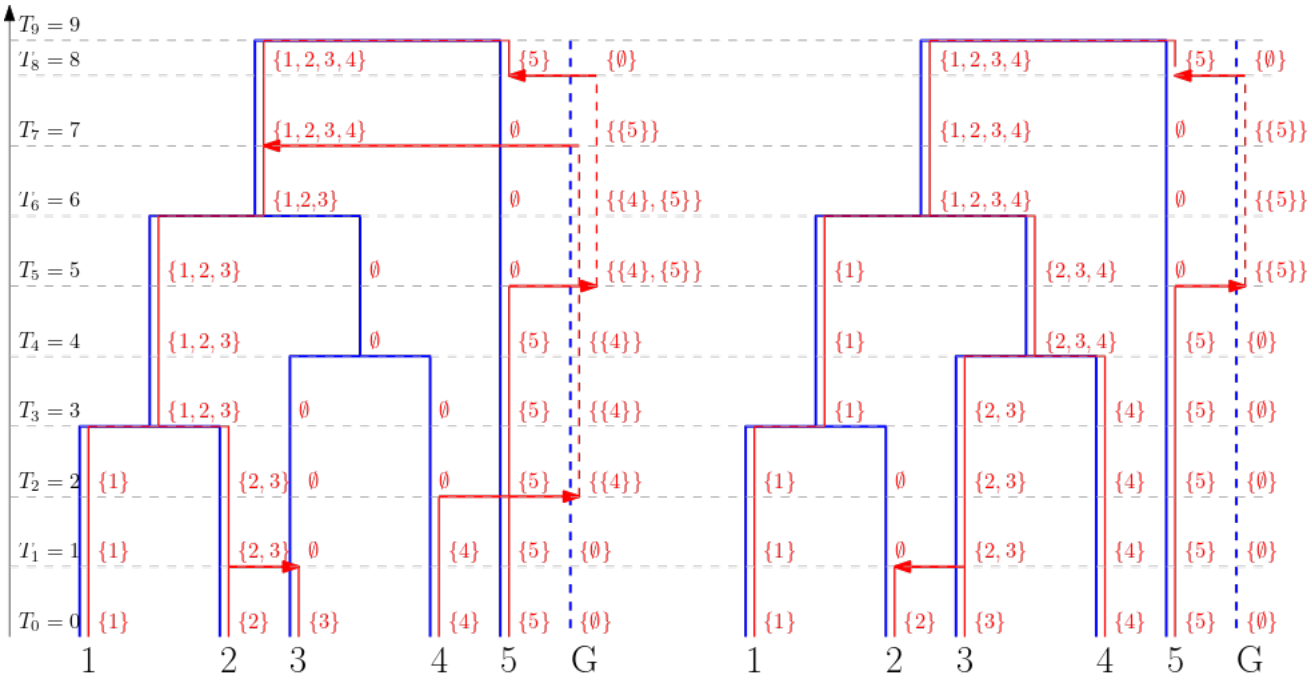


Figure 36: An example to calculate joint distance under Backward model II

- On the interval $[T_2, T_3)$ we have the same difference as on the previous interval, plus the parasite block $\{4\}$ is on the ghost line in the first scenario, while it is still on host branch $\{4\}$ in the second one. So we need two moves here: swap $\{2,3\}$ and the empty set (that is on host branch 3 in the first scenario), and move the element 4 to the ghost line, so $\tilde{N}_3 = 2$.

- On the interval $[T_3, T_4)$, the corresponding ordered parasite partitions are $\{\{1,2,3\}, \emptyset, \emptyset, \{5\}, \{4\}\}$ and $\{\{1\}, \{2,3\}, \{4\}, \{5\}, \emptyset\}$, respectively. One move we certainly need to make is to move block $\{4\}$ from the ghost line to host branch $\{4\}$, and we need to make two steps to account for the block on host branches $\{1,2\}$ and $\{3\}$: either we move elements 2 and 3 from the first block to the second, or we move only element 1 to the second block and then swap the order of the two blocks. In any case, $\tilde{N}_4 = 3$.

- On the interval $[T_4, T_5)$, we need to move elements 2, 3 and 4 to the second host branch, which would mean $\tilde{N}_5 = 3$.

- On the interval $[T_5, T_6)$ we need to move elements 4 and 5 from/to the ghost line, while we need to perform the same operations with the remaining blocks to get from $\{1,2,3\}, \emptyset$ to $\{1\}, \{2,3\}$. So we get that $\tilde{N}_6 = 4$.

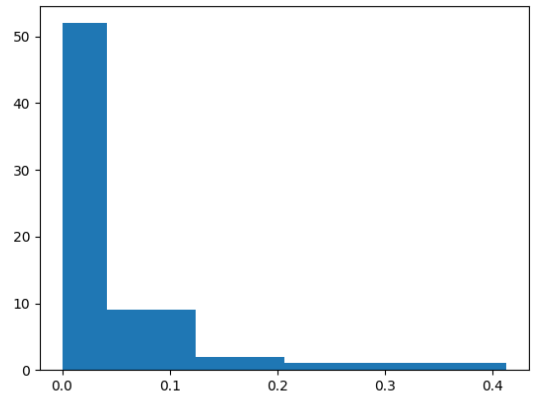
- On the interval $[T_6, T_7)$ we just need to move element 4 to the correct block, everything else is identical in the two matchings. We have $\tilde{N}_7 = 1$.

- On the intervals $[T_7, T_8)$ and $[T_8, T_9)$ the two matchings are identical, so $\tilde{N}_8 = \tilde{N}_9 = 0$.

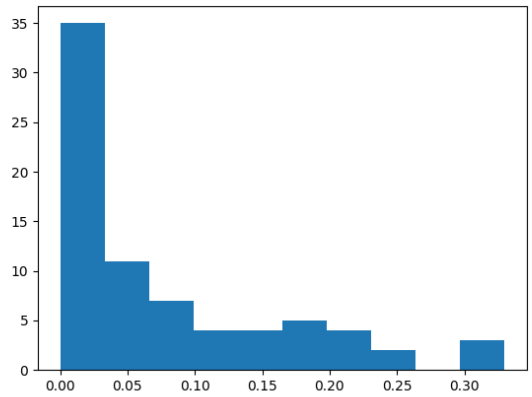
Using that in our (simplified) example each interval has length 1, the distance is given by the sum of all $\tilde{N}_i$ divided by the height of the tree:

$$d = \frac{0+1+2+3+3+4+1+0+0}{9} = \frac{14}{9}.$$

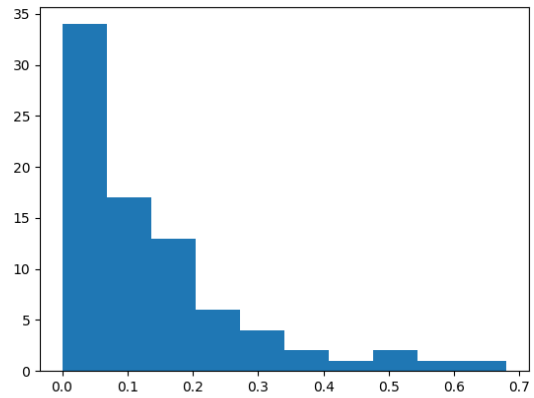
It is easy to see that this version of the distance is also a true distance measure itself, provided we identify those matchings that only differ in the order of blocks along the ghost line. In the following we demonstrate on simulated data how the reconciled scenario differs from the true scenario, for different values of the parameters τ, p and c.



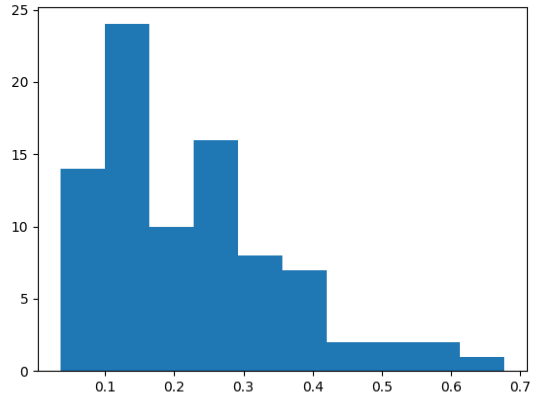
(a) $\tau = 0.1$



(b) $\tau = 0.5$



(c) $\tau = 1$



(d) $\tau = 2$

Figure 37: Distribution of the distance of the reconstructed scenario from the true scenario under Backward model II with $N = 10$ for different values of τ with $p = 0.25$ and $c = 1$

Figure 37 shows the histogram of the distances between the true simulated scenario and the scenario recovered by the algorithm for $N = 10$ with different values of τ. The histograms

are again based on 100 simulated scenarios. Note the different scalings on the y axis for the different histograms. We can observe that the performance of the algorithm using the ghost line is not very different form the performance without the ghost line. For $\tau = 0.1$ and $\tau = 0.5$, the majority of the scenarios are recovered correctly. For $\tau = 1$ and $\tau = 2$ there is a similar trend as in the case without the ghost line.

To sum up, we can say that the algorithm performs generally very well for small rates of horizontal transfer, both with and without the ghost line. For higher rates of horizontal transfer the reconstruction becomes more challenging and the outputs generated by the algorithm tend to have a larger distance to the true scenario.

6 Application to biological data

In this chapter we apply the algorithm to real-life biological data. We will run the algorithm on two distinct datasets. One describes the evolutionary relationships between hosts from the genus Cimex and their Wolbachia bacteria, which we will call the Cimex dataset. The dataset and access to it is described in [BRTR18]. The other describes evolutionary relationships between catelunid flatworms from the genus Paracatenula as hosts and their bacteria, which will be referred to as the Paracatenula dataset. The dataset is described in and available from [GVDL$^+$11]. All steps will be illustrated using the Cimex dataset, but of course the results are given for both. First, we summarize how separate trees were obtained from DNA sequences. Then we explain the process of converting the separate trees to the format our algorithm from Chapter 4 uses, as well as some necessary pruning of the data. Moreover, we analyze several issues that come up now that we did not see when working with simulated data. In the end, we compare the result given by our algorithm with results given by `eMPRess` [SYL$^+$21], a commonly used reconciliation software that we described earlier in Chapter 2.4.

6.1 Obtaining the separate trees

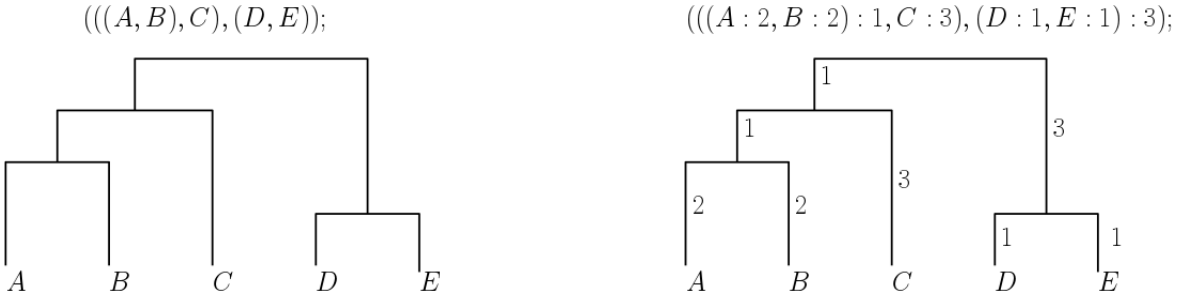
The separate unrooted phylogenetic trees of the two levels were obtained by using `IQ-TREE` [NSvHM15]. The trees were rooted with midpoint rooting, meaning the root is chosen to be at the midpoint of the path that connects the pair of leaves with the biggest distance between them. As our algorithm works with ultrametric trees (all leaves are at an equal distance from the root), we used the PAML software [Yan97] to do the conversion, assuming a strict molecular clock. In the end we have our separate trees, which are rooted and ultrametric. However, as the output of phylogenetic softwares (in particular, also `IQ-TREE`) is given in Newick format, now we need to convert them into a set of colaescence times and set of partitions, and we also need the matching of the leaves of the two trees.

6.2 Data pruning

Before starting to convert trees into our format, we might need to prune our trees so that they are not only ultrametric but also binary rooted trees. As trees are obtained from DNA sequence data, it is possible that even though two sequences come from different species, the sequences itself are identical and hence the distance between them will be 0. If such branches occur, we need to remove them as our reconciliation algorithm cannot handle branch lengths of 0. This goes for inner branches and also outer (connecting a leaf with an internal node) branches as well. The other case when we need to prune some branches is when there are multifurcations in the tree - in this case we also need to remove one or more branches. However, when removing such branches, it means we also need to remove the complete subtree below the newly removed branch.

6.3 Converting trees to the right format

First let us recall how a tree in the Newick format actually looks. The Newick format was invented by James Archie, William H. E. Day, Joseph Felsenstein, Wayne Maddison, Christopher Meacham, F. James Rohlf and David Swofford in 1986 in order to represent phylogenetic trees in computer-readable forms. As there has been no formal publication of the Newick Standard, we shortly explain it according to J. Felsenstein's own description on `https://phylipweb.github.io/phylip/newicktree.html`. In Newick format, trees are represented with the help of nested parentheses. Each tree begins with an opening paranthesis '(' and ends with a semicolon ';'. Tips (leaves) of the tree are represented by their names, which can be any string of printable characters with the exception of blanks, colons, semicolons, parentheses and square brackets. Internal nodes of the tree are represented by a pair of matched parentheses '()', between them are representations of the nodes that are immediately descend from that node, separated by commas. Note that these descendants can also be other internal nodes and so there would be further nestings of parentheses, to any level. Internal nodes can also have names (if needed). In this case the names follow the right parenthesis for the given internal node. Branch lengths can also be incorporated into the tree by putting a real number (with or without decimal point) after a node, preceded by a colon. This number represents the length of the branch immediately below that node. Figure 38 shows an example with and without branch lengths. As we are using branch lengths (i.e. the coalescence times) in our algorithm, we will work with the version that uses branch lengths.



(a) Only topology is considered

(b) Branch lengths are also considered

Figure 38: Example for trees in Newick format

Now we will describe the conversion to our format (see Definition 2.8). We use the ETE Toolkit [HCSB16], with which first we convert trees in Newick format into tree (Python) objects. These tree objects consist of nodes, and each node has two basic attributes to establish its position in the tree: a pointer to the parent node and a list of its children nodes. The (unique) tree node without a parent is called the root node, and those nodes without children nodes are the leaves of the tree. Moreover, each node has three more attributes with additional information. One of them is the name of the node, the second is the distance from the given node to its parent node (with a default value 1, if no branch lengths were given in the Newick format), and the third one is the support value of the node (reliability of the partition defined by the node), which is not relevant for us. Using these attributes and a postorder traversing of the tree we

can obtain the set of coalescence times and the partitions at each coalescence time.

