Estimating Phylogenetic Networks from Concatenated Sequence Alignments

By
Cora Allen-Savietta

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Statistics)

at the
UNIVERSITY OF WISCONSIN-MADISON

2020

Date of final oral examination: 2020-12-10

The dissertation is approved by the following members of the Final Oral Committee:

Cécile Ané, Professor, Statistics and Botany
Bret Larget, Professor, Statistics and Botany
Ron Gangnon, Professor, Biostatistics & Medical Informatics, Population Health Sciences, and Statistics
Michael Newton, Professor, Statistics and Biostatistics & Medical Informatics
Karl Rohe, Associate Professor, Statistics

To my grandmothers: Luella, for encouraging me to seek the beauty in this world and, Lucille, for showing me how much of that beauty could be found in mathematics.

Acknowledgements

I'm grateful for financial support from the National Institutes of Health Data Science and Biostatistics traineeships and from the National Science Foundation through the "Scalable Model-Based Reconstruction of Network Evolution" grant and the University of Wisconsin Institute for the Foundations of Data Science.

I would like to thank Cécile Ané, my primary advisor, for her contagious enthusiasm, wise mentorship, immense knowledge, and generous support, and all my committee members, Professors Bret Larget, Ron Gangnon, Michael Newton, and Karl Rohe, for their guidance and generosity. Thank you to my dissertation writing group, for their friendship and wisdom, and the University of Wisconsin's Writing Center for their excellent advice on writing throughout the years.

Finally, thank you to my parents, my generous role models, who taught me to greet each day with curiosity; my siblings, who make me wonder how I got so lucky; and my wife, who is both my solid ground and the wind under my wings. This accomplishment is yours as well as mine.

CONTENTS

ABSTRACT

In this dissertation, I present five chapters. The first chapter stems from a rotation project. In it, we explore a clustered ranking method for small area health outcomes in several applied data sets. By applying clustering to these rankings, we hope to improve rankings' effectiveness as a public health communication tool.

In the remaining four chapters, I present an estimation method for phylogenetic networks. Evolution is often depicted as a tree, with an ancestral species at the root and modern species branching out to portray a leaf-filled canopy of biological diversity. While this model of evolution fits some species, evidence now indicates that many lineages cannot be well represented by a tree. Reticulate events such as hybridization and horizontal gene flow create connections between branches of this ancestral tree, turning it from a tree into a network. Even when scientists are mainly interested in the tree-like components of inheritance, ignoring these reticulate events has been shown to lead to inaccurate estimation. Without a network-based model of evolution, we miss these important events in evolutionary history and may even incorrectly estimate a species' primary ancestry.

In this work, we develop a new method to extend phylogenetic analysis from trees to networks, allowing analysts to infer both the tree-like and reticulate events in evolutionary history with concatenated sequence data. In chapter two, we introduce the graphical model for estimating parameterized semi-directed phylogenetic networks. Then, in chapter three, we present identifiability results, as well as practical consequences for network estimation. Finally, we combine our identifiability results to define a canonical phylogenetic network from concatenated sequence data.

In chapters four and five, we introduce our phylogenetic network estimation method, which we call PhyLiNC. PhyLiNC estimates a fully parameterized network topology with edge lengths, hybrid edge inheritance percentages, and rates of DNA evolution from concatenated DNA sequences. We describe our method's topology estimation algorithm and introduce our extension to nearest neighbor interchange moves for parameterized semi-directed networks. We conclude, in chapter five, with a user-friendly manual, which includes examples and step-by-step instructions for estimating networks with our software.

INTRODUCTION

In this dissertation, I present five chapters. The first stems from a rotation project with Professor Ron Gangnon. In it, we explore a clustered ranking method for small area health outcomes in several applied data sets. Ranking small areas, such as a county or city, for example, help communicate important health findings to the medical community, policy makers, and the general public. By adding clustering to these rankings, we hope to more clearly communicate the natural variability that appears in the underlying data and improve the effectiveness of rankings as a public health communication tool.

In the remaining four chapters, I present work on phylogenetic networks with Professor Cécile Ané. I will give a brief background and motivation for this work in the following paragraphs.

Phylogenies depict evolutionary relationships between species over time, generally as a tree or tree-like network. The ancestral lineage serves as the root of this tree, with the recent species appearing as leaves. Figure 0.1 shows a simple example, with the modern species, labeled as A, B, C, D, and E, on the right and the most recent common ancestor of these five species on the left. Interior nodes represent speciation or other important events during these species' evolutionary history. Because this phylogeny is a tree, every interior node represents a speciation event, the juncture when one ancestral species split into two species. In a phylogenetic network, interior nodes can also represent hybridization or horizontal gene transfer events. Edges represent times between these events, with longer edges representing longer stretches of time.

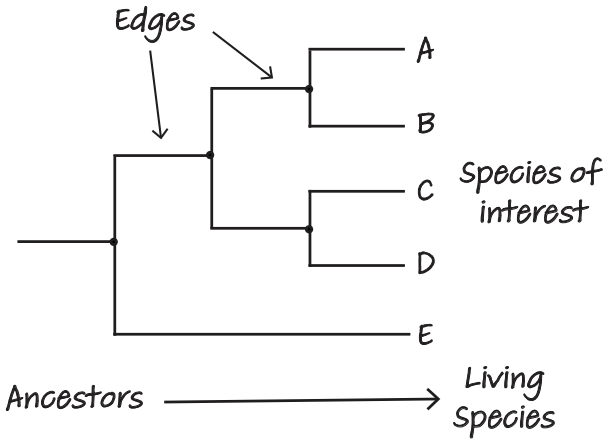

Figure 0.1: A Phylogenetic History for Species A, B, C, D, and E

Phylogenetic analysis is used to address a wide range of inquiries across fields, from research on ancient

human ancestry to urgent questions about modern medicine. For example, a wide variety of recent papers have tracked the spread and evolution of the SARS-CoV-2 virus around the world using phylogenetic tools. These analyses have provided answers to a wide range of important questions, from how the virus moved across continents (Candido *et al.*, 2020; Isabel *et al.*, 2020; Forster *et al.*, 2020) to how quickly it evolves (Pereson *et al.*, 2020). They have revealed the existence of hidden, early super-spreader events that set off the pandemic (Wang *et al.*, 2020) and uncovered evidence that the precursor of SARS-CoV-2 has been incubating in bat populations for decades, unnoticed (Boni *et al.*, 2020). This is just a small sampling of the ways phylogenetic analysis can answer important questions in human health, as well as in ecology, language evolution, human history, and many other areas.

In chapters two, three, four, and five of this dissertation, we present a model, identifiability results, and implementation for phylogenetic inference from sequence data. We hope our contributions can move the field forward, allowing scientists to address these important questions even more effectively.

Chapter 1

Simultaneous Clustering and Ranking of Small Area Health Outcomes with Nonparametric Empirical Bayes Mixture Models

Abstract In this section, I describe a clustered ranking method developed after discussions with collaborators at Wisconsin's County Health Rankings. Unlike their current methods, which use ranking exclusively, this method would incorporate clustering of ranks to clarify the variability between counties and aid in the interpretation of health ranking for researchers, policy-makers, and clinicians. In this chapter, I explore several ideas for how to improve rankings for county health applications, from improving interpretation using clusters to improving behavior around heterogeneity with weighting.

Keywords Bayesian methods; public health; ranking; clustering

Contribution I implemented the method, conducted analyses, produced visualizations, and wrote the first draft. Ron Gangnon and I contributed to this paper and serve as second and first authors, respectively. This work was completed as part of a Biostatistics training rotation with Professor Ron Gangnon.

1.1 INTRODUCTION

In the course of communicating health findings with the public, researchers are often asked to rank small areas. We may need to identify the ten healthiest counties in a state or five lowest-performing hospitals. Often, in these settings, researchers find that clear rankings of items obscure more subtle messages in the data. For example, two counties might have very similar percentages of residents with high blood sugar. Then, spaced significantly farther away, the third ranked county might have a much lower percentage. When we rank them as first, second, and third counties, analysts of this information would be excused from thinking that the first is equally far from the second, which is equally far from the third. Ranking items often obscures relationships, rather than clarifies them: we often lose a sense of the spacing between items. In these cases, instead of clearly communicating findings, rankings can muddy them.

This work was motivated by a collaboration with researchers at county health rankings. They develop ranking of counties on health indicators and aid in the interpretation of these rankings. They seek to use these as an educational tool and as a tool to guide healthcare improvement across the county. They find that rankings are an effective data-dissemination tool. They also find that ranking can lead to an overly simplified conversation about healthcare across counties within a state. More nuanced ways to handle uncertainty and uneven spacing between items within a ranking could help them communicate the nuances in the data. To that end, we sought to develop and explore several suggestions that might be of use to them.

Many researchers have sought to improve rankings for various applications, from school quality to genetic markers to health indices (Gibbons *et al.*, 1979; Aitkin and Longford, 1986; Goldstein and Spiegelhalter, 1996). One of the major challenges associated with ranking is bias related to variance (Paddock and Louis, 2011). Methods like maximum likelihood tend to overrepresent units with large variance in the highest and lowest ranks. Conversely, competing methods like testing tovercorrect by pushing units with low variance toward these extreme ranks (Louis and Shen, 1999). Biases in ranking due to heteroskedasticity are a particular concern when sample sizes vary widely across the items to rank. This is a consistent challenge in ranking health indices by county, our application of interest.

Researchers have taken several tacks to address these bias challenges. To address the problem of high variance units, Laird and Louis (1989) propose an empirical Bayesian method to shrink estimates toward

a common mean. Building on these developments, Henderson and Newton (2016) address both these problems with a ranking variable that aligns unit-specific posterior probabilities with empirical Bayesian distributions. With this method, they aim to reduce both kinds of bias: the local maximum likelihood bias toward items with high uncertainty and the testing approaches bias for items with low uncertainty, achieving better performance than both types of method. Jewett *et al.* (2019) offered another approach to reducing these biases: they propose that analysts should focus on the appropriate choice of loss function and offer a range of loss functions to support diverse ranking goals. Others have sought to improve the precision of ranks for units with small sample sizes through hierarchical empirical Bayesian methods and modeling outcomes jointly (Athens *et al.*, 2013, 2015).

Beyond the challenge of bias due to differences in variance, we are also faced with the challenge of multiple objectives. Ranking methods often try to accomplish two or even three goals: estimate parameters, determine overall rank position, and present pairwise comparisons of all ranked units (Shen and Louis, 1998). In most cases, it is not possible to achieve all of these goals. Due to these often conflicting aims, even after addressing bias concerns, accurate interpretation of rankings is often challenging.

While optimal ranking may not be able to meet all three of these inferential goals, we can improve the ease of interpretation and communication. Ranking emphasizes ordering of units. In contrast, clustering emphasizes natural groupings of units. Together, they may be able to clarify communication. Clusters complement rankings by clarifying the confusion about spacing that rankings can create: clusters highlight the non-uniform spacing present in most real-world rankings (Jacques *et al.*, 2014).

Mixture model clustering is a particularly appealing choice because it allows for high levels of flexibility to adapt to the problem at hand. While parametric mixture models are commonly used, this technique can be extended (Gelfand *et al.*, 2005; Chung and Dunson, 2009; Lee and Kim, 2020). Nonparametric approaches allow for high flexibility to learn from the data with fewer assumptions than parametric versions (Laird, 1978).

In summary, while rankings are an important tool in communicating health research, often the message cannot be clearly conveyed with a ranking alone. Clustering can compliment a ranking by clarifying the relative position of items and maintaining the easy, appealing interpretability of a ranking. Together, ranking and clustering provide a strong inferential and communication tool. In this work, we seek to build

on recent innovations in Bayesian ranking methods by combining these methods with the interpretability of clustering. We hope that this combination will give analysts an interpretable, precise tool to accomplish their multiple inference goals for small area health ranking.

1.2 METHODS

1.2.1 RANKING A SET OF SMALL AREA ESTIMATES

In this section, we describe our method for estimating ranking of small area estimates. Generally, to estimate ranks for N item, we first choose a parameter of interest. This measure v_j is indexed for item by $j = 1,2,...,N$ and estimated by the data for each item j. We assume that the distribution of data for each unit is known, but do not require independence between these units. We will draw samples from the Bayesian joint posterior distribution of these measures $v_1,...,v_j$. When we order these measures, we will refer to them as $v_{(j)}$.

In later real data analysis, we'll explore how the percent (p) of infants born at a low birth weight compares across counties within a state. We can assume that this measure could be binomially-distributed, with n births within each county and p percent of infants born at a low birth weight (LBW). Throughout this section, we'll use this binomial count data example to make our explanations more concrete.

We create the joint posterior by selecting a sensible prior and combining it with out data to create a posterior distribution. I'll make this more concrete with LBW data from N counties with y infants born at a low birth weight and n total births. In this case, we choose a Beta prior with parameters $\alpha_p = y$ and $\beta_p = (n-y)$. We assume each of the county's count data y are drawn from a Binomial distribution with n births and probability of low birth weight p. The posterior is then a Beta distribution with $\alpha = \alpha_p + y = 2y$ and $\beta = \beta_p + (n-y) = 2(n-y)$. This will be a proper posterior as long as y and $(n-y)$ are both non-zero.

We then draw 10,000 samples from this posterior distribution for each county and, for each of these sample, calculate the county's rank. This creates an N by 10,0000 matrix of ranks, where N is the number of counties. From this matrix of ranks, we can calculate the mean rank for each county. This gives us the individually optimal rank. However we are interested in a jointly optimal ranking. To get a jointly optimal ranking, we need to calculate the loss associated with assigning each rank to each county.

Loss Functions for Ranking

Loss functions allow us to consider the variance in the joint posterior distribution in a principled and mathematically optimal way. To assign ranks to items, we use a loss function to handle overlapping distributions and varying levels of uncertainty inherent in small area estimates, as described by Athens *et al.* (2013) and Jewett *et al.* (2019). To assign a rank to an item, we use the loss function to calculate the distance between the assigned rank and the optimal rank.

We estimate this optimal rank by sampling from each item's posterior distribution. Then, we can calculate the loss associated with assigning estimated rank position $\hat{k}$ to item j generally as

$$loss(j,k) = |\hat{\theta}_j - \hat{\theta}_{(\hat{k}_j)}|^\rho$$

where $\hat{\theta}_j$ is the estimated rank for item j, $\hat{\theta}_{\hat{k}_j}$ is the optimal rank for item j, and ρ gives the type of loss. For example, $\rho = 1$ for absolute value loss and $\rho = 2$ for square error loss.

By minimizing the loss for each item (or county, in our case), we can estimate loss-optimal ranks for each item individually. The individually-optimal rank for county j is the rank k that minimizes the individual risk for each county. This method can create an incomplete ranking: when multiple items have the same rank, some ranks will necessarily be skipped, so the ranking will be incomplete. To resolve these conflicts and guarantee a complete ranking, we instead calculate the loss over all N items. To guarantee a complete ranking of ranks $k_1, k_2, ..., k_N$ to items $j_1, j_2, ..., j_N$, we minimize the joint posterior loss:

$$loss_u(j,k) = \sum_{j=1}^{N} |\hat{\theta}_j - \hat{\theta}_{(\hat{k}_j)}|^\rho$$

In the case of binomially-distributed count data with percent parameter p_j:

$$loss_u(j,k) = \sum_{j=1}^{N} |\hat{p}_j - \hat{p}_{(\hat{k}_j)}|^\rho$$

We then find the risk, the mean loss over all posterior samples. For our binomial count data by county, we

would calculate this risk as:

$$R(j,k) = \mathbb{E}\sum_{j=1}^{N} |\hat{p}_j - \hat{p}_{(\hat{k}_j)}|^{\rho}$$

Using this posterior distribution, we obtain an N by N matrix of expected losses for all possible rankings of the N items.

To get a jointly optimal ranking for the binomial example, we calculate the mean square error loss distance between each possible rank and each county's assigned rank in the 10,000 samples. This provides a estimate of the loss incurred by assigning each rank to each county.

To construct a complete ranking from this matrix of losses, we need to minimize the overall loss to assign a rank to each county. In 1955, Kuhn proposed the Hungarian algorithm to solve the problem of assigning multiple items to a set of spots to minimize loss (Kuhn, 1955). We apply Kuhn's solution using the clue package implementation (Hornik, 2005, 2019). This produces a complete ranking, minimizing the loss across all counties.

1.2.2 Nonparametric Empirical Bayes Mixture Model for Clusters

To aid in interpretation of the rankings, we estimate a nonparametric empirical Bayes mixture model to create clusters of ranks. In constrast to the Beta-distributed priors used in the ranking sampling, the clusters are not assumed to come from a known distribution. Instead, they are centered on nonparameteric empirically-derived priors, γ_l. Each cluster, indexed by l, is centered on a support point, θ_l. The nonparametric priors for each cluster are

$$Pr(p_j = \theta_l) = \gamma_l$$

where $l = 1, 2, ..., m \leq N$, θ_l is the center of the cluster, and m is the number of clusters.

We find the nonparametric maximum likelihood (NPML) for the number of mixture components m, support points $\theta_1, \theta_2, ..., \theta_m$ and prior probabilities $\gamma_1, \gamma_2, ..., \gamma_m$ using the EM algorithm.

To make this more concrete, we present the case of clustering binomially-distributed count data from counties within a state, as in the ranking method description. As in the ranking, the likelihood for each of

ranked county's count y_j is estimated as

$$y_j \sim \text{Binomial}(n_j, p_j), j = 1, 2, ..., N$$

where n_j is the total number of births and p_j is probability of low birth weight for each ranked county.

Estimating Clusters

We start with an analyst-defined number of clusters. Each of these clusters has a point masses centered on θ_l, spaced evenly across the range of the measure. Each of these clusters is assigned equal probability ($\frac{1}{m}$ where m is the number of clusters).

Then, we alternate between the expectation and maximization steps for a set number of iterations. During the E step, we estimate the probability that each rank fits into each of the m clusters. In our binomial example, for each of the clusters, that probability is

$$P(\theta_l) * \binom{n_j}{y_j} * \theta_l^{y_j} * (1-\theta_l)^{n_j - y_j}$$

where $P(\theta_l)$ is the point mass for each cluster, y_j is the observed number of events in each ranked county, and n_j is the population of each ranked county.

During the M step, we reestimate the optimal θ_l for each cluster by calculating the mean likelihood over iterations for each ranked county fitting within a cluster. This mean is then multiplied by the observed, empirical probability for each ranked county ($\frac{y_j}{n_j}$).

After repeating the EM steps for a set number of iterations, we reduce the number of clusters by looking for duplicates to a specified number of significant digits. With fewer significant digits, close clusters are collapsed, while more significant digits allows close clusters to remain distinct. For example, with two significant digits, if two clusters are estimated at 0.162 and 0.235, they would be collapsed into one cluster centered at 0.20. With three significant digits, these would remain two separate clusters at 0.16 and 0.24. With more clusters, ranks fit their clusters more closely, but a high number of clusters reduces their usefulness. In the most extreme case, if the number of clusters is equal to the number of items, clustering will not add any useful information. Keeping the number of clusters small increases the usefulness of the

clustering.

After removing duplicates, we calculate the final posterior probabilities that each ranked county appears in each cluster. To reduce the possibility of under or overflow, we normalize the probability, then return the list of cluster centers, θ_l, and posterior distribution of θ assignments for the items, p_θ. Finally, we assign each rank to its individually-optimal cluster under square error loss.

In this section, we described how we estimate clusters for small area estimates using a Bayesian non-parametric mixture model. In the next section, we'll discuss how we combine this clustering with the ranking.

1.2.3 Adding Clustering to a Ranking

In our ranking formulation, we first optimize an item's ranks, then assign these ranks to clusters (Option One in Figure 1.1). We choose to use this approach because our collaborators at County Health Rankings prioritize rankings, seeing clusters only as a supplementary interpretation tool. Assigning ranks to clusters, as in option one, ensures that clusters should compliment, not conflict with, the ranking. Because the ranking is the priority, this is the best option for our application.

In an alternative formulation, one could assign items to ranks, then independently assign these items to a cluster (see Option Two in Figure 1.1). Option two might be appropriate for a setting in which we wish to interpret clusters and ranks independently. However, in our setting, we hope clusters serve as a useful complement to ranking, so option one is most appropriate.

1.2.4 Adding Weighting to the Ranking: Weighted Loss Ranking

In many cases, an analyst is primarily interested in certain ranks: Which small areas rank highest or lowest? We might also want to weight by these areas: Can we reduce the uncertainty of our ranking by weighting counties on their inverse variance? If might also care particularly about the rank position of more populous counties or target areas. To accommodate these goals, we propose two types of weighted loss: on the rank positions and on the items (in our case, counties).

$$R(j,k) = \mathbb{E} \sum_{j=1}^{N} w_{\hat{k}_j} w_j |\hat{\theta}_j - \hat{\theta}_{(\hat{k}_j)}|^2$$

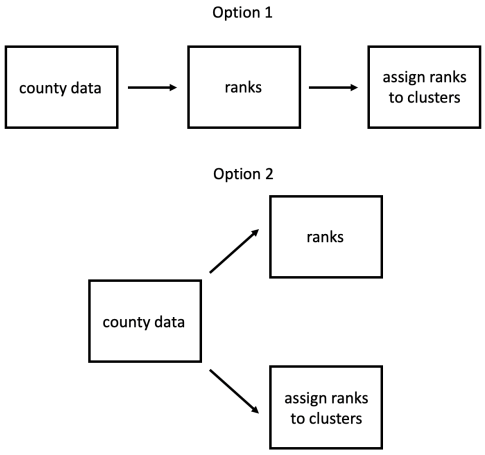

Figure 1.1: Clustering and Ranking. Option one assigns small areas to rank, then assigns these ranks to clusters. Finally, it assigns small areas to clusters by their ranks. Option two assigns small areas to rank and, independently assigns small areas to clusters.

For the binomial case:

$$R(j,k) = \mathbb{E} \sum_{j=1}^{N} w_{\hat{k}_j} w_j |\hat{p}_j - \hat{p}_{(\hat{k}_j)}|^2$$

Weighting on Rank Position

Often health ranking creators informally focus their ranking interpretation on the highest and lowest counties. They may want to target interventions or program funding toward counties with lower health indicators. Similarly, they may want to study the programs that are working well in healthier counties. To make these priorities explicit, we can weight the highest and lowest ranks most highly, then gradually decrease this weight as we move toward middle ranks. With this weighting, incorrect assignment of the highest and lowest ranks will contribute more to the loss, placing higher importance on the extreme ranks of interest.

For the binomial case, the weighted loss ranking with weight w_k for each rank is:

$$L_u(j,k) = \sum_{j=1}^{N} w_{\hat{k}_j} w_j |p_j - p_{(\hat{k}_j)}|^\rho$$

We then calculate the posterior expected risk over this weighted loss function using samples from a

posterior distribution. Generally,

$$R(j,k) = \mathbb{E} \sum_{j=1}^{N} w_{\hat{k}_j} w_j |\hat{\theta}_j - \hat{\theta}_{(\hat{k}_j)}|^{\rho}$$

For the binomial case, we use the proportion p for each county, j:

$$R(j,k) = \mathbb{E} \sum_{j=1}^{N} w_{\hat{k}_j} w_j |p_j - p_{(\hat{k}_j)}|^{\rho}$$

Creating a Partial Ranking with Weights

This framework could also be used to create a partial ranking. We could assign a weight of one to the ranks of interest and zero to all others. If an analyst cares about ranks 1 through n_k, we would assign a weight of 1 to ranks 1 through n_k and a weight of zero to ranks $n_k + 1$ to N. This assigns a weight of one to high priority ranks and zero weight to low priority ranks, giving us a partial ranking for only our focus ranks. This creates a top n_k ranking, giving meaningful ranks for the only top n_k items and leaving the rest unranked. In many appplied cases, as in the case of our application, county health rankings, a partial ranking will be unacceptable, so we proposed a more gradual weighting in the next section.

Gradual Weighted Loss Ranking

To weight by rank position while maintaining a complete list, we propose a gradual weighted loss ranking for small area estimates.

A gradual weighted loss ranking would allow analysts to prioritize the highest rank positions (or lowest), while maintaining a complete ranking. This family of weights w_k is based on a steepness criterion ϵ, where $0 < \epsilon < 1$. Depending on the ranking's purpose, this steepness criterion can be adjusted to prioritize extreme ranks more or less aggressively. We can prioritize top ranks, bottom ranks, or both symmetrically.

To prioritize top ranks $(1, 2, 3, ...)$, we suggest assigning w_k as

$$w_k = (1 - \epsilon)^{k-1}$$

To prioritize the lowest ranks, we suggest assigning a weight w_k to each rank as

$$w_k = (1-\epsilon)^{N-k}$$

To prioritize both ends symmetrically, assign a weight w_k to each rank as

$$w_k = \max\{(1-e)^{k-1}, (1-e)^{N-k}\}$$

The steepness of the weighting is controlled by the choice of ϵ. Applying a moderate ϵ in the above formulas leads to prioritizing the lowest or highest ranks, as desired. A more extreme ϵ moves closer to familiar, unweighted loss cases. For example, as ϵ goes to 1, the gradual weighted loss becomes a zero-one weighting. When ϵ is zero, the ranking is fully unweighted.

Inverse Variance Weighting on Small Area

To handle heterogeneity of variance from county to county, we also propose to weight items by variance. As discussed in the introduction, bias toward high variance items is a challenge in ranking. High certainty items are often pushed from their optimal rank in favor of a high variance items. In general, it would be ideal for rankings to place well-inferred items in their correct rank, letting items with lower certainty make the greater compromises. Weighting by variance could move us closer to this goal. These weights, w_{var}, are calculated as the inverse of an item's posterior variance

$$w_{var_j} = \frac{1}{var(\hat{\theta}_j)}$$

By weighting each item's estimate by the inverse of its variance, we hope to reduce the bias toward high variance items in our ranking.

1.3 Real Data Examples

To explore how our ranking and clustering method behaves in real data, we show rankings of counties for three states: Connecticut, Arizona, and Wisconsin. We first explore how our ranking and clustering method behaves alone using a small state with eight counties. Then, we explore how this method behaves

in a larger state, Arizona, which has 15 counties. Last, we explore how our ranking and clustering method behaves when combined with inverse variance weighting and gradual rank weighting in a larger state, Wisconsin, which has 71 counties.

In each of these examples, we examine a measure of infant health: the proportion of infants born at a low birth weight (LBW). Because low birth weight can be an indicator or risk factor for other health conditions, counties aim for a small percentage of infants born at a low birth weight. A higher percentage of infants born at a low birth weight indicates higher risk of illness among newborns in the county. In this way, low birth weight is often used a proxy for infant health.

For each county, we use publicly available data for the proportion of infants born at a low birth weight. We estimate each county's sample proportion p, assuming that each birth in a county, j, is an independent sample from the jth county, where low birth weights are distributed binomially with parameters (n_j, p_j).

1.3.1 Connecticut: Unweighted Clustering and Ranking for Eight Counties

To illustrate the ranking and clustering method concretely, we rank Connecticut's eight counties by their percent low birth weight. We used data from 2012 to 2017. During those six years, each of these counties ranged widely in population size and, accordingly, in the number of births. In Windham, the smallest county, there were 8,277 births. Fairfield county, the closest to New York City area and containing the state's largest city, had the highest number, with 73,350 births. The proportion of infants born at a low birth weight during those six years ranged from 6.1 to 8.6.

The following table shows raw data for counties in Connecticut. They are ranked by the percentage of infants born at a low birth weight between 2012 and 2017. The last column shows their assigned cluster.

This clusters adds context to the ranking. The clustering shows that the county with the lowest LBW percentage, Middlesex, is distinct from those that follow it: at 6.1%, it is grouped into its own cluster. The three counties that follow, at ranks 2, 3, and 4, hover near 7% and are grouped into cluster two. The next two are grouped into a third cluster, as they both have a LBW percentage around 7.5%. The last two counties, with LBW percentages 8.2% and 8.6% are separated into two distinct clusters. This clusters gives

a sense of how the counties are spaced within the context of their ranking.

rank	county	infants at LBW	births	percent LBW	cluster
1	Middlesex	628	10296	6.099	1
2	Tolland	582	8321	6.994	2
3	Litchfield	745	10650	6.995	2
4	Windham	601	8277	7.261	2
5	New London	1454	19610	7.415	3
6	Fairfield	5587	73350	7.617	3
7	New Haven	5306	64670	8.205	4
8	Hartford	5806	67908	8.550	5

Table 1.1: Connecticut Counties Ranked by Low Birth Weight, lowest to highest rate

.

Because the sample size varies greatly between counties, the levels of uncertainty in our estimates vary greatly as well. To give a sense of the uncertainty in each county's rank assignment (by square error loss), see Figure 1.2.

The first ranked county, Middlesex, is ranked with a high level of certainty: it is ranked in the first position almost 100% of the time. Similarly, New Haven and Hartford are ranked in the 7th and 8th positions, respectively, in almost all samples. Conversely, counties ranked in the middle positions are often interchanged: their rankings are much more unclear. The picture shown in Figure 1.2 aligns with the cluster results shown in the table above. Middle ranks show overlap and, indeed, are grouped together: ranks 2, 3, 4, 5, and 6 are grouped in only two clusters. As desired, the addition of clustering here adds nuance to the interpretation of the county ranking, underscoring that there many not be large differences between the middle ranked counties.

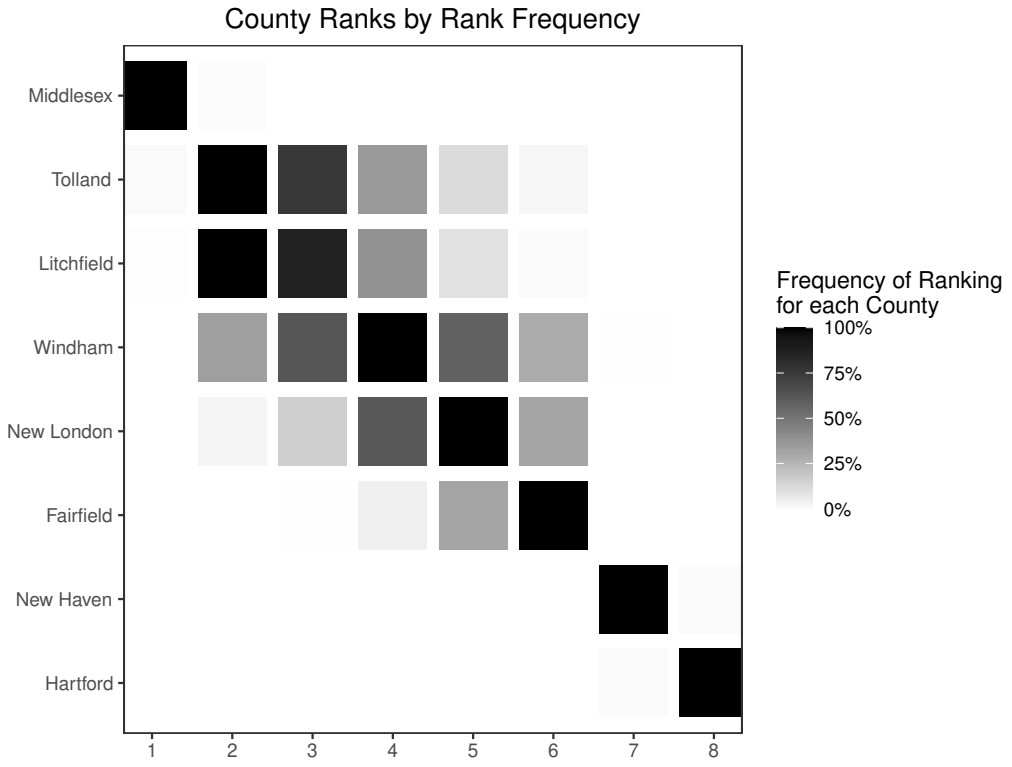

Figure 1.2: Frequency of Connecticut County Ranks

1.3.2 Arizona: Clustering and Ranking 15 Counties by Two Methods

In this second real data example, we show how our ranking and clustering method performs in a slightly larger state, with 15 counties. In Figure 1.3, Arizona's counties are displayed in optimal rank order by percent LBW. From these credible intervals, we can see that, as in the previous example, there are varying levels of uncertainty among counties.