We are going to add new features to each of the nodes ourselves. To define the partitions, we need to assign the blocks to all nodes, in particularly $\{1\}, \{2\}, ..., \{N\}$ to all the leaves. Then, as we go up from the leaves to the root, each internal node's block is defined as the union of its children's blocks. For the coalescence times, we add a new feature 'height' to all of the nodes. The height is the distance from the node to the closest leaf. Then, the set of all node heights is exactly the set of coalescence times we need.

If we take the tree depicted in Figure 38b, the set of coalescence times is given by the sequence of times $[0, 1, 2, 3, 4]$. In order to specify the partitions we give labels 1,2,3,4,5 to the nodes A,B,C,D,E, respectively. The corresponding partitions are then given by

$$\mathcal{P}_0 = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\},$$
$$\mathcal{P}_1 = \{\{1\}, \{2\}, \{3\}, \{4, 5\}\},$$
$$\mathcal{P}_2 = \{\{1, 2\}, \{3\}, \{4, 5\}\},$$
$$\mathcal{P}_3 = \{\{1, 2, 3\}, \{4, 5\}\},$$
$$\mathcal{P}_4 = \{\{1, 2, 3, 4, 5\}\}.$$

6.4 Adjusting coalesence times in the two trees

One of the main differences compared to working on simulated data is that biological data are not precise. The data is, by its very nature, noisy, and so the reconstructed trees are only approximations of the true underlying trees. Note that different softwares can produce different trees, and even the same software can give different outcomes depending on what model of molecular evolution is assumed. Moreover, the strict molecular clock assumption that we are using to convert trees to ultrametric trees is not entirely realistic, hence it can produce even more deviation from the true scenario.

6.4.1 Time rescaling

In order to identify coalescence events on the host and parasite trees, we need that the two trees have approximately the same timescale. As we can see in Figure 39, our data originally does not fulfill this condition, the lineages in the parasite tree merge significantly faster than in the host tree.

To rescale, we multiplied the branch lengths (i.e. the coalescence times) in the parasite tree with a certain factor while keeping the topology (i.e. the partitions) the same. To decide which scaling factor would be optimal, we create an array of numbers from 1 up until the last coalescence time in the host tree, with a step of 0.01 between them. Then, for each factor ϱ in the array, we calculated the 2-norm distance of the matrices $H = (h_{ij})_{i,j=1,2,...,N}$ and

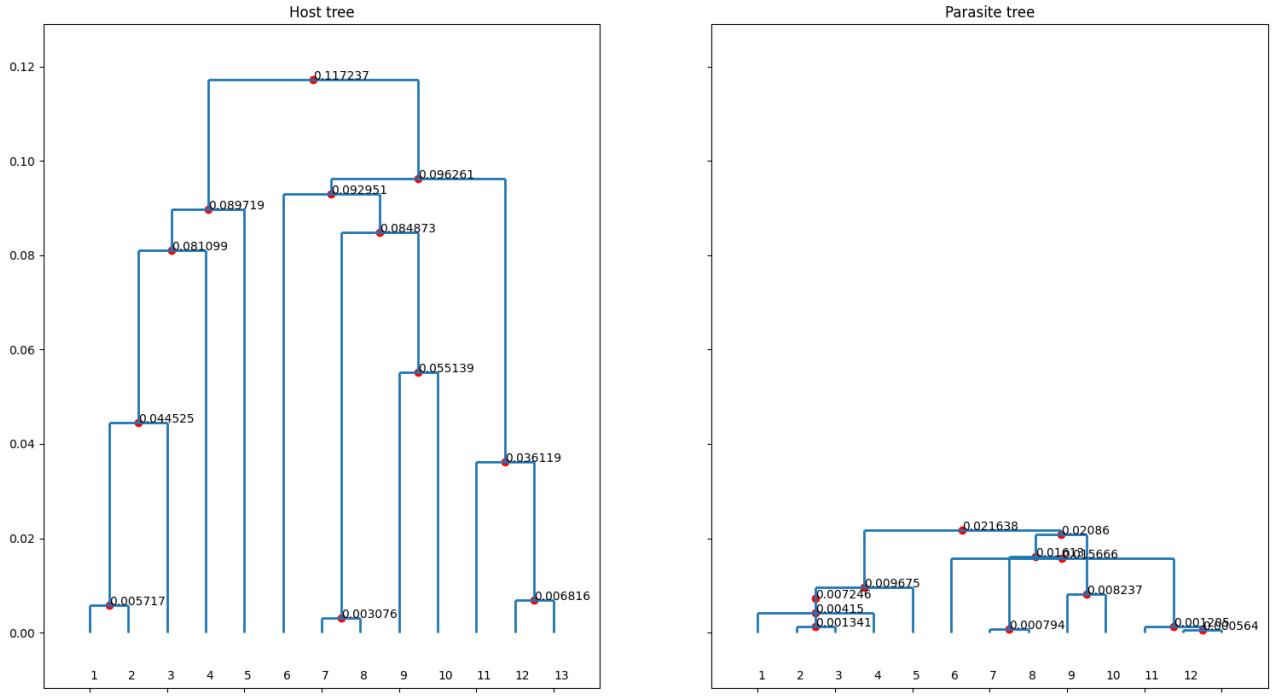


Figure 39: Separate trees without time rescaling in the Cimex dataset

$P = (\varrho \cdot p_{ij})_{i,j=1,2,\ldots,N}$. Here, h_{ij} is the distance from leave i to leave j in the host tree and p_{ij} being the same in the parasite tree without rescaling. We chose the ϱ which minimizes this distance between the matrices. In the Cimex dataset we have $N = 13$, and the separate trees before rescaling are depicted in Figure 39. By plotting the distance for all investigated scaling factors ϱ (see Figure 40), it turns out that the optimal value coincides with the rescaling that puts the last coalescence event in the two trees to the same time (indicated by a red vertical line).

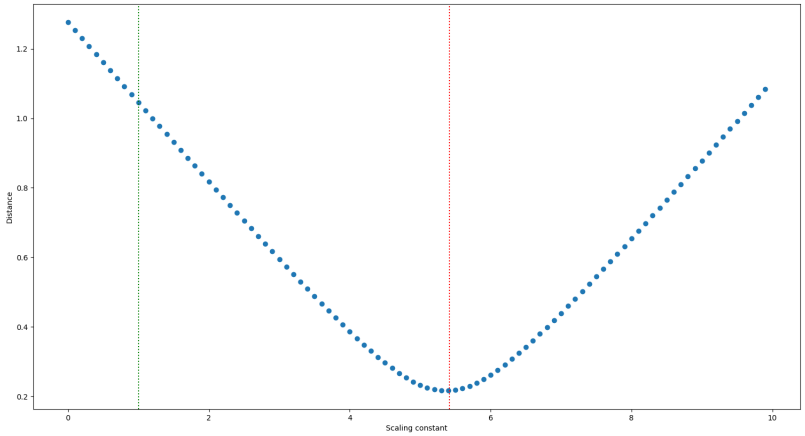


Figure 40: Distance of distance matrices corresponding to the separate trees with different rescalings in the Cimex dataset

The rescaled trees are shown in Figure 41. By visual inspection we can tell that there are similarities between the topologies of the separate trees, but they are certainly not identical.

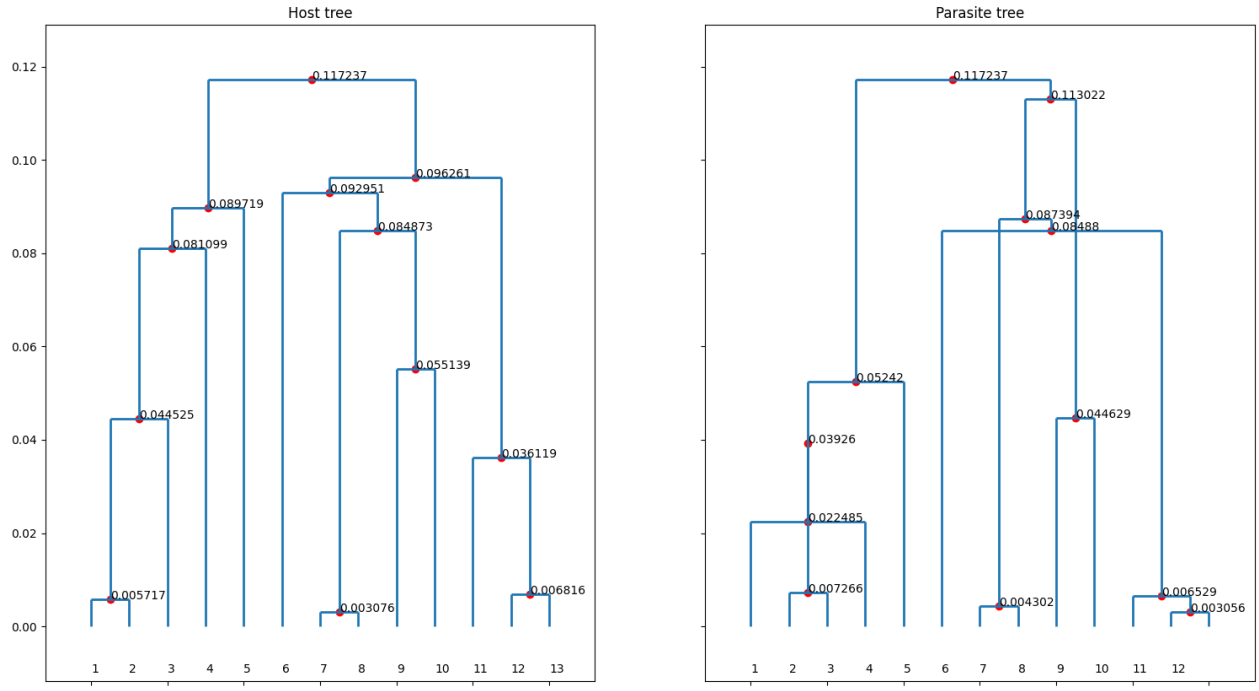


Figure 41: Separate trees with optimal time rescaling in the Cimex dataset

6.4.2 Matching coalescence times together

Now that the two trees are on the same timescale, we need to decide which coalescence events will we consider as the same event. As our model does not allow multiple coalescence events at the same time, we do not never consider two coalescence times in the host tree as the same, and the same goes for the parasite tree.

First, for each host coalescence time, we consider all those parasite coalescence times that are within a previously set distance δ. A reasonable choice is to consider a fraction of the smallest distance between the host coalescence events, or we can fix a small number as δ. Here we use $\delta = 0.008$. So for each host coalescence time T_H^i we keep track of the parasite coalescence times $T_P^{i,1}, T_P^{i,2}, ..., T_P^{i,j_i}$ that are within distance δ. We also take into account that it does not make sense to match two coalescence events if the blocks that are merging in the two trees are completely different. Suppose that the blocks merging in the host tree at time T_H^i are b_1^i and b_2^i, and the parasite blocks merging at time $T_P^{i,j}$ are $d_1^{i,j}$ and $d_2^{i,j}$ for each $i = 1, 2, ..., 12$ and $j = 1, ..., j_i$. For each $j = 1, ..., j_i$ we consider the number of common leaf descendants in the merging events. We keep those for which this number is maximal, i.e. for each i we determine

$$\max_{j \in \{1,...,j_i\}} \left| (b_1^i \cup b_2^i) \cap (d_1^{i,j} \cup d_2^{i,j}) \right|.$$

If there is a unique index j at which the maximum is obtained, we assume that times T_H^i and $T_P^{i,j}$ are the same. If there are several indices $j^1, j^2, ..., j^l$ where the maximum is obtained, then we choose the index j^k among them for which T_P^{i,j^k} is closest to T_H^i in order to avoid further complications.

In this way, each host coalescence event has at most one parasite coalescence event assigned to it. But it is still possible that a parasite coalescence event is assigned to multiple host coalescence events. As we can only allow one to be assigned, we use the same method as before: if a parasite coalescence event is assigned to several host coalescence events, we again look at the merging blocks and see how many leaf descendants the merging blocks have in common, and we choose the one with the largest number. If there are still several possibilities, we choose the one closest to the parasite coalescence event.

Now we have at most one parasite coalescence event assigned to each host coalescence event and at most one host coalescence event assigned to each parasite coalescence event. Next we need assign a single coalescence time to these pairs so that the algorithm work properly. We have again multiple options here, for each coalescence event we could fix the common coalescence time either as it appears in the host tree, as it appears in the parasite tree or somewhere between the two values. As a convention we will always use the timepoint exactly halfway between the host and the parasite coalescence times. This comes of course with possible complications - namely, it is possible that by such an assignment the order of the coalescence times changes in the separate trees, which leads to a change in the topology of the trees. We need to account for that by modifying the partitions of the separate trees accordingly. Figure 42 shows the separate trees after pairing and matching up some coalescence events in the separate trees with red dashed horizontal lines at the timepoints where two different coalescence events were matched.

6.4.3 Unmatching coalescence times

In the previous section we introduced a way to match up coalescence times in the separate trees in order for the algorithm to work well. Of course, this matching is not completely reliable. It is possible (and given the approximate nature of the reconstructed and rescaled separate trees, not unlikely) that we identify two coalescence events that are actually two different events. This will usually become apparent while we are running the algorithm and at a certain timepoint the algorithm cannot resolve a conflict - neither with transfers to empty lineages nor with correcting previous mistakes. In this case we consider unmatching the previously matched coalescence times and trying again.

More precisely, suppose we are at time T_i where T_i is a common coalescence event in both trees. Suppose that originally the coalescence time in the host tree was T_H^i and in the parasite tree T_P^i. We need to remove T_i from the set of all merging times and add back T_H^i and T_P^i. Here, it is possible that by reinserting the old coalescence times, their index in the ordering of the coalescence times (separately for the two trees) change, which means we need to account for

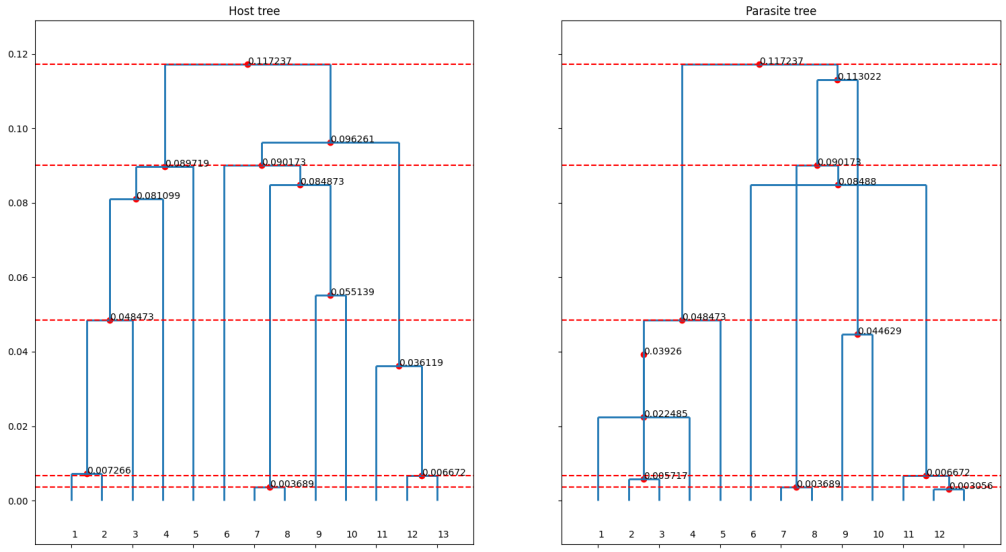


Figure 42: Separate trees with matched up coalescence times in the Cimex dataset. At dashed red lines, the corresponding host and parasite coalescence times are matched.

the change in the list of partitions as well. After that, let $t = \min(T_H^i, T_P^i)$. As we obtained $T_i = (T_H^i + T_P^i)/2$, time t must be smaller than T_i and so we delete each pairing that we have assigned that happened after time t, and start again from time t.

6.5 Results of the algorithm

In this chapter we show the results that our algorithm gives when applied to the Cimex dataset. One reconstructed joint tree is depicted in Figure 43.

As previously mentioned, the leaves 1-13 correspond to the leaves of the host tree according to the legend on the right side of the figure, and the additional blue, dashed vertical line represents the ghost line.

Before running the algorithm, we needed to estimate the transfer rate in order to be able to run the algorithm. We used the method described in Chapter 3 and the estimated transfer rate came out as 4.5. Here we would like to note that in this case, the estimator gives an approximation of the ratio between the transfer rate and the coalescence rate, as in our models and simulations the coalescence rate was set to 1 but here it is not fixed to that value.

We can observe that the algorithm decided for cospeciation in 5 cases, and it added 11 transfers to the scenario, with one-one transfer going to and from the ghost line.

Here we would like to emphasize again that if we run the algorithm several times with the same separate trees, it is possible that we get different outputs. This is due to the random choices

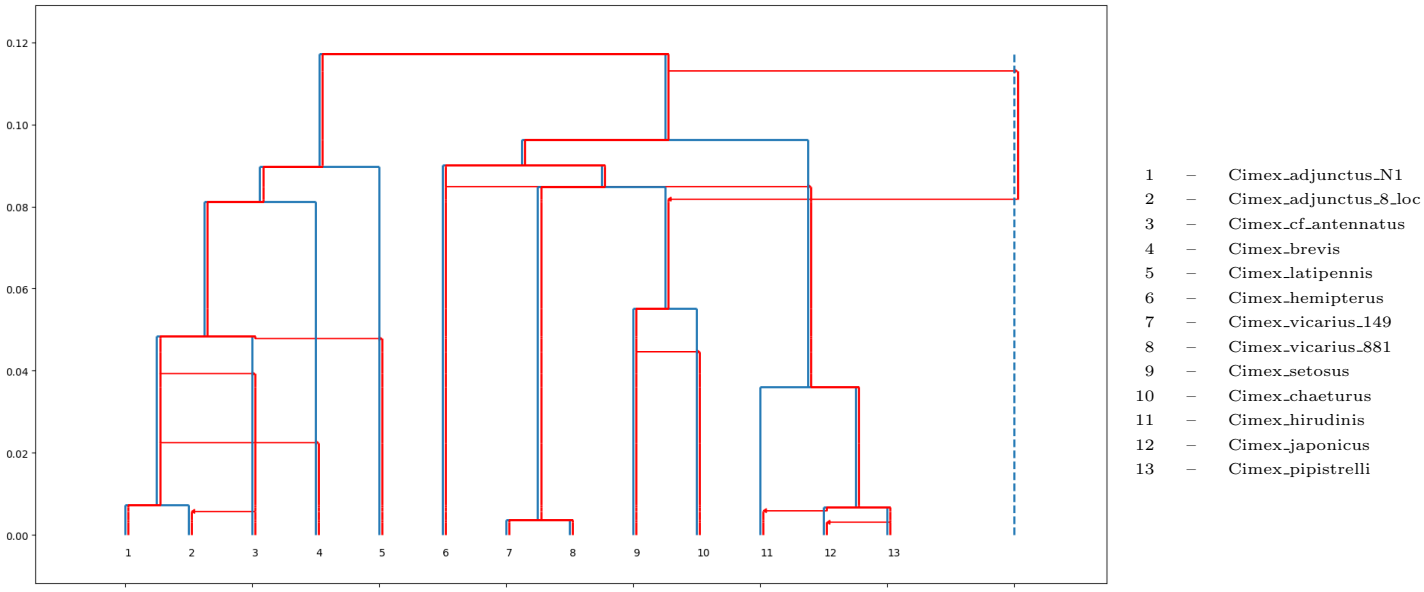


Figure 43: An output of the tree reconciliation algorithm on the Cimex dataset

made by the algorithm. In the case of the Cimex dataset, the depicted scenario indicates that when host branches $\{7, 8\}$ and $\{9, 10\}$ merge, there is no parasite merging event, even though both of them have a parasite lineage associated to them previously. As no possible global mistakes affect these host branches, the only possibility is to transfer one of the parasite lineages away to either an empty host branch or to the ghost line. As the current empty host branches are not compatible, the algorithm decides to transfer one of them to the ghost line. In the scenario depicted in Figure 43 the transferred parasite block was $\{9, 10\}$. When the algorithm decides to transfer parasite block $\{7, 8\}$ away, we get a different output which is depicted in Figure 44.

We can observe that in the second output, an additional transfer to the ghost line is needed in order to place the correct parasite blocks to the merging host branches $\{6\}$ and $\{7, 8, 9, 10\}$. The number of cospeciations remains the same as in the first output.

As the only random choice that the algorithm needs to make that can lead to a global mistake (the other random choices such as the direction of the transfer between branches $\{12\}$ and $\{13\}$, or the time of the transfer between branches $\{1, 2\}$ and $\{5\}$, etc, only lead to local mistakes) is when deciding whether parasite block $\{7, 8\}$ or $\{9, 10\}$ should be transferred to the ghost line, the global structure of any reconstructed scenario based on the same input trees will be similar to either the output depicted in Figure 43 or to the one depicted in Figure 44 . When we later compare the results from our algorithm with the results of `eMPRess`, we will (following the principle of Occam's razor) use the simpler scenario (i.e. the one that is depicted in Figure 43) that can explain the same separate trees.

Now we demonstrate the results when the algorithm is applied to the Paracatenula dataset. We applied the same methods to get the separate trees, to get the optimal scaling and adjust the coalescence times in the two trees. The separate trees before rescaling and adjustment, after

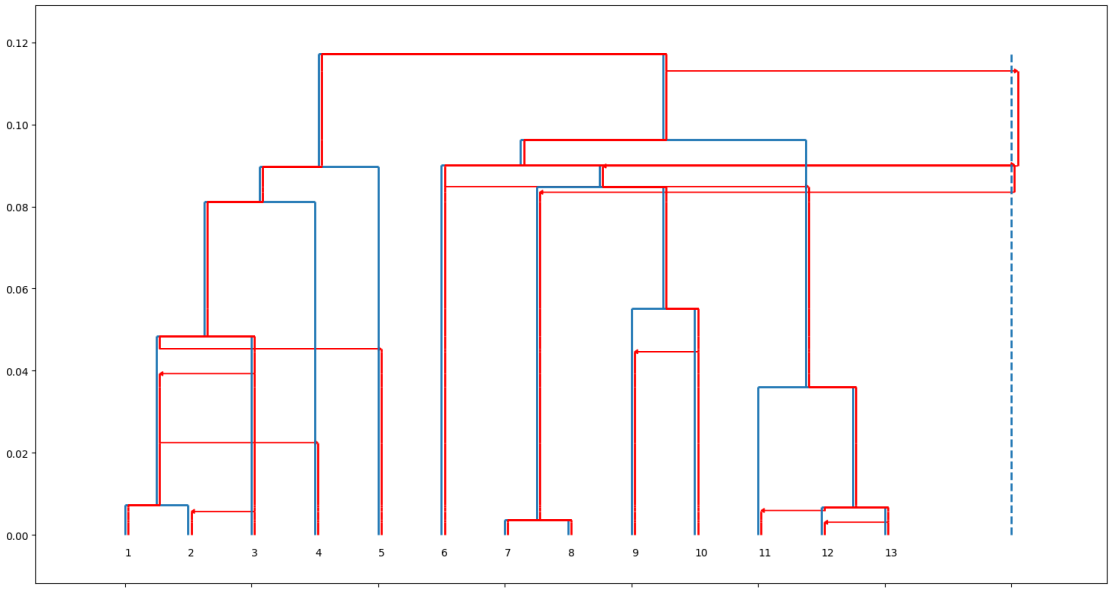


Figure 44: A different output of the tree reconciliation algorithm on the Cimex dataset

rescaling and adjustment are depicted in Figures 45, 46 respectively.

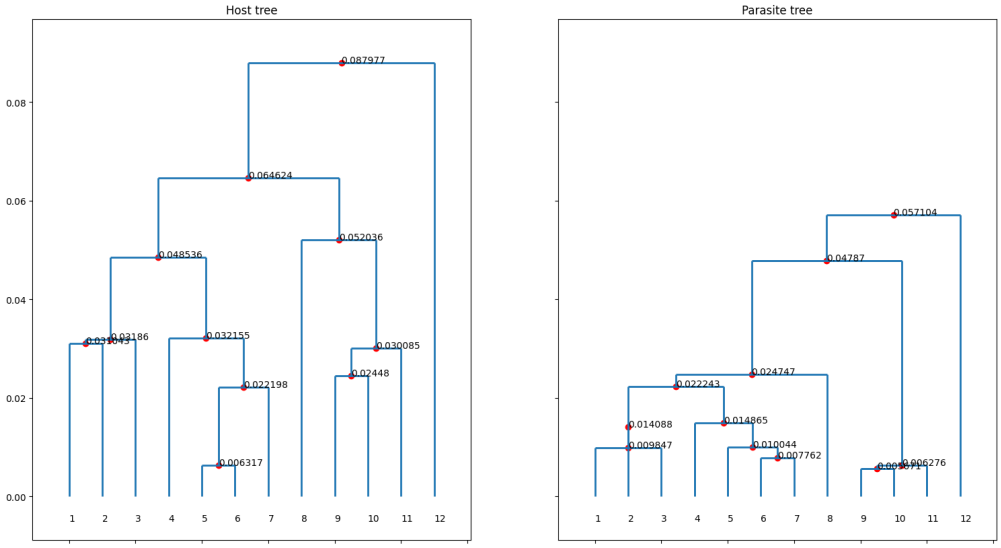


Figure 45: Separate trees before rescaling and adjusting coalescence times in the Paracatenula dataset

We can observe that the topologies of the separate trees do not seem very similar. The estimated value of τ based on them came out as 8.04, which suggests that reconstruction of the joint scenario will be difficult.

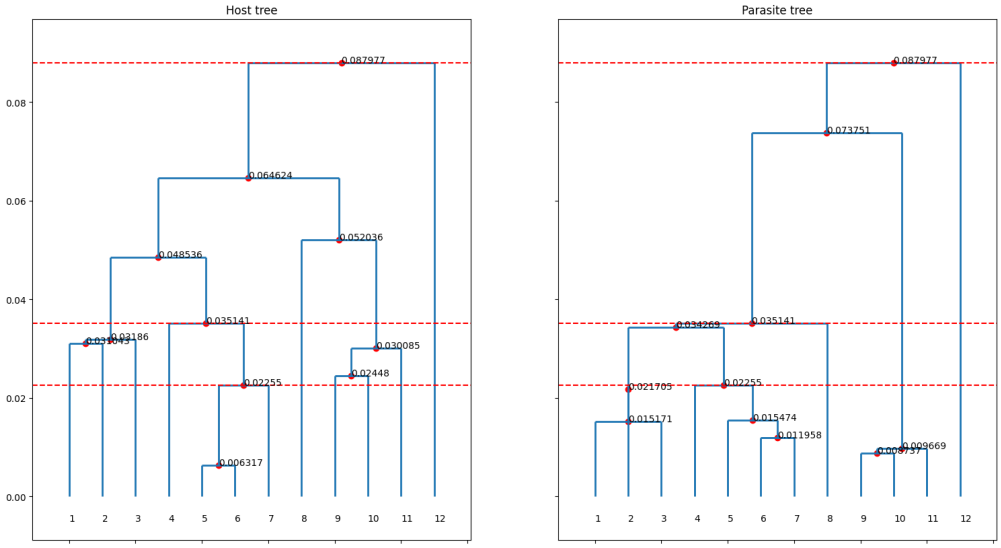


Figure 46: Separate trees after rescaling and adjusting coalescence times in the Paracatenula dataset

Again, when applying the algorithm to these separate trees, we can get different outputs. The first output is depicted in Figure 47.