In this example, we'll focus on the process of assigning clusters in order to clarify this process. To assign clusters according to our method, we estimate the empirical Bayes posterior across clusters for each order statistic (Figure 1.4). From these distributions, we can see that, with high certainty, the county ranked at position one should be placed into cluster one. With slightly less confidence, we can place county two into this cluster as well. More uncertainty remains about which counties should be placed in clusters 2, 3, and 4. There is stronger indication that counties 11, 12, 13, and 14 should be placed into cluster 5 and county 15 should be placed into cluster 6.

Through the ranking, counties are linked to their optimal order statistics, so they now take their order

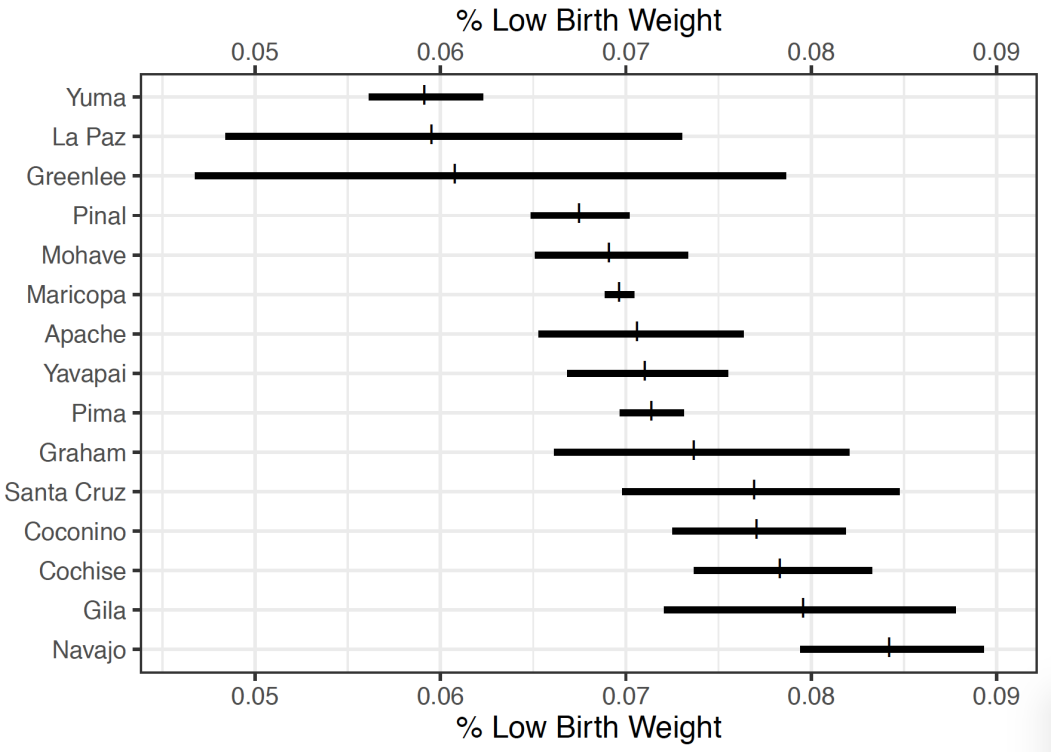

Figure 1.3: Arizona's Counties in Optimal Rank Order by Percent Low Birth Weight

statistic's optimal cluster. Figure 1.5 shows joint ranking and clustering for Arizona counties, with credible intervals. Cluster assignment is emphasized by color in this figure. The size of the dot for each county indicates the certainty of the rank assignment.

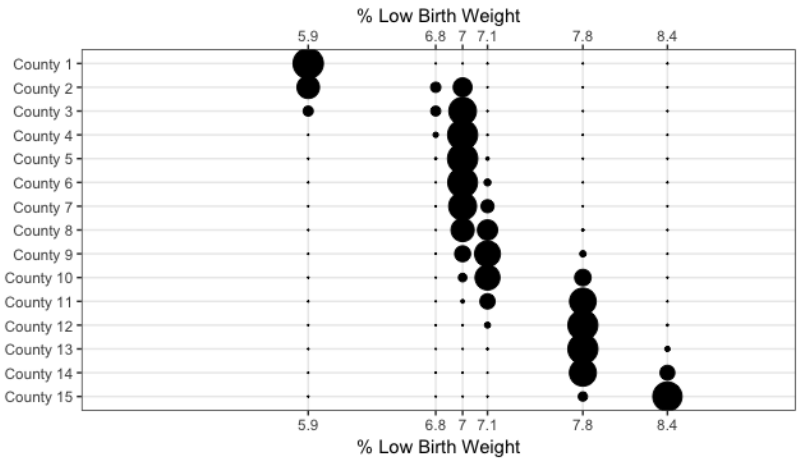

Figure 1.4: Empirical Bayes posterior distributions for order statistic

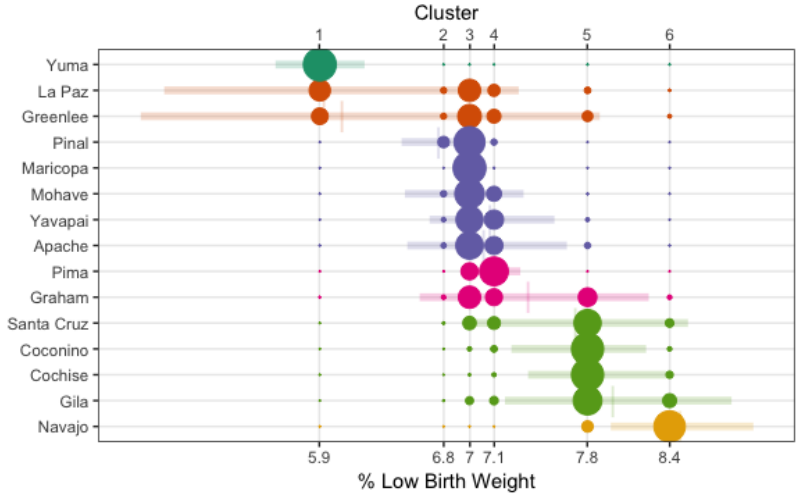

Figure 1.5: Ranked Arizona counties in clusters with credible intervals

Exploring an Alternative Method: Separate Ranking and Clustering Assignment

We also show how the second clustering methods described in Figure 1.1 would affect results in real data. In the second option, we would assign ranks and clusters independently. To illustrate this version of the method, we estimate the posterior distributions of cluster assignments for each county (Figure 1.6). If we were assigning cluster assignments to the counties separately from their ranks, we would assign them according to these distributions. Some counties have clear cluster assignments (Yuma county), while others, like Graham County, have posterior distributions spread across multiple clusters.

In contrast, our method of assigning clusters to order statistics produces much less uncertainty in this ex-

ample, as shown in Figure 1.7. Assigning clusters to counties directly creates more uncertainty, potentially complicating the interpretation. The intention behind adding clusters is to aid in interpretation, so we choose to assign clusters to ranks, rather than to counties directly.

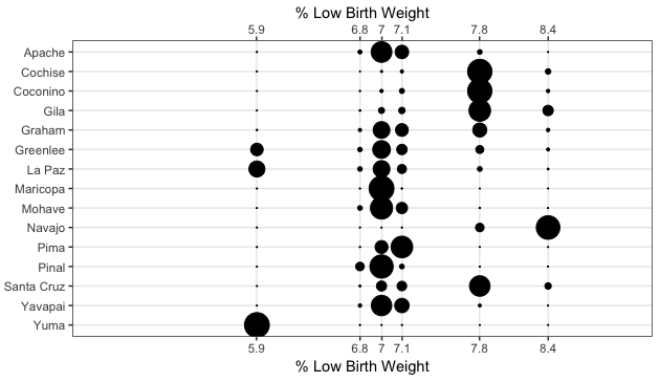

Figure 1.6: Cluster Assignments for each Arizona county

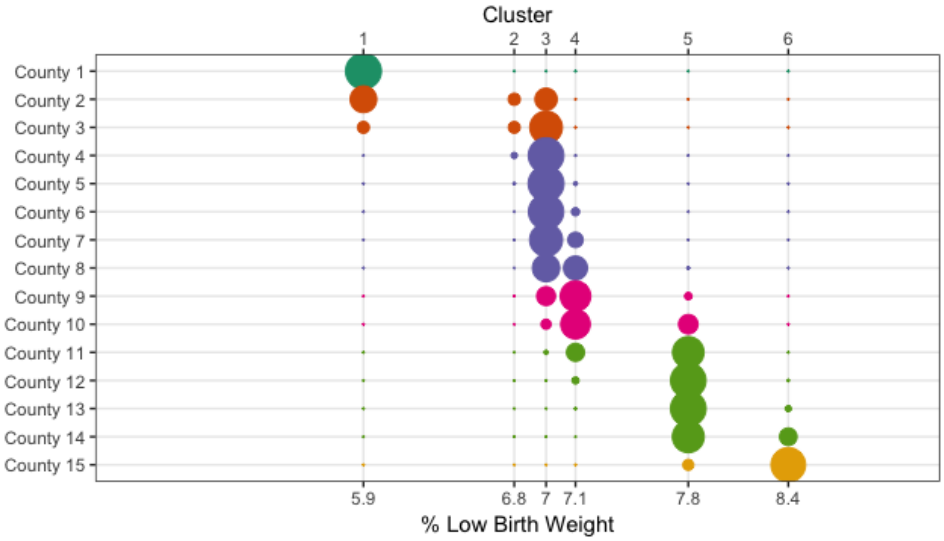

Figure 1.7: Cluster assignments for each order statistic

1.3.3 Wisconsin: Weighted Ranking for 71 Counties

Next, we explore how our weighting ideas perform in the context of ranking. Specifically, we are interested in how gradual weighting on ranks and inverse variance county weighting methods might perform for a larger state with high variability in sample size between counties. We apply our weighted loss ranking method to County Health Ranking's estimates of low birth weight prevalence in Wisconsin's counties, first in an unweighted way, then with weights.

Like Connecticut and Arizona, Wisconsin counties vary drastically in population size so there is a high level of heterogeneity among counties. In contrast to Connecticut and Arizona, Wisconsin has a large number of counties: 72 in all. The posterior distribution of ranks in our sample (Figure 1.8) displays this clearly: some counties have relatively certain ranks (e.g. Milwaukee), while some counties have so much uncertainty that their distributions give very little information about their rank (e.g. Green Lake).

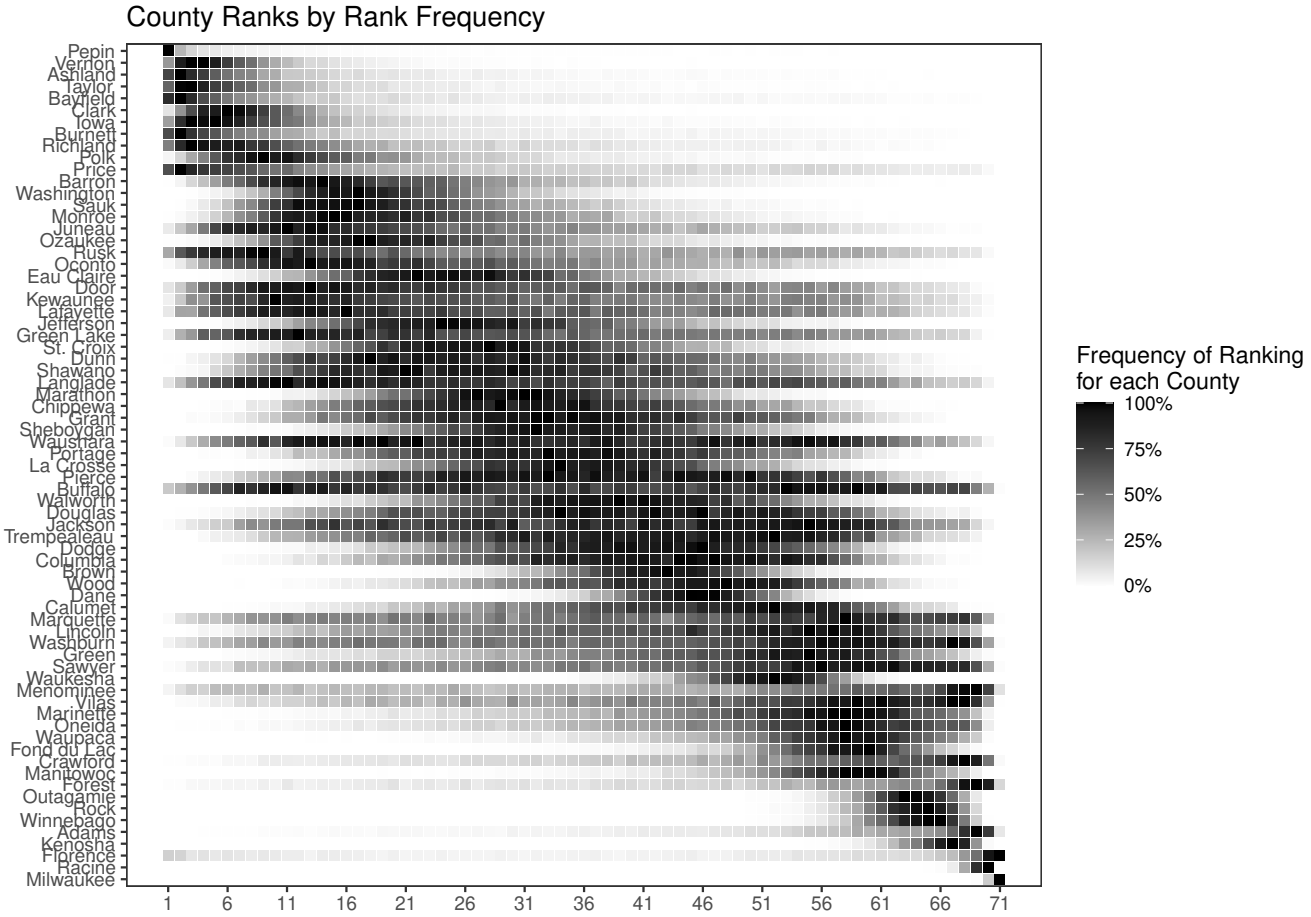

Figure 1.8: Distributions of Ranks for Wisconsin Counties (unweighted)

To account for this uncertainty, we may want to prioritize correct rankings for counties with high certainty. Table 2 shows the distribution of ranks for the top five highest counties for LBW when each county is weighted by its inverse variance. This changes the rankings, especially in places where two counties were in conflict over a rank position. In these cases, weighting counties by inverse variance prioritizes correct rank assignment for low variability counties.

Table 3 shows the distribution of ranks when counties are weighted by their inverse variances and ranks

rank	county	percent LBW
71	Milwaukee	9.37
70	Racine	8.27
69	Kenosha	7.56
68	Winnebago	7.39
67	Rock	7.36

Table 1.2: Wisconsin County Ranked by Low Birth Weight. Counties are weighted by their inverse variance. Ranks are unweighted.

rank	county	percent LBW
71	Dane	6.48
70	Milwaukee	9.37
69	Brown	6.39
68	La Crosse	6.22
67	Racine	8.27

Table 1.3: Wisconsin County Ranked by Low Birth Weight. Counties are weighted by their inverse variance. Top ranks are weighted more highly, with weights gradually decreasing in lower ranks ($\epsilon = 0.9$).

are weighted gradually with $\epsilon = 0.9$. Ideally, weighting both ranks and counties allows analysts to make their priorities about ranks explicit and to control small area estimate heterogeneity. In this case, gradual rank weighting combined with inverse variance weighting places low uncertainty counties at the top of the list much more than would be desired. While it is useful to prioritize high certainty counties, we do not want counties with high certainty, but low LBW to appear in the top five, as in the case of Dane, Brown, and La Crosse counties here. This behavior may indicate that further tuning of ϵ parameter is needed. It could also indicate unpredictable behavior when these two types of weighting are combined.

This is a small case study of inverse variance and gradual weighting on rankings: there is still much work to be done to clarify how these methods behave more broadly. Further exploration of how these multiple weightings work together, as well as tools to choose appropriate ϵ weight values would be particularly useful. To that end, i the next section, we present a small exploration of our ranking method's behavior with various ϵ weights.

1.4 Exploring Appropriate Values for Gradual Rank Weighting: A Simulation Study

In this section, we explore the behavior of various choices of ϵ weight in a small set of simulations. We sample from a simplified but realistic model, based on our real data example, with a moderate numbers of items, ranked on a binomially-distributed measure. As in the real data, we include areas with large variations in sample size, leading to heteroskedasticity similar to that seen in our real examples.

In this experiment, we rank 50 items on binomially-distributed data. For each item, data are drawn from a binomial distribution. Each item's true percentage p varies from 0 to 1 and its sample size varies from 100 to 1000.

To estimate the accuracy of the ranking, we use two metrics. In the first, we estimate the percent of true top N ranked within the top N spots. This first metric measures the percentage of true top N_k items appearing in the method's top N_k ranks.

$$\text{Percent Top } N_k \text{ Ranking Quality Metric} = \frac{\text{true top } N_k \text{ items ranked in top N}}{N}$$

In the second metric, we measure how many units in the top N of the ranking are ranked perfectly. This second metric measures the percentage of true top N items at their correct rank.

$$\text{Strict Top } N_k \text{ Ranking Quality Metric} = \frac{\text{true top } N_k \text{ items ranked at their correct ranks}}{N}$$

We chose to focus on the accuracy of the top part of the ranking because that is the focus on our weighting scheme. However, we do acknowledge that this is only one way to measure the accuracy of a ranking and perhaps only one component of accuracy. Still, these two metrics allow us to measure if a particular choice of ϵ weight achieves our goal: high accuracy top ranks while maintaining reasonable ranking within the higher part of the list.

We evaluate accuracy for varying choice of ϵ, from no weighting ($\epsilon = 0$) to aggressive weighting ($\epsilon = 0.99$). In addition to exploring ϵ weight choices, we also define the top of the list in varying way, from a Top 1

(N_1) to a Top 20 (N_20) ranking. For all cases, we evaluate the ranking on both metrics: the top N_k ranking metric and strict top N_k metric.

1.4.1 Simulation Results

Results in Figure 1.9 show the percentage of true top N_k items that appeared in our method's ranked top N_k, where N_k again varies from 1 to 20. The colors indicate the number of items included in the top count, with gray ($N_k = 1$) and blue ($N_k = 20$). By this more relaxed metric, all top N_k rankings do well, with mean top N_k in ranked top N_k over 96% for all N_k greater than 20. It appears that smaller values of ϵ perform better by this metric, with performance slipping as epsilon increases to 0.9, especially for the top 20 items in the ranking.

The more strict metric indicates the same trend: lower values of ϵ result in higher accuracy. Figure 1.10 shows the percentage of true top N items that were corrected ranked, where N_k varies from 1 to 20. The ranking's accuracy is high for the top 1 and 3 items, but reduced significantly for larger N_ks. This reduction becomes more dramatic as ϵ increases.

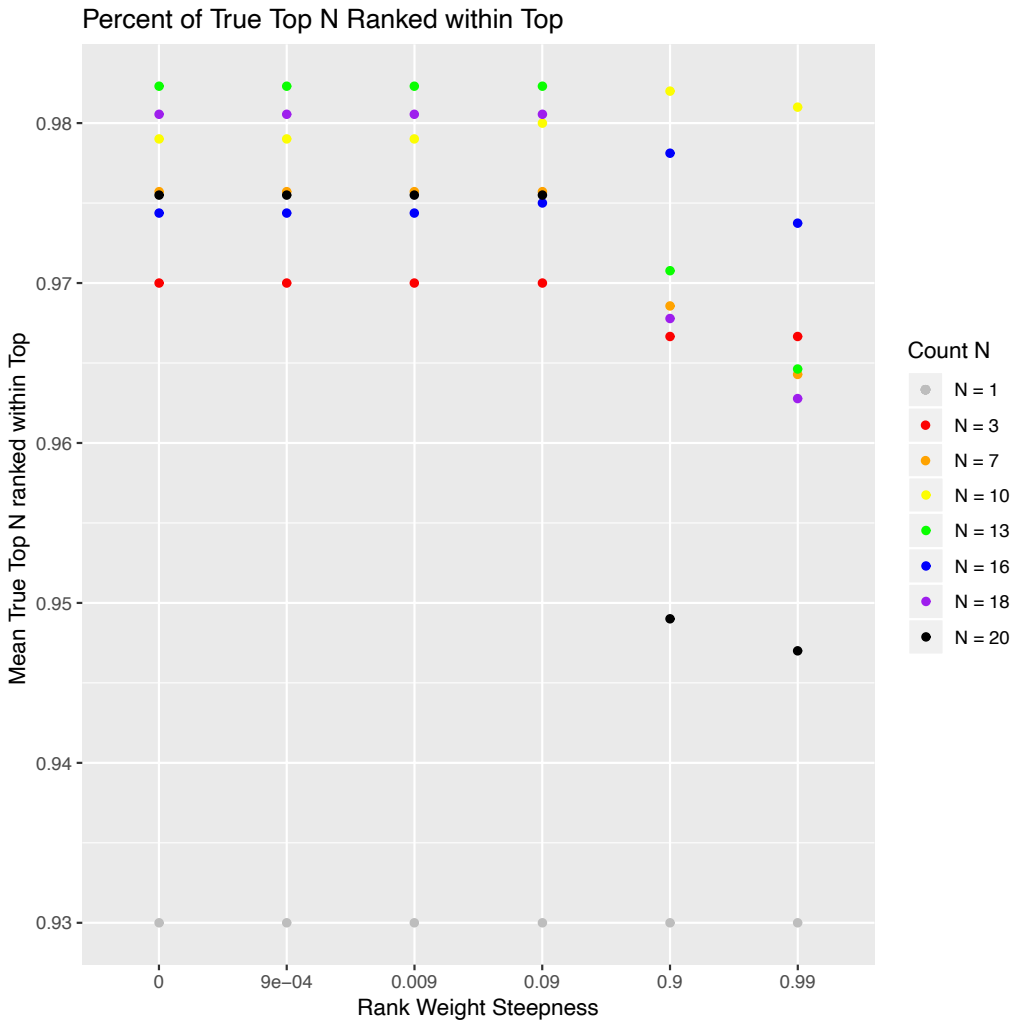

Figure 1.9: Percent True Top N_k in Ranked Top N_k for Varying Weight Steepness ϵ

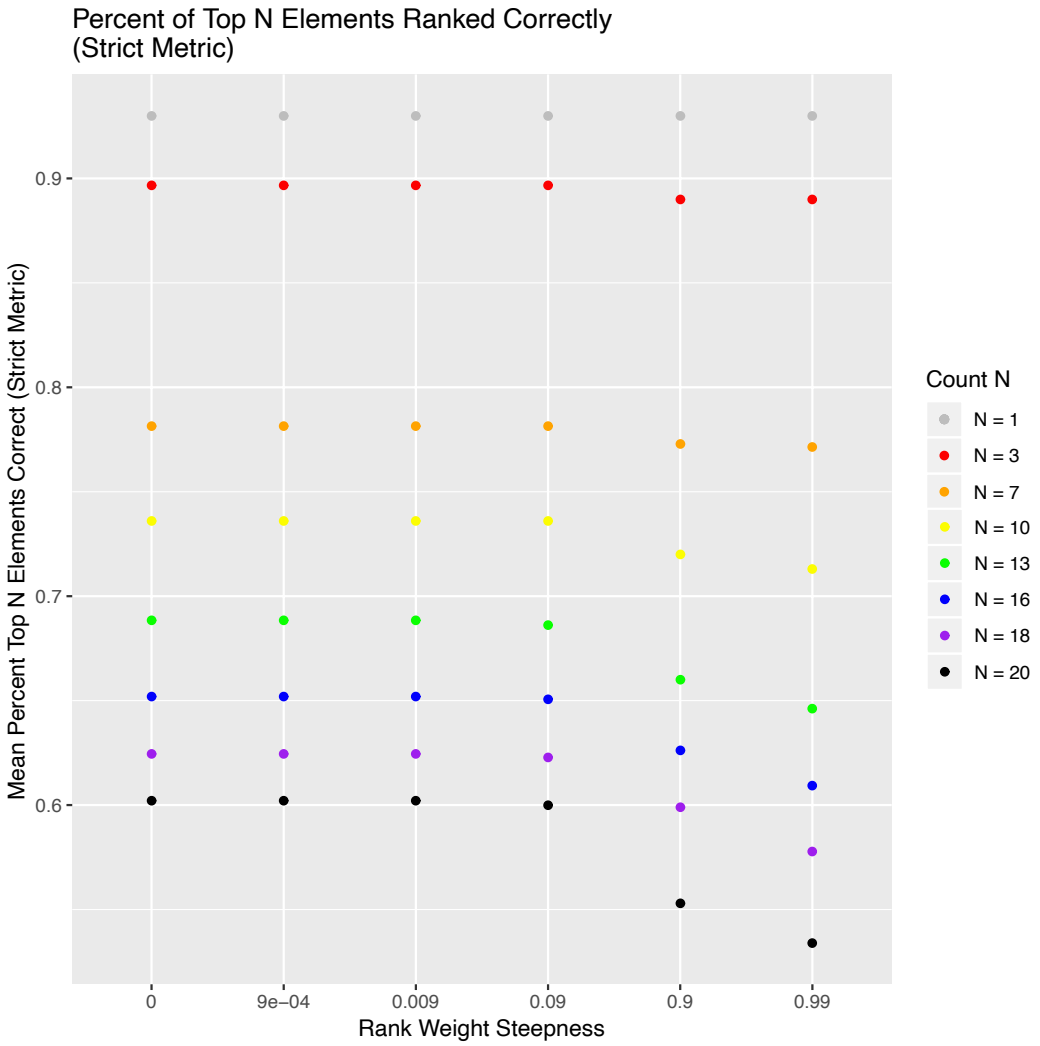

Figure 1.10: Percentage of True Top N_k Items Corrected Ranked for Varying Weight Steepness ϵ

1.5 Summary and Suggestions for Future Work

We present a nonparametric empirical Bayes mixture model method for simultaneous clustering and ranking of small area health estimates. For our clustering and ranking strategy, we present illustrating examples for counties within Connecticut and Arizona. We also present some weighting methods that may help analysts create rankings that better align with their priorities around uncertainty. We explore how these weighting methods behave using real data for counties within the state of Wisconsin. We also explore the behavior of gradual weighting in the ranking setting in a simulation study.

The question of how to handle uncertainty within a ranking is a challenging problem. Here, we explore two ideas to address this. First, we describe a clustering method that aims to clarify a ranking's interpretation. We hope the addition of clusters can emphasize the varying spacing within most real-world rankings. Here we have presented two case studies showing how this might be applied in count-based healthcare outcomes for two states. However, more work is needed to explore how it might behave in other types of data.

Second, we explore how weighting might control the uncertainty faced when ranking small areas on health measures. We present results showing mixed accuracy in a real-world case study and a simulation study. With these mixed accuracy results, many questions about the appropriate use of weights remain. If gradual weight ranking were to be pursued, we would need to know what values of epsilon might be appropriate. Our simulation study suggests that smaller values of ϵ might lead to highest accuracy. This presents additional questions for study. First, would this weighting ranking behave better than an unweighted ranking and, second, do the benefits of these rankings outweigh the potential cost of placing high certainty-low estimate items at the top of the list? In our case study with real data in the state of Wisconsin and simulation study, it is unclear when this weighing might be more effectively deployed. More work is needed to understand when it might be effectively used to handle varying uncertainty when ranking.

Differences in variability between items in a ranking is a major challenge in developing quality ranking methods (Shen and Louis, 1998; Henderson and Newton, 2016; Athens *et al.*, 2013; Jewett *et al.*, 2019). In this work, we explore a clustering method to clarifying interpretation of rankings with uncertainty and uneven spacing and two weighting ideas to handle uncertainty, as well as interesting questions for future

work.

Acknowledgements. Thank you to my dissertation committee for their helpful suggestions on this chapter. This work was supported by the National Institutes of Health award T32 HL 83806-10.

CHAPTER 2

MODEL FOR DNA EVOLUTION ON PHYLOGENETIC NETWORKS

Abstract In this chapter, I introduce a graphical model for estimating parameterized semi-directed phylogenetic networks. I present the model in two ways: first, from a mathematical perspective, then focusing on the biological interpretation. I discuss statistical and computational challenges in calculating the network's likelihood, then present our solutions. Bringing the model and computational tools together, I present our methods for estimating a phylogenetic network's numerical parameters: branch lengths, inheritance values, and evolutionary rates. Finally, I demonstrate the method on real data by estimating evolutionary rates for a set of 23 snake species.

Keywords speciation; gene flow; rate variation; hybridization; phylogenetic networks

Contribution I implemented the Markov and rate variation models. I conducted the snake data analyses, visualized results, and wrote the first draft. Cécile Ané and I serve as second and first authors, respectively.

2.1 INTRODUCTION

Phylogenetic trees have been useful tools for understanding evolutionary history for more than 150 years. Though a tree structure can convey much about current species and their ancestors, it also misses some important aspects of these relationships. Though much of heredity is shared vertically, from parent to offspring within a species line, some genetic information is passed between species (Mallet *et al.*, 2016; Mindell, 2013). In eukaryotes, this genetic sharing between species commonly happens through hybridization. In fact, about 10% of animals and 25% of plants can hybridize with other species (Mallet, 2007). In Archaea and Bacteria, this occurs most commonly through horizontal gene transfer (also known as lateral gene transfer) (Fontaine *et al.*, 2015; Cui *et al.*, 2013; Ochman *et al.*, 2000; Burmeister, 2015; Syvanen, 1994). (Horizontal gene transfer is an important though rare occurrence in eukaryote history as well (Keeling and Palmer, 2008).)

Ignoring gene flow between species, whatever its mechanism, can lead us to miss these important events in species' evolutionary history and often leads to inaccurate estimates. Even if scientists are mainly interested in vertical inheritance, ignoring these events has been shown to lead to inaccurate tree reconstruction (Leache *et al.*, 2014; Solís-Lemus *et al.*, 2016). In these cases, ignoring hybrid events can create non-meaningful taxonomic groupings, blurring species' true ancestral relationships. Extending phylogenetic tree models to include these horizontal events is crucial to fully understanding evolutionary history.

Over the past two decades, many new methods have extended evolutionary reconstruction methods to include these horizontal events. Generally, these events are included as reticulation edges between lineages within a phylogenetic tree. In a typical phylogenetic tree, each node can have at most one parent. Reticulation nodes break that rule: they have two parents edges. These reticulate parent edges point to this hybrid child node, indicating that it received genetic information from two parental lineages, rather than one.

In the following section, I describe the most popular reconstruction methods, from pattern frequency to Bayesian and maximum likelihood approaches. As I summarize these methods, I will describe gene tree approaches before introducing site-based likelihood methods in more detail.

2.1.1 Existing Reconstruction Methods

While existing phylogenetic network reconstruction methods vary, they all estimate the network topology: the way edges and nodes are connected. In addition, some also estimate branch lengths, the percentage of inheritance flowing through each reticulate parent into a hybrid, and rates of evolutionary change.

Examining frequencies of patterns in sequence data to test for reticulation events can be a powerful method for topology estimation. While these methods are fast, they rely on simplifying assumptions that cannot be met in realistic biological situations (Green *et al.*, 2010; Felsenstein, 2004; Blischak *et al.*, 2018). For more accurate network reconstruction, we will need more complex tools.

One way to reconstruct a species network is to first estimate trees for subsections of the sequence data. These subsections can be called loci or, if they are specific to protein-coding section, genes (Pamilo and Nei, 1988). These methods are popular and allow for complex models that tend to perform accurately, if slowly. As a group, they are generally referred to as multi-locus methods. Some of them take gene trees as input. Others take the multiple alignments as input, and integrate over the gene trees. The techniques they use include maximum parsimony, maximum likelihood, maximum pseudolikelihood, and Bayesian inference (Meng and Kubatko, 2009; Kubatko *et al.*, 2009; Yu *et al.*, 2012; Wen *et al.*, 2016). In general, these methods first estimate a tree for each gene (or loci), then reconstruct a species network from this forest of gene trees.

While the more complex models used in multi-locus methods allow for better reconstruction with fewer assumptions than previous approaches, these methods suffer from high computational cost. The likelihood-based gene tree methods are generally considered too slow to use with more than five to ten species (Hejase and Liu, 2016; Wen and Nakhleh, 2017; Yu *et al.*, 2014; Wen *et al.*, 2018; Steel and Penny, 2000; Kamneva and Rosenberg, 2017).

Bayesian methods, while suffering from many of the same computational time challenges as full likelihood-based methods, can use priors to lower the risk of overly complex results (Mueller *et al.*, 2018; Wen and Nakhleh, 2017; Bouckaert *et al.*, 2019). They compute posterior distributions of possible networks from sequence data and can also give confidence estimates for the number of reticulations. However, they can encounter mixing challenges, especially when adding and removing reticulate edges (Huelsenbeck *et al.*, 2002; Zhang *et al.*, 2018; Elworth *et al.*, 2019). Bayesian tools also have scalability problems and,

in general, are only practical for small numbers of species with a few dozen loci each, like the gene tree methods.

Pseudo-likelihood multi-locus methods show more promise: they seem to be fast and accurate, accommodating up to about 30 species (Solís-Lemus and Ané, 2016; Yu and Nakhleh, 2015). However, they estimate likelihoods over sub-networks, pretending that these sub-networks are independent. A full likelihood model could perhaps yield even more accurate results.

The fastest and most popular approaches for phylogenetic tree reconstruction use concatenated aligned genomic sequences. Concatenation methods estimate species networks directed from concatenated sequence data, aligned gene or loci sequences concatenated into one long sequence. While this erases the grouping information that was present in the gene grouping information, it allows for estimation in one step rather than two, which can significantly speed estimation. Concatenated sequence data can be from multiple genes, multiple proteins, alignments gathered with next-gen sequencing methods like RAD-seq, or other kinds of genetic marker data like many individual SNPs. Concatenated sequence methods do not use information about the site's location on the genome. Here, the unit of data is a randomly sampled site from the genome, whereas in the gene tree case, the unit of data is a randomly sampled gene. Concatenation methods are popular among biologists due to their speed and relative accuracy, even in the presence of moderate incomplete lineage sorting (Tonini J, Moore A, Stern D, Shcheglovitova M, 2015; Gadagkar *et al.*, 2005; Thiergart *et al.*, 2014; Chifman and Kubatko, 2014; Bryant and Hahn, 2020).

While methods for reconstructing trees and networks vary in model and approach, the most accurate among them generally suffer from one major limitation: scalability. Evolutionary scientists are often interested in constructing phylogenies for many species, but current methods cannot accommodate this. For example, in the Tree of Life project, scientists are creating a phylogenetic history for all species on earth, living and extinct (Maddison and Maddison, 1996). In order to model biological realities like horizontal gene transfer, hybridization, and introgression in larger projects like this, biologists need network methods. But, in order to accommodate this many species, we need an approach that is radically more efficient.

Despite the computational challenges of maximum likelihood methods, there are several reasons to pursue them. First, these methods use all of the available data. They choose the network that maximizes the

likelihood of the all data, without subsampling or simplifying. While methods like maximum parsimony can be shown to do this in some carefully-defined cases, they are not guaranteed to do so in most cases (Kolaczkowski and Thornton, 2004; Steel and Penny, 2000). For these reasons, maximum likelihood methods are generally considered most accurate, but efficiency improvements are needed.

Most likelihood-based methods assume independence between sites (Felsenstein, 1981). Multi-locus methods can make a smaller assumption: that sites are independent after grouping on their gene. Here, a subset of the sequence is used to infer a gene tree and each subset is assumed to be independent of the others, given its true tree. When using concatenated data, we lose the gene location information for each site. Therefore, concatenated methods make a stronger assumption: that sites are independent without any conditioning. This assumption is rather strong, and cannot be met when there are strong differences between sites based on their gene location.

Because of this assumption, concatenation methods tend to work best in data where gene location conveys a relatively small amount of information. This is typically true of deeper-time phylogenetic inference, for alignments with low rates of variable sites, and for single-nucleotide polymorphism (SNP) data. Data with a high proportion of constant sites are a good fit for the concatenation model, because they have decreased overall dependence across sites. Even though sites within each locus are not independent, any site with a substitution rate of zero, generating constant patterns, will be independent from any other. Only variable sites are informative and can affect the likelihood difference between any two network topologies. Because high levels of constant sites are common in phylogenetic problems, concatenated sequence methods are a good tool for many problems. This independence between sites assumption is also sensible for short genes, biological traits, and even words within languages, making these data also a good fit for concatenation methods.

While location and gene information does encode useful information in some cases, often an assumption of independence between sites is appropriate. Furthermore, while an assumption of independence among sites is not always warranted, these methods have been shown to perform well even in DNA sequences with some dependence (Chifman and Kubatko, 2014; Bryant and Hahn, 2020).

To reduce the complexity of our model and ease computational time, we do not include incomplete lineage sorting (ILS) in our model. As summarized in the introduction, ILS occurs when there are short

internal branches in a network, meaning that some lineages do not have time to resolve before the next ancestral node. In data where ILS is known to be the main source of variability, concatenation methods may be inconsistent. However, in deeper time scales, error due to a lack of coalescence is swamped by other forms of variability, so models that do not include ILS work well (Bryant and Hahn, 2020).

A method that combines the precision of maximum likelihood models with the speed of concatenation could be both accurate and fast. Recent work in Bayesian concatenation approaches for biallelic markers data suggest that this may be a promising approach (Rabier *et al.*, 2020). Strimmer and Moulton (2000) presented a likelihood-based graphical model framework to infer phylogenetic networks from concatenated DNA. However, we are not aware of an existing implementation of this framework or of any existing likelihood-based concatenation methods to estimate phylogenetic networks using four-state DNA sequences. In the following section, I describe the graphical model that underlies our concatenation-based maximum likelihood method and efficient implementation.

2.2 A Graphical Model for Phylogenetic Networks

Phylogenetic networks are graphs depicting the evolutionary history of a set of species or taxa. Their leaf vertices are bijectively labeled with taxa for which we have data (typically, extant), while the interior vertices represent ancestral species, for which we do not have data. Generally, the length of edges in the graph represent time, with longer edges representing longer stretches of evolutionary time.

The root represents the most recent common ancestor of the taxa in the network. Phylogenetic networks are generally considered either unrooted, rooted, or semi-directed (Huson and Scornavacca, 2011; Nakhleh, 2010; Solís-Lemus and Ané, 2016).

Definition 1. Unrooted Phylogenetic Network.
An *unrooted phylogenetic network* is an undirected graph with leaves bijectively labeled by a set of taxa (Huson and Scornavacca, 2011). Leaf vertices have degree one and other vertices have degree three. Edges are undirected.

Definition 2. Rooted Phylogenetic Network.
A *rooted phylogenetic network* is a rooted directed acyclic graph (DAG) with leaves bijectively labeled by a set of taxa (Huson and Scornavacca, 2011). Leaf vertices have degree one, the root has degree two, and

other interior vertices have degree three. Edges within the graph are directed: information flows from the root to the leaves and from parent to child node, representing inheritance over evolutionary time.

Definition 3. Semi-Directed Phylogenetic Network.
A *semi-directed phylogenetic network*, introduced by Solís-Lemus and Ané (2016), is a directed acyclic graph with leaves bijectively labeled with a set of taxa. It lacks a root node. Edges within the graph are categorized as tree or hybrid edges. Only hybrid edges have an explicit direction; tree edges have no direction. This constrains the placement of a root: it cannot be the descendant of any hybrid node.

In this work, we focus on semi-directed phylogenetic networks. Our semi-directed phylogenetic networks represent a hybridization event with a hybrid node, an interior vertex with in-degree two (two parent edges directed in) and out-degree one (one child edge directed out). The two parent hybrid edges a and b are directed into a hybrid vertex H (Figure 2.1). These hybrid edges each have an inheritance weight, denoted generically by γ, indicating what percentage of the child's ancestry is inherited from each parent edge. These inheritance weights must add up to 1 across all parent edges, meaning that $\gamma_a + \gamma_b$ must equal 1. Non-hybrid, non-root tree vertices have in-degree one and out-degree two. The parent edge e of a tree vertex v is called a tree edge. It always has an inheritance weight $\gamma = 1$, since e is the only parent of its child v: all of v's ancestry must trace back through e.

After establishing our network's definition, let's turn out attention to the data. In this work, we consider DNA alignments with the nucleotide alphabet, but the model described below could be easily extended to any other alphabet, to use fewer states, as in binary data, or more states, as in codon data.

At each site in the concatenated DNA alignment, each vertex in the graph has four possible states for each of the nucleotides that make up DNA: A, C, G, or T. For leaf vertices, these states are known from the sequence data. For each of the interior vertices, these states are unknown, so the probability of each state is calculated. Each edge in the graph has a transition probability, representing the probability of a change in state from the beginning of the edge to the end of the edge. These transition probabilities are modeled by a continuous-time Markov process. To model the state at a tree node conditional on its parent node, it suffices to know the 4×4 transition probability matrix P_e for its parent edge e. The probability of a change in state at a given site along an interior edge e is a function the branch length t_e, and the site's evolutionary rate. This 4×4 matrix is assumed of the form $\exp(t_e r Q)$, where t_e is the edge's length, r is the

rate multiplier from the rate variation model, and Q is the rate matrix, shared across all edges and all sites. The model underlying these probabilities is described in more detail in the next section.

To model the state distribution of a hybrid node conditional on its two parent nodes, we need a more complex transition probability matrix. We combine the transition probabilities along the two parent edges and the γ inheritance weights to produce a 16×4 matrix of conditional state probabilities for the hybrid node given the 16 combinations of states at its parent nodes. Figure 2.1 illustrates this case for hybrid node H in a subgraph of a larger directed acyclic graph (DAG). I will focus on the Markov process and hybridization represented by solid lines, because the graphical model is fully characterized by the conditional probabilities of nodes conditional on their parents. Capital letters refer to vertices, while small letters refer to edges. The two edges a and b are parents to hybrid node H. They have inheritance weights γ_a and γ_b, such that $\gamma_a + \gamma_b = 1$.

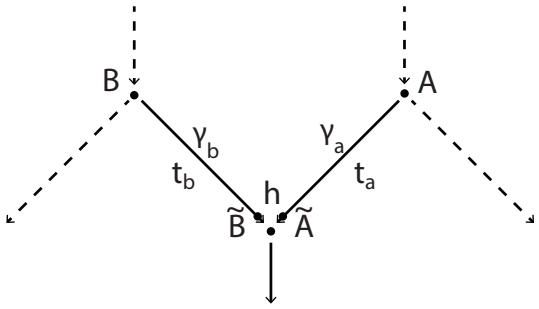

Figure 2.1: The graphical model at a hybridization event combines a Markov process along edges and a discrete mixture model.

To simplify the model description, I consider two additional vertices of degree two, $\tilde{A}$ and $\tilde{B}$ at the end of edges a and b, respectively, so that the length between them and vertex H is zero. $\tilde{A}$ (resp. $\tilde{B}$) is a tree node with a single parent A (resp. B), so its distribution conditional on its parent is described by $P_a = \exp(t_a r Q)$ (resp. $P_b = \exp(t_b r Q)$). Next, we model the state at H as being equal to the state at $\tilde{A}$ with probability γ_a, and the state at $\tilde{B}$ with probability $1 - \gamma_a = \gamma_b$. In other words, the distribution at H conditional on $\tilde{A}$ and $\tilde{B}$ is given by the following 4×16 matrix (transpose of the transition matrix for presentation purposes), in which the column headers indicate the states of the immediate parent vertices $\tilde{A}$ (first letter) and $\tilde{B}$ (second letter):

$$\begin{bmatrix} & AA & CA & GA & TA & AC & CC & GC & TC & AG & CG & GG & TG & AT & CT & GT & TT \\ A & 1 & \gamma_b & \gamma_b & \gamma_b & \gamma_a & 0 & 0 & 0 & \gamma_a & 0 & 0 & 0 & \gamma_a & 0 & 0 & 0 \\ C & 0 & \gamma_a & 0 & 0 & \gamma_b & 1 & \gamma_b & \gamma_b & 0 & \gamma_a & 0 & 0 & 0 & \gamma_a & 0 & 0 \\ G & 0 & 0 & \gamma_a & 0 & 0 & 0 & \gamma_a & 0 & \gamma_b & \gamma_b & 1 & \gamma_b & 0 & 0 & \gamma_a & 0 \\ T & 0 & 0 & 0 & \gamma_a & 0 & 0 & 0 & \gamma_a & 0 & 0 & 0 & \gamma_a & \gamma_b & \gamma_b & \gamma_b & 1 \end{bmatrix}$$

Components of this model were presented by Strimmer *et al.* (2001) and implemented by Jin *et al.* (2006), though we were not able to find existing software implementing their approach. We build upon their work by exploring identifiability of this model from concatenated sequence data, by defining a canonical network under this model, and by adding more complexity to the probability model to better fit the evolutionary process. We also explicitly link our work to the underlying graphical model, the reversible continuous-time Markov process, and evolutionary processes in our likelihood formulation.

In the following section, we describe Markov models for evolutionary change, from a simple model to more complex models that fit biological data more closely. To align with the terminology used by evolutionary scientists and biologists, we move from typical graph theory terminology of graphs, vertices and edges to the more commonly-used phylogenetic terminology of networks, nodes, and branches.

2.3 Markov Models of Evolutionary Change

To develop a model of evolutionary change, we examine how a string of sites in a strand of DNA might change over time. Each site in the sequence of nucleotides has a certain probability of mutating from its current state into a new one. A site can mutate from one state to another (from A to G, for example) or it can stay the same. In Figure 2.2, we represent one possible evolutionary history for one site along a sequence. In the ancestor species, this site has state A. As time progressed, this site changes states from A to C in one of its descendants. For the other two descendants, the site remained in the same state (or mutated from away from state A, then back again).

The simplest model assumes that the rates of change from one state to another are equal. All possible transitions happen with the same probability; the probability of changing from an A to a C is equal to the probability of changing from A to G. It is also reasonable to assume that the probability of a change in the first generation is equal to a probability of a change in the second, making the process time homogeneous.

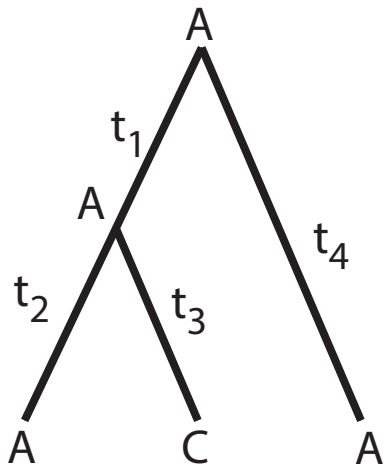

Figure 2.2: One possible evolutionary history for one site with data ACA

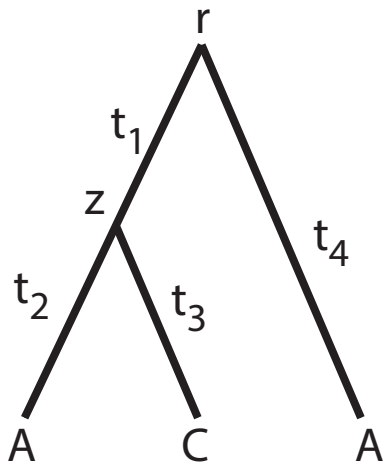

Figure 2.3: Network with interior nodes r, z, states unknown

Perhaps the most important trait of this process is its memorylessness. In this setting, it reasonable to assume that the probability of change in the next generation only depends on the current state, not on the history that came before: the site's current state of A affects the probability of its future possible states, but a previous state of G does not. Finally, as time is continuous, we can assume that this mutation process is also continuous. By combining these characteristics, this process can be well-described by a continuous-time Markov model, as first proposed by Kaplan and Langley (1979) and refined by Felsenstein (1981).

With equal rate α, we can describe the probabilities of each of these changes using a matrix of transition

rates, Q:

$$Q = \begin{bmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{bmatrix}$$

Here, in the simplest, equal rate setting, the probabilities of transition from one state to another is $P(t) = e^{Qt}$. The branch or edge length, t, indicates how much evolutionary time has passed between two neighbor nodes in a phylogenetic tree. This gives us a matrix of transition probabilities, $P(t)$, for an edge of length t.

$$P(t) = \begin{bmatrix} \frac{1}{4} + \frac{3}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} \\ \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} + \frac{3}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} \\ \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} + \frac{3}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} \\ \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & \frac{1}{4} + \frac{3}{4}e^{-4\alpha t} \end{bmatrix}$$

Let us pretend for a moment that we know the state of the interior nodes of the network, as shown in Figure 2.2. Using the transition probabilities in the P matrices, we can calculate the likelihood of the observed data at the tips for this history. Given the network topology and branch lengths, t_i, the probability of these data is:

$$\pi_A P_{AA}(t_1) P_{AA}(t_2) P_{AC}(t_3) P_{AA}(t_4) \tag{2.1}$$

The first term in this equation, π_A, is the stationary frequency, the proportion of each DNA base expected from the model at stationarity.

However, we do not know the states of the interior, ancestral nodes in a network. We have only site data for the species on the external nodes. Therefore, Figure 2.2 describes only one possible history for this site. So, instead of calculating the probability of one history for these nodes, we must consider the probability of all 16 possible histories, as show in Figure 2.4. To find the probability of each of these histories, we consider the branch lengths, transition probabilities, and stationary frequencies for each possible ancestral history

in equation (2.1). To consider all these possible histories at once, we take the sum over all histories:

$$P = \sum_{r \in A,C,G,T} \sum_{z \in A,C,G,T} \pi_r \, P_{rz}(t_1) \, P_{zA}(t_2) \, P_{zC}(t_3) \, P_{rA}(t_4)$$

where, with some abuse of notation, r is the state at the root node and z is the state at the interior node.

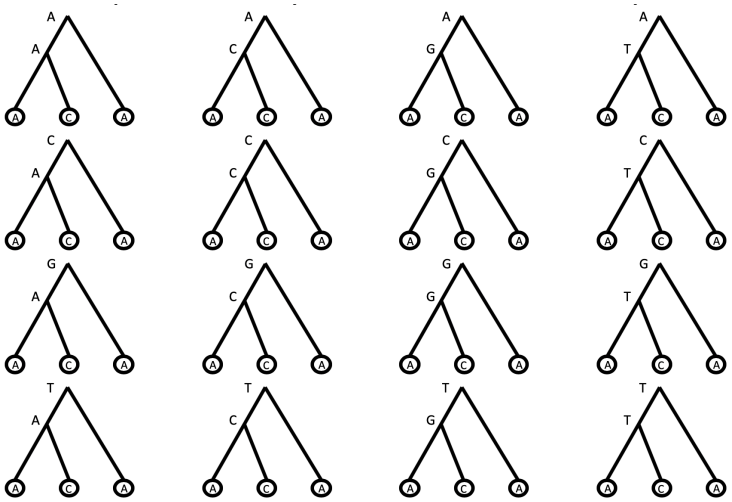

Figure 2.4: Possible evolutionary histories for one site on a candidate phylogenetic network structure

2.3.1 Jukes and Cantor Model

The preceding section with equal rates describes the model introduced by Jukes and Cantor (1969), often denoted as the JC69 model. This model has few parameters, making it efficient to estimate. In fact, in the relative version, there are no parameters to estimate if the Q matrix is normalized to having ones on the diagonal, such that there is an average of one change per unit of time:

$$Q = \begin{bmatrix} -1 & 1/3 & 1/3 & 1/3 \\ 1/3 & -1 & 1/3 & 1/3 \\ 1/3 & 1/3 & -1 & 1/3 \\ 1/3 & 1/3 & 1/3 & -1 \end{bmatrix}$$

In the absolute version, the Q matrix is the same except that, in the above fractions, 1 is replaced with a

rate parameter α:

$$Q = \begin{bmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{bmatrix}$$

The first row, for example, indicates how quickly a site starting at state A will change into states C, G, and T respectively. In this model, the stationary distribution is uniform: $\pi_i = 1/4$ for all states $i \in \{A, C, G, T\}$.

2.3.2 Hasegawa Kishino Yano Model

The more flexible and complex Hasegawa Kishino Yano (HKY85) substitution model introduces two new elements (Hasegawa *et al.*, 1985). First, it lets base frequencies be any arbitrary frequency vector $\pi = (\pi_A, \pi_C, \pi_G, \pi_T)$. In empirical studies, base frequencies are often estimated by the empirical frequencies, the proportion of each state seen within the sequence data.

Second, the HKY85 model recognizes that transitions are generally more likely than transversions. For the relative version of the HKY85 model, only one rate is used. This parameter represents the ratio between the rate of change from a state that is similar to the current state (e.g. A to G, both purines), and the rate of change to a different state (e.g. A, a purine, to C, a pyrimidine). This parameter is called the transition-transversion ratio and is denoted κ.

For the absolute version of the HKY85 model, there are two rates to estimate: one for the transition rate, α, and one for the transversion rate, β. For the relative version, the rate transition matrix Q is normalized to give an average of one change per unit of time. To do this, the absolute version of Q is divided by:

$$2(\pi_T\pi_C + \pi_A\pi_G)\alpha + 2(\pi_Y\pi_R)\beta$$

where π_Y represents the proportion of purines (A, G) such that $\pi_Y = \pi_A + \pi_G$. Similarly, π_R represents the proportion of pyrimidines (C, T) such that $\pi_R = \pi_C + \pi_T$.

The additional flexibility of this model means that it provides a better fit than the JC69 model for most data sets, though it does require parameter estimation. The relative HKY85 model is more complex than

JC69, with exactly two unique eigenvalues not equal to zero. Despite this additional complexity, the model is computationally feasible because the eigenvalues are known analytically.

2.3.3 Generalized Time Reversible Model

The Generalized Time Reversible (GTR) model is even more flexible than the HKY85 model (Tavaré, 1986). In empirical studies, HKY85 generally fits well for moderately-long genes, but the GTR model fits better for most longer multi-gene concatenated data. Despite these benefits, there are reasons to use simpler models. Unlike the case of JC69 and HKY85 models, there is no analytical formula for the GTR model's eigenvalues. Therefore, any software that uses it needs to do an eigenvalue decomposition for every change in parameters during optimization, which increases the computational time significantly. Due to this computational consideration, I did not implement this model in our software. However, it could be easily added if needed, given the existing data structures.

As mentioned above, each Markov substitution model has two versions. The relative version is obtained from the absolute version by normalizing the Q matrix to an average of one substitution per unit of time. To estimate branch lengths in substitutions per site, we use the relative rate version of each model.

2.3.4 Rate Variation Across Sites

Evolutionary rates are know to vary across a genome. Generally, biologists attribute this variation in the evolutionary pressures upon different parts of the genome. We model this changeability using Yang's discrete gamma model for variable substitution rates across sites (Yang, 1994). This is often referred to as the "+Γ model" because it add rate variation to existing models: JC69 or HKY85 becomes JC69 +Γ or HKY85 +Γ.

To set the mean of the desired continuous Gamma distribution to 1, we estimate the shape parameter α and set scale parameter $\theta = 1/\alpha$ (so $\beta = \alpha$). The Gamma distribution is discretized into k categories, with the default number of categories set to four. In each category, the rate multiplier is a normalized quantile of the gamma distribution. The probability of each site is calculated under under a mixture model, for we need to calculate the probability of each site assuming each category. Therefore, the computational time to compute the likelihood is multiplied by k, the number of discrete categories. The k likelihooods, under the k rate categories, are averaged with the weight of each category, which is $1/k$ in the discretized

Γ model.

We also implemented the "+ I" model and the "+Γ + I" models, in which one rate category has rate $r_0 = 0$ with weight p_{inv}. Any site with this 0 rate is invariable: mutations cannot occur, and the data for this site must be constant. In the I model, there is one other category with rate $r_1 = 1/(1 - p_{\text{inv}})$ and weight $1 - p_{\text{inv}}$, such that the average rate is 1. In the Γ + I model, there are k non-zero rate categories with rates $r_i = \tilde{r}_i/(1 - p_{\text{inv}})$ and weights $(1 - p_{\text{inv}})/k$, where $\tilde{r}_1, \ldots, \tilde{r}_k$ are the weights for the discretized Γ model with shape parameter α, as described above. By rescaling the $\tilde{r}_i$ rates, the weighted average of $r_0, r_1, \ldots, r_k$ remains normalized to 1.

When optimizing the log likelihood, we optimize the α parameter for the Γ model, the p_{inv} parameter for the I model, and both parameters for the Γ+I model. This flexibility explains biological variation in rates, but only increases the number of parameters to estimate by one or two, which can be considered computationally efficient. I will discuss how this model is combined with the other components to create a full likelihood for the network.

In the next section, I describe challenges in calculating the likelihood of phylogenetic networks, solutions to these challenges, and our approach to extending that calculation to networks.

2.4 Calculating the Likelihood

2.4.1 Efficiently Calculating a Tree's Likelihood

In a network with n nodes and 4 possible states, as in the DNA case with the A, C, G, T alphabet, calculating the likelihood is a nontrivial task. With n interior nodes, there are 4^n possible histories. Each of these histories has its own probability, which can be calculated using the Markov models described in the previous section.

However, as the number of species scientists want to include grows, the number of possible histories grows exponentially. In a rooted fully resolved tree, there are $2N_{\text{species}} - 1$ nodes, with $n = N_{\text{species}} - 1$ of them interior nodes. Therefore, there are $4^{N_{\text{species}}-1}$ possible histories. In calculating the probabilities for each of these histories, we could traverse the tree $4^{N_{\text{species}}-1}$ times. Because the likelihood needs to be calculated repeatedly, this time complexity is not workable. Below, we describe the algorithm we use to

compute the state probabilities at each of these interior nodes in linear time instead of exponential time using a message-passing algorithm.

Message-passing algorithms efficiently calculate the likelihood on a tree, but using them to calculate a network's likelihood is more complex. We will first describe how message-passing algorithms can calculate the likelihood of a tree, then We will extend the calculation to networks.

Felsenstein's message-passing pruning algorithm calculates the likelihood of a tree quickly and exactly and has been widely used to calculate a tree's likelihood since Felsenstein introduced it in 1973 (Felsenstein, 1973). This algorithm is a recursive sum-product message passing algorithm, also sometimes known as belief propagation. For each site in the data, his algorithm moves through the tree from the tips to root, in a post order tree traversal, computing the marginal distribution at each interior node, for its descendant observed leaf nodes (Pearl, 1982). For a tree, the complexity of this algorithm for each site scales linearly, with the number of nodes in the network.

Starting at the leaves, we calculate the probability of the data below each node, given each possible state (A, C, G, or T) at that node. For the leaves, calculating the probability is straightforward: it is simply one for the observed state and zero for all others. Figure 2.5 shows step one of this algorithm, the calculation at the leaf nodes, for a simple network with tree species and only one site. For this site in the data, the first species shows state A; the second species shows state C, and the third shows state A.