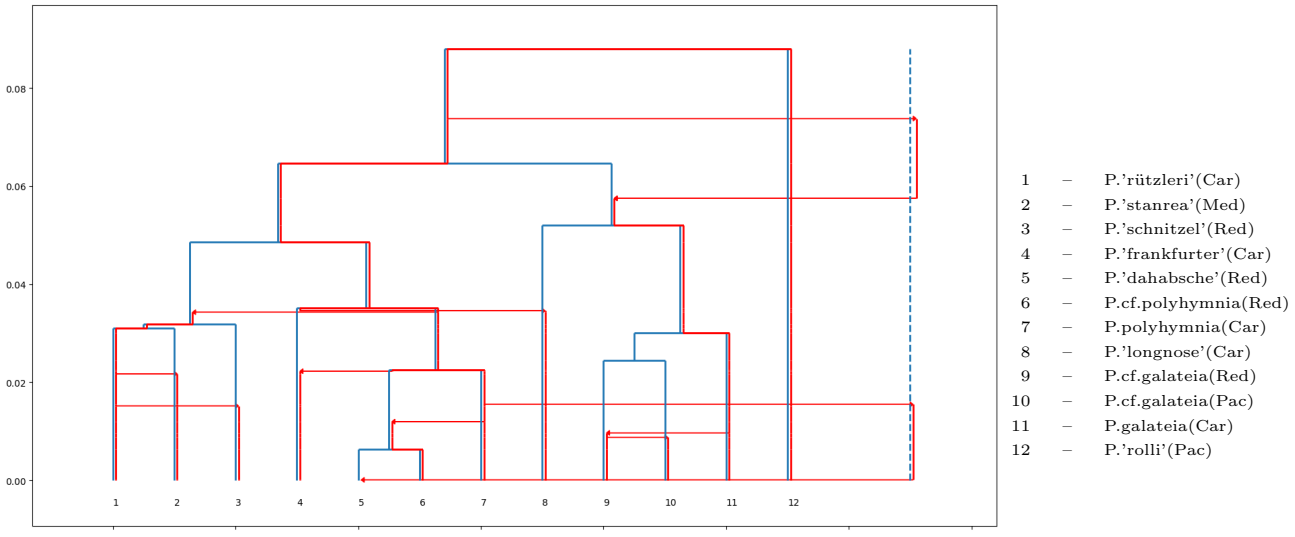
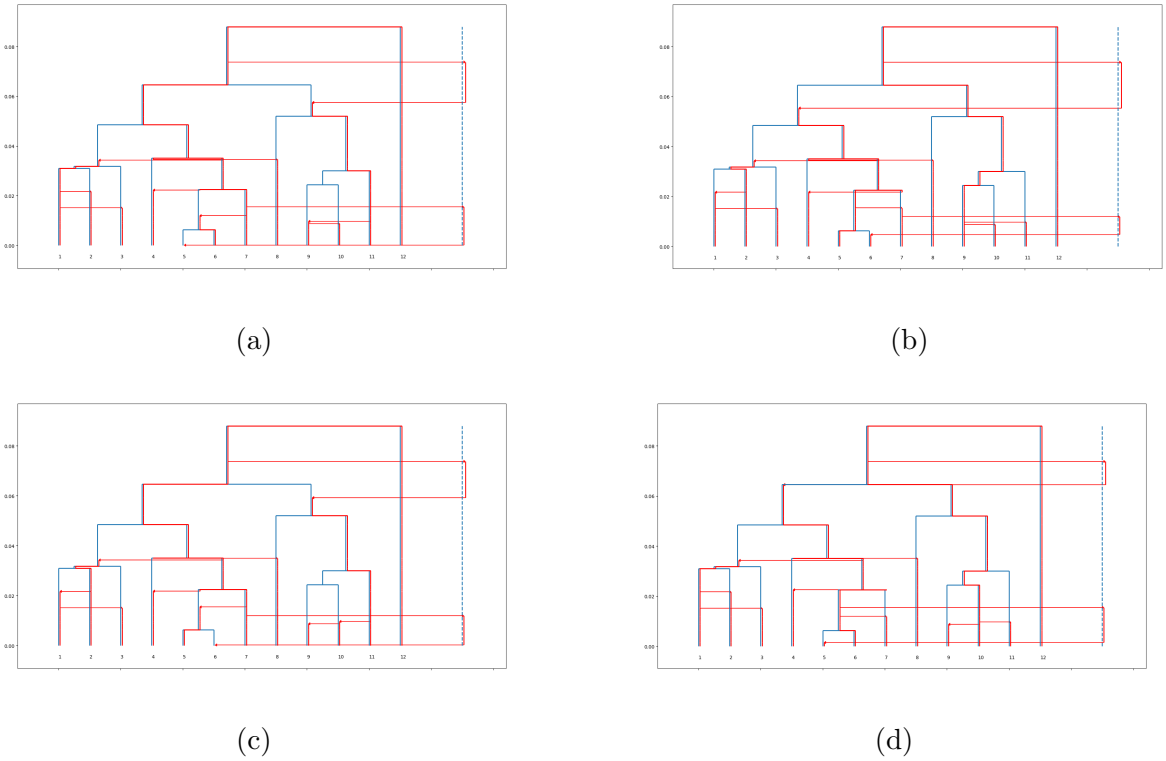


Figure 47: First output of the tree reconciliation algorithm on the Paracatenula dataset

It is possible to read off of this scenario at which times the algorithm needs to make a random choice that leads to a possible mistake (again, we do not count those random choices that result only in local mistakes). In this case, there are two such points, namely, at the merging of host branches $\{5\}$ and $\{6\}$, and at the merging of host branches $\{1, 2, 3, 4, 5, 6, 7\}$ and $\{8, 9, 10, 11\}$. That would result in four topologically different outputs (including the one already depicted in Figure 47). All four outputs are demonstrated in Figure 48:



(a) (b)

(c) (d)

Figure 48: The four topologically different outputs for the Paracatenula dataset

In all four scenarios we can see that the algorithm has recovered 3 cospeciation events and 11 transfer events, with two transfers to and two from the ghost line.

The four possible outputs are similar in the sense that each of them has the same number of cospeciations and transfers, it is not obvious which scenario is the 'simplest' (as in the case of the Cimex dataset). When comparing the results with `eMPRess`, for the sake of simplicity we use the representation corresponding to Figure 48a. Of course, all four outputs could be compared to `eMPRess` analogously.

6.6 Comparison with eMPRess

In this chapter we compare the results obtaine dby our algorithm in Chapter 6.5 with the output of the tree reconciliation software eMPRess [SYL+21], both with simulated and biological datasets. Before comparing the results, let us give a briefly recall eMPRess, which was already mentioned in Chapter 2.4.

The software eMPRess belongs to the class of maximum parsimony reconciliation (MPR) methods, where the goal is to find an evolutionary scenario with the lowest possible overall cost. It works under the assumptions of the DTL model, where 4 kinds of events are allowed: cospeciation, duplication, transfer and loss. The overall cost is calculated as the sum of the costs of the events happening in the scenario, which are pre-set by the user, except for cospeciation, which comes at zero cost. The aim is to find a scenario which results in the separate trees given by the input, with the lowest possible cost. It is important to note here that only the ratio of different event costs matters, not the actual value of them. For example, the triples of costs for duplication, transfer and loss $(1, 1, 2)$ and $(10, 10, 20)$ will give the same result.

As finding optimal cost scenarios with replacing transfers is NP-hard, it is important to note that it is not guaranteed that we indeed get the most parsimonius (i.e. with globally lowest cost) reconciliation, nor that the most parsimonius reconciliation is time-consistent. It is also often the case that there are more than one MPRs (maximum parsimony reconciliations), in this case eMPRess outputs one median reconciliation. Whenever working with eMPRess, we will use this median reconciliation representing the MPRs.

To compare the performance of our algorithm and the eMPRess software, first we used simulated data, and set the cost so that eMPRess would only use transfers instead of duplications and losses. The costs we set for duplication, loss and transfer were 100, 100 and 1, respectively. (The much larger costs for duplication and loss makes sure that these are essentially not used in the reconstruction.) We generated scenarios for different values of the parameter τ under Backward model I with $\tau = 0.5$ and $\tau = 2$. We did not compare scenarios generated under Backward model II since it is known that eMPRess does not consider parasite lineages leaving the host tree which would surely result in eMPRess not arriving to the true scenario. In all of the simulations we used $N = 10$ leaves for the sake of better visualization of the results. In order to compare the two outputs, we use two kinds of distance measures: the symmetric set distance d_S of two reconciliations introduced by [NRBS13], i.e. the number of events that appear in exactly one of the two considered scenarios, and the distance measure defined in Chapter 5 in Definition 5.6, which we denote by d_T and refer to as the time-sensitive distance.

First, we needed to create the input files in order to be able to run eMPRess: the two separate trees in Newick format (without branch lengths), and a file which gives the mapping between the leaves of the two trees. Similarly to the case when converting trees from Newick format to the set of coalescence times and partitions, we use the ETE3 package to do the conversion backwards.

Let us first the case of $\tau = 0.5$. We start by visually comparing the outputs of our algorithm

and eMPRess with each other, and also with the true (simulated) scenario, which is depicted in Figure 49. We can see that there are 3 simulated transfer events, but one of them (drawn in green) is not identifiable since its recipient is an empty branch. From the 9 host merging events there is cospeciation in 7 cases.

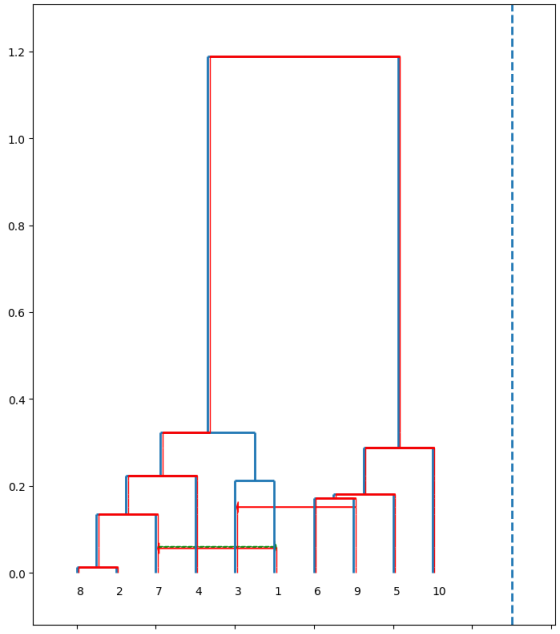
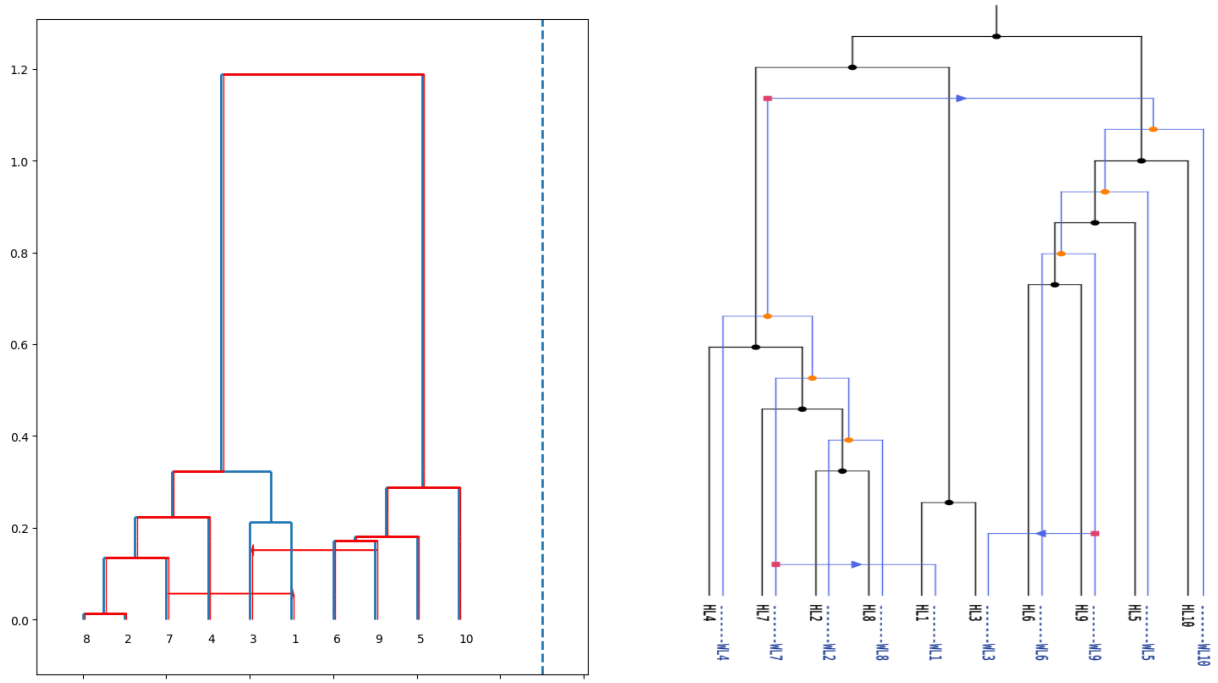


Figure 49: A simulated scenario with $N = 10$ and $\tau = 0.5$

Now, let us look at the reconciled scenarios, which are depicted in Figure 50. First, note that the order of the branches are not the same in the two scenarios, however, it is easy to identify the leaves in the two scenarios by their numbers. In the output of eMPRess, transfer events are denoted also by arrows, but the arrowhead is in the middle of the edge rather than being at one end. Cospeciation events are all the speciation events that have both a black and an orange circle associated with it. We can see that eMPRess has recovered exactly the same transfer events as our algorithm, and both of them were in the true scenario as well. However, eMPRess has added an extra transfer event between host branches $\{2, 4, 7, 8\}$ and $\{5, 6, 9, 10\}$. This is due to the fact that eMPRess considers transfers to be additive, not replacing, i.e. when a parasite lineage is being transferred to a host branch, the current parasite on the recipient species is not being replaced, but both of the infections continue along the given host branch.

Next, we are going to calculate the symmetric set distances between the true scenario and the recovered scenarios. In this setting, two transfers are considered equivalent if both the donor and recipient lineages are the same in the two scenarios (which means that the direction of the transfer also counts). From the output of our algorithm, only the single transfer event is missing that is not identifiable, everything else is the same, so the symmetric distance is 1. When comparing with the output of eMPRess, the first two transfers are equivalent in the two scenarios. Besides the unidentifiable transfer event, there is an additional transfer event recognized by eMPRess that was not part of the original scenario, which results in the last host



(a) Output of our algorithm

(b) Output of `eMPRess`

Figure 50: The two reconciled scenarios for $\tau = 0.5$

merging event not to be a cospeciation event. This means that we have 3 events that can be found in one scenario but not in the other.

It is not easy to quantify the time-sensitive distance between the two recovered scenarios due to them carrying different meanings. Our algorithm identifies horizontal transfer events based on the time differences of the coalescence times in the two trees, while `eMPRess` does not consider branch lengths (times) at all. The order in which the events are visualized does not represent the real order in which they have happened. Nevertheless, we can try to calculate the distance described in Chapter 5.2 with the following adjustment: whenever a cospeciation event is detected by `eMPRess`, we just give it a time identical to the true scenario. Whenever there is a transfer event recovered by `eMPRess`, we can give it a time that is uniformly distributed on the time interval between their previous and next merging event.

The coalescence times (omitting the present time 0) in the host tree are given by the sequence

$$0.013, 0.135, 0.172, 0.181, 0.213, 0.223, 0.289, 0.323, 1.189$$

and for the parasite tree we have the sequence of merging times given by

$$0.013, 0.056, 0.135, 0.151, 0.172, 0.181, 0.223, 0.289, 1.189$$

with the common merging times (that are indicating cospeciation events) written in blue. Both of the recovered horizontal transfers cause a merging in the parasite tree, and so their times are given by the merging times in the parasite tree that are not present in the host tree: 0.056

and 0.151. To give a timing to the transfer event in the scenario recovered by `eMPRess`, we consider the previous and next merging events of the branches participating. For the transfer event between host branches $\{1\}$ and $\{7\}$, their starting point is at time 0 as they are both external branches (i.e. starting from a leaf of the tree), and they end at the first merging events of those branches, at times 0.135 and 0.213, respectively. Thus the time of the transfer is given by a uniformly distributed random variable on the time interval $(0, 0.135)$, which we represent here by its expectation, i.e. the middle point of the interval, 0.0675. Similarly, the transfer between host branches $\{3\}$ and $\{9\}$ is given by a uniformly distributed random variable on the time interval $(0, 0.172)$, which is represented by its expectation 0.086. For the third transfer recovered by `eMPRess` that is between host branches $\{2, 4, 7, 8\}$ and $\{5, 6, 9, 10\}$, its time is given by a uniformly distributed random variable on the time interval $(0.289, 0.323)$, which is again represented by its expectation 0.306.