Figure 2.6 shows the next step of the algorithm, for interior node z. To calculate the probability of its descendant conditional on state A at node z, we multiply contributions from the left and right descendants. To calculate these contributions, we will need a rate matrix Q, introduced in the previous section. For clarity, I will use the simplest Markov model, JC69, with equal rates across the Q matrix:

$$Q = \begin{bmatrix} -1 & 1/3 & 1/3 & 1/3 \\ 1/3 & -1 & 1/3 & 1/3 \\ 1/3 & 1/3 & -1 & 1/3 \\ 1/3 & 1/3 & 1/3 & -1 \end{bmatrix}$$

With edge length $t = 0.1$, we can calculate the matrix of transition probabilities:

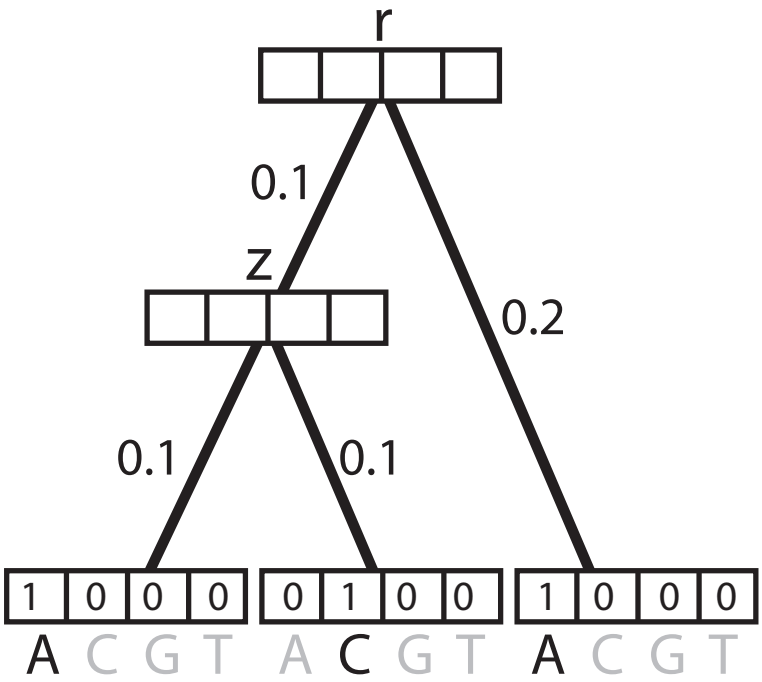

Figure 2.5: Pruning Algorithm Step 1

$$P(0.1) = e^{0.1Q} \simeq \begin{bmatrix} 0.91 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.91 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.91 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.91 \end{bmatrix}$$

The contribution from the left is the probability of the node starting at A at the leaf and also showing state A at interior node z. Based on the rate matrix and edge length, this probability is about 0.91. The contribution from the right for state A is the probability of starting at state C then changing to A over the edge, about 0.03. These two contributions are multiplied to obtain the probability of state A at interior node z:

$$P(\text{data below node } z|A \text{ at } z) = P_{AA}(1)P_{AC}(1) \simeq 0.03$$

Similarly, the probabilities for states C, G, and T are calculated as the product of the contributions from the left and right descendants:

$$P(\text{data below node } z|C \text{ at } z) = P_{CA}(1)P_{CC}(1) \simeq 0.03$$

$$P(\text{data below node } z|G \text{ at } z) = P_{GA}(1)P_{GC}(1) \simeq 0.001$$

$$P(\text{data below node } z|T \text{ at } z) = P_{GA}(1)P_{GC}(1) \simeq 0.001$$

These probabilities align with our intuition: the states A and C are most likely at this interior node, considering that its descendants' data show states A and C.

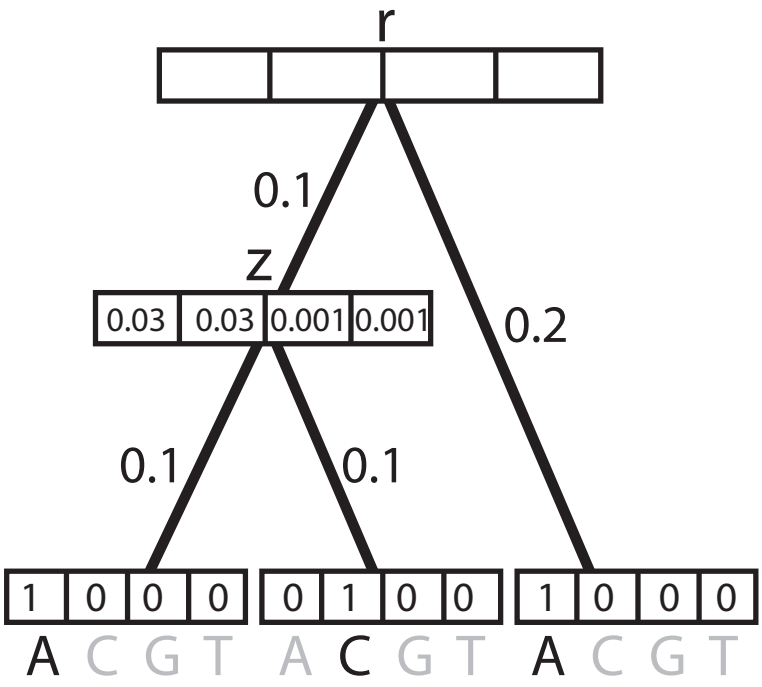

Figure 2.6: Pruning Algorithm Step 2

After calculating all interior non-root nodes, we calculate the probability of the data at the full set of leaves, conditional on the state at the root node. After we process the root probabilities, we perform a final step to combine the 4 conditional probabilities into a final, unconditional, probability (Figure 2.7). This calculation requires considering the data for species 3. Like for z, the probability conditional on each state at the root has two contributions: from the left and from the right. The probability from the left is the probability of the data at the leaves below the left child z, based on the interior node z and branch length of length 0.1. Again, we use the transition probability matrix $P(0.1)$ above. Conditional on A, the probability of the data below z is the sum of the probabilities of remaining at state A at node z, mutating to C, mutating to G, or mutating to T at node z, and then obtaining the leaf data conditional on the state at z:

$$P(\text{left descendants}|\text{A at the root}) \simeq 0.91*0.03+0.03*0.03+0.03*0.001+0.03*0.001 \simeq 0.03$$

$$P(\text{left descendants}|\text{C at the root}) \simeq 0.03*0.03+0.91*0.03+0.03*0.001+0.03*0.001 \simeq 0.03$$

$$P(\text{left descendants}|\text{G at the root}) \simeq 0.03*0.03+0.03*0.03+0.91*0.001+0.03*0.001 \simeq 0.003$$

$$P(\text{left descendants}|\text{T at the root}) \simeq 0.03*0.03+0.03*0.03+0.03*0.001+0.91*0.001 \simeq 0.003$$

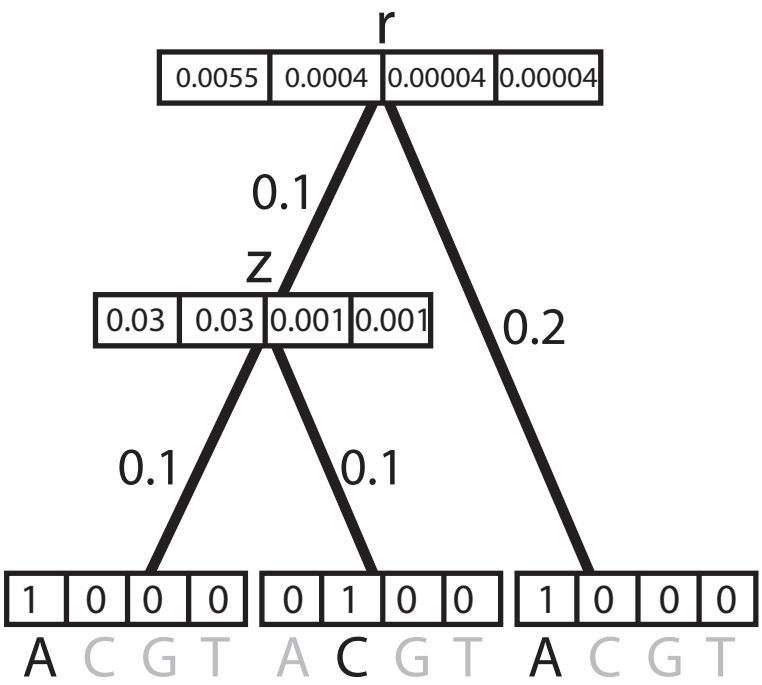

Figure 2.7: Pruning Algorithm Step 3

For the right child's contribution, we need the transition matrix for edge length $t = 0.2$.

$$P(0.2) = e^{0.2Q} \simeq \begin{bmatrix} 0.82 & 0.06 & 0.06 & 0.06 \\ 0.06 & 0.82 & 0.06 & 0.06 \\ 0.06 & 0.06 & 0.82 & 0.06 \\ 0.06 & 0.06 & 0.06 & 0.82 \end{bmatrix}$$

These contributions are simply the A column of $P(0.2)$:

$$\begin{aligned} P(\text{right descendants}|\text{A at the root}) &\simeq 0.82 \\ P(\text{right descendants}|\text{C at the root}) &\simeq 0.06 \\ P(\text{right descendants}|\text{G at the root}) &\simeq 0.06 \\ P(\text{right descendants}|\text{T at the root}) &\simeq 0.06 \end{aligned}$$

Multiplying probabilities from the right and left descendants gives these likelihoods for each state:

$$[0.022, 0.002, 0.0002, 0.0002]$$

Finally, these conditional probabilities are combined in a last step to provide the likelihood of the data at the leaves, unconditional on the root state. To marginalize out the root state, we simply take the weighted average of the conditional probabilities, weighted by the stationary distribution. In this simplest model, this distribution is even, 25% for each state:

$$[0.25, 0.25, 0.25, 0.25]$$

Multiplying the contributions from descendants with this stationary distribution gives a probability of each state at the root of

$$[0.0055, 0.0004, 0.00004, 0.00004]$$

By using this algorithm, the exponential problem of calculating the likelihood becomes a linear one. To calculate the probability of each these histories without this method, we would calculate 4^n probabilities. As the tree grows, even with a small number of species, the time needed to calculate the likelihood would grow impractically large. Using this pruning algorithm instead, we calculate the probability of each site using only one traversal of the network. To take advantage of this fast algorithm, we extend this calculation to networks by breaking them down into its component parts. In the next section, we describe this process in detail.

2.5 Calculating a Network's Likelihood

2.5.1 A Network's Likelihood from its Weighted Displayed Trees

To take advantage of Felsenstein's linear-time pruning algorithm for networks, we break each network into its components. Each network is made up of displayed trees, created by dropping hybrid edges in the network. After extracting a network's displayed trees, we can apply the pruning algorithm to each displayed tree, then recombine these likelihoods to get the likelihood of the network.

This displayed tree approach is made possibly because of our mixture model for the state at each hybrid node, described in section 2.2. At each hybrid node, the state is inherited from its left parent with some γ value. With this probability γ, the right parent edge can be deleted from the network. With probability $1-\gamma$, the left parent edge can be ignored and deleted from the network to calculate the site probability.

I will describe this process step by step. First, we extract all displayed trees within each network. To do this, we choose one parent edge for each hybrid node and delete its partner edge. A network with no reticulations has one displayed tree, while a network like Figure 2.8 with one hybridization has two displayed trees (Figure 2.9). In general, if the network has h hybrids, it has 2^h displayed trees. There are a few notable exceptions to this rule (for which we will present examples later).

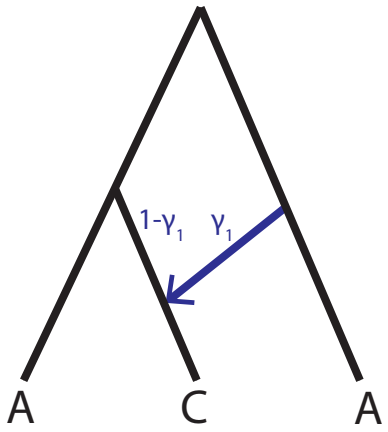

Figure 2.8: Network with one hybridization

As discussed in the model summary, a hybrid parent edge e above a hybrid node has inheritance weight γ_e. These weights indicate the proportion of heritage that the child node received from e. For all tree edges, this weight equals one. For hybrid edges, inheritance weights are between zero and one, with weights summing to one across all parents of a given hybrid node.

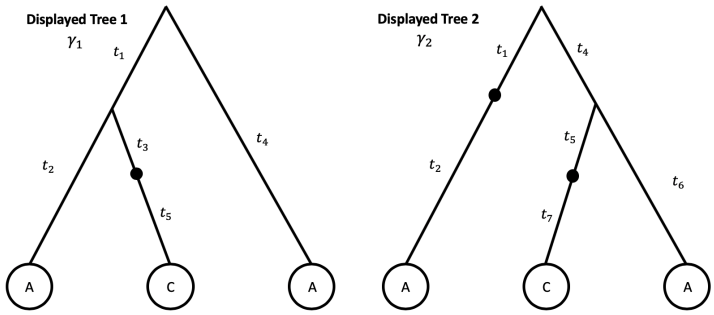

Figure 2.9: Displayed trees for a network with one hybridization

After breaking a network into its displayed trees, we use Felsenstein's pruning algorithm to calculate the probability of the data conditional on each displayed tree, that is, conditional of the parent that each hybrid node inherited from.

Finally, the network likelihood is calculated by combining these conditional likelihoods, to marginalize out the displayed tree. Each of the displayed trees within a network carries a proportion of inheritance, based on its edges' γ inheritance weights. For example, if a network has one reticulation event, the probability that the state S at the hybrid node was inherited from each of the parents is

$$S = \begin{cases} P_1 & \text{with probability } \gamma_1 \\ P_2 & \text{with probability } \gamma_2 \end{cases}$$

In networks with more than one hybridization, let $\mathscr{T}$ be the set of all displayed trees in the network. For $T \in \mathscr{T}$, the inheritance weight of T, $\gamma(T)$, is the product of all inheritance weights in the tree:

$$\gamma(T) = \prod_{e \in T} \gamma_e$$

The likelihood of the data D_i at site i is then a weighted sum of the conditional likelihoods $L(D_i|T)$, conditional on each displayed tree T. Combining over all sites as a product of the site likelihoods, the likelihood can be expressed as

$$L(N) = \prod_{i=1}^{n} \sum_{T \in \mathscr{T}(N)} \gamma(T) L(D_i|T)$$

where n is the number of sites, $L(D_i|T)$ is the tree likelihood of the data at site i given the tree T, $\gamma(T) = \prod_{e \in T} \gamma_e$, D_i are the data at site i, and $\mathscr{T}$ is the set of all displayed trees in the network N.

To account for rate variation with our discrete rate distribution model, I sum over all rates r_j, weighted by the probability p_j of each rate category. For example, with the discretized Γ model allowing for four rate categories, $p_1 = \cdots = p_4 = 1/4$. The likelihood is now expressed as

$$L(N) = \prod_{i=1}^{n} \sum_{T \in \mathscr{T}(N)} \sum_{j=1}^{k} p_j \gamma(T) L(D_i|T, r_j)$$

where $L(D_i|T, r_j)$ is the conditional probability of the data D_i at site i, given a displayed tree T and rate r_j.

For improved efficiency, our software simplifies the data by eliminating duplicate site patterns. We do this by counting repeated site patterns, removing duplicates of each site pattern (which decreases the total number of sites n), then weight each site pattern with this count, denoted as w_i. With this additional complexity, the likelihood is

$$L(N) = \prod_{i=1}^{n} \Bigg(\sum_{T \in \mathscr{T}(N)} \sum_{j=1}^{k} p_j \gamma(T) L(D_i|T, r_j) \Bigg)^{w_i}.$$

2.5.2 Challenges in Calculating the Network's Likelihood

Even after traversing trees efficiently, this problem is still very time consuming. For each site, we need to traverse 2^h displayed trees. After the pruning algorithm is completed for each site, we still need to take a weighted sum over displayed trees and multiply these site likelihoods.

Because these probabilities can be very small, we calculate likelihoods on the log scale to avoid underflow problems. Note that maximizing the likelihood of the network is equivalent to maximizing the log-likelihood:

$$\log L(N) = \sum_{i=1}^{n} w_i \log \Bigg(\sum_{T \in \mathscr{T}(N)} \sum_{j=1}^{k} p_j \gamma(T) L(D_i|T, r_j) \Bigg). \tag{2.2}$$

2.6 Optimizing Numerical Parameters Efficiently

Estimating parameters within the network is often one of the greatest barriers to improving speed. Every time a parameter is changed, the network's likelihood must be recalculated to see if this new parameter has improved the model's fit of the data. As discussed in the previous section, this is a lengthy calculation, even

after implementing clever algorithms. To avoid repetitive calculation and recalculation of the network's likelihood during numerical optimization, we present gradient-based methods for optimizing hybrid inheritance weights and branch lengths.

2.6.1 Efficient, Quadratic-Time Optimization of Hybrid Inheritance Weights

First, I will describe how our method estimates inheritance values using a gradient-based optimization. As presented earlier, each hybrid node has two hybrid edges, e and $\tilde{e}$. Their inheritance weights meet the constraints $0 \leq \gamma_e, \gamma_{\tilde{e}} \leq 1$ and $\gamma_e + \gamma_{\tilde{e}} = 1$. This complementary relationship allows us to optimize only one: after we know the first γ, we can easily calculate the second. After optimization the inheritance weight γ for one edge at each hybrid, we assign the complement, $1 - \gamma_e$, to its partner hybrid edge $\tilde{e}$.

By deconstructing the likelihood into components that rely only on edge e or edge $\tilde{e}$, we can avoid less efficient gradient-free optimizers. We prove below that the network likelihood is convex as a function of a single γ parameter, which enables use to use the efficient Newton-Raphson method.

Deconstructing the Likelihood

To optimize γ_e, we must find the value that maximizes the network likelihood $L(N)$. Before we can maximize this likelihood, we partition the set of displayed trees into two subsets: the trees that contain edge e, and the trees that contain edge $\tilde{e}$ instead.

To avoid underflow, we divide the likelihood by a constant, m^N where $N = \sum_i w_i$ and

$$m = \max_{i,T,j} \{p_j \gamma(T) P(D_i|T, r_j)\}.$$

Because we use the log likelihood in (2.2) instead of the likelihood, dividing the likelihood by m^N is equivalent to subtracting $N\log(m)$ from $\log(L)$, making the expression of interest:

$$\log(L) - N\log(m) = \sum_i w_i \log\Big(\sum_{j=1}^{k} \sum_{T \in N} p_j \gamma(T) P(D_i|T, r_j)/m \Big)$$

We decompose the terms and $\gamma(T)$ in this expression into components that involve either edge e, or its partner edge $\tilde{e}$, or neither edge. If a network has more than one hybrid, there will be additional hybrid

edges, here denoted e', which also contribute weights to the displayed trees.

$$\log(L) - N\log(m) = \sum_i w_i \log\Big\{\gamma_e \sum_{j=1}\sum_{T:e\in T} p_j \Big(\prod_{e'\in T, e'\neq e} \gamma(e')\Big) P(D_i|T,r_j)/m + (1-\gamma_e)\sum_{j=1}\sum_{T:\tilde{e}\in T} p_j \Big(\prod_{e'\in T, e'\neq \tilde{e}} \gamma(e')\Big) P(D_i|T,r_j)/m + \sum_{j=1}\sum_{T:e,\tilde{e}\notin T} p_j\,\gamma(T)P(D_i|T,r_j)/m\Big\}$$

The first line sums over all displayed trees containing edge e. The second includes all displayed trees containing e's hybrid partner edge e'. The third includes displayed trees that show neither. The third component of this expression is non-zero only in networks that are not tree-child, that is, if some node does not have any child that is a tree node. In that case, some displayed trees have neither edge e nor edge $\tilde{e}$. For tree-child networks, all displayed trees will have either edge e or $\tilde{e}$, and this third component will be zero.

We simplify the expression by renaming the sums that do not depend on γ_e:

$$f(\gamma_e) = \log(L) - N\log(m) = \sum_i w_i \log(\gamma_e A_i + (1-\gamma_e)B_i + C_i) \tag{2.3}$$

where

$$\begin{aligned} A_i &= \sum_{j=1}^{k}\sum_{T,e\in T} p_j \Big(\prod_{e'\in T, e'\neq e} \gamma(e')\Big) P(D_i|T,r_j)/m \\ B_i &= \sum_{j=1}^{k}\sum_{T,\tilde{e}\in T} p_j \Big(\prod_{e'\in T, e'\neq \tilde{e}} \gamma(e')\Big) P(D_i|T,r_j)/m \\ C_i &= \sum_{j=1}^{k}\sum_{T:e,\tilde{e}\notin T} p_j\,\gamma(T)P(D_i|T,r_j)/m\,. \end{aligned}$$

The first and second derivatives of this rescaled log-likelihood with respect to γ_e are

$$f'(\gamma_e) = \sum_i w_i \frac{A_i - B_i}{\gamma_e A_i + (1-\gamma_e)B_i + C_i}$$

$$f''(\gamma_e) = -\sum_i w_i \frac{(A_i - B_i)^2}{(\gamma_e A_i + (1-\gamma_e)B_i + C_i)^2} \leq 0$$

These derivatives show that the function f is concave, which will greatly facilitate optimization.

Optimizing γ_e

Before beginning the optimization, we check if the maximum likelihood is attained at either boundary $\gamma_e = 1$ or $\gamma_e = 0$:

$$f'(0) = \sum_i w_i \frac{A_i - B_i}{B_i}$$

$$f'(1) = \sum_i w_i \frac{A_i - B_i}{A_i}\,.$$

If $f'(1)$ is greater than zero, then the maximum likelihood is achieved at $\gamma_e = 1$. Inversely, if $f'(0)$ is less than zero, then the maximum is at $\gamma_e = 0$. If either of these are the case, we assign γ_e to its optimum and $\gamma_{\tilde{e}}$ to $1 - \gamma_e$ and delete the edge with $\gamma = 0$. (If $\hat{\gamma}_e = 0$, we delete edge e. If $\hat{\gamma}_e = 1$, we delete $\tilde{e}$.)

If neither 0 nor 1 maximizes the function f, we optimize γ_e using the Newton-Raphson algorithm. At each step, we propose a new $\hat{\gamma}_e$ by subtracting the ratio of the first and second derivative from the current γ_e:

$$\hat{\gamma}_e = \gamma_e - \frac{f'(\gamma_e)}{f''(\gamma_e)}$$

If this new $\hat{\gamma}_e$ is inside the bounds $(0, 1)$, γ_e is assigned to the new $\hat{\gamma}_e$. If the new $\hat{\gamma}_e \geq 1$, $\hat{\gamma}_e$ is recalculated as the average of the original γ_e and 1. If $\hat{\gamma}_e \leq 0$, the proposed γ is adjusted to be the average of the original γ and zero. We recalculate the likelihood with the updated γ_e using equation (2.3).

These steps are repeated until the difference between the new likelihood and the previous likelihood is less than a specified tolerance or for a maximum of ten iterations. Once this occurs, the current γ_e is assigned to edge e and $1 - \gamma_e$ is assigned to the partner hybrid edge $\tilde{e}$. This process may be repeated for each hybrid node in the network or may be applied to a single hybrid node adjacent to re-arranged edge (changed via a nearest neighbor interchange, a hybrid addition, or a hybrid flip).

Because the Newton-Raphson method is quadratic in speed and, in this case, guaranteed to converge to the global optimum, this method converges quickly to the best estimate of the inheritance weights for each hybrid event in the network, conditional on all other parameters. Even more importantly for

computational speed, it avoids repetitive likelihood calculations on each displayed tree. Because the sums A_i, B_i and C_i are unchanged throughout the optimization, the likelihood of each displayed tree must be computed only once. In contrast, nonlinear methods require recalculating likelihoods of all displayed trees for each new proposed γ_e. By decomposing the likelihood into displayed trees this way, our optimization method avoids these costly computations, greatly improving the speed and quality of hybrid inheritance estimation on a phylogenetic network.

2.6.2 Efficient, Quadratic-Time Optimization of Branch Lengths

Using a similar technique to the one described above for inheritance weights, we can estimate branch lengths individually using a gradient-based method. To improve the efficiently of calculating the network likelihood during branch length optimization, we use a method described by Hoang *et al.* (2018). This method allows us to isolate the contribution of the single edge length to the network likelihood. Unfortunately, the log-likelihood is not guaranteed to be concave, as a function of a single edge length. Still, we find significant computational time savings by isolating the contribution of a branch's edge length.

We take advantage of the fact that, on a tree, the contribution of the length t_e of an edge e can be easily isolated by calculating the following quantities: the probability of leaves below the child node of e conditional on the state at that child node (which is calculated as part of the pruning algorithm), and the probability of all other leaves conditional on the parent node of e (which can be calculated by one more tree traversal after the pruning algorithm). These probabilities do not depend on the branch length t_e. Using these stored probabilities, the number of calculations needed to calculate the likelihood for each candidate t_e value decreases dramatically. With this reduction in computational burden, we can try new branch lengths and test if each one improves the model's fit much faster than if we had to recompute the entire log-likelihood.

Since the log-likelihood is not concave as a function of a single edge length, we use the sequential quadratic programming (SQP) algorithm for nonlinearly constrained gradient-based optimization developed by Kraft (1988) and implemented in the Julia package NLopt (Johnson, 2020).

Because branch length optimization is costly and needs to be done repeatedly on new topologies, this addition does decrease the time cost noticeably. Its implementation is included in PhyLiNC's current version.

2.7 Estimating Evolutionary Rates

In this section, I showcase our model's method of estimating evolutionary rates for two Markov models using real data.

To estimate the rate of evolution on a network, we use a derivative-free, local optimizer developed as implemented in the Julia package NLopt (Powell, 1994; Johnson, 2020). This estimates a set of rates concurrently. Constrained optimization by linear approximations (COBYLA) creates linear approximations using a simplex of $p+1$ points in the branch length parameter space, where p is the number of parameters. So, if we optimize 3 rates simultaneously, the simplex will have 4 points. It assigns one of these points' values as the parameter value, then recalculates the full likelihood. The simplex then removes the point that fits least well and adds a new point in the area of the simplex with the highest likelihood. With this method, estimating the rates is computationally costly: each attempt with new rate parameters requires a full likelihood recalculation.

In the next section I show a rate estimation example using real data. For clarity, I discuss our method for estimating rates before discussing the full network in chapters three and four. However, it is important to note that high quality rate estimation relies on a correct network topology, branch lengths, and inheritance weights. It is not possible to accurately estimate evolutionary rates alone. In the following example, I ensure that all other network parameters are well estimated by using a network previously estimated with SNaQ (Solís-Lemus *et al.*, 2017), then estimate evolutionary rate using our method.

2.7.1 Example with Kingsnake Genetic Sequences

To demonstrate our method, I estimate evolutionary rates using sequence data from 22 species of kingsnakes and milksnakes (genus *Lampropeltis*) and one species of scarlet snake (genus *Cemophora*). This scarlet snake serves as the outgroup.

Kingsnakes and milksnakes are endemic to North America and are thought to have originated in temperate forests of North America and spread to the tropics later. While the relationships between species in the *Lampropeltis* genus are thought to be mainly tree-like, there is evidence of recent hybridization between species in close geographic proximity (Burbrink and Gehara, 2018). In 2017, Chen et al. aligned sequences for 23 species, resulting in 304 long loci, concatenated into 453 kilobase pairs (Chen *et al.*, 2017). When

summarized, the concatenated alignment contains 6737 unique site patterns. Figure 2.10 shows the

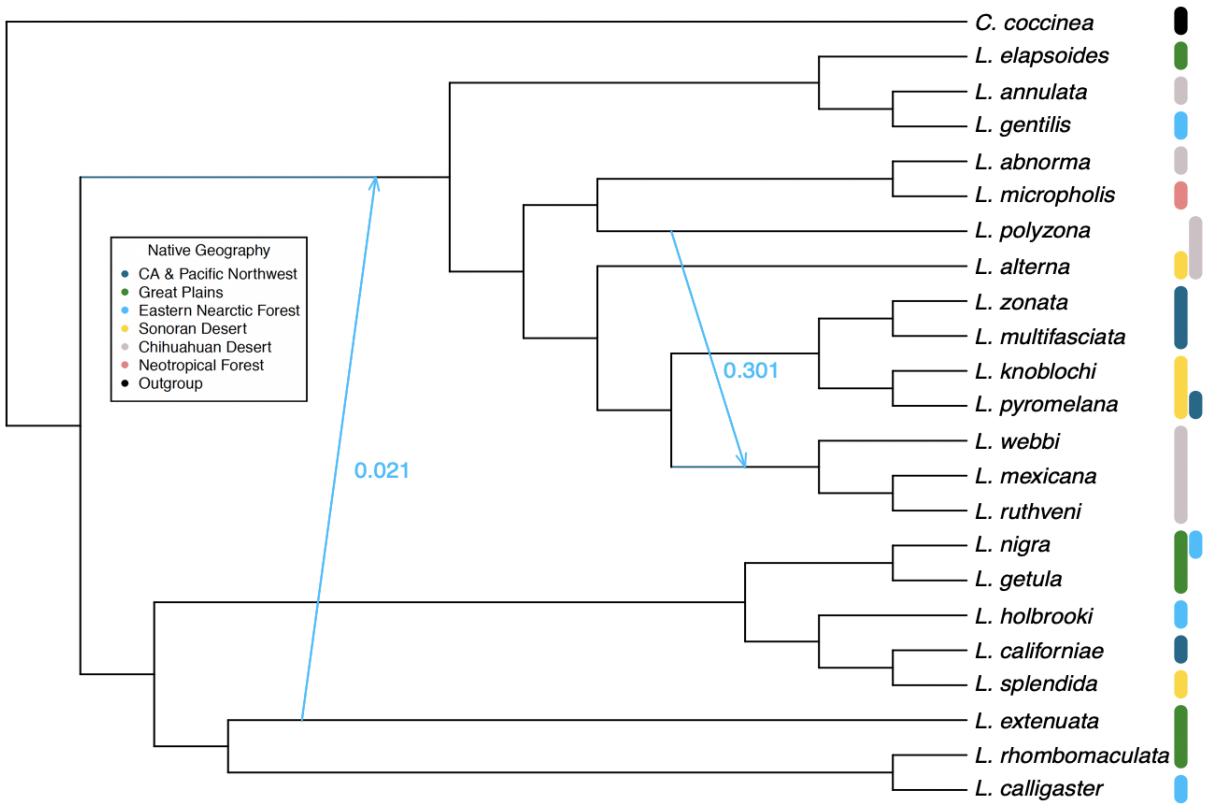

Figure 2.10: *Lampropeltis* phylogenetic network with two hybridizations, one estimating 30% lineage through the hybrid edge, one estimating 2% lineage, from Burbrink & Gehara (2018)

snakes in a network estimated using SNaQ by Burbrink & Gehara (2018). This network shows two modern hybridization events. The first is between a Chihuahuan desert snake population, *L. polyzona*, and the group of Chihuahuan desert-dwelling snake populations *L. webbi, L. mexicana, and L. ruthveni*. SNaQ estimated that 30% of their present-day lineage was passed through this hybridization. The second hybridization is estimated to be much earlier in time, before many of the branching events that led to today's species, and only 2% of genes are estimated to have been affected by this gene flow event.

Branch lengths were estimated for this network to fit pairwise distances between pairs of tips, then considered fixed. I estimated evolutionary rates and rate variation across sites using the absolute version of two Markov substitution models: the simple JC69 and the more complex HKY85.

The more complex HKY85 model fits better than the simpler JC69 model. On average, the HKY85 model increased the model's log likelihood by 10^4 compared to the JC69 model. Figure 2.11 shows that the HKY85 model improves the model fit regardless of the number of allowed reticulations in the network.