Now we can calculate the distance of the scenario recovered by `eMPRess` and the true simulated scenario. Let us here summarize all the differences between the trees:

- On time interval $[0, 0.056)$ the two scenarios are identical.
- At time 0.056 there is the transfer between branches $\{7\}$ and $\{1\}$ in the true scenario, while it happens at time 0.0675 in the recovered. On this small time interval between them, the difference in the two scenarios is given by the parasite blocks on host branches $\{7\}$ and $\{1\}$: in the simulated scenario it is $\{1, 7\}, \emptyset$, while in the recovered scenario it is $\{7\}, \{1\}$, which can be made equal by moving element 1 into the same block as 7.
- On interval $(0.0675, 0.086)$, the two scenarios are the same.
- The second transfer event happens at time 0.151 in the true scenario, while it already happens at 0.086 in the recovered scenario. Note that there is a host merging event at time 0.135, but since the host branches merging at this time are not involved in the second transfer event, it would not cause any further issues. The difference of the two scenarios on this time interval are given by the different parasite blocks on host branches $\{9\}$ and $\{3\}$: in the recovered scenario, it is $\{3, 9\}, \emptyset$, while in the true scenario it is still given by $\{9\}, \{3\}$, where we again only need to move element 3 together with 9 to arrive to the other scenario.
- On interval $(0.151, 0.306)$ the two scenarios are identical.
- At time 0.306 there is an additional transfer in the `eMPRess` scenario, which would cause all existing parasite lineages merging together at that time. On time interval $(0.306, 0.323)$ there are still 3 host branches, with parasite blocks corresponding to $\{1, 2, 4, 7, 8\}, \emptyset, \{3, 5, 6, 9, 10\}$ in the true scenario and $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}, \emptyset, \emptyset$ in the `eMPRess` scenario. We would need to move 5 elements to get from one partition to the other. Similarly, on interval $(0.323, 1.189)$ the parasite blocks along the host branches are given by $\{1, 2, 4, 7, 8\}, \{3, 5, 6, 9, 10\}$ in the true scenario and $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}, \emptyset$

in the recovered scenario, where again we would need to move 5 elements to resolve the differences.

By calculating the distance between the two scenarios, we get

$$d = \frac{1}{1.189}(1 \cdot (0.0675 - 0.056) + 1 \cdot (0.151 - 0.086) + 5 \cdot (1.189 - 0.306)) = 3.7775$$

To sum up, Table 2 gives us the most important data about the three scenarios.

$\tau = 0.5$	True scenario	Our algorithm	`eMPRess`
Cospeciations	7	7	7
Transfers	3	2	3
Symmetric set distance to true scenario	0	1	3
Time-sensitive distance to true scenario	0	0	3.7775

Table 2: A summary of the different scenarios for $\tau = 0.5$

We can observe that in case of simulated data with low transfer rate, the time-sensitive algorithm seems to perform better, both with respect to the time-sensitive and symmetric set distances. It is not entirely surprising since the time-sensitive algorithm is based on the same model where simulations come from, and it also uses more information than `eMPRess`.

Let us move to now the case of $\tau = 2$. Here we will also give the visualization of all three scenarios, however we omit the details of how we calculate the distances as it is done using the same principle as for the previous case. The simulated scenario can be seen in Figure 51.

We can see that in the simulated scenario there are 8 transfer events in total, with 4 cospeciation events. Now let us compare it with the two recovered scenarios, depicted in Figure 52. As before, the costs set in `eMPRess` for duplication, loss and transfer was set as $(100, 100, 1)$, respectively.

Again let us visually inspect the two recovered scenarios. The scenario recovered by our algorithm contains all 4 cospeciation events, as well as 6 transfer events, and 5 of them can be found in the simulated scenario at the exact same time, while the 6th one is also present in the simulated scenario but at a different time. The scenario recovered by `eMPRess` contains only 3 cospeciation events, with two of them appearing in both the simulated and the first recovered scenario (merging of host branches $\{9\}$ and $\{10\}$, as well as $\{1\}$ and $\{4\}$). The third cospeciation event is assigned at the merging of host branches $\{2\}$ and $\{3\}$, which is actually not a cospeciation event in the simulated scenario. `eMPRess` also has recovered 6 transfer events, however only 4 of them appear in the simulated scenario: one between host branches $\{1\}$ and $\{7\}$, another between host branches $\{5\}$ and $\{1, 4\}$ (though it is $\{1, 4, 7\}$ in the simulated scenario) and the third is between host branches $\{9, 10\}$ and $\{1, 4\}$ (again, $\{1, 4, 7\}$ in the simulations). All the other transfer events involve host branches $\{2\}$ or $\{3\}$. Namely, there is one between host branches $\{2\}$ and $\{8\}$, that is the 4th transfer that is appearing in the true scenario, but

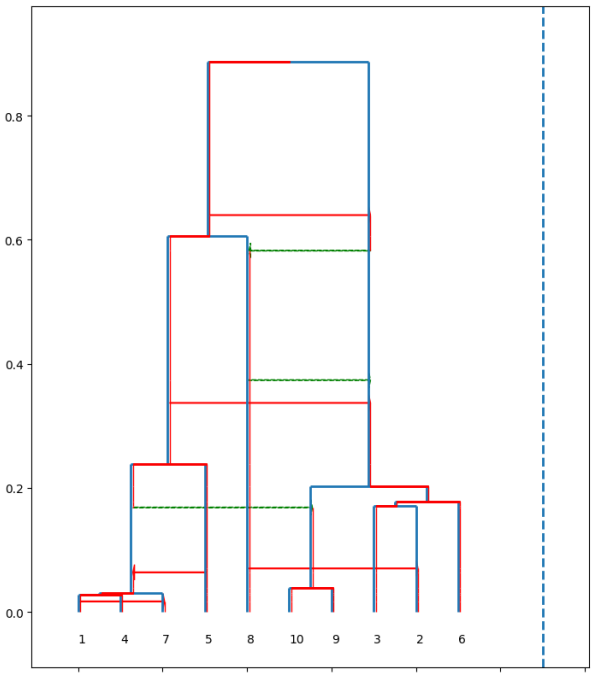
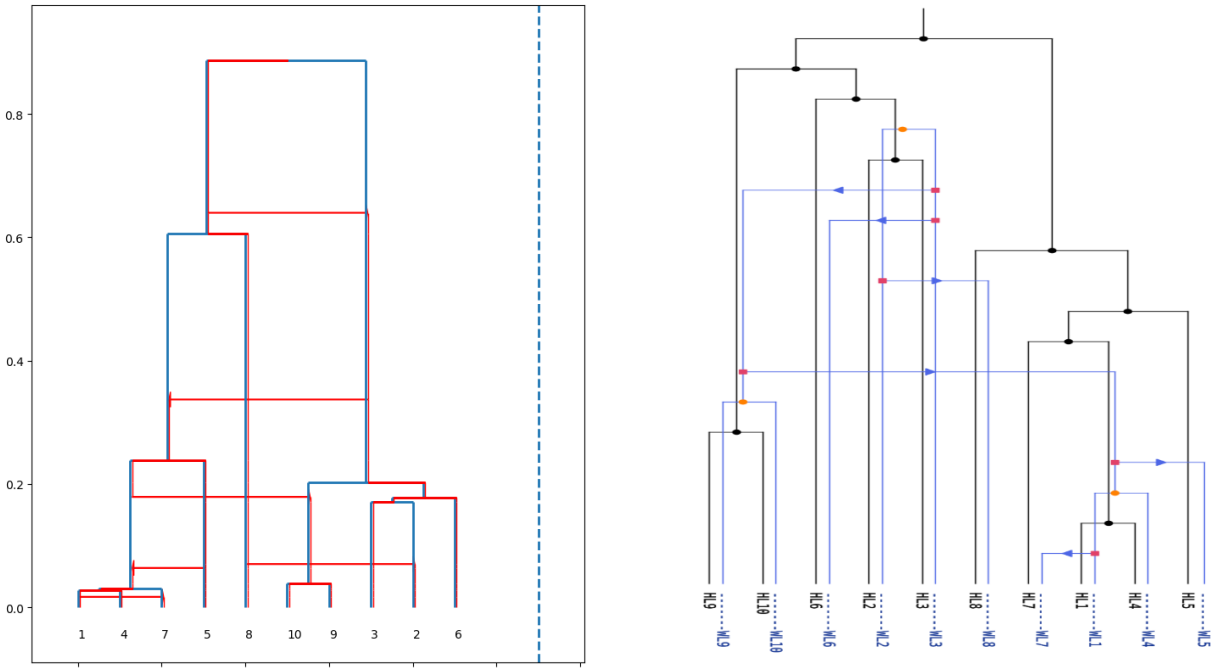


Figure 51: A simulated scenario with $N = 10$ leaves with $\tau = 2$



(a) Output of our algorithm

(b) Output of `eMPRess`

Figure 52: The two reconciled scenarios for $\tau = 2$

its direction is different. Another transfer is between host branches $\{3\}$ and $\{6\}$ and the last is between host branches $\{3\}$ and $\{9, 10\}$.

Note that now we cannot calculate the time-sensitive distance using this reconciliation given

by eMPRess as it assigns a transfer event between branches $\{9, 10\}$ and $\{1, 4\}$, even though these two branches are not coexistent, meaning that there was no time in the past when both of the branches existed simultaneously. This makes the scenario recovered by eMPRess time-inconsistent. The symmetric set distance can still be calculated. As previously mentioned, there are 3 cospeciation events that appear only in one scenario, and from the transfer events there is only one (between branches $\{1\}$ and $\{7\}$) that appears in both trees. All the other transfers are either assigned between different branches in the two scenarios or the wrong direction is given. It means that we have 13 transfer events contributing to the symmetric set distance. Putting everything together, the total symmetric set difference is 16. Table 3 summarizes the most important facts about the 3 scenarios.

$\tau = 2$	True scenario	Our algorithm	eMPRess
Cospeciations	4	4	3
Transfers	8	6	6
Symmetric set distance to true scenario	0	4	16
Time-sensitive distance to true scenario	0	0.4925	Not defined

Table 3: A summary of the different scenarios for $\tau = 2$

We can still observe that the time-sensitive algorithm does better with respect to both of the here considered distance measures, however, as seen already in Chapter 5, the reconstructed scenarios are now further from the true scenario than for a smaller transfer rate.

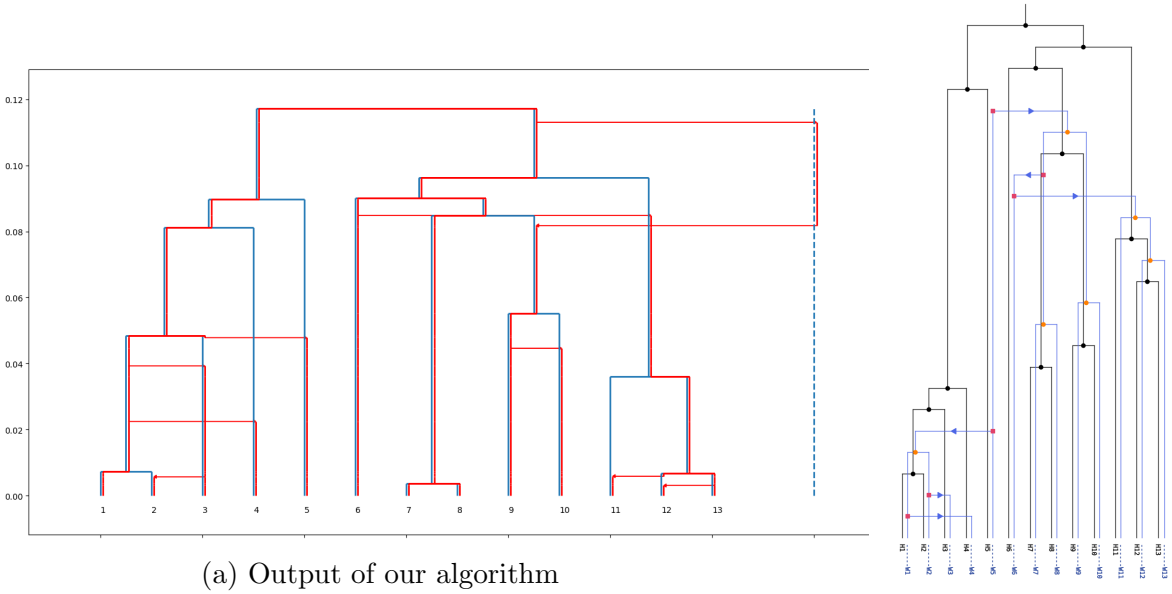
Now let us compare the two reconciled scenarios for the two biological datasets we have introduced before. As now the true scenario is not known, it is not possible to calculate the distance of the two scenarios from it. Instead we calculate the distance of the two different reconciled scenarios.

First let us start with the Cimex dataset. The two recovered scenarios are depicted in Figure 53.

As described in Chapter 6.5, our algorithm recovered 5 cospeciation events and 11 transfer events, from which 2 involve the ghost line. On the other hand, eMPRess has recovered 6 cospeciation events and 6 transfer events. All 6 transfer events are feasible in the sense that they always happened between two coexisting branches of the host tree. However, our scenario has recovered more transfer events because it considers the time differences between certain events, for example the transfer between host lineages $\{9\}$ and $\{10\}$ is only there to account for the different merging times in the two trees. However, there are only two common cospeciation events and only one common transfer event in the two scenarios, which gives us a symmetric set distance of 22.

Now let us demonstrate the results for the Paracatenula dataset. The two recovered scenarios are depicted in Figure 54.