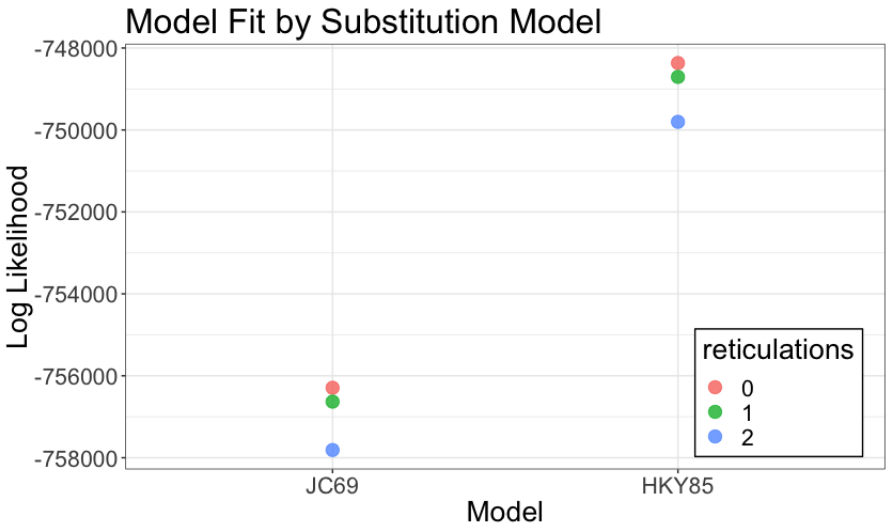

Figure 2.11: Model fit for two Markov substitution models

It is interesting to note that both models fit less well with more reticulations. While this would generally not be expected with a maximum likelihood model (more parameters should always lead to a better fit), in this case, it is likely because the branch lengths were estimated beforehand then were considered fixed. Once these branch lengths are optimized in Chapter Three alongside the network structure and model parameters, I expect that the likelihood will increase with each additional reticulation node.

Figure 2.12 shows how model fit changes with the inclusion of variation across sites. As described earlier in this chapter, the rate variation + Γ model allows some sites to evolve slowly, while other sites evolve quickly, according to a discretized Gamma distribution. Adding this flexibility improved the fit significantly, increasing the log likelihood by 10^5 on average.

In Figure 2.13, I show the computational time required to estimate evolutionary rates and rate variation. On a shared server, our method showed fast computational times for simple models and moderately fast times for more complex models. For our simplest model, JC69, I estimated evolutionary rates in an average of 5 minutes. The more complex HKY85 model required an average of 24 minutes. This

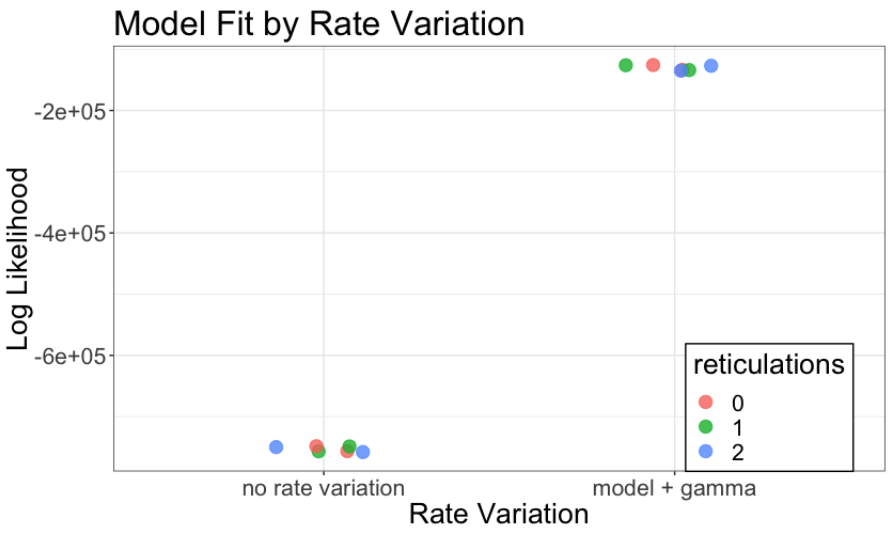

Figure 2.12: Model fit with and without rate variation across sites

increase was expected, given that the JC69 model estimates one parameter while HKY85 estimates two. The rate variation models required estimating one new parameter and optimizing the likelihood over four additional categories of the Gamma distribution. As expected, this increased the estimation time by a multiplier of four. These models were estimated in an average of 80 minutes. Increasing the number of reticulations also increased computation time, due to the increased computational time of computing the likelihood of the 2^h displayed trees within the network, where h is the number of reticulations.

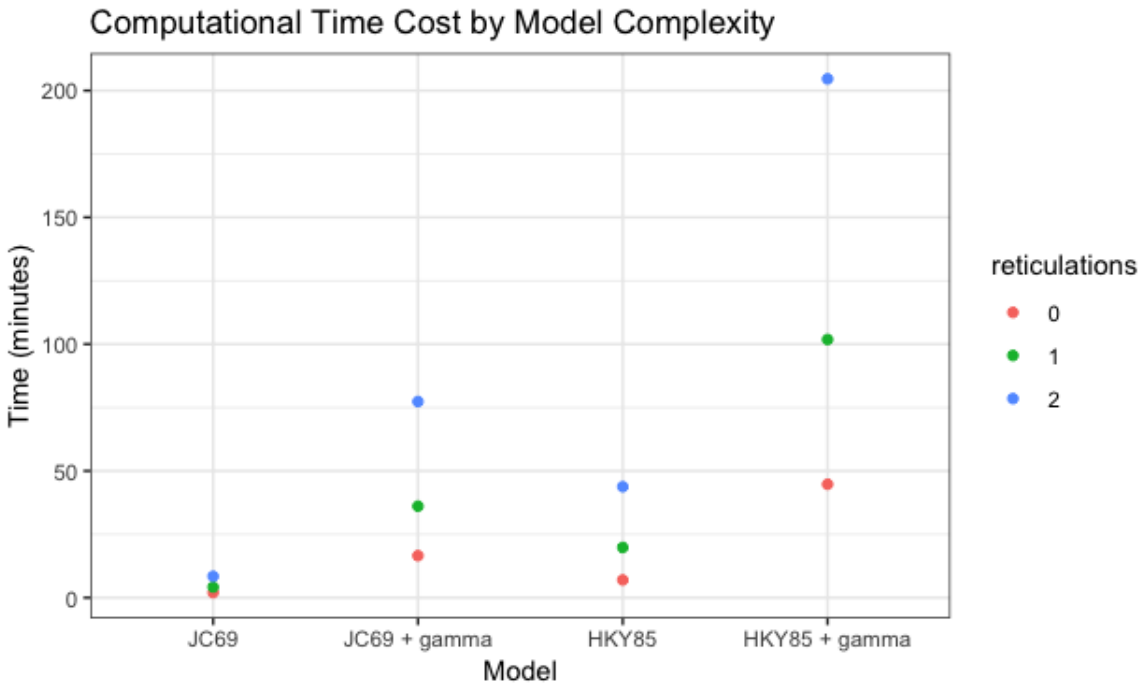

Figure 2.13: Computational time for estimating evolutionary rates and rate variation across sites

2.8 Conclusion

Our method, the first maximum likelihood concatenation approach for inferring phylogenetic networks, quickly estimates evolutionary rates from genetic data for a given network. It incorporates realistic biological assumptions about rate variation across sites and differences between transitions and transversions while allowing horizontal gene flow between species. While we have emphasized its use for concatenated sequence data here, the same theory and model can also be used for a single site or trait.

We hope this method will be a useful addition for deeper time applications, larger sets of taxa, and for biologists who prefer fast concatenation methods for phylogenetic reconstruction, allowing efficient and accurate inference of evolutionary history while acknowledging histories of horizontal gene transfer, hybridization, and introgression.

In the coming chapters, we will show how this phylogenetic network model, numerical parameter optimization, and rate estimation fit into our method's network topology search to estimate a fully parameterized phylogenetic network.

Acknowledgements. Thank you to Jingcheng Xu for his implementation of the branch length optimization.

This work was supported by National Institutes of Health award T32 HL 83806-10 and National Science Foundation awards DMS-1902892 and DMS-2023239.

CHAPTER 3

IDENTIFIABILITY OF PHYLOGENETIC NETWORKS FROM CONCATENATED SEQUENCE ALIGNMENTS

Abstract In this chapter, we present identifiability results for phylogenetic network parameter and topology estimation. We first describe the current state of knowledge in phylogenetic network identifiability from various data types. Then, we focus on the specific advantages and challenges of concatenated sequence data. Based on the model described in Chapter Two, we present five theorems on identifiability. First, we discuss branch length parameter identifiability. Next, we focus on the shape of the network, starting with root non-identifiability in semi-directed phylogenetic networks. After this, we prove that two- and three-cycle structures are not or nearly not detectable from sequence data. Finally, we discuss non-tree-child structures and the number of hybridizations in a network. After each result, we discuss practical consequences and make recommendations for network estimation. Lastly, we combine these results to present a definition for a canonical phylogenetic network from concatenated sequence data.

Keywords identifiability; Markov models; evolution; parameter estimation; speciation; gene flow; hybridization; phylogenetic networks

Contribution I led this work: theoretical proofs of identifiabilty, practical applications for topology

estimation, and I wrote the first draft. Cécile Ané and I serve as second and first authors, respectively.

3.1 IDENTIFIABILITY OF PHYLOGENETIC NETWORKS

Identifiability is a persistent challenge in phylogenetic networks reconstruction. While the identifiability of phylogenetic trees under a Markov model is well-understood, theoretical questions remain for phylogenetic networks (Chang, 1996; Allman and Rhodes, 2006; Allman *et al.*, 2008, 2011). Trying to detect and construct non-identifiable structures or numerical parameters can easily lead to infinite loops in network search optimization, wasting time and reducing the quality of the final estimate. Given the computational complexity of estimating networks, it is vital to avoid wasting time on non-identifiable parameters. To do this, we need to carefully understand which components are detectable from our data. Phylogenetic networks have many components: topology of the network, root placement, reticulation locations, branch lengths, and inheritance weights for reticulation edges. Generally, the identifiability challenges in reconstructing phylogenetic networks can be broken into two components. First, the topology of the network describes the way species are related to each other through connections between nodes and edges. This includes root location and locations of reticulations. Second, parameters within the network describe how closely each of the species, past and present, are related to each other. These estimates include branch lengths, the length of time or number of generations between nodes, and reticulation weights, how much genetic inheritance is passed on by each hybrid parent. Depending on the number of taxa, data available, and model used, some of these components may be detectable, while others may not.

Before defining identifiability of these various components, we must decide how to define a phylogenetic network. Moret *et al.* (2004) proposed a definition for the “reconstructable" phylogenetic network, a network reduced to only its identifiable components. In their formulation, a reconstructable phylogenetic network follows four rules. First, it must be a rooted, directed, acyclic graph. Second, its nodes must have certain in- and out-degree, as follows. The root node should have in-degree zero and out-degree ≥ 1. Any other node should either be a leaf, a tree node, or a reticulation node: tree nodes and leaf nodes should have in-degree of 1, while reticulation nodes should have in-degree ≥ 2 and an out-degree of one. The third condition concerns timing: two nodes must not have a conflicting placement in time. And, fourth, every edge in the network must have at least one tree node endpoint, forbidding hybrid ladders. Moret et al. also point out that if a sample is missing taxa, whether because these species were not sampled or went extinct, then certain reticulation structures will not be identifiable. While their definition of a reconstructable network does not cover all currently-understood identifiability restrictions, many of their

ideas serve as a foundation for current guidance.

Pardi and Scornavacca (2015) built on Moret *et al.* (2004)'s idea of a reconstructable network by introducing the useful definition for a canonical network. Roughly, they define a canonical network as one whose non-identifiable branches have been collapsed. Then they prove that, with data from gene trees, a network is indistinguishable from its canonical version. While some of their proofs may not apply to all situations because they require a biologically unrealistic time condition, they do prove a network can be reduced to its canonical network without reducing fit to gene tree data. They urge network reconstruction method developers to restrict their search to this canonical network to avoid attempting to "distinguish the indistinguishable." Their proofs and definition provide strong identifiability restrictions to guide topology search methods.

After defining networks overall, it is useful to turn to each component within the network in turn. First, we will focus on numerical parameters in the network: branch lengths and inheritance weights. The length of a branch corresponds to time or generations and the inheritance weight indicates how much inheritance is passed along it. Branch lengths and reticulation inheritance weights are challenging to disentangle in all settings, though their identifiability varies depending on the type of data used. When they are estimated, branch lengths and inheritance weight parameters often appear only as a linear combination.

To handle this problem, some methods ignore these branch parameters completely. Other methods set some of these parameters to zero, a constraint that makes other nearby parameters estimable. For example, in the network reconstruction tool *TreeMix*, branch lengths above a hybrid node and inheritance weights are not separately identifiable from the frequency of haplotypes (Pickrell and Pritchard, 2012). (Though their model is different from ours, their identifiability problem is similar.) To handle this, they constrain two edges adjacent to the hybrid node: the child edge and one of the parent edges. While this approach addresses identifiability challenges, the authors suggest that it may introduce bias in estimating inheritance percentages. In their simulation studies, Pickrell and Pritchard (2012) found that constraining branch lengths this way caused a downward bias in the inheritance estimate from the major parent. They hypothesize that this bias is due to constraining branches around a hybrid node to zero, especially when one or more of these branches may be large.

Some methods disentangle branch lengths and inheritance weights further. When estimating phylogenetic

networks from quartet concordance factors on gene trees, Solís-Lemus and Ané (2016) can identify hybrid parent branch lengths and inheritance values, given enough species in the network (Solis-Lemus *et al.*, 2020). Using quartet concordance factors (percentage of gene trees displaying a particular tree topology for four taxa), they estimate network topology simultaneously with gene flow events, accounting for incomplete lineage sorting. They are able to estimate inheritance weights and lengths for hybrid parent branches and all interior branches.

Let us turn our focus to the second component of a network: reticulate events. The first question is, how many hybrid events occurred during this history? Under a maximum likelihood model, every additional reticulate event adds to the model complexity and its likelihood. As the model complexity grows, the model fit, its likelihood given the data, will always improve, because the model with k reticulations is nested in the model with $k+1$ reticulations. Therefore, some penalty should be paid for each reticulation. Therefore, it is not possible to identify the number of reticulate events under a maximum likelihood setting, without some penalization procedure.

Most maximum likelihood methods avoid this problem by searching for hybridizations up to a analyst-provided threshold. Analysts generally estimate the network several times with multiple reticulation count thresholds, then compare the resulting networks.

After concluding that reticulations have occurred in our species' history, analysts will naturally wonder, when did these reticulate events happen and between which lineages? In the next section, we show that answering these two questions is easier in some cases than in others and that this difference in identifiability depends largely on one important component of the network's structure: reticulation's cycls size.

By their nature, reticulations introduce cycles, converting a tree structure into a network. A cycle is a set of nodes and edges which create an undirected circuit, wherein the first and last nodes are repeated. From a graph theory perspective, these would be termed undirected cycles, but in the phylogenetics literature, these are simply called cycles. For consistency with the phylogenetics literature, we will use the term cycle.

We categorize cycles by the number of nodes that comprise them. For example, two-cycles have two nodes: two edges originate at the same parent node, then point at the same hybrid child node (Figure 3.2).

A three-cycle has three nodes, forming a triangle. Three-cycles can be formed in two ways: with three hybrid edges (Figure 3.4) or with two hybrid edges and one tree edge (Figure 3.5).

Smaller cycles, of two and three nodes, are hard to identify, while larger cycles are easier to detect. When estimating networks with DNA sequences in the absence of incomplete lineage sorting, cycles of size four or greater were proved to be identifiable under simple models. Gross and Long (2018) prove that, if a level-1 network has only one reticulation, and it its cycle is of size four or greater, then the semi-directed network topology (without root and without parameters) can be fully estimated under the Jukes and Cantor model (Jukes and Cantor, 1969). Gross *et al.* (2020) extended this to triangle-free level-1 networks with any fixed number of reticulations under Jukes and Cantor and Kimura models (Kimura, 1980). Gross *et al.* (2020) prove that a two-cycle is not distinguishable from a tree structure and that different types of three-cycles are not distinguishable from each other. Given these findings, they suggest all small cycles should collapsed to their respective tree topology. However, they also note that the likelihood of a tree with three-cycles is not necessarily identical to that of with the cycle shrunk to the corresponding tree structure, without quantifying these potential differences.

Small cycles are challenging to detect from quartet concordance factor data as well. In the setting of level-1 networks, quartet concordance factor equations for two-cycles are identical to their corresponding tree equations, making these structures impossible to detect. Three-cycles can be detected in many but not all cases, depending on taxon sampling. Cycles of four or larger are identifiable (Solís-Lemus and Ané, 2016; Solis-Lemus *et al.*, 2020).

Pardi and Scornavacca (2015) found similar results. When reconstructing a network using perfect gene trees with branch lengths and assuming incomplete lineage sorting and no rate variation across genes, cycles of size three and larger are identifiable. Whether estimating with sequence data, concordance factors, or gene trees, detecting and handling small cycles is not yet a fully solved problem.

While small cycles are hard to detect, non-tree-child topologies are even more challenging. A network is said to be *tree-child* if every internal node has a child who is a tree node. Non-tree-child topologies break this rule: either because they contain a *W structure* (in which both children of a tree node are hybrid nodes), or a *hybrid ladder* (in which the only child of a hybrid node is itself a hybrid node). The identifiability of non-tree-child networks varies depending on model and data type. Pardi and Scornavacca (2015) show

that the order of consecutive reticulate events is not identifiable from perfectly-estimated gene trees, making some of these tree-child structures potentially challenging to fully identify. Depending on the data type, non-tree child structures can cause identifiability challenges, but their detectability from DNA sequences is not fully understood.

In summary, much progress has been made in understanding the identifiability of phylogenetic networks from various data types. The identifiability challenges fall into two main categories: parameters and topology. Branch lengths and inheritance weights are often entangled, making them challenging to estimate. Small cycles and non-tree-child structures make reconstructing network topology challenging as well. Depending on these topological structures surrounding it, each reticulation affects the identifiability of a network's topology and branch lengths differently.

In the following sections, we build on the current knowledge to further our understanding of identifiability from concatenated sequences. First, we show which parameters adjacent to reticulations are identifiable with the zipper theorem. Next, we examine the identifiability of the root placement and small cycle structures in semi-directed phylogenetic networks from sequence or single nucleotide polymorphism (SNP) data. Following these cycle identification results, we provide practical formulas for branch length and inheritance probability adjustment after shrinkage. We extend current knowledge of non-tree-child network structure to the concatenated sequence setting. After each result, we discuss how our network reconstruction software incorporates these results.

3.2 Parameter Identifiability

Branch lengths and inheritance weights make up the numerical parameters in a phylogenetic network. Respectively, they denote evolutionary time (such as calendar time, number of substitutions, or number of generations) and proportion of inheritance.

To explore the identifiability of branch lengths and inheritance values near reticulate nodes, we extend the idea of a canonical network to the concatenated sequence setting and propose the following theorem.

3.2.1 Zipper Theorem for Reticulate Node Height

I propose the following theorem on the identifiability of branch lengths below reticulate nodes.

Zipper Theorem. *From concatenated sequence data evolving under a Markov process, the length of the three edges adjacent to a hybrid node are not separately identifiable. The length of the child edge below the hybrid node may be set to 0 without changing the fit to the data.*

Let h be a hybrid node connected to hybrid parent edges e_1, e_2 with lengths t_1, t_2. Let e_c be the child edge of h with length t_c (Figure 3.1). The likelihood of the network is unchanged if t_1, t_2, and t_c are replaced by any $\tilde{t}_1, \tilde{t}_2, \tilde{t}_c$ such that $\tilde{t}_1 + \tilde{t}_c = t_1 + t_c$ and $\tilde{t}_2 + \tilde{t}_c = t_2 + t_c$. The hybrid node is said to be unzipped if the child edge has its length t_c reduced to zero and the length of each of the parent edges, t_1 and t_2, are increased by the same amount.

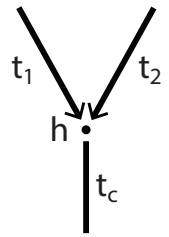

Figure 3.1: Reticulation at hybrid node h

Proof. We partition the trees displayed in the network into 3 sets, depending on whether a displayed tree has e_1, e_2, or neither: $\mathscr{T}_1 = \{T \in \mathscr{T}(N); e_1 \in T\}$, $\mathscr{T}_2 = \{T \in \mathscr{T}(N); e_2 \in T\}$ and $\mathscr{T}_3 = \{T \in \mathscr{T}(N); e_1 \notin T$ and $e_2 \notin T\}$. Note that $\mathscr{T}_3$ is empty if the network is tree-child.

Recall that the weight of a tree is the product of all edge weights in the tree: $\gamma(T) = \prod \gamma_e \in T$. Using this notation, we write the likelihood of the network as a sum of weighted likelihoods of all trees, as discussed in Chapter Two.

$$L(N) = \sum_{T_1 \in \mathscr{T}_1} \gamma(T_1)L(T_1) + \sum_{T_2 \in \mathscr{T}_\epsilon} \gamma(T_2)L(T_2) + \sum_{T_3 \in \mathscr{T}_3} \gamma(T_3)L(T_3).$$

Consider the network $\tilde{N}$ created by modifying the lengths of t_1, t_2, and t_c in the original network N to $\tilde{t}_1, \tilde{t}_2$, and $\tilde{t}_c$. For each tree T displayed in N, there is a modified tree $\tilde{T}$ displayed in $\tilde{N}$. If $T_1 \in \mathscr{T}_1$, then $e_1 \in T_1$ can be simplified by deleting node h (of degree 2 in T_1) and merging e_1 and e_c into a single edge of

length $t_1 + t_c$. Since $t_1 + t_c = \tilde{t}_1 + \tilde{t}_c$, the simplified versions of T_1 and $\tilde{T}_1$ are identical, so the likelihoods are equal: $L(T_1) = L(\tilde{T}_1)$. Similarly, since $t_2 + t_c = \tilde{t}_2 + \tilde{t}_c$, $L(T_2) = L(\tilde{T}_2)$ for any $T_2 \in \mathcal{T}_2$. Finally, if $T_3 \in \mathcal{T}_3$, then T_3 contains none of e_1, e_2, e_c or h, so it is unaffected by changes in their branch lengths: $T_3 = \tilde{T}_3$, and $L(T_3) = L(\tilde{T}_3)$. Since the likelihoods of the displayed trees are identical, their weighted sum is identical. Thus, the likelihood of the network $\tilde{N}$ is equal to that of the original network N: $L(N) = L(\tilde{N})$. This shows that while the combinations $t_1 + t_c$ and $t_2 + t_c$ might be identifiable, t_c, t_1, and t_2 are not separately identifiable. □

Practical Consequences

The height of a hybrid node cannot be identified. Therefore, attempting to identify these branch lengths when reconstructing a phylogenetic network from concatenated sequences is likely to impede the quality and speed of parameter estimation. Instead, our estimation software sets the length of any edge below a hybrid node to zero ($t_c = 0$ in Figure 3.1) and we communicate this to analysts. We refer to this format as the network's canonical, "unzipped" form, after the theorem's name.

Note: If the child edge is assumed to have length $t_c = 0$ and if the rest of the network is known, then the branch lengths of the two parent edges (t_1 and t_2) and their inheritance weights is identifiable from gene trees (Pardi and Scornavacca, 2015). However, it is unknown if this identifiability is maintained with concatenated sequence data instead of gene tree data. We conjecture that it is indeed the case.

3.3 Topological Identifiability

The identifiability of the topology is the main focus of most phylogenetic history reconstruction. In the following section, we focus on four components of topology: root placement, cycle detection, non-tree-child structures, and choosing a total number of hybridizations.

3.3.1 Root Placement in Semi-Directed Networks

Networks can be rooted, unrooted, or semi-directed. As introduced by Solís-Lemus and Ané (2016), a semi-directed network gives direction only to hybrid edges, not tree edges, and suppresses the root node. While the root is undetermined in a semi-directed network, its placement is restricted because it cannot be the descendant of any reticulation.

The root placement in phylogenetic trees is not identifiable under reversible models of sequence evolution (Felsenstein, 2004). Gross and Long (2018) claim that the root is not identifiable in phylogenetic networks with a single reticulation, but others have said that the root placement may be be partially identifiable (Pickrell and Pritchard, 2012). In the following theorem, we prove that the location of the root within a network is constrained but not identifiable from concatenated sequence data.

Root Placement Theorem. *Under a reversible substitution model with a stationary distribution, the root of a phylogenetic network is not identifiable.*

Proof. Consider a network N_1. Next consider network N_2 obtained from N_1 by changing the root position to a different node, not below any reticulation. In other words, N_1, N_2 are two rooted networks with identical semi-directed versions. They have the same hybrid edges and the same hybrid nodes. Therefore, they have the same *unrooted* displayed trees.

Since the likelihood of N_1 and N_2 are calculated using a time-reversible model and stationary distribution at the root, the likelihood of the displayed trees within these networks, $\mathscr{T}_1$ and $\mathscr{T}_2$, respectively, does not depend on the root placement in the trees, so these trees can be considered as unrooted (Felsenstein, 2004). After removing their root, the displayed trees and inheritance weights within the first network are identical to those of the second. Because the likelihood of a network is simply the weighted sum of the likelihood of the displayed trees, the likelihoods of the two networks are identical. If the likelihoods of two networks with different roots are identical, then the root placement cannot be identified. □

Practical Consequence

As a consequence of this theorem, our software estimates a semi-directed network without identifying one root placement. Our software accounts for this restriction during topology optimization, moving the root only to permissible locations in the network. The estimated output network is output with an arbitrary root. This fact is communicated to analysts, who are advised to root their network using a known outgroup species.

3.3.2 TWO-CYCLE IDENTIFIABILITY

In a two-cycle, two nodes are connected by parallel edges. The following theorem shows that these cycles cannot be detected in phylogenetic networks from sequence or single nucleotide polymorphism (SNP) data. We prove this by showing that the likelihood of the network is unchanged (or nearly unchanged, depending on the model) by shrinking a two-cycle down to a single edge. This holds for networks with and without time consistency. First, we define time consistency.

Definition 4. Time-Consistency

1. A rooted network is *time consistent* if, for every node, all paths from the root to that node have equal length.

2. An semi-directed network is *time consistent* if there exists a node such that the network rooted at that node is time consistent.

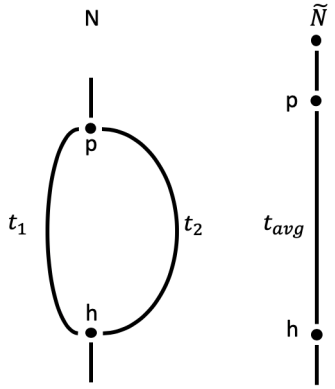

Figure 3.2: Full two-cycle and collapsed two-cycle in a semi-directed network

Two Cycle Identifiability Theorem. *1. Let N be a semi-directed network with a two-cycle and let p, h be the nodes forming this cycle. These nodes are connected by edges e_1 and e_2 with lengths t_1 and t_2. We can create a corresponding network, $\tilde{N}_t$, by replacing e_1 and e_2 in N with a single edge e from nodes p to h, with new length t.*

(a) If $t_1 = t_2$, then $L(N) = L(\tilde{N}_{t_1})$ and the two-cycle is not identifiable.

(b) *Under the Jukes-Cantor substitution model,* $L(N) = L(\tilde{N}_{\tilde{t}})$ *for*

$$\tilde{t} = -\frac{3}{4}\ln(\gamma_1 e^{-\frac{4}{3}t_1} + \gamma_2 e^{-\frac{4}{3}t_2}).$$

Consequently, the two-cycle is not identifiable regardless of t_1 *and* t_2*. Note that* $\tilde{t} \sim \gamma_1 t_1 + \gamma_2 t_2$ *as* $t_1 - t_2 \to 0$*.*

(c) *The two-cycle is nearly not identifiable as the difference between* t_1 *and* t_2 *is relatively small, in the following sense: If* $t_{av} = \gamma_1 t_1 + \gamma_2 t_2$*, then* $L(N) = L(\tilde{N}_{t_{av}}) + O(t_2 - t_1)^2$*.*

2. *Let N be a semi-directed network with one or more two-cycles. If the network N is time-consistent, then the hybrid edge branch lengths in each two-cycle are not identifiable and any two-cycle structure is not identifiable. Similarly, if the network is nearly time consistent, two-cycles within the network are nearly non-identifiable.*

Proof. As in the proof of the zipper theorem, we partition the set of displayed trees $\mathcal{T}(N)$. This time, we pair trees created by the reticulation node of interest. To make this pairing precise, we introduce additional notation.

First, we order the r hybrid nodes in such a way that the first node is the hybrid node creating the two-cycle of interest. For each reticulation node, its two hybrid parent edges are given an arbitrary order, 1 and 2. For each $(j_1, ..., j_r) \in \{1,2\}^r$, let $T_{j_1,...,j_r}$ be the displayed tree obtained from N by keeping parent edge j_k for hybrid node number k.

To focus on the two-cycle within the network, we pair the two displayed trees created by choosing e_1 versus e_2, in the cycle of interest. These two trees, $T_1 = T_{1,j_2,...,j_r}$ and $T_2 = T_{2,j_2,...,j_r}$, have the same topology, since edges e_1 and e_2 run parallel, sharing parent and child nodes. The only possible difference between T_1 and T_2 is their branch lengths: T_1 has e_1 connecting p and h, with length t_1, whereas T_2 has e_2 connecting p and h, with length t_2.

On the network $\tilde{N}_{\tilde{t}}$ with the parallel edges collapsed into a single edge, one can similarly list all displayed trees:

$$\mathcal{T}(\tilde{N}_{\tilde{t}}) = \{\tilde{T}_{j_2,...,j_r}; (j_2, ..., j_r) \in \{1,2\}^{r-1}\}.$$

This ordering of displayed trees provides a pairing with displayed trees from N.

Note that if N is not tree-child, it is possible that for some choice of $j_2, ..., j_r$, neither $T_{1,j_2,...,j_r}$ nor $T_{2,j_2,...,j_r}$ have the edges and nodes of interest, e_1, e_2, h and p. In this case, $T_{1,j_2,...,j_r} = T_{2,j_2,...,j_r}$ and they are equal to the corresponding $\tilde{T}_{j_2,...,j_r}$ in the network $\tilde{N}_{\tilde{t}}$ with parallel edges collapsed, so their likelihoods will be equal.

Using this pairing notation, the full likelihood of the network is

$$L(N) = \sum_{(j_2,...,j_r)\in\{1,2\}^{r-1}} \Big(\prod_{k=2}^{r} \gamma_{k,j_k}\Big)\Big[\gamma_{1,j_1} L(T_{1,j_2,...,j_r}) + (1-\gamma_{1,j_1}) L(T_{2,j_2,...,j_r})\Big] \tag{3.1}$$

Summing from $j_2, ..., j_r$ excludes the parent edges involved in the two cycle, allowing us to explicitly combine the contribution of the two-cycle edges to the likelihood as a sum of the paired displayed trees.

(1a) If $t_1 = t_2$, then $T_{1,j_2,...,j_r}$ and $T_{2,j_2,...,j_r}$ have not only the same topology, but also the same branch lengths, so they have the same likelihood, and also the same likelihood as $\tilde{T}_{j_2,...,j_r}$ with the choice $\tilde{t} = t_1 = t_2$. Therefore:

$$\begin{aligned} L(N) &= \sum_{(j_2,...,j_r)\in\{1,2\}^{r-1}} \Big(\prod_{k=2}^{r} \gamma_{k,j_k}\Big)\Big[\gamma_{1,j_1} L(T_{1,j_2,...,j_r}) + (1-\gamma_{1,j_1}) L(T_{2,j_2,...,j_r})\Big] \\ &= \sum_{(j_2,...,j_r)\in\{1,2\}^{r-1}} \Big(\prod_{k=2}^{r} \gamma_{k,j_k}\Big) L(\tilde{T}_{j_2,...,j_r}) = L(\tilde{N}_{\tilde{t}})\,, \end{aligned}$$

proving part (a). This likelihood equality between N and $\tilde{N}_{\tilde{t}}$ proves that the topology is not identifiable. Further, even if we had additional information to guarantee that a two-cycle did exist here within the network, the inheritance weights and branch lengths would be non-identifiable. Instead, only their combination $t_a + \tilde{t} + t_b$ with the branch lengths above and below the two-cycle would be identifiable. The topology, branch lengths and γ inheritance values within the two-cycle are not identifiable, finishing the proof of (a).

(1b) Under the Jukes-Cantor model, the transition rate matrix is $Q = U\Lambda U^{-1}$ with

$$\Lambda = \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & \lambda \end{bmatrix}$$

and $\lambda = -\frac{4}{3}$. Let $\tilde{t}$ such that $e^{\lambda\tilde{t}} = \gamma_1 e^{\lambda t_1} + \gamma_2 e^{\lambda t_2}$, that is:

$$\tilde{t} = -\frac{3}{4}\ln(\gamma_1 e^{\frac{-4}{3}t_1} + \gamma_2 e^{-\frac{4}{3}t_2}).$$

Then, $e^{\tilde{t}Q} = \gamma_1 e^{t_1 Q} + \gamma_2 e^{t_2 Q}$ so $P_{\tilde{t}} = \gamma_1 P_{t_1} + \gamma_2 P_{t_2}$ and then

$$\gamma_1 L(T_{1,j_2,\ldots,j_r}) + \gamma_2 L(T_{2,j_2,\ldots,j_r}) = L(\tilde{T}_{j_2,\ldots,j_r}).$$

Consequently, $L(N) = L(\tilde{N}_{\tilde{t}})$, which finishes the proof of (b).

(1c) In general, the trees $T_{1,j_2,\ldots,j_r}$ and $T_{2,j_2,\ldots,j_r}$ have the same topology, but may differ in their branch lengths (unless e_1 and e_2 are absent from both). To focus on these two displayed trees and simplify notations, we drop the subscripts $j_2,\ldots,j_r$ here. Because this is the only component of the likelihood in equation 3.1 affected by the two-cycle topology, we just need to prove that this component is almost unchanged by the shrinking the two-cycle, with

$$\gamma_{j_1} L(T_1) + (1-\gamma_{j_1}) L(T_2) = L(\tilde{T}) + O(t_2 - t_1)^2.$$

Since the root position does not affect the likelihood of a tree (Felsenstein, 2004), I calculate the likelihood of T_1, T_2 and $\tilde{T}$ by rooting them at node p, without loss of generality (see Figure 3.3).

Let D_h (and D_p) be the data at the tips descending from h (and from the sister of h below p) in T_1. They are the same in T_2 and in $\tilde{T}$. Then, let

$$f_h(x) = P\{D_h | x_h = x\} \text{ and } f_p(x) = P\{D_p | x_p = x\},$$

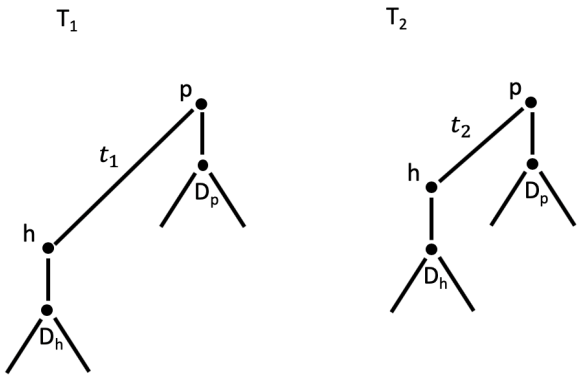

Figure 3.3: Displayed trees T_1 and T_2, rooted at node p

where $f(x)$ are directional likelihoods, calculated with Felsenstein's pruning algorithm. These directional likelihoods depend on $(j_2, ..., j_r)$, but are the same in T_1, T_2 and $\tilde{T}$ since they depend on subtrees that are identical in all three trees. With these directional likelihoods, the full likelihood of each displayed tree, T_i, can be expressed as

$$L(T_i) = \sum_{x,y \in \{A,C,G,T\}} \pi(x) f_p(x) P_{t_i}(x,y) f_h(y).$$

Replacing P_{t_i} by $e^{t_i Q}$ gives

$$L(T_i) = \sum_{x,y \in \{A,C,G,T\}} \pi(x) f_p(x) e^{t_i Q}(x,y) f_h(y).$$

After taking a weighted sum of $L(T_1)$ and $L(T_2)$ and combining like terms, their contribution $\gamma_{1,j_1} L(T_1) + (1-\gamma_{1,j_1}) L(T_2)$ to the likelihood reduces to

$$\sum_{x,y \in \{A,C,G,T\}} \pi(x) f_p(x) f_h(y) [\gamma_{j_1} e^{t_1 Q} + (1-\gamma_{j_1}) e^{t_2 Q}]_{x,y}.$$

Replacing the two branch lengths t_1 and t_2 with their weighted average, $t_{\text{av}} = \gamma_1 t_1 + (1-\gamma_1) t_2$, gives

$$\gamma_{j_1} e^{t_1 Q} + (1-\gamma_{j_1}) e^{t_2 Q} = e^{t_{\text{av}} Q} + O(t_2 - t_1)^2. \tag{3.2}$$

Therefore, $L(N) = L(\tilde{N}_{t_{\text{av}}}) + O(t_2 - t_1)^2$, which completes part (c).

To prove equation 3.2, we use a Taylor expansion of $e^{t_1 Q}$ and of $e^{t_2 Q}$ when $t_i \sim t_{\text{av}}$, shown below. We can write $t_1 = t_{\text{av}} + \gamma_2(t_1 - t_2)$ so

$$e^{t_1 Q} = e^{t_{\text{av}} Q}[e^{I + \gamma_2(t_1 - t_2)Q}] = e^{t_{\text{av}} Q}[I + \gamma_2(t_1 - t_2)Q + O(t_1 - t_2)^2].$$

Similarly, $t_2 = t_{\text{av}} + \gamma_1(t_2 - t_1)$ so

$$e^{t_2 Q} = e^{t_{\text{av}} Q}[I + \gamma_1(t_2 - t_1)Q] + O(t_1 - t_2)^2.$$

Together,

$$\gamma_1 e^{t_1 Q} + \gamma_2 e^{t_2 Q} = e^{t_{\text{av}} Q}[I + \gamma_1\gamma_2(t_1 - t_2)Q + \gamma_2\gamma_1(t_2 - t_1)Q] + O(t_1 - t_2)^2$$

The expression within the bracket cancels out, leaving

$$\gamma_1 e^{t_1 Q} + \gamma_2 e^{t_2 Q} = e^{t_{\text{av}} Q} + O(t_1 - t_2)^2.$$

Therefore, the two-cycle is nearly non-identifiable under any Markov model, even if $t_1 \neq t_2$. The difference between the original network's likelihood and the shrunk likelihood is $O(t_1 - t_2)^2$.

(2) If a network is time-consistent, then the branch lengths t_1 and t_2 must be equal. Because $t_1 = t_2$, this is proved by the above proof 1(a).

□

Note on Rate Variation These results assume a model without rate variation across sites in the data. Under a model with rate variation across sites, parts (a) and (c) still hold. In parts (a) and (b), the branch lengths t_1 and t_2 can be replaced by $r\,t_1$ and $r\,t_2$, where r may depend on the site, without affecting the proof. In (1a), $r\,t_1 = r\,t_2$ holds if $t_1 = t_2$. In (1c), the weighted average of $r\,t_1$ and $r\,t_2$ is $r\,t_{\text{av}}$ where t_{av} is independent of the site, so the proof holds in this case as well.

To summarize, two-cycles and their numerical parameters are not identifiable in time consistent networks. For networks without time consistency, this remains true up to the first order approximation. To show this, we proved that networks with and without a two-cycle yields identical or nearly identical likelihoods.

In other words, the likelihood is flat or nearly flat over this topology change.

PRACTICAL CONSEQUENCES

Distinguishing between networks with and without two-cycles is impossible (or nearly impossible). Attempts to identify these structures may lead to persistent switching from the two-cycle structure to its corresponding shrunk structure and back. Due to this switching, any topology search allowing for two-cycles will be inefficient and could lead to optimization slowdown or failure. To avoid this, we restrict our software's topology search to the space of networks without two-cycles. We describe the process of removing and avoiding two-cycles in detail in Chapter Four.

3.3.3 THREE CYCLE IDENTIFIABILITY

A three-cycle structure is created when a hybrid node h has parents that are connected but not identical. Two types of three-cycles can arise in a phylogenetic network, which we call type 1 and type 2. A *type-1 three-cycle* is made up of one tree node and two hybrid nodes (Figure 3.4). A *type-2 three-cycle* contains two tree nodes and one hybrid node (Figure 3.5). Like two-cycles, three-cycles are a challenging topology to identify in phylogenetic networks. When using pairwise distances, they are fully non-identifiable, but, in the case of concatenated sequence data, they are only nearly non-identifiable. Removing a three-cycle can change the network's likelihood, but that change is often imperceptibly small.

In the following section, we prove theorems on the identifiability of three-cycles from concatenated sequence data. We use ideas similar to those used in the two-cycle theorem above, comparing intact three-cycle structures with those that have been collapsed or shrunk. While this process of shrinking a three-cycle has been called a triangle operation (Francis *et al.*, 2018), we call it "shrinking" for clarity, in parallel with the two-cycle case.

In addition to proving that these structures are nearly non-identifiable, we give branch length conversions for use in estimation methods. The proposed post-shrinkage edge lengths have been set such that pairwise distances are identical under N and the shrunk network $\tilde{N}$. Therefore, N and $\tilde{N}$ are perfectly indistinguishable with pairwise distances and nearly indistinguishable under full alignment likelihood.

Three Cycle Identifiability. *(a) Type-1 Three-Cycle Identifiability*

Let N be a semi-directed network with a three-cycle containing two hybrid nodes (Figure 3.4). Let $\tilde{N}$

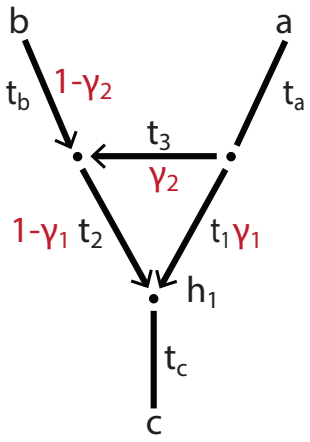

Figure 3.4: Type-1 Three-Cycle Structure

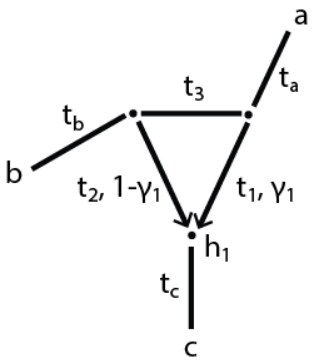

Figure 3.5: Type-2 Three-Cycle Structure

be the network obtained from N by shrinking the cycle. That is, remove edge t_3 into the top hybrid node, set the inheritance weight of parent a to

$$\tilde{\gamma} = \gamma_1 + (1-\gamma_1)\gamma_2, \text{ that is, } 1-\tilde{\gamma} = (1-\gamma_1)(1-\gamma_2)$$

and adjust the branch lengths (Figure 3.6) using these pairwise distance-based conversions:

$$\begin{aligned} \tilde{t}_a &= t_a + \frac{\gamma_1 t_1 + (1-\gamma_1)\gamma_2(t_2+t_3)}{\tilde{\gamma}} \\ \tilde{t}_b &= t_b + t_2\,. \end{aligned}$$

Then

$$L(N) = L(\tilde{N}) + O((t_1 - t_2 - t_3)^2).$$

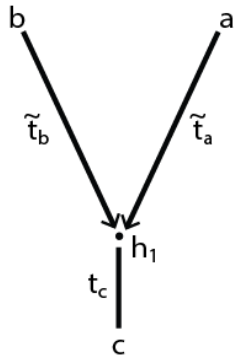

Figure 3.6: Modified Network $\tilde{N}$ with Collapsed Type-1 Three-Cycle

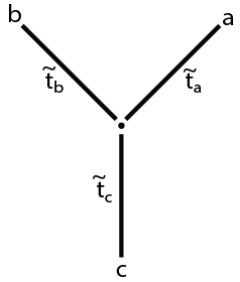

Figure 3.7: Modified Network $\tilde{N}$ with Collapsed Type-2 Three-Cycle

(b) Type-2 Three-Cycle Identifiability

Let N be a semi-directed network with a type-2 three-cycle containing one hybrid node (Figure 3.5). Let $\tilde{N}$ be the network obtained from N by shrinking the cycle. That is, add a new node in the center and replace the cycle edges with new edges connected to this central node (Figure 3.7), with branch lengths adjusted using the following pairwise distance-based conversions:

$$\tilde{t}_a = t_a + (1-\gamma)\,t_3$$

$$\tilde{t}_b = t_b + \gamma\, t_3$$

$$\tilde{t}_c = t_c + \gamma\, t_1 + (1-\gamma)\,t_2.$$

After shrinking, all three edges become tree edges, with γ values equal to 1 (Figure 3.7). Then

$$L(N) = L(\tilde{N}) + O(\max\{(t_1 - t_2)^2, t_3\}).$$

To a first order approximation in branch lengths, N and $\tilde{N}$ are not distinguishable, so both type-1 and type-2 three-cycle structure are nearly not identifiable.

Note on the Approximation Accuracy When $t_3 = 0$, the 3-cycle degenerates into a 2-cycle and the approximation term becomes $O((t_2 - t_1)^2)$, as in the two-cycle theorem. In the type-1 three-cycle, t_3 does not need to be small for the approximation to hold. In fact, if the network is time consistent, then we must have that $t_1 = t_2 + t_3$, so $L(N) = L(\tilde{N})$ exactly. Consequently, type-1 three-cycles are non-identifiable in time-consistent networks.

Note on Distances Branch lengths $\tilde{t}$ on $\tilde{N}$ are called "pairwise-distance-based" conversions because they maintain average distances between a and b, b and c, and a and c when converting N into $\tilde{N}$. Here, we consider distances on each displayed tree and average across all displayed trees with the same topology. For example, on a type-2 three-cycle in which all displayed trees share the same topology, the new branch lengths $\tilde{t}$ are determined by pairwise distances using

$$\begin{aligned}
\tilde{t}_a &= d(a,c) + d(a,b) - d(b,c) = t_a + (1-\gamma_1)\,t_3, \\
\tilde{t}_b &= d(b,c) + d(b,a) - d(a,c) = t_b + \gamma_1\,t_3, \\
\tilde{t}_c &= d(c,a), + d(c,b) - d(a,b) = t_c + \gamma_1\,t_1 + (1-\gamma_1)\,t_2.
\end{aligned}$$

Note on Rate Variation As with two-cycle topologies, this three-cycle theorem holds for models incorporating rate variation, because the branch length transformations are linear: multiplying the original branch lengths t_1, t_2, t_a, t_b, and t_c by a rate multiplies the new branch lengths $\tilde{t}_a, \tilde{t}_b$, and $\tilde{t}_c$ by that same rate.

Proof. I pair the displayed trees, as in the two-cycle theorem proofs. First, we order the r hybrid nodes so that the nodes creating the three-cycle of interest are first. For a type-1 three-cycle, two nodes, h_1 and h_2, appear in the three-cycle structure. For a type-2 three-cycle, only node h_1 appears in the three-cycle structure. For each reticulation, the two parent edges are given an arbitrary order 1 and 2. For each $(j_1, ..., j_r) \in \{1,2\}^r$, let $T_{j_1,...,j_r}$ be the displayed tree obtained from N by keeping parents edges j_k for hybrid node number k.

To focus on a given three-cycle within the network, we group the displayed trees created by the focus node or nodes. In a type-1 three-cycle structure, there are two hybrid nodes, h_1 and h_2, and four displayed trees: $T_{1,1,j_3,...,j_r} = T_{1,2,j_3,...,j_r}$, $T_{2,1,...,j_r}$ and $T_{2,2,j_3,...,j_r}$ (see Figure 3.8). Of the three distinct displayed trees, the first two have the same topology, with possibly different branch lengths. The third one has a different topology. For clarity, we will refer to these three trees as T_1, $T_{2,1}$, and $T_{2,2}$ with associated γ inheritance values γ_1, $(1-\gamma_1)\gamma_2$ and $(1-\gamma_1)(1-\gamma_2)$.

In a type-2 three-cycle, there is one hybrid node, h, and two displayed trees (see Figure 3.9). These two trees, $T_{1,j_2,...,j_r}$ and $T_{2,j_2,...,j_r}$, have the same topology. The only possible difference between these two trees is their branch lengths: $T_{1,j_2,...,j_r}$ has e_1 connecting p and h, with length t_1, whereas $T_{2,j_2,...,j_r}$ has e_2 connecting p and h, with length t_2. For clarity, we refer to these two displayed trees as T_1 and T_2 with inheritance values γ_1 and $(1-\gamma_1)$.

(a) Type-1 Three-Cycle Identifiability

Figure 3.8 shows the local topology of the displayed trees and their inheritance weights. While

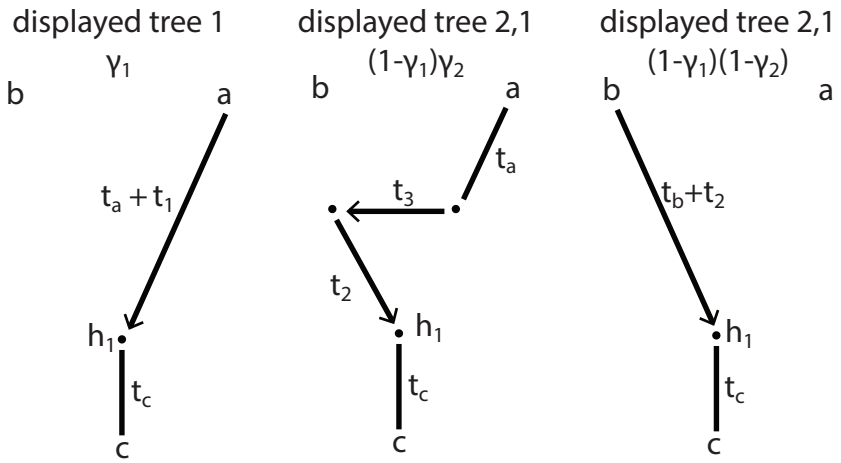

Figure 3.8: Displayed Trees within a Type-1 Three Cycle Structure

the topologies of T_1 and $T_{2,1}$ are identical, they may have different branch lengths. To show that the likelihood of the network is nearly equivalent after shrinking the three-cycle, we focus only on the local contribution of the three-cycle to the likelihood, called L_{3c}. Using the pairing notation introduced above and the decomposition from Chapter Two, the full likelihood of the network is as shown below. Summing from $j_2, ..., j_r$ excludes the parent edges involved in the three cycle, allowing us to explicitly combine the contribution of the edges in the three-cycle as a sum of likelihoods from

the three local displayed trees.

$$L(N) = \sum_{(j_3,...,j_r)\in\{1,2\}^{r-2}} \Big(\prod_{k=3}^{r} \gamma_{k,j_k}\Big) L_{3c}(N, j_3, ..., j_r)$$

where

$$\begin{aligned} L_{3c}(N, j_3, ..., j_r) &= \gamma_1 L(T_{1,\cdot,j_3,...,j_r}) + (1-\gamma_1)\gamma_2 L(T_{2,1,j_3,...,j_r}) + \\ &\quad (1-\gamma_1)(1-\gamma_2) L(T_{2,2,j_3,...,j_r}) . \end{aligned}$$

After shrinking the three-cycle, this contribution is denoted $L_{3c}(\tilde{N}, j_3, ..., j_r)$. To ease notations, we drop the $j_3, ..., j_r$ conditioning below. We need to show that the likelihood is nearly unchanged by shrinking the type-1 three-cycle in N to create network $\tilde{N}$, with the choice of $(\tilde{t}_a, \tilde{t}_b)$ in (a), in the sense that:

$$L_{3c}(N) = L_{3c}(\tilde{N}) + O(t_1 - t_2 - t_3)^2 . \tag{3.3}$$

Since h_1 is a reticulation, the root of N cannot be located below h_1. Therefore, there must be a undirected path between a and b that does not include any of the edges shown in Figure 3.4. We can consider the bipartition of tips created by removing edge t_c. In all three displayed trees T_1, $T_{2,1}$ and $T_{2,2}$, this bipartition separates data D_c below node c, from data D_{ab} connected to a and b. Note that D_c and D_{ab} depend on the choice of $j_3, \cdots, j_r$, which we consider as fixed here. Since the likelihood of a tree does not depend on the root position, we may consider T_1, $T_{2,1}$ and $T_{2,2}$ to be rooted at h_1. We decompose their likelihoods into the directional likelihood along edge t_c

$$f_c(x) = P\{D_c | x_{h_1} = x\},$$

and the marginal likelihoods of data connected to a and b, D_{ab},

$$g_a(x) = P\{D_{ab} | x_a = x\} \text{ and } g_b(x) = P\{D_{ab} | x_b = x\}.$$

Considering components $f_c(x)$, $g_a(x)$, and $g_b(x)$, the likelihoods for each displayed tree are

$$\begin{aligned} L(T_1) &= \sum_{z\in\{A,C,G,T\}} \pi(z) f_c(z) P_{t_1+t_a}(z,x) g_a(x) \\ L(T_{2,1}) &= \sum_{z\in\{A,C,G,T\}} \pi(z) f_c(z) P_{t_2+t_3+t_a}(z,x) g_a(x) \text{ and} \\ L(T_{2,2}) &= \sum_{z\in\{A,C,G,T\}} \pi(z) f_c(z) P_{t_2+t_b}(z,y) g_b(y)\,. \end{aligned}$$

Note that f_c, g_a and g_b only depend on $j_3,\cdots,j_r$, but are the same on T_1, $T_{2,1}$ and $T_{2,2}$, and are also the same on the two displayed trees $\tilde{T}_1$ and $\tilde{T}_2$ on the reduced network $\tilde{N}$. The likelihood of $\tilde{T}_1$ and $\tilde{T}_2$ can similarly be expressed:

$$\begin{aligned} L(\tilde{T}_1) &= \sum_{z\in\{A,C,G,T\}} \pi(z) f_c(z) P_{\tilde{t}_a}(z,x) g_a(x) \\ L(\tilde{T}_2) &= \sum_{z\in\{A,C,G,T\}} \pi(z) f_c(z) P_{\tilde{t}_b}(z,y) g_b(y)\,. \end{aligned}$$

Since $\tilde{t}_b = t_b + t_2$, $L(T_2) = L(\tilde{T}_2)$. Using the approximation derived in the proof of the two-cycle theorem and $\tilde{\gamma} = \gamma_1 + (1-\gamma_1)\gamma_2$, we get

$$\begin{aligned} \frac{\gamma_1}{\tilde{\gamma}} P_{t_1+t_a} + \frac{(1-\gamma_1)\gamma_2}{\tilde{\gamma}} P_{t_2+t_3+t_a} &= P_{\frac{\gamma_1}{\tilde{\gamma}}(t_1+t_a)+\frac{(1-\gamma_1)\gamma_2}{\tilde{\gamma}}(t_2+t_3+t_a)} + O((t_1+t_a)-(t_2+t_3+t_a))^2 \\ &= P_{\tilde{t}_a} + O(t_1-t_2-t_3)^2 \end{aligned}$$

which implies that

$$\gamma_1 L(T_1) + (1-\gamma_1)\gamma_2 L(T_{2,1}) = \tilde{\gamma} L(\tilde{T}_1) + O(t_1-t_2-t_3)^2\,.$$

Finally,

$$L_{3c}(N) = L_{3c}(\tilde{N}) + O(t_1-t_2-t_3)^2$$

which completes the proof of (a) for type-1 three-cycles.

(b) Type-2 Three-Cycle Identifiability

First, we apply the zipper theorem to zip the reticulation fully at h_1 until the smaller branch length of t_1 or t_2 becomes 0, and the larger of t_1, t_2 becomes $|t_1 - t_2|$. As shown in the Zipper Theorem proof, this modification of branch lengths does not change $L(N)$, so we only need to prove part (b)

of the theorem with an approximation term of $O(\max\{t_1, t_2, t_3\}^2)$.

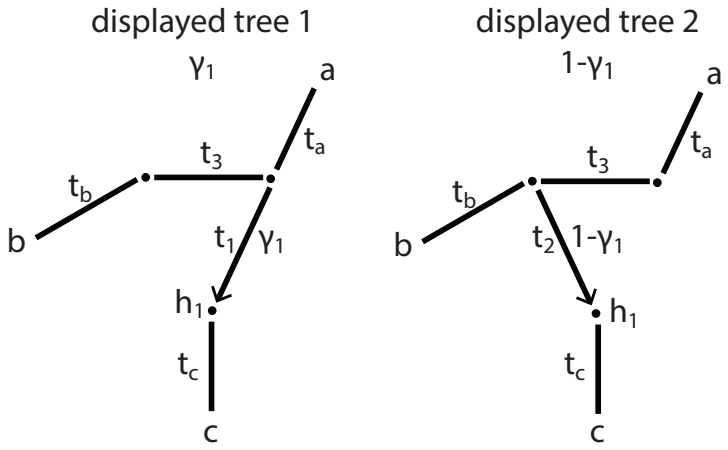

Figure 3.9: Displayed Trees within a Type-2 Three Cycle Structure

Each type-2 three-cycle has two displayed trees, T_1 and T_2, one for each of the inheritance weights (Figure 3.9). Using the pairing notation, the full likelihood of the network is

$$L(N) = \sum_{(j_2,\ldots,j_r)\in\{1,2\}^{r-1}} \Big(\prod_{k=2}^{r} \gamma_{k,j_k}\Big) L_{3c}(N, j_2, \ldots, j_r)$$

where $L_{3c}(N, j_2, \cdots, j_r)$ as defined, similarly to part (a), as

$$L_{3c}(N, j_2, \cdots, j_r) = \gamma_1 L(T_{1,j_2,\cdots,j_r}) + (1-\gamma_1) L(T_{2,j_2,\cdots,j_r}).$$

To prove (b), we show that, for every $j_2, \ldots, j_r$,

$$L_{3c}(N, j_2, \ldots, j_r) = L_{3c}(\tilde{N}, j_2, \ldots, j_r) + O(\max\{t_1, t_2, t_3\})^2$$

where $\tilde{N}$ is as in Figure 3.7, obtained from N by replacing the three-cycle with a single node and three branches with length $\tilde{t}_a = t_a + (1-\gamma_1)t_3$, $\tilde{t}_b = t_b + \gamma_1 t_3$, and $\tilde{t}_c = t_c + \gamma_1 t_1 + (1-\gamma_1)t_2$, and for which

$$L_{3c}(\tilde{N}, j_2, \cdots, j_r) = L(\tilde{T}_{j_2,\cdots,j_r}).$$

As before, we drop the j_k indices for convenience and denote $T_j = T_{j,j_2,\cdots,j_r}$ and $\tilde{T} = T_{j_2,\cdots,j_r}$.

Without loss of generality, we can consider the displayed trees T_1, T_2 and $\tilde{T}$ rooted at node a. Let D_c

denote data at tips descending from c in T_1 (or T_2 or $\tilde{T}$), D_b denote data at tips descending from b, then $D_a = D \setminus \{D_b \cup D_c\}$. Similarly to two-cycle theorem and three-cycle part (a) proofs, we let f_c, f_b and f_a denote the following forward likelihoods, which are identical on T_1, T_2 and $\tilde{T}$:

$$f_u(x) = P\{D_u | x_u = x\} \text{ for nodes } u = a, b, c\,.$$

Then we can write

$$\begin{aligned} L(T_i) &= \sum_{x,y,z \in \{A,C,G,T\}} \pi(x) f_a(x) f_b(y) f_c(z) X_i(x,y,z) \text{ and} \\ L(\tilde{T}) &= \sum_{x,y,z \in \{A,C,G,T\}} \pi(x) f_a(x) f_b(y) f_c(z) \tilde{X}(x,y,z) \end{aligned}$$

where

$$\begin{aligned} X_1(x,y,z) &= \sum_{w \in \{A,C,G,T\}} P_{t_a}(x,w) P_{t_b+t_3}(w,y) P_{t_c+t_1}(w,z) \\ X_2(x,y,z) &= \sum_{w \in \{A,C,G,T\}} P_{t_a+t_3}(x,w) P_{t_b}(w,y) P_{t_c+t_2}(w,z) \\ \tilde{X}(x,y,z) &= \sum_{w \in \{A,C,G,T\}} P_{\tilde{t}_a}(x,w) P_{\tilde{t}_b}(w,y) P_{\tilde{t}_c}(w,z)\,. \end{aligned}$$

Since $L_{3c}(N) = \gamma_1 L(T_1) + (1-\gamma_1) L(T_2)$, it suffices to show that, for every x, y, z,

$$\gamma_1 X_1(x,y,z) + (1-\gamma_1) X_2(x,y,z) = \tilde{X}(x,y,z) + O(\max\{t_1, t_2, t_3\})^2\,.$$

This approximation is obtained by writing the Taylor expansion of the transition probability matrix: $P_t = I + tQ + O(t^2)$. Applying this Taylor expansion to the expression gives the same constant term, $1_{\{x=y=z\}}$, in both $\gamma_1 X_1(x,y,z) + (1-\gamma_1) X_2(x,y,z)$ and $\tilde{X}(x,y,z)$, and the same first order term:

$$\begin{aligned} \sum_{w \in \{A,C,G,T\}} & (t_a + (1-\gamma_1) t_3) Q(x,w) 1_{\{w=y=z\}} + (t_b + \gamma_1 t_3) Q(w,y) 1_{\{w=x=z\}} \\ & + (t_c + \gamma_1 t_1 + (1-\gamma_1) t_2) Q(w,z) 1_{\{w=x=y\}}\,, \end{aligned}$$

hence the second-order approximation of $O(\max\{t_1, t_2, t_3\}^2$.