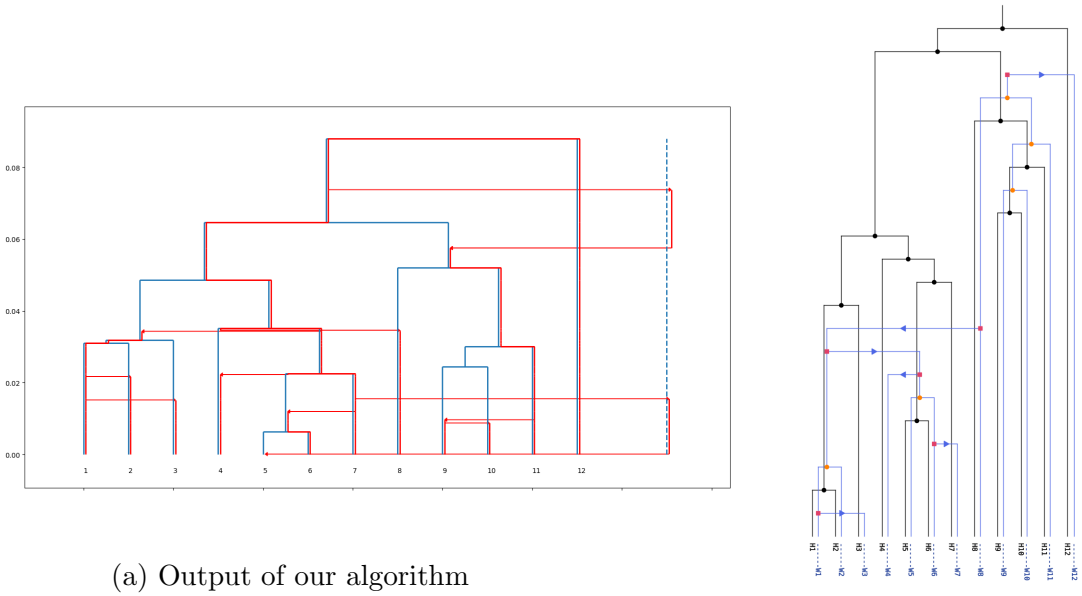
Our algorithm identified 3 cospeciation events and 12 transfer events, from which 4 involves



(a) Output of our algorithm

(b) Output of eMPRess

Figure 53: The two reconciled scenarios for the Cimex dataset



(a) Output of our algorithm

(b) Output of eMPRess

Figure 54: The two reconciled scenarios for the Paracatenula dataset

the ghost line. On the other hand, eMPRess identified 5 cospeciation events together with 6 transfer events. However, note that in this scenario the transfers recovered by eMPRess are not all feasible: there is a transfer between host branches $\{1, 2\}$ and $\{5, 6\}$, but there is no timepoint at which both of the branches existed together. Checking for events happening in both of the scenarios, we find that there are no common cospeciation events in the two scenarios,

however, there are two common transfer events, namely the transfers between branches 1 and 3, and between $\{5, 6\}$ and 4. This gives us a symmetric set difference distance of 22. The time-sensitive distance of the two recovered scenarios is not defined due to the time inconsistency of the recovered scenario by `eMPRess`.

To sum up, we can see significant differences in the two suggested reconstructions for both of the biological datasets. From Chapter 6.5 and the estimates of the (relatively high) transfer rates we already know that obtaining a feasible reconciliation is a challenging task. However, our algorithm uses additional information on the timing of the merging events that `eMPRess` does not use. This means that our algorithm always provides a (strongly) time-consistent scenario, while `eMPRess` might find a reconciliation optimal that is not time-consistent. Moreover, our algorithm outputs the exact timepoints when horizontal transfers happened, which can be used (or tested for) in further (co)phylogenetic studies.

7 Conclusions and open questions

In this chapter we conclude this thesis by summarizing and discussing the most important findings. We also mention some related open questions.

We defined a general stochastic model for the joint evolution of a fixed number N of hosts and parasites in Definition 2.11. We also defined two backward models (Backward model I in Definition 2.13 and Backward model II in Definition 2.15) inspired to this forward dynamics. In all of our models we assume that horizontal transfer of parasites happen at a constant rate τ for each pair of branches of the host tree independently. Our main goal was to reconstruct the joint evolutionary history of hosts and parasites from the information coming from the separate phylogenetic trees of the hosts and the parasites.

In Chapter 3 we introduced a new estimator for τ under the assumptions of Backward model I, which only uses information that can be obtained from the separate phylogenetic trees of the hosts and parasites. We have proven some of its mathematical properties, such as existence (Theorem 3.6), consistency (Theorem 3.7) and under an additional assumption, asymptotic normality (Theorem 3.10), all meant as $N \to \infty$. We would like to discuss two questions regarding this estimator that are still left open.
The first and most important open question is whether Assumption (3.29) always hold. As mentioned in Chapter 3 and shown in Figure 16, simulations support a positive answer for this question, and so one would expect that this could be mathematically confirmed. The main difficulty here is the complex dependence of the constants c_i^N on the topology of the joint tree, as well as the implicit dependence of the constants c_i^N on the transfer rate τ.
The second question one could still address is how to define a good estimator for τ under Backward model II. One requirement for that would be to have information on the rates p, c and r, or make them also dependent on τ. Another major challenge here is that we do not have information on the number of parasite lineages associated with the ghost line, as this is something we cannot obtain from the separate trees alone. However, this knowledge would be essential in order to define a Markov process similarly to the one in Proposition 3.1 in Chapter 3 and derive an estimator based on its jump times.

In Chapter 4 we introduced and described a new reconciliation algorithm. It takes dated ultrametric trees (that represent the separate host and parasite phylogenies) as an input and it tries to identify the horizontal transfers between the branches of the host tree based on differences in the coalescence times and topologies in the host and parasite tree. When it is not possible to fully identify the transfer, the algorithm makes an informed random choice out of the feasible possibilities, where the different outcomes are weighted by the probability of their occurrence. These probabilities depend on the horizontal transfer rate τ, so that an a priori estimate of τ based on the separate trees (which was provided in Chapter 3) is relevant for the algorithm. Here we would like to discuss some possible modifications and extensions to this algorithm.
The first is regarding the complexity of the algorithm. As stated and discussed in Chapter 4.6

previously, it has a complexity of $O(N^4)$. Even though this is a polynomial runtime, for large trees it can take long to produce an output. It remains an open task to optimize the runtime. The following possible extensions are based on biological observations. The assumption that transfers happen at an identical rate τ between each pair of branches of the host tree makes the mathematical analysis much simpler. However, it has been shown that more transfers tend to happen between branches that correspond to closely related species than between those branches that correspond to distantly related species [EF19]. This is called preferential host switching. This could be incorporated in our models and the algorithm by assuming that transfers are attempted at the same rate τ for each pair of host branches, but not all of them will be successful. The probability of a transfer being successful should be given as a decreasing function of the distance between the branches participating in the transfer.
Given our focus on application to systems where one host is most commonly infected by a single parasite, our models and algorithm also allow at most one parasite on a host lineage. However, as it is not uncommon that different parasite species can infect the same host species, another extension is to allow more than one parasite lineages along a single host lineage. In this case, the use of additive transfers are required instead of replacing transfers, where the original parasite lineage will not be overtaken after a transfer, but both of them would continue to exist along the recipient host branch.

In Chapters 5 and 6 we applied the algorithm both on simulated and biological datasets, and investigated its performance. We also compared the algorithm with the well established reconciliation method `eMPRess`. For the evaluation of the performance and for the comparisons we have used two distance measures between joint trees. One of them is the symmetric set distance defined by Ngyuen et al. [NRBS13], which only considers topologies. Our time-sensitive distances defined in Definition 5.6 and Definition 5.9 are distance measures that explicitly take into account not only the topological information, but also the time differences in the two compared scenarios. They are true distance measures in a mathematical sense, however their biological interpretation is not clear yet. These measures only can be used to compare two scenarios where the host tree is the same. Theoretical results involving this distance measure, such as obtaining an expected distance between the true and a reconciled scenario seem very hard to prove, as one would need to consider all possible joint topologies. However, we can see from the simulated scenarios that the algorithm does generally well in finding the true scenario for small values of the transfer rate ($\tau < 1$), when there are more host coalescence events than horizontal transfers. For $\tau > 1$ the performance starts to decrease due to the more complex scenarios happening in the presence of many horizontal transfers. This outcome could be improved by changing the strategy when assigning directions to horizontal transfers (see Chapter 4.2.1): instead of looking at the next merging event of a certain branch in the parasite tree, we can also look at the 2nd next event. With this we would get more information and thus would be able to decide more accurately. But of course, it would come with a higher complexity.
Another question that needs to be addressed concerns the difference between the different outputs the algorithm may produce for the same input (same separate trees). While with simulated scenarios we could directly compare the different outputs with the true scenario by computing

their distances and select the one closest to the true scenario, it is not possible anymore when working with biological datasets. In Chapter 6 we have investigated two datasets. For the Cimex dataset there were two topologically different outputs, from which, by visual inspection, we chose the 'simpler' scenario, i.e. the scenario with the lower number of transfer events. For the Paracatenula dataset we found four topologically different outputs, but each of had the same number of transfers, so by the same visual inspection we could not choose one that was the 'simplest'. Here we just picked a certain output from the four possibilities. However, assessing visually how complicated the scenarios are is not always a good option, especially if the trees in question are large. Thus, we need to find a universal method to select a reasonable output to work with. One idea would be to select the scenario with the least number of transfers. A more sophisticated approach is to define a median output similarly as Ngyuen et al. [NRBS13] did for reconciliations under the DTL model, using the symmetric set distance measure d_S. Here, for each input we could run the algorithm n times (for example $n = 100$) with outputs given as matchings $M_1, M_2, ..., M_n$, then use the time-sensitive distance d_T of joint trees given in Definition 5.6 (or Definition 5.9 if the ghost line is allowed) to find the matching M_i for which the expression $\sum_{j=1}^{n} d_T(M_i, M_j)$ is minimal over all $i = 1, 2, ..., n$. This has two limitations, namely it is possible that this minimum is obtained at several outputs, not at just a single one. The other issue is computational: we described in Chapter 5 that computing the distance between two reconciliations with N leaves is of complexity $O(N^4)$, which means that finding the median output has a complexity of $O = (N^4 \cdot n^2)$ which results in a long runtime for larger trees. Hence, it still remains an open question to find a computationally faster method to select a good representative output.

When comparing the outputs from `eMPRess` and our algorithm, we can say that on both simulated scenarios our algorithm did better regarding both the symmetric set distance and the time-sensitive distance. For the second simulation dataset `eMPRess` even produced a time-inconsistent scenario. For the biological datasets, our algorithm and `eMPRess` produced very different scenarios. Here we could not compare them with the true scenario as it is unknown. We could only calculate the distance between the two different outputs. There are several factors behind the difference of the outputs. The first and most important is the way we obtain the separate trees. During reconstruction of phylogenetic trees (see Chapter 2.3), the tree is usually unrooted and branch lengths represent the number of mutations happening along them, not time. To convert them to ultrametric trees where branch lengths represent the time, there are several available methods, and each of them are based on the molecular clock hypothesis. Here, it is assumed that mutations happen according to a Poisson process with a certain parameter λ, which is assumed to be constant over time. With this assumption and using the information regarding when species were sampled, it is possible to convert the tree to a form where branch lengths represent time. For us, we know that all species were sampled at the same time, which means that in the end we get an ultrametric tree. The root of the tree then will be chosen as the midpoint of the path between the two most distantly related leaves of the tree. The assumption that the rate λ stays constant in time is strong, and has proven to be unrealistic, as mutation rates can vary between species and genes [Kum05]. There are several methods that relax this strong assumption, for example the rate does not need to be constant over time, but

can be chosen as an i.i.d. copy of a certain random variable along each branch of the inferred tree. There are also some Bayesian methods which allow the use and comparison of alternative models of substitution changes over time, implemented for example in `BEAST` [BVBS$^+$19]. For more details see Chapter 8.7 of [Ble17]. So the use of a strict molecular clock may result in dated trees with inaccurate timing information. Another possible source of inaccuracy is the adjustment of the coalescence times of the two trees, described in Chapter 6.4.2. Here, we compared the set of coalescence times in the two trees, and matched those which were close according to some measure. There is no guarantee that these events actually happened together, and vice versa, it is possible that there are cospeciation events that were not recovered by this method. The third source of difference between the outputs of our algorithm and `eMPRess` is the usage of the ghost line, which is not possible when using `eMPRess`. Considering all these factors, we can say that currently there are some limitations in using the algorithm for biological datasets. However, our algorithm has the advantage that it always produces a (strongly) time consistent scenario. And as new methods will be available which result in more accurate ultrametric trees, the performance of our algorithm will also improve on biological datasets.

A Appendix

Here we describe some results in detail that we use in the proofs in Chapter 3, but which are not integral parts of the proofs.

A.1 A modified version of convergence in probability

In this appendix we consider a modified version of convergence in probability, where the random variables are not all defined on the same probability space. After introducing the concept, we prove that some of the classical results (especially the Slutsky lemma) still remain valid. All the proofs are adaptations of the proofs for the classical results, see e.g. [].

When talking about convergence in probability, we assume that the sequence of random variables and the limiting random variable are all defined on the same probability space. However, when the limit is a constant, we can allow that the random variables are defined on different probability spaces as constants (degenerative random variables) are defined on all probability spaces. To be more precise, for all $n \in \mathbb{N}$ let $(\Omega_n, \mathcal{A}_n, \mathbb{P}_n)$ be a probability space, and let $X_n : \Omega_n \to \mathbb{R}^d$ be a random vector. Similarly to the classical setting, we say that the sequence of random variable $(X_n)_{n\in\mathbb{N}}$ converges in probability to a constant c if for all $\varepsilon > 0$ we have that $\lim_{n\to\infty} \mathbb{P}_n(\|X_n - c\| \geqslant \varepsilon) = 0$. We still denote this by $X_n \xrightarrow{\mathbb{P}} c$ when there is no confusion with the classical setting.
Before stating any of the results concerning this convergence in probability, we want to emphasize that results concerning weak convergence of probability measures (i.e. convergence in distribution), like the Portmanteau theorem or the continuous mapping theorem, still remain valid as for convergence in distribution it is not required that all random variables are defined on the same probability space. Convergence in distribution is denoted as usual by $X_n \xrightarrow{\mathcal{D}} X$. First we prove some auxiliary lemmas. Their proofs are essentially the same as in the classical case, with minor modifications.

Lemma A.1 *(Cramér-Slutsky) Let $(X_n)_{n\in\mathbb{N}}$ and $(Y_n)_{n\in\mathbb{N}}$ be two sequences of random vectors in $\mathbb{R}^d$ such that for each n, X_n and Y_n are defined on $(\Omega_n, \mathcal{A}_n, \mathbb{P}_n)$, and let X be another random vector defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$. If $X_n \xrightarrow{\mathcal{D}} X$ and $X_n - Y_n \xrightarrow{\mathbb{P}} 0$, then $Y_n \xrightarrow{\mathcal{D}} X$.*

Proof. We show that for each closed set $F \in \mathcal{B}(\mathbb{R}^d)$ it holds that

$$\limsup_{n\to\infty} \mathbb{P}_n(Y_n \in F) \leqslant \mathbb{P}(X \in F),$$

then the statement follows from the Portmanteau theorem. Using that the function $x \mapsto d(x, F) := \inf\{\|x - y\| : y \in F\}$ is continuous, we get that for any $\varepsilon > 0$ the set $F_\varepsilon := \{x \in \mathbb{R} : d(x, F) \leqslant \varepsilon\}$ is closed. Moreover, for any $\varepsilon > 0$ we have that

$$\begin{aligned}\mathbb{P}_n(Y_n \in F) &= \mathbb{P}_n(Y_n \in F, \|X_n - Y_n\| \geqslant \varepsilon) + \mathbb{P}_n(Y_n \in F, \|X_n - Y_n\| < \varepsilon) \\ &\leqslant \mathbb{P}_n(\|X_n - Y_n\| \geqslant \varepsilon) + \mathbb{P}_n(X_n \in F_\varepsilon),\end{aligned}$$

where we used that in case of $Y_n \in F$ and $\|X_n - Y_n\| < \varepsilon$ we have that

$$d(X_n, F) \leqslant \|X_n - Y_n\| + d(Y_n, F) = \|X_n - Y_n\| < \varepsilon.$$

Using the assumption $X_n \xrightarrow{\mathcal{D}} X$ and the Portmanteau theorem, we conclude that

$$\limsup_{n\to\infty} \mathbb{P}_n(Y_n \in F) \leqslant \limsup_{n\to\infty} \mathbb{P}_n(X_n \in F_\varepsilon) \leqslant \mathbb{P}(X \in F_\varepsilon). \tag{A.1}$$

Since F is closed, $d(x, F) = 0$ holds if and only if $x \in F$, which implies that $F_{1/k} \downarrow F$ as $k \to \infty$. Then by the continuity of the probability we get that $\lim_{k\to\infty} \mathbb{P}(X \in F_{1/k}) = \mathbb{P}(X \in F)$. Now, going back to (A.1), with the choice of $\varepsilon := 1/k$ and taking the limit $k \to \infty$, we get that $\limsup_{n\to\infty} \mathbb{P}_n(Y_n \in F) \leqslant \mathbb{P}(X \in F)$. □

With this lemma we can prove that convergence in distribution to a constant is equivalent to convergence in probability (in the new setting) to the same constant.

Lemma A.2 *Let $(X_n)_{n\in\mathbb{N}}$ be a sequence of random vectors in $\mathbb{R}^d$ such that for each n, X_n is defined on $(\Omega_n, \mathcal{A}_n, \mathbb{P}_n)$. Then convergence in distribution to a constant is equivalent to converging to the constant in probability:*

$$X_n \xrightarrow{\mathbb{P}} c \iff X_n \xrightarrow{\mathcal{D}} c$$

Proof. First, let us suppose that $X_n \xrightarrow{\mathbb{P}} c$. For the sequence $X'_n := c$ it holds trivially that $X'_n \xrightarrow{\mathcal{D}} c$, and since $X_n - c \xrightarrow{\mathbb{P}} 0$, we get by Lemma A.1 that $X_n = X'_n + X_n - c \xrightarrow{\mathcal{D}} c$. Now for the other direction, let us suppose that $X_n \xrightarrow{\mathcal{D}} c$ and let $\varepsilon > 0$. The set $F_\varepsilon = \{y \in \mathbb{R}^d : \|y - c\| \geqslant \varepsilon\}$ is closed and $c \notin F_\varepsilon$. By the Portmanteau theorem we get that

$$\limsup_{n\to\infty} \mathbb{P}_n(\|X_n - c\| \geqslant \varepsilon) = \limsup_{n\to\infty} \mathbb{P}_n(X_n \in F_\varepsilon) \leqslant \mathbb{P}(c \in F_\varepsilon) = 0,$$

which implies $X_n \xrightarrow{\mathbb{P}} c$. □

Another result that we are going to use is the following:

Lemma A.3 *For all n, let X_n and Y_n be random vectors in $\mathbb{R}^d$ defined on a probability space $(\Omega_n, \mathcal{A}_n, \mathbb{P}_n)$, and let X be a random vector in $\mathbb{R}^d$ defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$, and let $c \in \mathbb{R}^d$. Then, if $X_n \xrightarrow{\mathcal{D}} X$ and $Y_n \xrightarrow{\mathbb{P}} c$, then $(X_n, Y_n) \xrightarrow{\mathcal{D}} (X, c)$.*

Proof. First we show that $(X_n, c) \xrightarrow{\mathcal{D}} (X, c)$. For that we will need to show that for all continuous and bounded functions $f : \mathbb{R}^{2d} \to \mathbb{R}$ we have that $\mathbb{E}_n(f(X_n, c)) \to \mathbb{E}(f(X, c))$. Let f be such a function, and let $g : \mathbb{R}^d \to \mathbb{R}, g(x) := f(x, c)$. Then also g is continuous and bounded, and by the Portmanteau theorem we have $\mathbb{E}_n(g(X_n)) \to \mathbb{E}(g(X))$, since $X_n \xrightarrow{\mathcal{D}} X$ holds. It means that $\mathbb{E}_n(f(X_n, c)) \to \mathbb{E}(f(X, c))$, and so $(X_n, c) \xrightarrow{\mathcal{D}} (X, c)$. As by the assumption $(X_n, Y_n) - (X_n, c) = (0, Y_n - c) \xrightarrow{\mathbb{P}} (0, 0) \in \mathbb{R}^{2d}$, Lemma A.1 yields that $(X_n, Y_n) \xrightarrow{\mathcal{D}} (X, c)$.

Next prove that a version of Slutsky's lemma holds also in this setting.

Lemma A.4 *(Slutsky) For all n, let X_n, Y_n and Z_n be random vectors in $\mathbb{R}^d$ defined on a probability space $(\Omega_n, \mathcal{A}_n, \mathbb{P}_n)$, and let X be a random vector defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$. Moreover, let a and b be deterministic vectors in $\mathbb{R}^d$. If $X_n \xrightarrow{\mathcal{D}} X$, $Y_n \xrightarrow{\mathbb{P}} a$ and $Z_n \xrightarrow{\mathbb{P}} b$, then $Y_n X_n + Z_n \xrightarrow{\mathcal{D}} aX + b$.*

Proof. As $Z_n - b \xrightarrow{\mathbb{P}} 0$, by Lemma A.1 it suffices to show that $Y_n X_n + b \xrightarrow{\mathcal{D}} aX + b$ for which, by the continuous mapping theorem, it suffices to show that $Y_n X_n \xrightarrow{\mathcal{D}} aX$. By Lemma A.3 we have that $(X_n, Y_n) \xrightarrow{\mathcal{D}} (X, a)$, and by the applying the continuous mapping theorem with the continuous function $h : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d, h(x, y) := yx$, we get that $Y_n X_n \xrightarrow{\mathcal{D}} aX$. □

A.2 Wasserstein distance of conditional measures

In this chapter we summarize the most important properties of the 1-Wasserstein distance, with emphasis on the Wasserstein distance of conditional measures. As we are only considering the 1-Wasserstein distance, it will be denoted as d_W throughout this chapter instead of d_W^1.

As a reminder, the 1-Wasserstein distance of two probability measures μ and ν on a space $(\Omega, \mathcal{A})$ are defined as

$$d_W(\mu, \nu) = \inf_{\substack{X \sim \mu \\ Y \sim \nu}} \mathbb{E}(|X - Y|), \tag{A.2}$$

where the infimum is taken over all possible couplings of the measures μ and ν. Another way to calculate this distance is given by the Kantorovich–Rubinstein duality:

$$d_W(\mu, \nu) = \sup_{\|f\|_L \leqslant 1} \mathbb{E}(f(X)) - \mathbb{E}(f(Y)) \quad \text{with } X \sim \mu \text{ and } Y \sim \nu, \tag{A.3}$$

where the supremum is taken over all Lipschitz continuous functions f with Lipschitz constant of maximum 1.

If μ and ν are probability measures on $\mathbb{R}$, then their 1-Wasserstein distance can be calculated using their distribution functions. In this case,

$$d_W(\mu, \nu) = \int_{\mathbb{R}} |F_\mu(x) - F_\nu(x)| dx \tag{A.4}$$

where F_μ and F_ν are the distibution functions of μ and ν, respectively (i.e. $F_\mu(x) = \mu((-\infty, x])$ and $F_\nu(x) = \nu((-\infty, x])$ for all $x \in \mathbb{R}$,).

However, having some additional information about the two distributions can change the distance between them. Suppose that we have a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ and two random variables $\xi : \Omega \to \mathbb{R}$ and $\eta : \Omega \to \mathbb{R}$ defined on them. Whenever we are referring to the distance of the distributions of ξ and η, we will denote it with $d_W(\xi, \eta)$ but we actually mean $d_W(\mathbb{P}_\xi, \mathbb{P}_\eta)$. Suppose we have a (sub) σ-algebra $\mathcal{F} \subset \mathcal{A}$ and suppose that we do not know the distribution of ξ, but only the conditional distribution $\xi|\mathcal{F}$. In that case note that $d_W(\xi|\mathcal{F}, \eta)$ is a random quantity itself. In this setting we need a result that compares the Wasserstein distance of the conditional and the unconditional distributions.

Proposition A.5 *Let ξ and η be two random variables on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ and let $\mathcal{F} \subset \mathcal{A}$ be a σ-algebra. Then*

$$d_W(\xi, \eta) \leqslant \mathbb{E}(d_W(\xi|\mathcal{F}, \eta)) \tag{A.5}$$

Proof. By (A.4) and the properties of the conditional expectation we have that

$$\begin{aligned}
d_W(\xi, \eta) &= \int_{\mathbb{R}} |F_\xi(x) - F_\eta(x)| \mathrm{d}x = \int_{\mathbb{R}} |\,\mathbb{E}(\mathbf{1}_{\{\xi \leqslant x\}}) - \mathbb{E}(\mathbf{1}_{\{\eta \leqslant x\}})| \mathrm{d}x \\
&= \int_{\mathbb{R}} |\,\mathbb{E}(\mathbf{1}_{\{\xi \leqslant x\}} - \mathbf{1}_{\{\eta \leqslant x\}})| \mathrm{d}x = \int_{\mathbb{R}} |\,\mathbb{E}\left(\mathbb{E}(\mathbf{1}_{\{\xi \leqslant x\}}|\mathcal{F}) - \mathbb{E}(\mathbf{1}_{\{\eta \leqslant x\}})\right)| \mathrm{d}x \\
&\leqslant \int_{\mathbb{R}} \mathbb{E}\left(|\,\mathbb{E}(\mathbf{1}_{\{\xi \leqslant x\}}|\mathcal{F}) - \mathbb{E}(\mathbf{1}_{\{\eta \leqslant x\}})|\right) \mathrm{d}x = \mathbb{E}\left(\int_{\mathbb{R}} |\,\mathbb{E}(\mathbf{1}_{\{\xi \leqslant x\}}|\mathcal{F}) - \mathbb{E}(\mathbf{1}_{\{\eta \leqslant x\}})| \mathrm{d}x\right) \\
&= \mathbb{E}\left(\int_{\mathbb{R}} |F_{\xi|\mathcal{F}}(x) - F_\eta(x)| \mathrm{d}x\right) = \mathbb{E}(d_W(\xi|\mathcal{F}, \eta))
\end{aligned}$$

where we also used the Fubini theorem. □

A.3 Source codes

A Python 3 implementation of the algorithm described in Chapter 4 is available at `https://gitlab.gwdg.de/boesze/tisetra`. A code for visualization of joint trees and computation of the time-sensitive distance between joint trees is provided. An example biological dataset is provided as well.

References

[AAR08] E.S. Allman, C. Ané, and J.A. Rhodes. Identifiability of a Markovian model of molecular evolution with gamma-distributed rates. *Adv. Appl. Prob.*, 40:229–249, 2008.

[AG16] G. Alsmeyer and S. Göttrup. Branching within branching: A model for host-parasite co-evolution. *Stochastic Processes and their Applications*, 126(6):1839–1883, 2016.

[AGM+90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[AMS+97] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[AN72] K.B. Athreya and P.E. Ney. *Branching Processes*. Springer-Verlag, New York, 1972.

[BAGFS12] M.S.C. Blagrove, C. Arias-Goeta, A.-B. Failloux, and S.P. Sinkins. Wolbachia strain wMel induces cytoplasmic incompatibility and blocks dengue transmission in Aedes albopictus. *Proc. Natl. Acad. Sci. USA*, 109:255–260, 2012.

[BAH+08] L. Baldo, N.A. Ayoub, C.H. Hayashi, J.A. Russell, J.K. Stahlhut, and J.H. Werren. Insight into the routes of wolbachia invasion: high levels of horizontal transfer in the spider genus Agelenopsis revealed by wolbachia strain and mitochondrial DNA diversity. *Molecular Ecology*, 17:557–569, 2008.

[Bai64] N.T.J. Bailey. *The elements of stochastic processes with applications to the natural sciences*. Wiley Publications in Statistics, 1964.

[BDLSJ18] A. Blancas, J.L. Duchamps, A. Lambert, and A. Siri-Jégousse. Trees within trees: simple nested coalescents. *Electronic Journal of Probability*, 23, 2018.

[BDM12] A. Boc, A.B. Diallo, and V. Makarenkov. T-REX: a web server for inferring, validating and visualizing phylogenetic trees and networks. *Nucleic Acids Research*, 40:Web Server Issue, W573–W579, 2012.

[Ber09] J. Berestycki. Recent progresses in coalescent theory. *Ensaios Matemáticos*, 16:1–193, 2009.

[BGCKWB16] J. Blath, A. González-Casanova, N. Kurt, and M. Wilke-Berenguer. A new coalescent for seed-bank models. *Annals of Applied Probability*, 26(2):857–891, 2016.

[BKKK18] M.S. Bansal, M. Kellis, M. Kordi, and S. Kundu. RANGER-DTL 2.0 : Rigorous reconstruction of gene-family evolution by duplication, transfer and loss. *Bioinformatics*, 34(18):3214–3216, 2018.

[Bla18] A. Blancas. Two contributions to the theory of stochastic population dynamics. (Doctoral thesis). *Available at* `https://www.repositorionacionalcti.mx/`, 2018.

[Ble17] C. Bleidorn. *Phylogenomics*. Springer Cham, 2017.

[BMMG24] M. Bernabeu, S. Manzano-Morales, and T. Gabaldón. On the impact of incomplete taxon sampling on the relative timing of gene transfer events. *PLoS Biology*, 22(3):e3002460, 2024.

[BPM10] A. Boc, H. Philippe, and V. Makarenkov. Inferring and validating horizontal gene transfer events using bipartition dissimilarity. *Syst. Biol.*, 59(2):195–211, 2010.

[Bre10] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 2010.

[BRTR18] O. Balvín, S. Roth, B. Talbot, and K. Reinhardt. Co-speciation in bedbug Wolbachia parallel the pattern in nematode hosts. *Scientific Reports*, 8(8797), 2018.

[BT11] V. Bansaye and V.C. Tran. Branching Feller diffusion for cell division with parasite infection. *ALEA: Latin American Journal of Probability and Mathematical Statistics*, 8:95–127, 2011.

[BVBS+19] R. Bouckaert, T.G. Vaughan, J. Barido-Sottani, S. Duchêne, M. Fourment, A. Gavryushkina, et al. BEAST2.5 : An advanced software platform for Bayesian evolutionary analysis. *PLoS Computational Biology*, 15(4):e1006650, 2019.

[Can74] C. Cannings. The latent roots of certain Markov chains arising in genetics: a new approach. I. haploid models. *Adv. Appl. Probab.*, 6:260–290, 1974.

[Can75] C. Cannings. The latent roots of certain Markov chains arising in genetics: a new approach. II. further haploid models. *Adv. Appl. Probab.*, 7:264–282, 1975.

[CDW12] Z.Z. Chen, F. Deng, and L. Wang. Simultaneous identification of duplications, losses, and lateral gene transfers. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 9:1515–1528, 2012.