□

In summary, both types of three-cycles are nearly non-identifiable from concatenated data for Markov models with and without rate variation. The presented pairwise distance-based branch length and inheritance weight conversions can be used to shrink three-cycles.

PRACTICAL CONSEQUENCES

As a consequence of these three-cycle findings, we anticipate convergence difficulty when switching between topologies with and without three-cycles. Attempting to distinguish between a three-cycle and its contracted version could slow or create failures during optimization. Though these structures look different, they have nearly identical or flat likelihoods. To prevent cycling between these two structures without gain, we allow the user to restrict the network topology search to networks without three-cycles. Under this restriction, we shrink these structures to nodes. We describe this method in detail in Chapter Four.

3.3.4 NON-TREE-CHILD STRUCTURES

In a tree-child network, every non-leaf node must have a tree node child, by definition. This rule is relatively restrictive and not likely to be true for real data. In reality, reticulation events can occur consecutively in time.

There are two ways the tree-child rule can be violated in a phylogenetic network: W structures and hybrid ladders. The first, W structures, have two reticulate nodes side by side, creating a W shape (see Figure 3.10).

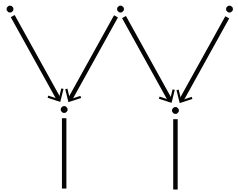

Figure 3.10: W Structure

The second violation creates a hybrid ladder. Hybrid ladders have two consecutive hybrid nodes (see Figure 3.11). We present formal definitions for each.

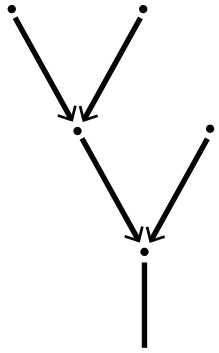

Figure 3.11: Hybrid Ladder Structure

Definition 5. W Structure

A network contains a *W structure* if two hybrid nodes share a parent node.

Definition 6. Hybrid Ladder

A network contains a *hybrid ladder* if it contains a hybrid node whose the parent node is also hybrid.

Because the tree-child rule is not likely to be met by real data, it is important to understand if W structures and hybrid ladders can be detected. The identifiability of these two structures is not well understood, but varies depending on model and data type.

Tree-child networks are known to be identifiable for certain types of data under some restrictions. Using multi-distances between taxa, a tree-child network can be fully determined, as long as the network is rooted at an outgroup and each reticulation has parent hybrid edges with equal inheritance weight (Bordewich *et al.*, 2018). For concatenated sequences under a recombination-mutation model of evolution (which, like our model, does not take incomplete lineage sorting into account) tree-child networks with a fixed number of reticulations have been shown to be identifiable, as long as none of the reticulations are adjacent to the network's root (Francis and Moulton, 2018; Thatte, 2013). For concatenated sequences under our model, Gross *et al.* (2020) show that level-1 networks with a fixed number of reticulations are generically identifiable under our model. In a level-1 network, each biconnected component can have at most one reticulation node (Gambette and Huber, 2012), so level-1 networks are tree-child. From trinet data (rooted 3-leaf network, for all sets of 3 leaves), van Iersel and Moulton (2014) show that tree-child networks are encoded, or determined, by these data, meaning that, there is only one phylogenetic network displaying a given set of trinets (Huber and Moulton, 2013).

In contrast, the identifiability of non-tree-child networks from sequence data are not yet well understood. (van Iersel and Moulton, 2014) extend their finding on trinets to level-2 networks, as long as the networks do not have strongly redundant biconnected components. This includes some, but not all, non-tree-child networks. For sequence data, the identifiability of non-tree-child structures is not well understood.

In this section, we extend the zipper theorem and prove that hybrid ladders are not identifiable from concatenated data, while acknowledging that much is still not understood about these structures.

Hybrid Ladder Reticulation Order. *The unresolved hybrid ladder structure in network $\tilde{N}$ has three possible resolutions, N_1, N_2, N_3, leading to different reticulation orders (Figure 3.12). For each of these resolutions, a different pair of parent edges make up the first hybridization. These three resolutions are not distinguishable from one another based on concatenated sequence data under a Markov model.*

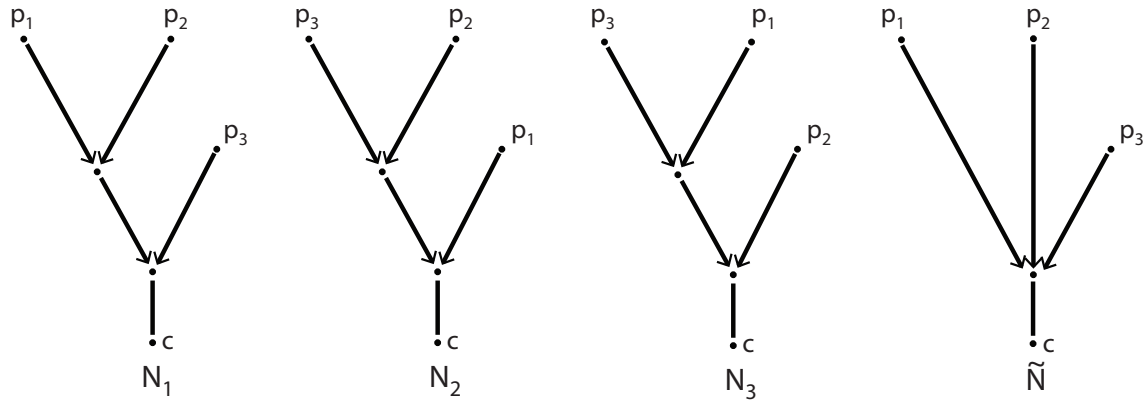

Figure 3.12: Hybrid Ladder Structure Resolutions

Proof. First, assume that network N_1 contains one or more hybrid ladders. Following directly from the zipper theorem, edge lengths directly below hybrid nodes are not identifiable. Thus, if one hybrid node leads to another, then the edge between them can be unzipped (its length set to zero) without changing the likelihood (see Figure 3.13). This new network is identical in topology to N_1, but has modified branch lengths. As we showed earlier in the zipper theorem, their likelihoods will be identical: the displayed trees in N_1 are equal to those in the unzipped version.

Now, consider N_2, a modification of N_1 in which the order of the two consecutive hybrid nodes has been reversed, as shown in Figure 3.13a. We unzip the edge between the reticulate nodes. Branch lengths and inheritance weights are updated as shown in Figure 3.13a. For network N_2, the new inheritance weights

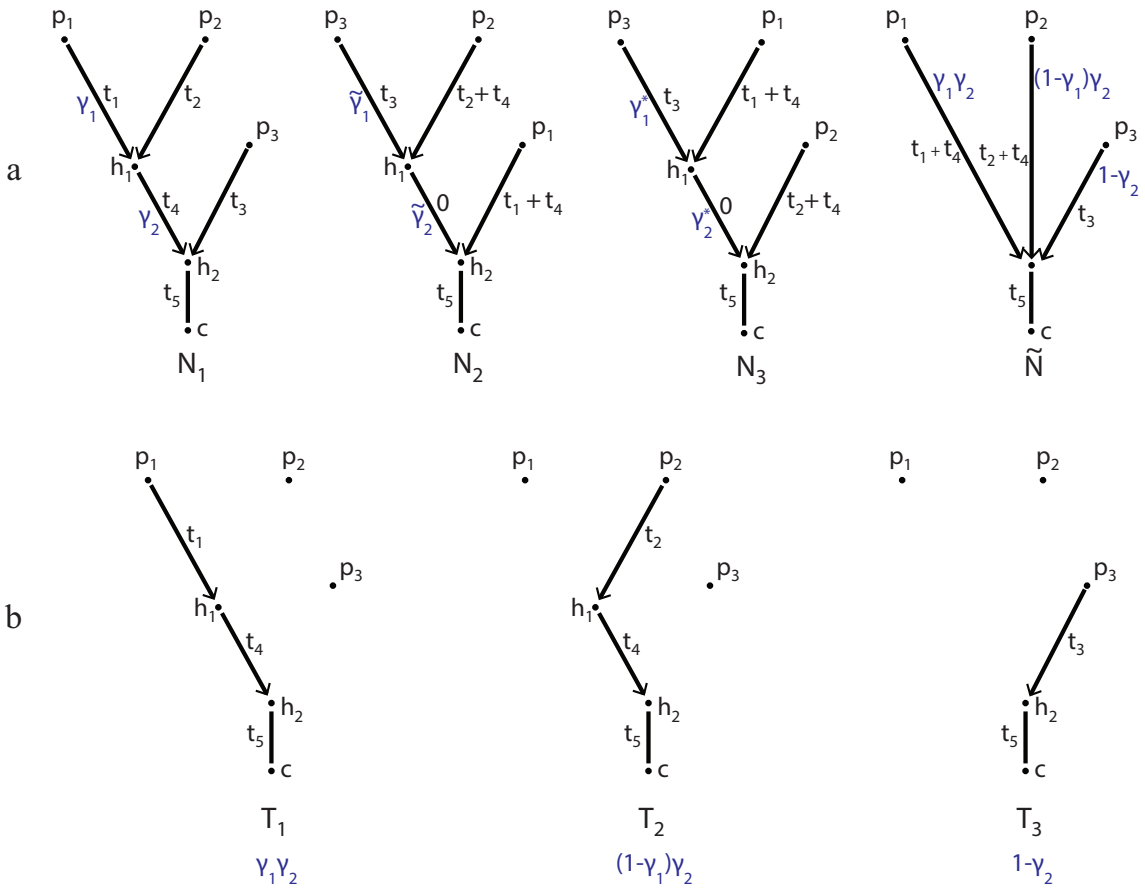

Figure 3.13: Hybrid Ladder Structure After Applying Zipper Theorem. For each reticulation, only one inheritance value is shown. Its partner branch has an inheritance weight such that the gamma values at each reticulation sum to one. a) Three networks created by switching reticulation order and the canonical network $\tilde{N}$, created by applying the zipper theorem and by adjusting inheritance weights. b) The three displayed trees contained within these networks (with inheritance weights)

are

$$\tilde{\gamma}_1 = \frac{1-\gamma_2}{1-\gamma_1\gamma_2}$$

$$\tilde{\gamma}_2 = 1-\gamma_1\gamma_2$$

To show that the ordering of these nodes is not identifiable, we first extract the displayed trees from N_1 and N_2. The set of displayed trees in N_1 is exactly equal to those in the network N_2, with reticulation order changed (shown in Figure 3.13b). After modifying the inheritance weights, the weights of these displayed trees are also equal. Because the likelihood of a network is simply a weighted sum of displayed trees, $L(N_1) = L(N_2)$.

The likelihood is unchanged by reordering reticulations, showing that the order of consecutive reticulations is not identifiable.

This result holds for the other reordering as well. The extracted displayed trees and their branch lengths are identical. To keep the same weights of displayed trees, the inheritance weights for the reordered network N_3 are

$$\gamma_1^* = \frac{1-\gamma_2}{1-\gamma_2+\gamma_1\gamma_2}$$

$$\gamma_2^* = 1-\gamma_2+\gamma_1\gamma_2\,.$$

□

In summary, we show that the order of consecutive reticulations in a hybrid ladder is not identifiable from concatenated sequences. After applying the zipper theorem, any hybrid ladder can be replaced with a polytomy to create the canonical form, $\tilde{N}$ (shown in Figure 3.13a.

It is still unknown if W structures are identifiable from concatenated sequences. In Figure 3.14, we show a W structure before and after applying the Zipper Theorem. After applying this branch length change, the placement of the reticulate nodes appears to remain distinct. Still, future work is needed to confirm the identifiability of W structures from concatenated sequence data.

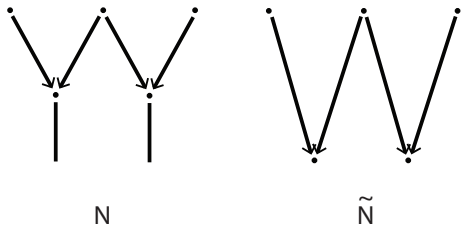

Figure 3.14: W Structure After Unzipping

3.3.5 Practical Consequences

Our topology search allows W structures and prevents hybrid ladders by default, although analysts can override this.

While it is not the default setting, we do let analysts choose to allow hybrid ladders. This provides the benefit of additional complexity of consecutive reticulations, even if their order is not identifiable. Although

W structures and hybrid ladders look quite different, they both share the non-tree-child property, they can both reduce the number of displayed trees, and one structure can be transformed into the other with a single nearest neighbor interchange (NNI). The NNI moves we propose in Chapter Four acknowledge the possibility of hybrid ladders and handle the branch lengths as described above. Inheritance weights are not updated in the NNI moves themselves, but they are optimized immediately afterward, which should set them appropriately. In most cases, the benefits of allowing hybrid ladders are outweighed by the possibility of the topology estimation cycling between two structures with equivalent likelihoods, although preventing hybrid ladders reduces the neighborhood of some networks, which could increase the occurrence of local optima.

3.3.6 Number of Hybridizations

As discussed in the introduction, the number of hybridizations within a network is not identifiable under plain maximum likelihood. Without a penalty for model complexity, the likelihood of a network estimator increases with the addition of a new hybridization, assuming that all branch lengths and inheritance weights can be perfectly optimized. Therefore, a penalty needs to be derived if one wants to estimate the number of hybridizations.

Practical Consequences

We handle this non-identifiability by passing decisions about the appropriate number of hybridizations to analysts. As in previous likelihood, pseudo-likelihood, and Bayesian network reconstruction methods, analysts choose a maximum number of reticulations (Solís-Lemus and Ané, 2016; Cao *et al.*, 2019; Yu *et al.*, 2014). The software adds reticulations up to this limit, estimating inheritance weights for each parent hybrid edge.

Analysts can use these inheritance weights to identify the most important hybridizations, those which account for a high percentage of inheritance. For example, some may be interested only in reticulation events where at least 10% of inheritance is passed down by the minor parent. In these cases, analysts can remove low-inheritance reticulation events during post-estimation processing.

3.3.7 Canonical Shrunk Semi-Directed Network from Sequence or SNP Data

Together, these results provide a path to define a canonical phylogenetic network from concatenated data. This canonical shrunk network is semirooted and all two- and three-cycles have been shrunk, recursively if need be. We conjecture that branch lengths are identifiable, except when below a reticulation node. While the root placement is not identifiable, there are clear restrictions on its placement. Two-cycle structures created by reticulation nodes are not identifiable in time consistent networks, no matter how much data are available. Three-cycles may be identifiable, though in practice they will very likely not be distinguishable. Lastly, some non-tree-child structures may be identifiable, while others must be collapsed. We believe it is possible that canonical shrunk networks may be fully identifiable from sequence data. We state this as a conjecture.

Definition 7. Canonical Shrunk Semi-Directed Phylogenetic Network

A *canonical shrunk semi-directed network* is a phylogenetic semi-directed network, as defined by Solís-Lemus and Ané (2016), such that any undirected cycle is of size four or greater, and such that $t_e = 0$ for any edge e whose parent is a hybrid node. In particular, any hybrid ladder is collapsed into a polytomy.

Conjecture 1. *The network topology, branch lengths, and inheritance weights of a canonical shrunk semi-directed phylogenetic network are fully identifiable from concatenated sequence data under a Markov model.*

3.4 Discussion

In this chapter, we explored parameter and topology identifiability challenges in estimating phylogenetic networks. First, we investigated root location, proving that the root placement in a semi-directed network is restricted, but not identifiable. We then explored topology and branch length identifiability with several theorems. The zipper theorem clarified branch lengths non-identifiability at reticulations and proposed a canonical solution. We then showed that two-cycles, three-cycles, and hybrid ladders are not identifiable.

After each of these findings, we discussed how we use the results to inform implementation choices and improve the efficiency of our network estimation software. When applicable, we also provided exact or pairwise-distance-estimated branch length and inheritance weight formulas for converting

non-identifiable structures to their canonical form. Finally, we gave a definition of a canonical shrunk semi-directed phylogenetic network for sequence data, building upon the definition proposed by Pardi and Scornavacca (2015). While all proofs use the network likelihood, these results transfer easily to the case of log likelihood, which is often more practical for software applications.

While we have made progress toward understanding the identifiability of semi-directed phylogenetic networks from sequence data, there are still many unanswered questions. First, our zipper theorem explores the identifiability of branch lengths and inheritance values near reticulations without a coalescent model. It would be interesting to explore how this result might change under a coalescent model. It might also be interesting to explore how non-identifiable three-cycle structures are in practice, by measuring much the likelihood changes during shrinkage with simulation experiments. Finally, while we made progress in understanding the non-identifiability of hybrid ladders, there are still many questions about the full identifiability of the network, and of non-tree-child structures in particular. Exploring these questions could be interesting and useful future contributions.

In work focused on gene trees or pairwise multi-distances, tree-child and level-1 networks have been shown to be identifiable under mild restrictions (Solis-Lemus *et al.*, 2020; Bordewich *et al.*, 2018; Bordewich and Semple, 2016). For sequence data, several recent papers have illuminated the positive identifiability of a full network. Gross *et al.* (2020) show that level-1 networks with any fixed number of reticulations and cycles of size four or greater are generically identifiable under the Jukes and Cantor, Kimura 2-parameter, and Kimura 3-parameter models of evolution, extending earlier work (Gross and Long, 2018). As in our case, their models do not consider incomplete lineage sorting. They exclude two-cycles without considering their identifiability. We build on this by showing that two-cycles are nearly not identifiable, in general. In time-consistent networks, two-cycles are fully not identifiable. We found that these results hold even after adding rate variation across sites under any Markov model of evolution.

In line with our findings on the non- and nearly non-identifiability of three-cycles, Gross *et al.* (2020) show that three-cycles are hard to identify. Our results agree that three-cycles are hard to detect in practice and we extend these findings to a broader class of networks and models. For three-cycles, they show that type-2 three-cycles are theoretically detectable in networks with a single reticulation, but that the hybrid node placement is not identifiable. We show that all three-cycles are nearly not identifiable under all Markov models of evolution when branch lengths are small. This finding is broader, covering any Markov

models with or without rate variation, includes networks with any fixed number of reticulations, and considers the identifiability of continuous parameters like branch lengths and inheritance weights.

Gross *et al.* (2020) also found that the root placement was not identifiable in their level-1 networks. We extend this finding to more complex networks of any level and non-tree-child networks.

In this chapter, we presented identifiability findings for branch length and inheritance weight parameters, non-tree-child structures, and small cycles to a broad set of networks and models. In Chapter Four, we present a software estimation tool developed with careful consideration of these identifiability results. We hope other methods developers will find these identifiability results, practical tools, and numerical parameter conversions useful as well.

Acknowledgements. This work was supported by National Science Foundation awards DMS-1902892 and DMS-2023239.

CHAPTER 4

PHYLINC: MAXIMUM LIKELIHOOD ESTIMATION OF PHYLOGENETIC NETWORKS WITH CONCATENATED MARKER DATA

Abstract In this chapter, we describe our phylogenetic network estimation method, PhyLiNC: Phylogenetic Likelihood-based Network from Concatenated data. This method uses the model described in Chapter Two to estimate a fully parameterized network topology with evolutionary rates, edge lengths, and hybrid edge inheritance percentages from concatenated DNA sequences. We first describe the topology estimation algorithm, then introduce the semi-directed nearest neighbor interchange moves developed for this method. We then present results from three simulation experiments.

Keywords network reconstruction; Markov models; evolution; parameter estimation; speciation; gene flow; hybridization; phylogenetic networks

Contribution I led all aspects of this work: designing the overall strategy, extending nearest neighbor interchange moves to semi-directed networks, implementing topology estimation moves, parallel computation implementation, and user-implemented constraints. I designed and implemented the simulation, collected and analyzed data, created visualizations, and wrote the first draft of this manuscript. Cécile Ané and I serve as second and first authors, respectively.

4.1 Optimizing over Semi-directed Network Space: Overall Strategy

Our method estimates a semi-directed phylogenetic network with branch lengths and hybridization inheritance weights for a set of DNA sequences. Implemented in Julia as part of the publicly-available PhyloNetworks package, it estimates a phylogenetic network by alternating topology change proposals and parameter optimization to improve the likelihood of observing the sequences given the candidate network. In this section, we describe our estimation strategy: the overarching algorithm, numerical parameter optimization, and horizontal and vertical moves used to search network space.

The space of all possible networks is very large, discrete, and computationally intractable to search globally. To ensure that the entire space of phylogenetic networks is theoretically reachable during topology estimation, this method combines horizontal and vertical topology changes (Huber *et al.*, 2016a,b; Gambette *et al.*, 2017). Horizontal changes modify the way nodes are connected while keeping the number of reticulations constant (Figure 4.1). Vertical moves change the number of reticulations, adding or removing reticulations to increase or decrease the complexity of a network (Figure 4.2).

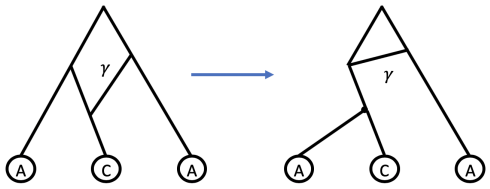

Figure 4.1: Horizontal Moves: Rearrangement at a Fixed Complexity

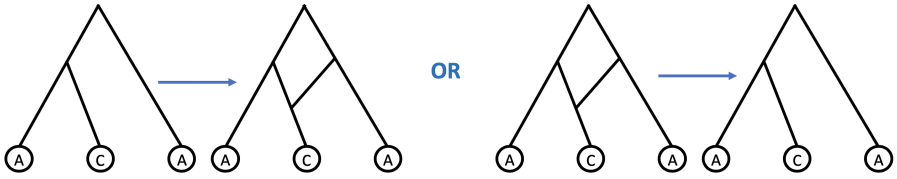

Figure 4.2: Vertical Moves: Complexity Increase or Decrease

Our method uses two vertical topology moves, to either add a new hybrid edge or remove an existing one, and two horizontal moves: semi-directed nearest neighbor interchange (sNNI) moves and hybrid edge flips. These horizontal moves explore the network space at a set number of reticulations.

Algorithm 1 PhyLiNC's Method for Reconstructing Semi-directed Networks

Read in concatenated data, starting network N_0, and species constraints (if any)
Remove non-identifiable structures in N_0
Estimate starting branch lengths from pairwise distances, if any is missing in N_0
Unzip edge lengths below hybrid nodes to 0 in N_0
Optimize substitution rates and rate variation given N_0
for r **do** in 1 through n_{runs}
    $N_r = N_0$
    transform N_r with a random number k of sNNIs, $k \sim \mathcal{G}(0.5)$
    $n_{\text{moves}} = 0$; $n_{\text{rejected moves}} = 0$
    while $n_{\text{moves}} < m_{\text{moves}}$ & $n_{\text{rejected moves}} < m_{\text{rejected}}$ **do**
        $\tilde{N} = N_r$ modified by a randomly-chosen topology move
        on $\tilde{N}$, optimize inheritance weights
        on $\tilde{N}$, optimize the lengths of and adjacent to a randomly-chosen branch
        n_{moves} += 1
        if the move was a root change **then**
            $N_r = \tilde{N}$ (accept the move)
        else if $L(\tilde{N}) \geq L(N_r)$ **then**
            $N_r = \tilde{N}$ (accept the move)
            $n_{\text{rejected moves}} = 0$
        else
            reject the move (keep N_r unchanged)
            $n_{\text{rejected moves}}$ += 1
        end if
        if $n_{\text{moves}} \geq m_{\text{moves}}$ **then**
            optimize substitution rates and rate variation parameter(s) given N_r
            on N_r, optimize branch lengths
            on N_r, optimize inheritance weights
            $n_{\text{moves}} = 0$
        end if
    end while
end for
$\hat{N} = \underset{N_1,\ldots,N_{n_{\text{runs}}}}{\arg\max}\ L(N)$
on $\hat{N}$, optimize all branch lengths and inheritance weights tightly
return $\hat{N}$, substitution rate matrix, and rate variation parameters

VERTICAL MOVES Vertical moves change the number of reticulations, or complexity, of a network. They can either add a new hybridization edge or remove an existing hybridization edge. Hybrid deletions are proposed 5% of the time, unless the current network is a tree (in which case hybrid deletions are not proposed). Hybrid additions are proposed 20% of the time, unless the current network has reached the maximum allowed number of reticulations (in which case hybrid additions are not proposed). Hybrid additions are proposed more frequently than hybrid deletions because the likelihood should increase with the number of reticulations, assuming that all numerical parameters are well optimized. During the search algorithm, branch lengths and gammas are not fully optimized to conserve computing time, so it is possible for hybrid additions to be rejected or for hybrid deletions to be accepted.

As discussed in Chapter Three, the ideal number of hybridizations in a network is not identifiable. Because of this, we ask analysts to provide a maximum number of hybrids. From the analyst-provided maximum number of hybrids, we calculate the number of edges and nodes in the most complex possible semi-directed network. These calculations are used to create space-efficient data structures for likelihood calculations. This strategy allows for efficient reuse of memory and faster estimation of the likelihood. In an unrooted tree, the maximum number of edges is $2n_t - 3$, where n_t is the number of tips, and the maximum number of nodes is $2n_t - 2$. Each reticulation contributes an extra 3 edges and an extra 2 nodes, such that, the maximum number of edges that can occur in the network, given a maximum of m_h hybrids, is

$$2n_t + 3(m_h - 1)$$

and the maximum number of nodes is

$$2n_t + 2(m_h - 1)\,.$$

HORIZONTAL MOVES Our algorithm searches network space using two types of horizontal changes: semi-directed nearest neighbor interchanges (NNIs), described in depth in the next section, and hybrid flips, described here. Gambette *et al.* (2017) showed that all rooted networks with a set number of reticulations are reachable with a finite number of rooted NNI rearrangements. However, to accommodate the semi-directed nature of our networks and increase the efficiency of our search, we also included hybrid edge flips.

Semi-directed NNIs are our main tool to search network space. They make local changes to the network

topology and are proposed most frequently: 60% of the time. They are used in two ways: first, to randomize the starting network and, second, to search network space during our hill-climbing topology optimization.

To randomize the starting network, we start with an analyst-provided starting tree or network, then, for each of the runs separately, apply a random number of sNNI moves. The number of moves is distributed as a Geometric random variable with parameter $p = 0.5$ by default. With the default p, an average of one sNNI moves is performed. The probability of no change in the starting network is p: a proportion p of runs is expected to start at the existing network. A proportion $1 - p$ of runs is expected to start from different topologies, reducing the risk of repeatedly estimating the same local optimum. The sNNI moves we developed for semi-directed parameterized networks are presented in detail in section 4.2.

Subtree prune and regraft (SPR) moves are more wide-ranging than NNIs: NNIs are special cases of SPRs. An SPR move takes a subnetwork attached by one edge and reconnects the origin of this edge to another edge of the network. This type of move allows for large jumps across network space. In our search, we did not implement SPR moves, but SPRs can in fact be proposed indirectly by our implementation: an SPR occurs if a reticulation edge is added, and its partner (the original edge) is estimated to have an inheritance value of zero. The original edge is then deleted and the new edge becomes a tree edge. After this move, the number of reticulations is actually preserved, and the subnetwork below the original edge is regrafted onto a new region of the network. An SPR move can make large changes to the network topology, increasing the range of our search. However, this move would only occur when hybrid additions are allowed, that is, when the current candidate network has fewer than the maximum allowed number of reticulations.

Hybrid edge flip moves also make large jumps in network space. These moves reverse the direction of a random minor hybrid edge, changing the location of a hybrid node and the identity of its partner parent hybrid edge. Hybrid edge flips are proposed for 5% of the moves.

We also implemented a move that simply re-assigns the root. It is used for technical reasons, such as for traversing the network with topological sorting. We propose moving the root to an admissible position despite the fact that it does not modify the semi-directed topology or the network likelihood, as shown in Chapter Three. This move changes the directionality of some tree edges, which affects the permissible

sNNI neighbors of each network. To ensure that all allowed sNNI moves can be performed, the algorithm proposes to move the root for 10% of the topology moves. Another reason for this move is to increase the relative time spent optimizing continuous parameters on the current topology, compared to the time spent proposing new semi-directed topologies. The inheritance weights and the lengths of one randomly-chosen edge and its neighbors are optimized after each move, including root moves, so proposals to move the root contribute to branch parameter optimization on the current topology.

NUMERICAL PARAMETER OPTIMIZATION A network's fit is affected by its numerical parameters as well as its topology, so they must be optimized simultaneously. Before starting the optimization process, we estimate good starting branch lengths using pairwise distances (or use existing lengths if given) and evolutionary rates. Between topology estimation moves, we optimize all inheritance weights, one at a time. We then optimize the length of an edge chosen at random and the lengths of adjacent edges, one at a time. After topology estimation is complete, all branch lengths and inheritance weights are optimized (again, one at a time) with tight tolerance parameters.

For both types of numerical parameters, the optimization uses an efficient gradient-based optimization method, based on the objective functions introduced in Chapter Two. For inheritance weights, the objective function is concave, so the Newton-Raphson method is used. For branch lengths, we use the sequential quadratic programming (SQP) algorithm for nonlinearly constrained gradient-based optimization (Johnson, 2020), as described in Chapter Two.

NETWORK RECONSTRUCTION ALGORITHM Using a reasonable starting network, such as a tree estimated with IQ-Tree (Nguyen *et al.*, 2015), we start a default of $n_{\text{runs}} = 10$ independent estimation processes. The analyst can modify n_{runs} depending on their time and computational restrictions. For efficiency, our implementation allows for running these processes in parallel. The user simply needs to give julia access to multiple workers. Each run then proceeds to optimize the network topology, by proposing neighboring topologies using the vertical and horizontal moves outlined earlier. We choose the final estimated network topology as that with the best likelihood among the n_{runs} topologies.

For moves that change the likelihood (sNNI and hybrid flip, addition or deletion), the likelihood is evaluated before accepting or rejecting the move. Some numerical parameters are optimized locally on the proposed network, then the new network's likelihood is compared with that of the current network. If

the proposed network has a better likelihood, the move is accepted. Otherwise, the move is rejected. In practice, the candidate network is modified in place to save memory and computing time. If the move is rejected, the topology modifications are undone and the original branch lengths and inheritance values are restored. A count of consecutive rejected moves is kept. Because root moves do not change the likelihood, they are automatically accepted without any likelihood calculation. They do not count towards the number of accepted moves, that is, they do not break a streak of rejected moves.

Every $m_{\text{moves}} = 100$ moves (by default), model parameters, edge lengths, and inheritance values are optimized globally. The optimization continues until some number of consecutive proposals are rejected ($n_{\text{rejected moves}} = 75$ by default) and the topology search is stopped. Section 4.3 details constraints, whose details were not included in this overview.