[CFOLH10] C. Cononw, D. Fielder, Y. Ovadia, and R. Libeskind-Hadas. Jane: a new tool for cophylogeny reconstruction problem. *Algorithms Mol. Biol.*, 5(16), 2010.

[Dar59] C. Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, London, 1859.

[Daw18] D. Dawson. Multilevel mutation-selection systems and set-valued duals. *J. Math. Biol.*, 76:295–378, 2018.

[DBS+15] B. Donati, C. Baudet, B. Sinaimeri, P. Crescenzi, and M.F. Sagot. EUCALYPT: Efficient tree reconciliation enumerator. *Algorithms Mol. Biol.*, 10(3), 2015.

[DEM24] M. Delabre and N. El-Mabrouk. Synesth: comprehensive syntenic reconciliation with unsampled lineages. *Algorithms*, 17:186, 2024.

[DPK+20] D. Darriba, D. Posada, A.M. Kozlov, A. Stamatakis, B. Morel, and T. Flouri. ModelTest-NG: a new and scalable tool for the selection of DNA and protein evolutionary models. *Mol. Biol. Evol.*, 37(1):291–294, 2020.

[DR09] J.H. Degnan and N.A. Rosenberg. Gene tree discordance, phylogenetic inference and the multispecies coalescent. *Trends in Ecology and Evolution*, 24:332–340, 2009.

[DSG+10] J.P. Doyon, C. Scornavacca, K.Y. Gorbunov, G.J. Szöllősi, V. Ranwez, and V. Berry. An efficient algorithm for gene/species trees parsimonius reconciliation with losses, duplications and transfers. *In Research in Computational Molecular Biology—Comparative Genomics; Tannier, E., Ed.; Springer: Berlin/Heidelberg, Germany*, pages 93–108, 2010.

[Dur08] R. Durrett. *Probability model for DNA sequence evolution*. Springer, New York, 2nd edition, 2008.

[EF19] J. Engelstädter and N. Fortuna. The dynamics of preferential host switching: host phylogeny as a key predictor of parasite prevalence and distribution. *Evolution*, 73:1330–1340, 2019.

[EK72] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[EK86] S.N. Ethier and T.G. Kurtz. *Markov Processes: Characterization and Convergence*. John Wiley & Sons, New York, 1986.

[Fel81] J. Felsenstein. Evolutionary trees from DNA sequences: a maximum-likelihood approach. *J. Mol. Evol.*, 17:368–376, 1981.

[FFRS20] A. Lambert F. Foutel-Rodier and E. Schertzer. Kingman's coalescent with erosion. *Electron. J. Probab.*, 25(56):1–33, 2020.

[FICD+14] T. Flouri, F. Izquierdo-Carrasco, D. Darriba, A.J. Aberer, L.T. Nguyen, B.Q. Minh, A. von Haeseler, and A. Stamatakis. The phylogenetic likelihood library. *Syst. Biol.*, 64(2):356–362, 2014.

[GB16] M. Gerth and C. Bleidorn. Comparative genomics provides a timeframe for Wolbachia evolution and exposes a recent biotin synthesis operon transfer. *Nat Microbiol.*, 2(16241), 2016.

[GRB13] M. Gerth, J.. Röthe, and C. Bleidorn. Tracing horizontal Wolbachia movements along bees (Anthophila): a combined approach using multilocus sequence typing data and host phylogeny. *Mol. Ecol.*, 22:6149–6162, 2013.

[Gus02] D. Gusfield. Partition-distance: A problem and class of perfect graphs arising in clustering. *Information Processing Letters*, 82:159–164, 2002.

[GVDL+11] H.R. Gruber-Vodicka, U. Dirks, N. Leisch, C. Baranyi, K. Stoecker, S. Bulgheresi, N.R. Heindl, M. Horn, C. Lott, A. Loy, M. Wagner, and J. Ott. Paracatenula, an ancient symbiosis between thiotrophic Alphaproteobacteria and catenulid flatworms. *PNAS*, 108(29):12078–12083, 2011.

[GWK05] F. Ge, L.-S. Wang, and J. Kim. The cobweb of life revealed by genome-scale estimates of horizontal gene transfer. *PLoS Biol.*, 3(10):e316, 2005.

[HBA+16] L. Hug, B. Baker, K. Ananthamaran, et al. A new view of the tree of life. *Nat. Microbiol.*, 1(16048), 2016.

[HCSB16] J. Huerta-Cepas, F. Serra, and P. Bork. Ete 3: Reconstruction, analysis and visualization of phylogenomic data. *Mol. Biol. Evol.*, 33(6):1635–1638, 2016.

[HH08] B.K. Hall and B. Hallgrímsson. *Strickberger's Evolution*. Jones and Bartlett Publishers, Sudbury, MA, 4th edition, 2008.

[HL01] M. Hallet and J. Lagergren. Efficient algorithms for later gene transfer problems. *In: El-Mabrouk N., Lengauer T., Sankoff D., editors. Proceedings of the Fifth Annual International Conference on Research in Computational Biology. New York: ACM Press*, pages 149–156, 2001.

[HT19] D. Hasić and E. Tannier. Gene tree reconciliation including transfers with replacement is NP-hard and FPT. *Journal of Combinatorial Optimization*, 38:502–544, 2019.

[JCS+16] E. Jacox, C. Chauve, G.J. Szöllősi, Y. Ponty, and C. Scornavacca. ecceTERA: comprehensive gene tree - species tree reconciliation using parsimony. *Bioinformatics*, 32(13):2056–2058, 2016.

[KA15] M. Kimmel and D.E. Axelrod. *Branching processes in biology*. Springer, New York, 2nd edition, 2015.

[Kin82a] J.F.C. Kingman. The coalescent. *Stochastic Processes and their Applications*, 13:235–248, 1982.

[Kin82b] J.F.C. Kingman. On the genealogy of large populations. *Journal of Applied Probability*, 19:27–43, 1982.

[KKL01] I. Kaj, S.M. Krone, and M. Lascoux. Coalescent theory for seed bank models. *Journal of Applied Probability*, 38(2):285–300, 2001.

[Knu97] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 1997.

[Kub09] L.S. Kubatko. Identifying hybridization events in the presence of coalescence via model selection. *Syst. Biol.*, 58(5):478–488, 2009.

[Kuh55] H.W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2, 1955.

[Kum05] S. Kumar. Molecular clocks: four decades of evolution. *Nat. Rev. Genet.*, 6:654–662, 2005.

[LDTB16] F. Leroy, J.Y. Dauxois, and P. Tubert-Bitter. On the parametric maximum likelihood estimator for independent but non-identically distributed observations with application to truncated data. *Journal of Statistical Theory and Applications*, 15(1):96–107, 2016.

[LGNH24] J. Lueg, M.K. Garba, T.M.W. Nye, and S.F. Huckemann. Foundations of the wald space for phylogenetic trees. *J. London Math. Soc.*, 109:e12893, 2024.

[LH22] R. Libeskind-Hadas. Tree reconciliation methods for host-symbiont cophylogenetic analyses. *Life*, 12(443), 2022.

[LHWBK14] R. Libeskind-Hadas, Y.C. Wu, M.S. Bansal, and M. Kellis. Pareto-optimal phylogenetic tree reconciliation. *Bioinformatics*, 30:i87–i95, 2014.

[LRvH07] S. Linz, A. Radtke., and A. von Haeseler. A likelihood framework to measure horizontal gene transfer. *Mol. Biol. Evol.*, 24(6):1312–1319, 2007.

[LS06] V. Limic and A. Sturm. The spatial λ-coalescent. *Electronic Journal of Probability*, 11(15):363–393, 2006.

[MACL20] J. Marin, G. Achaz, A. Crombach, and A. Lambert. The genomic view of diversification. *J. Evol. Biol.*, 33:1387–1404, 2020.

[MCDB05] D. MacLeod, R.L. Charlebois, F. Doolittle, and E. Bapteste. Deduction of probable events of lateral gene transfer through compariosn of phylogenetic trees by recursive consolidation and rearrangement. *BMC Evolutionary Biology*, 5(27), 2005.

[Mei18] R. Meizis. Metric two-level measure spaces: A state space for modeling evolving genealogies in host-parasite systems. (Doctoral thesis). *Available at* ***https://duepublico2.uni-due.de/receive/duepublico_mods_00070033***, 2018.

[MIOJ+09] L.A. Moreira, I. Iturbe-Ormaetxe, J.A. Jeffery, et al. A Wolbachia symbiont in Aedes aegypti limits infection with dengue, chikungunya and plasmodium. *Cell*, 139:1268–1278, 2009.

[MKV+21] S. Mishra, U. Klümper, V. Voolaid, T.U. Berendon, and D. Kneis. Simultaneous estimation of parameters governing the vertical and horizontal transfer of antibiotic resistance genes. *Science of The Total Environment*, 798:149174, 2021.

[MMW10] D. Merkle, M. Middendorf, and N. Wieseke. A parameter-adaptive dynamic programming approach for inferring cophylogenies. *BMC Bioinform.*, 11:S60, 2010.

[Mor62] P.A.P. Moran. *The statistical processes of evolutionary theory.* Clarendon Press, Oxford University Press, 1962.

[MR13] S. Méléard and S. Roelly. Evolutive two-level population process anf large population approximations. *Annals of the University of Bucharest*, 4(LXII):37–70, 2013.

[MS01] M. Möhle and S. Sagitov. A classification of coalescent processes for haploid exchangeable population models. *The Annals of Probability*, 29(4):1547–1562, 2001.

[Mun57] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[Not90] M. Notahara. The coalescent and the genealogical process in geographically structured population. *J. Math. Biol.*, 29(1):59–75, 1990.

[NRBS13] T.H. Nguyen, V. Ranwez, V. Berry, and C. Scornavacca. Support measures to estimate the reliability of evolutionary events predicted by reconciliation methods. *PLoS ONE*, 8:e73667, 2013.

[NSvHM15] L.-T. Nguyen, H.A. Schmidt, A. von Haeseler, and B.Q. Minh. IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Mol. Biol. Evol.*, 32:268–274, 2015.

[OFCLH11] Y. Ovadia, D. Fielder, C. Conow, and R. Libeskind-Hadas. The cophylogeny reconstruction problem is NP-complete. *J. Comput. Biol.*, 18:59–65, 2011.

[OW20] L. Osorio and A. Winter. Two level branching model for virus population under cell division. *Preprint. Available at* `https://arxiv.org/abs/2004.14352`, 2020.

[Pag95] R.D.M Page. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syt. Biol.*, 43(1):58–77, 1995.

[PG19] C. Pokalyuk and I. Görzer. Diverstiy patterns in parasite populations capable for persistence and reinfection. *Preprint. Available at* `https://www.biorxiv.org/content/10.1101/512970v2`, 2019.

[PW20] C. Pokalyuk and A. Wakolbinger. Maintenance of diversity in a hierarchical host-parasite model with balancing selection and reinfection. *Stoch. Proc. Appl.*, 130:1119–1158, 2020.

[RK12] M.D. Rasmussen and M. Kellis. Unified modeling of gene duplication, loss and coalescence using a locus tree. *Genome Res.*, 22:755–765, 2012.

[Ros11] N. Ross. Fundamentals of Stein's method. *Probability Surveys*, 8:210–293, 2011.

[RvLD15] M. Ravenhall, N. Škunca, F. Lassalle, and C. Dessimoz. Inferring horizontal gene transfer. *PLoS Computational Biology*, 11(5):e1004095, 2015.

[SLS+21] D. Schaller, M. Lafond, P.F. Stadler, N. Wieseke, and M. Hellmuth. Indirect indentification of horizontal gene transfer. *Journal of Mathematical Biology*, 83(10), 2021.

[SLYA16] C. Solís-Lemus, M. Yang, and C. Ané. Inconsistency of species tree methods under gene flow. *Syst. Biol.*, 65(5):843–851, 2016.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.

[Sta14] A. Stamatakis. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30:1312–1313, 2014.

[STDB14] G.J. Szöllősi, E. Tannier, V. Daubin, and B. Boussau. The inference of gene trees with species trees. *Syst. Biol.*, 64(1):e42–e62, 2014.

[STLD13] G.J. Szöllősi, E. Tannier, N. Lartillot, and V. Daubin. Lateral gene transfer from the dead. *Systematic Biology*, 62(3):386–397, 2013.

[SYL+21] S. Santichaivekin, Q. Yang, J. Liu, R. Maworther, J. Jiang, T. Wesley, Y.C. Wu, and R. Libeskind-Hadas. empress: A systematic cophylogeny reconciliation tool. *Bioinformatics*, 37(16):2481–2482, 2021.

[Tak89] N. Takahata. Gene genealogy in three related populations: consistency probability between gene and population trees. *Genetics*, 122:957–966, 1989.

[Tav86] S. Tavaré. Some probabilistic and statistical problems in the analysis of DNA sequences. *Lectures on Mathematics in the Life Sciences*, 17:57–86, 1986.

[THL11] A. Tofigh, M.T. Hallett, and J. Lagergren. Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 8:517–535, 2011.

[Wak08] J. Wakeley. *Coalescent Theory: An Introduction*. Roberts and Company Publishers, Greenwood Village, CO, 2008.

[WB21] S. Weiner and M.S. Bansal. Improved duplication-transfer-loss reconciliation with extinct and unsampled lineages. *Algorithms*, 14:231, 2021.

[WBC08] J.H. Werren, L. Baldo, and M.E. Clark. Wolbachia: master manupulators of invertebrate biology. *Nature Reviews Microbiology*, 6:741–751, 2008.

[WMSS20] Y. Wang, A. Mary, M.F. Sagot, and B. Sinaimeri. Capybara: Equivalence class enumeration of cophylogeny event-based reconciliations. *Bioinformatics*, 36:4197–4199, 2020.

[Wu93] Y. Wu. A multilevel birth-death particle system and its continuous diffusion. *Adv. Appl. Prob.*, 25(3):549–569, 1993.

[Yan97] Z. Yang. PAML: a program package for phylogenetic analysis by maximum likelihood. *Bioinformatics*, 13:555–556, 1997.

[YDN12] Y. Yu, J.H. Degnan, and L. Nakhleh. The probability of a gene tree topology within a phylogenetic network with applications to hybridization. *PLoS Genetics*, 8(4):e1002660, 2012.

[ZDOKH16] S.A. Zamani-Dahaj, M. Okasha, J. Kosakowski, and P.G. Higgs. Estimating the frequency of horizontal gene transfer using phylogenetic models of gene gain and loss. *Mol. Biol. Evol.*, 33(7):1843–1857, 2016.

[ZH12] R. Zug and P. Hammerstein. Still a host of hosts for Wolbachia: Analysis of recent data suggests that 40% of terrestrial arthropod species are infected. *PLoS ONE*, 7(6):e38544, 2012.

[ZRSM18] C. Zhang, M. Rabiee, E. Sayyari, and S. Mirarab. ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics*, 19(153), 2018.