4.2 Semi-directed Network Nearest Neighbor Interchange Rearrangements (sNNIs)

Semi-directed NNI moves make up the largest component of our network topology search. In this section, we describe here how we extended the *rooted* NNIs by Gambette *et al.* (2017) to our setting on *semi-directed* phylogenetic networks.

Nearest neighbor interchange (NNI) moves are a subset of subtree pruning and regraft (SPR) moves. Subtree pruning and regraft moves detach a subtree and reattach it somewhere else in the network. As a restricted class of SPR moves, NNIs do this with one restriction: they must reattach the subtree to an edge adjacent to its original neighbor. While SPR moves can make larger or small changes, nearest neighbor interchange moves make only local rearrangements of the network topology. In essence, NNIs exchange the neighbors around an edge. In Figure 4.3, we show a basic semi-directed network NNI (sNNI) around the red focus edge.

4.2.1 Expanding SPRs and NNIs to Phylogenetic Networks

Nearest neighbor interchange (NNI) moves have long been used to search phylogenetic space in phylogenetic tree methods (Stamatakis and Kozlov, 2020). The local changes made by NNIs allow algorithms to search tree space without altering the likelihood too much. First, we discuss the current state of knowledge

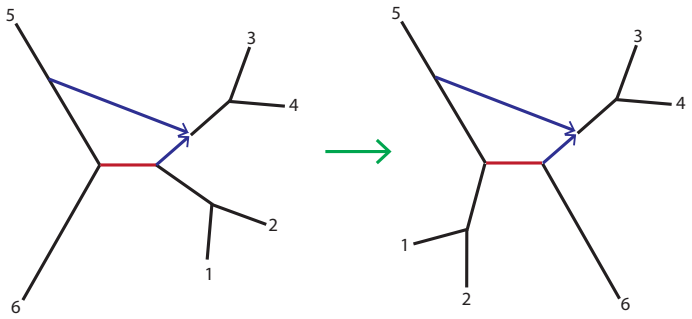

Figure 4.3: NNI moves prune an edge adjacent to the central edge and regraft it onto another adjacent edge

in extending NNIs and the broader set of SPR moves from trees to networks.

NNIs are more complex on rooted networks than on trees because the modified network needs to remain a phylogenetic network, and in particular, it needs to remain a directed acyclic graph (DAG). Huber *et al.* (2016b) extended NNIs to unrooted level-1 networks (in which all edges are undirected, and the identity of hybrid edges and nodes is unknown). Soon after, Gambette *et al.* (2017) extended NNIs and SPRs to all rooted networks: their rooted NNIs serve as a basis for our work. Bordewich *et al.* (2017) and Gambette *et al.* (2017) developed extensions of SPRs to rooted networks, which the former called subnet pruning and regraft (SNPR) moves and the latter called rooted SPR (rSPR) moves. Both Gambette *et al.* (2017) and Bordewich *et al.* (2017) show that their moves are reversible, meaning that every move has an opposite move, which can undo it and revert the network to its original state. In his recent thesis, Klawitter (2020) provides a clear and concise summary of SPR and NNI moves for rooted and unrooted networks. In a hill-climbing network search, having the option to easily undo the move comes in handy. It lets us to try a new topology while retaining the ability to revert to a previous one, if the fit is not sufficiently improvement. This allows us to avoid copying the network each time we try a new move, leading to reduced memory use.

4.2.2 Connectedness of the Network Space with SPRs and NNIs

With NNIs and SPRs adapted to networks (as opposed to trees), a question remains: how effectively do these moves search the space of networks? If we are to use these moves to search the space of possible networks, it is important to understand not only how to perform these moves on networks, but also whether they can search network space thoroughly.

Bordewich *et al.* (2017) show that reticulation-visible networks, networks in which each reticulation has at least one leaf for which every path from root to that leaf goes through that reticulation, are connected under their SNPR moves (reticulation-visible networks are defined nicely in Klawitter (2020)). Reticulation-visible networks are tree-child, meaning that this connectedness result does not apply to networks containing hybrid ladders and W structures. Erdős *et al.* (2020) show that NNI moves connect the space of tree-based rooted phylogenetic networks. Huber *et al.* (2016a) and Huber *et al.* (2016b) show that unrooted and rooted level-1 networks are connected by a finite number of NNI moves. Gambette *et al.* (2017) took this further, showing that their rooted SPRs and NNIs (called rSPRs and rNNIs) connect all networks with the same number of hybridizations. Combining these results, we can see that the space of general networks is connected under both NNIs and SPRs (Klawitter, 2020).

Francis *et al.* (2018) developed a bound on the number of NNIs needed to transform one unrooted network into another of the same reticulation number, showing that the number of unrooted NNI moves needed to make this transformation is bounded above by a term of order v^2, where v is the number of nodes in a network.

ORDERING NNI MOVES In the context of trees, there is a computational advantage to doing NNIs in an ordered way (e.g. depth first traversal, as in RAxML) (Stamatakis and Kozlov, 2020; Kozlov *et al.*, 2019). This savings comes from storing the forward likelihood of the tree at each node, then only computing updates to those parts of the tree changed by the NNI. Because calculating the likelihood is the most intensive part of the structure optimization process, this cuts computing time significantly.

In contrast, this computational gain seems difficult on networks. In a network, there are many more nodes: 2^h displayed trees within each network, leading to 2^h times as many nodes and as many forward likelihood values to store. In a network with rate variation across sites, 10,000 sites, v nodes, h hybrids, and four rate categories, there would be $10,000 \times 4 \times 2^h \times v$ forward likelihoods to store. So far, we have not attempted this increase in memory requirement, and the associated implementation to re-use stored partial likelihoods for speeding up ordered NNI moves.

Instead of doing NNIs on an ordered set of edges, we choose edges randomly. Unfortunately, this means that, no matter how many moves we do, we cannot guarantee an exhaustive local search (such as trying all NNIs on every edge until refusing all of them, confirming that no further improvement to the likelihood is

possible). Future work on the optimal ordering of branches and smart storage solutions for likelihoods could lead to improvement in computational efficiency.

In general, existing network SPRs and NNIs focus on topology changes without considering how branch length or inheritance weights may be affected. In the next section, we describe how we extend the rooted NNIs by Gambette *et al.* (2017) to semi-directed networks, and to handle branch lengths and inheritance weights.

4.2.3 Extending NNIs to Parameterized Semi-directed Networks

For our phylogenetic networks, we chose to infer neither a rooted nor unrooted network. Instead, we infer semi-directed networks, based on our finding from Chapter Three that the root is unidentifiable.

Semi-directedness confers advantages both biologically and practically, while also introducing additional complexity for NNI rearrangements. Semi-directed networks are more biologically realistic than unrooted networks because they encode information about the direction of gene flow and the identity of hybrid nodes (Solís-Lemus and Ané, 2016). This empowers analysts to root the network upstream of any gene flow, using external biological knowledge to root the network using a known outgroup. On a semi-directed network, the position of the root determines the direction of all edges. In contrast, on an unrooted network, the position of the root is insufficient to recover the rooted network.

Semi-directedness also confers methodological advantages. First, if there is an uncontroversial outgroup, post-inference rooting is an easier and less error-prone task (Degnan, 2018). Second, there are fewer semi-directed networks than rooted networks. Therefore, estimating a semi-directed network reduces computational complexity because the space to explore during topology optimization is smaller.

Using semi-directed networks introduces additional complexity for NNI rearrangements. Existing rooted and unrooted network NNI moves are not automatically transferable to semi-directed networks. In a rooted network, all edge directions are fixed. In an unrooted network, no edges have direction. In contrast to both of these, semi-directed networks have both directed edges and undirected edges, often in close proximity to one another. This distinction leads to varying restrictions on NNI move types allowed for each edge.

4.2.4 SEMI-DIRECTED NNIS

To accommodate the additional complexity of semi-directed networks with branch lengths and weights, we make significant changes to Gambette *et al.* (2017)'s rNNI moves. Their rNNI moves restrict their discussion to rooted networks without numerical parameters, while we extend them to semi-directed networks with branch lengths and inheritance weights. In this section, we propose an extended and novel set of semi-directed NNI (sNNI) moves. To our knowledge, previous network NNIs have not explicitly accounted for these.

Extending NNIs to semi-directed networks with branch lengths and inheritance weights is significantly more complex, increasing the number of NNIs from Gambette *et al.* (2017)'s 11 rNNI types to our 25 sNNI types. We also handle parameter identifiability issues that arise with some NNI rearrangements, especially in non-tree-child networks. Last, we briefly describe our algorithm to implement and easily undo these moves.

Semi-directed NNIs start by choosing an interior focus edge e. Given some arbitrary placement of the root, let e be directed away from the root as $e = (u, v)$, u being the parent and v the child of e. This edge is placed into one of four categories based on whether u and v are tree or hybrid nodes. We re-use the notations by Gambette *et al.* (2017), where a tree node is of type B (bifurcation) and a hybrid node is of type R (reticulation). Edge e is then of type BB, RR, BR or RB depending on type of its nodes (u, v). In a rooted network, e is directed, therefore types BR and RB are obviously distinct and well-defined.

For semi-directed network, we now prove that the type of an edge is also well-defined, that is: the classification of the edge into the BR or RB category does not depend on the root position. If e is a hybrid edge, then its direction is fixed and constrains the set of possible root positions. We cannot flip the identity of its adjacent nodes u and v, so its classification does not depend on the root. It can be of type BR, but not RB. In fact, if e is of type BR for one root position, then it must be a hybrid edge, and must be of type BR for any admissible root position. If e is a tree edge with $e = (u, v)$ for some admissible root position, then v must be a tree node. Therefore, e cannot be of type BR. If it is of type RB, then u is a reticulation, e is the child edge of this reticulation, and its direction is implied: from u to v necessarily. In summary, the classification into 4 types BB, RR, BR and RB from Gambette *et al.* (2017) is independent of the root position on rooted networks, so it is well defined on semi-directed networks.

In addition to the classification above, an edge is categorized as either *unrooted* or *rooted*, depending on whether it *may* or *may not* contain the root, if one wanted to root the network along the interior of the edge. Any edge downstream of a hybrid node may not contain the root, and is therefore considered as rooted. A hybrid edge may contain the root if it is not downstream of any other reticulation. In this case, the edge is said to be unrooted, despite being directed. Note that RR and RB edges must be rooted. BB and BR edges can be either unrooted or rooted. Our set of sNNI moves on e are determined by these two categorizations. When e is rooted, our moves are identical to the rooted NNIs from Gambette *et al.* (2017), with an additional assignment of edge lengths and inheritance weights. In what follows, we present NNI moves for each of these edge types.

Generic sNNI with branch length modification

Every type of sNNI can be obtained by applying the generic sNNI described here, in which we focus on an edge e adjacent to nodes u and v, a node $\beta \neq v$ connected to u and an edge $v\delta$ adjacent to e. Let αu be the third edge adjacent to u. We say that we "graft β onto $v\delta$" if we fix $\alpha u, e, v\delta$ as a backbone and slide the subnetwork attached to u and β along this backbone beyond v. To do this, we disconnect αu and u from each other, we disconnect v and $v\delta$ from each other, then attach αu to v and attach u to $v\delta$ (see Figure 4.4).

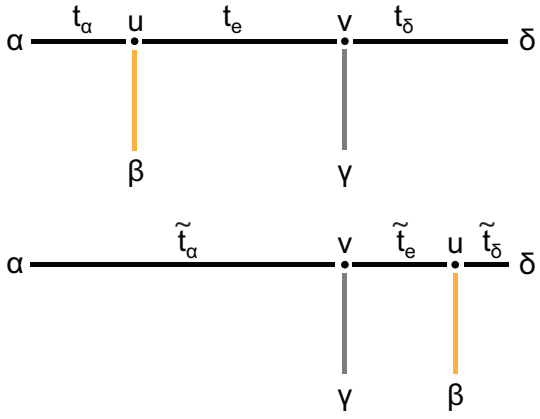

Figure 4.4: Generic NNI move that encompasses all sNNI types. For each sNNI, we choose a backbone (denoted by black edges) by choosing one of the two neighbors from each end of focus edge e. Of the other neighbors, one is grafted (yellow) and one remains fixed (grey). Here, β is grafted onto $v\delta$. The grey and colored edges are unchanged during the move. The black backbone edges may have their lengths and inheritance weights modified. All edges have their directions unchanged, except for edge e if the backbone is a directed path, and the tree/hybrid type of nodes is unchanged.

The type of each labelled node (tree node versus hybrid node) does not change: u (resp. v) remains a tree node after the sNNI if it was a tree node originally; or it remains a hybrid node after the sNNI if it was a hybrid node originally.

Only three edges are modified by this move: the backbone edges αu, e and $v\delta$. Edges connecting u with β and v with γ have their direction, lengths and inheritance weights unchanged. The directions of αu and $v\delta$ are also unchanged. To maintain the direction of edges away from the current root, the direction of e is flipped if the backbone forms a directed path before the sNNI. Specifically, e is flipped in two cases only: either if u is the child of αu, $e = (u, v)$ and v is the parent of $v\delta$; or if v is the child of $v\delta$, $e = (v, u)$ and u is the parent of αu.

The lengths of backbone edges can be modified in multiple ways. They can be modified randomly, such as for a Bayesian framework, as in Larget and Simon (1999). In their move, often called the Larget-Simon move, they slide node u along the backbone edge, making u's position random uniform along this edge. When node u's new position is between node v and node δ, an NNI in performed. Alternatively, node u's position can remain between nodes α and v, providing modified branch lengths but no topology change. Notably, after the Larget-Simon move, distances between nodes α, v, γ, and δ are preserved.

Alternatively, the lengths of backbone branches can be modified deterministically. This can be done in multiple ways: first, with *generic assignment*, which simply assigns edges their original lengths or, second, with *distance-based assignment*, which preserves distances between the four nodes α, v, γ, and δ, as in the Larget-Simon move. With generic assignment, we maintain the original branch lengths, with $\tilde{t}_\alpha = t_\alpha$, $\tilde{t}_e = t_e$, and $\tilde{t}_\delta = t_\delta$. With distance-based assignment, we reassign branch lengths in the modified network to maintain existing distances between α, v, γ, and δ. The distance-based new branch lengths are

$$\tilde{t}_\alpha = t_\alpha + t_e$$

$$\tilde{t}_e = r\, t_\delta$$

$$\tilde{t}_\delta = (1 - r)\, t_\delta$$

where $r = \frac{t_e}{t_\alpha + t_e}$. Figure 4.4 shows this distance-based assignment.

Generic assignment of branch lengths is used below unless stated otherwise (such as to keep the network

unzipped). The assignment of inheritance weight values will be described for each sNNI type separately, along with checks that the new network is a DAG.

BB sNNI Moves

An edge with two bifurcations (BB) can be either rooted or unrooted. In the rooted case, we use the same two moves as in Gambette *et al.* (2017), with branch length assignment as described in section 4.2.4 (Figure 4.5).

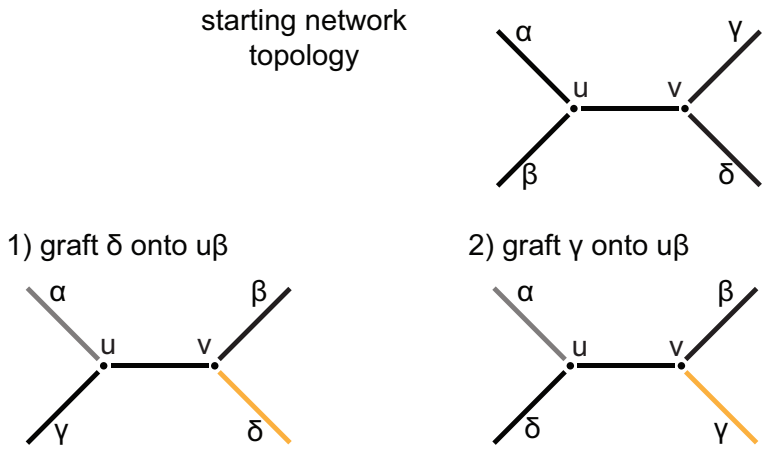

Figure 4.5: BB rooted moves. Since the edge (u, v) may not contain the root, its direction is constrained by reticulation(s) upstream of u, and the parent of u does not depend on the root position. We label this parent node α. The child of u other than v is labelled β. The children of v are interchangeably labelled δ and γ. In each case, the backbone edges are shown in black. (u, v) remains a tree edge. All other edges also retain their original tree/hybrid status

In the unrooted case, applying the two rooted options above to all admissible placements of the root position leads to eight possible moves around this edge (Figure 4.6). These eight cases lead to only two distinct topologies, but they may differ in how branch lengths are assigned. In all moves of type BB, inheritance weights are unchanged (and is 1 for the focus edge, which is must be tree edge). The modified network is guaranteed to be a valid semi-directed network before and after each move. In particular, with the current root position, the modified network is guaranteed to be a DAG.

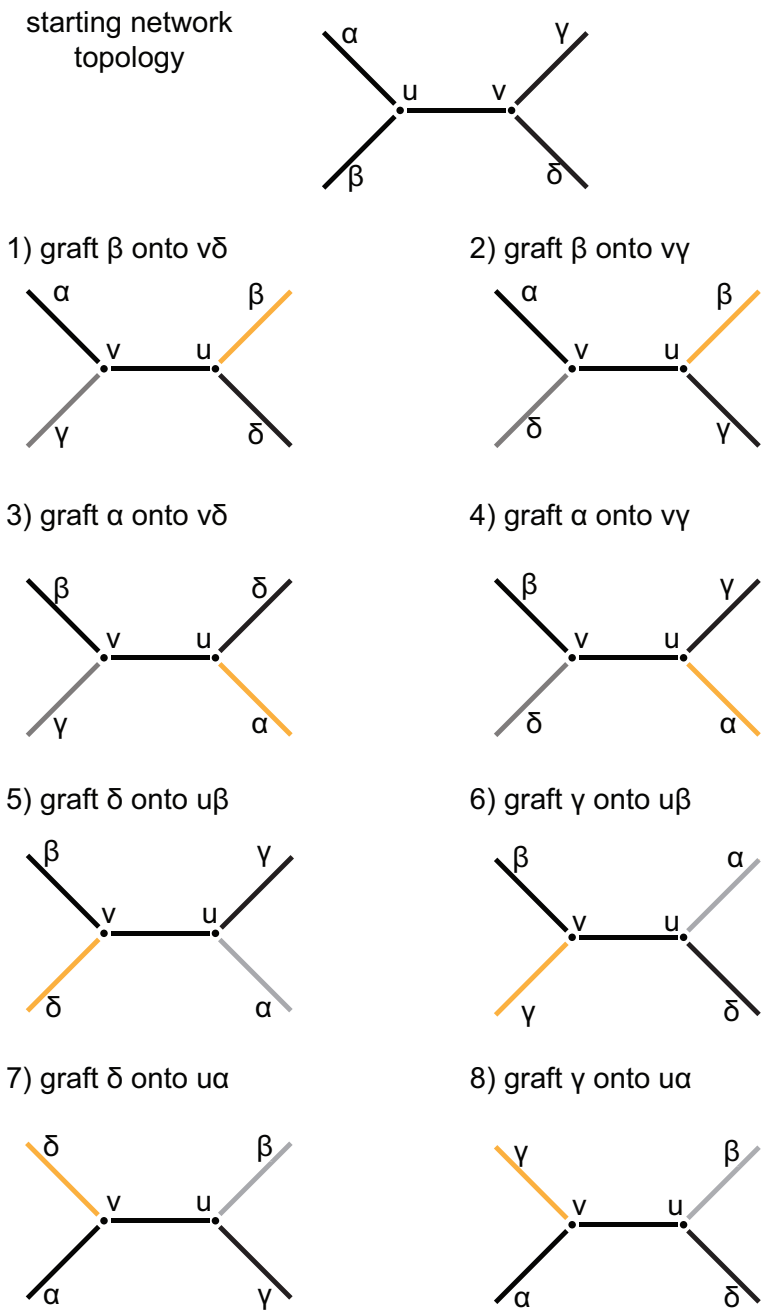

Figure 4.6: BB unrooted moves on an edge $e = (u, v)$ or $e = (v, u)$, depending on the root position. The backbone edges are shown in black. The fixed non-backbone edge is grey and the grafted edge is yellow. Cases 5-8 replicate the topology in cases 1-4, up to relabeling nodes u and v, but edge lengths may be different. Only black backbone edges may have their lengths modified, in a way that need not be symmetric in the grey and colored edges.

RR sNNI Moves

An edge with two reticulations (RR) is always rooted because the focus edge e is downstream of the first reticulation. In this RR case, we use the same two moves as in Gambette *et al.* (2017) as shown in Figure 4.7, with edge lengths assigned as in the generic description (section 4.2.4). Changes to the inheritance weights are also shown, denoted on each hybrid edge by γs. The modified network is guaranteed to be a valid semi-directed network.

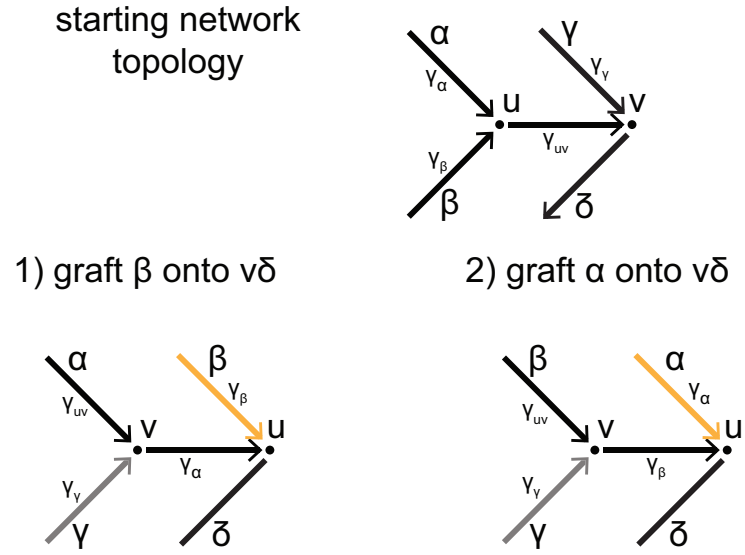

Figure 4.7: RR moves (rooted). The two parents of u are interchangeably labelled α and β. The second parent of v, other than u, is labelled γ. The only child of v is labelled δ. The pre- and post-NNI inheritance weights are denoted by γs. Note that only the backbone edges (in black) may have their parameters modified.

An RR configuration corresponds to a hybrid ladder: with a hybrid node having another hybrid node as a child. Recall from Chapter Three that the lengths of edges adjacent to a hybrid node are not identifiable. We introduced canonical networks in which reticulations are unzipped, to impose the constraint that the child of a hybrid node has length 0 and restore the hope of identifiability. For the assignment of branch lengths, the generic description in section 4.2.4 works well: if the network is unzipped at u and v, then it remains unzipped after an sNNI of type RR, because the edge to δ is unchanged (of length 0), and the focus edge has its length unchanged (at 0).

However, recall from Chapter Three that the three resolutions of the hybrid ladder are non-distinguishable from each other. In Figure 4.7, the original network (before the NNI) and the two modified networks (after the NNI) cover the three resolutions described in Chapter Three. The NNI-modified networks have the

same likelihood as the original network provided that the inheritance weights are set appropriately to maintain the same forest of weighted displayed trees. For instance, in RR case 1 (shown in the first NNI in Figure 4.7), the likelihood would remain unchanged if we set inheritance parameters as:

$$\begin{aligned} \tilde{\gamma}_\beta &= \gamma_\beta(1-\gamma_\gamma) \\ \tilde{\gamma}_\gamma &= \gamma_\gamma/\tilde{\gamma}_\beta\,. \end{aligned}$$

We did not implement this assignment of inheritance weights, however, to implement generic NNIs, in which the fixed edges (in grey in Figure 4.7) are left unchanged.

RB sNNI Moves

An edge $e = (u, v)$ of type RB is necessarily rooted. If this RB case, we use the same four moves as in Gambette *et al.* (2017) as shown in Figure 4.8. The modified network is guaranteed to be a valid semi-directed network, without any further checks.

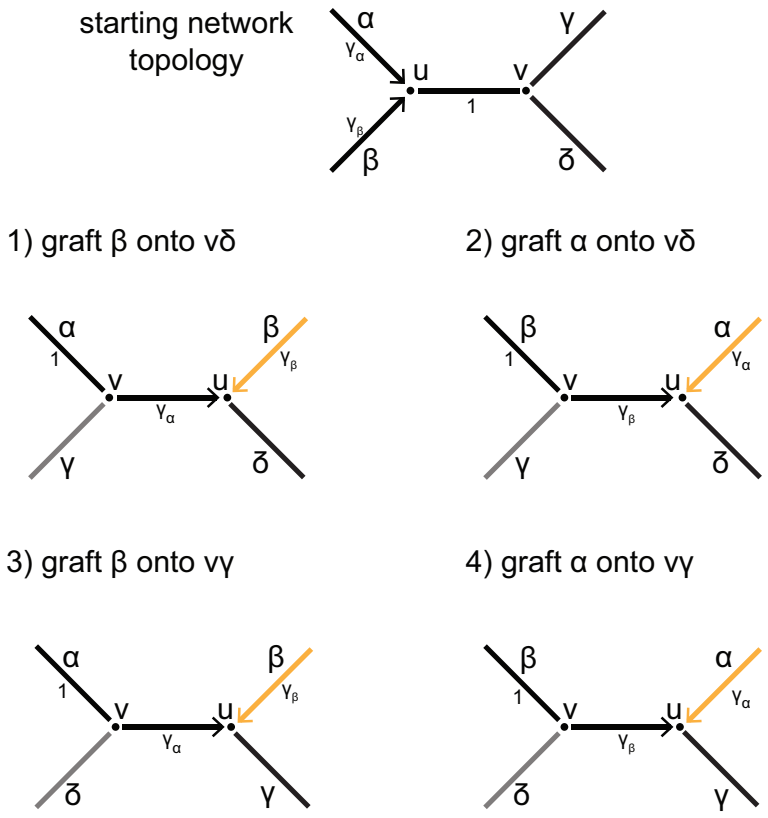

Figure 4.8: RB moves (rooted). The parents of u are interchangeably labelled α and β. The children of v are interchangeably labelled γ and δ. Backbone edges are shown in black. The grafted edge is shown in yellow. Edges are annotated with inheritance weights for edges that may change status. Node u remains a hybrid node; node v remains a tree node. The direction of the focus edge is flipped from (u, v) to (v, u). (The backbone is a directed path.) Two of the backbone edges change type: the focus edge becomes a hybrid edge and its parent edge on the backbone becomes a tree edge.

Keeping the network unzipped takes special care in this case. If the network is unzipped before the move, then (u, v) has length zero. After the move, a different edge needs to have length zero: edge to δ for cases 1-2, edge to γ for cases 3-4. Therefore, our current NNI implementation deviates from the generic assignment of edge length, in RB cases: it swaps the length of (u, v) with that of the child edge of u after the move.

BR sNNI Moves

The BR case can be either rooted or unrooted. It is the most complex, in part because the modified network is not necessarily a DAG. We give criteria to easily check if a given sNNI is valid, depending on the rest of the network. When rooted, we use the same three moves as in Gambette *et al.* (2017) as shown in

Figure 4.9. As in the RB case, we deviate from the generic edge length assignment in order to keep the network unzipped. That is, we swap the length of the edge to δ (which is zero if the original network is unzipped) with that of the child edge of v after the move.

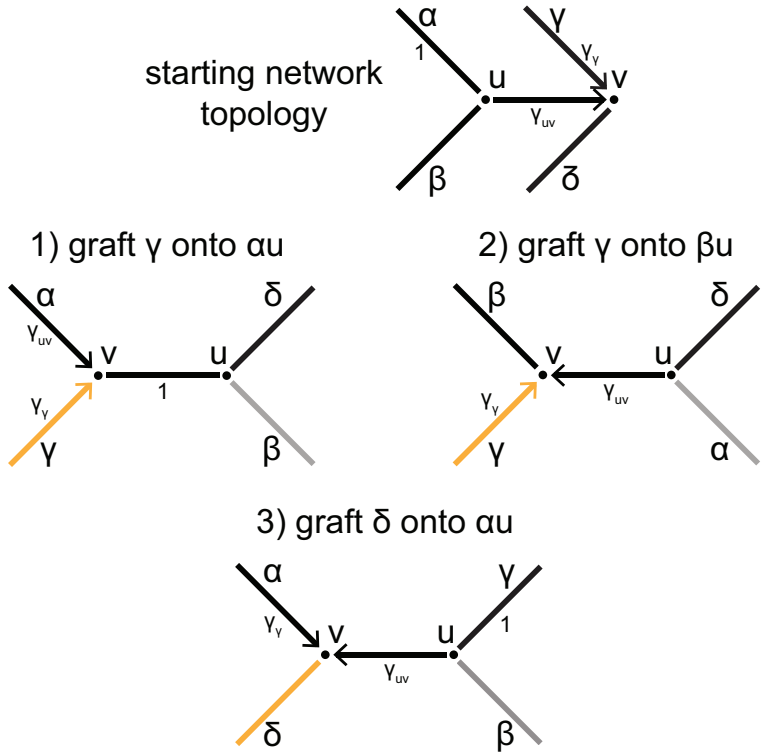

Figure 4.9: BR rooted moves on $e = (u, v)$, which must be a hybrid edge. Since the e may not contain the root and u is a tree node, the parent of u does not depend on the root position. We label this parent node α. The child of u other than v is labelled β. The parent of v other than u is labelled γ. The child of v is labelled δ. Backbone edges are shown in black. Edges are annotated with inheritance weights for edges that may change status. In case 1, e become a tree edge and its direction is flipped. (The backbone is a directed path.) In cases 2 and 3, e remains a hybrid edge.

These moves require that there is no directed path from node β to node γ in the rest of the network prior to performing the move (Gambette *et al.*, 2017). If such a path exists, then none of the three NNIs in Figure 4.9 are valid: they all lead to a modified network that is not a DAG. If no such path exists, then all three NNIs in Figure 4.9 are valid, as the modified network is guaranteed to be a DAG.

If the BR edge is instead unrooted, applying the three rooted options above to all admissible placements of the root position leads to six sNNI moves. As in the rooted case, the edge length assignment deviates from the generic description to maintain unzipping. Figure 4.10 shows the six moves, differentiated by the direction of the edges adjacent to the tree node in e under the current root position. Using notations

from Figure 4.10, one may apply the first three moves (left column) after re-rooting the network at node α. This re-rooting is necessarily possible if the edge between u and α is a tree edge, because $e = (u, v)$ may contain the root, so u can serve to root the network. Similarly, if the edge between u and β is a tree edge, then β can also serve as root and moves 4-6 (right column) may be applied.

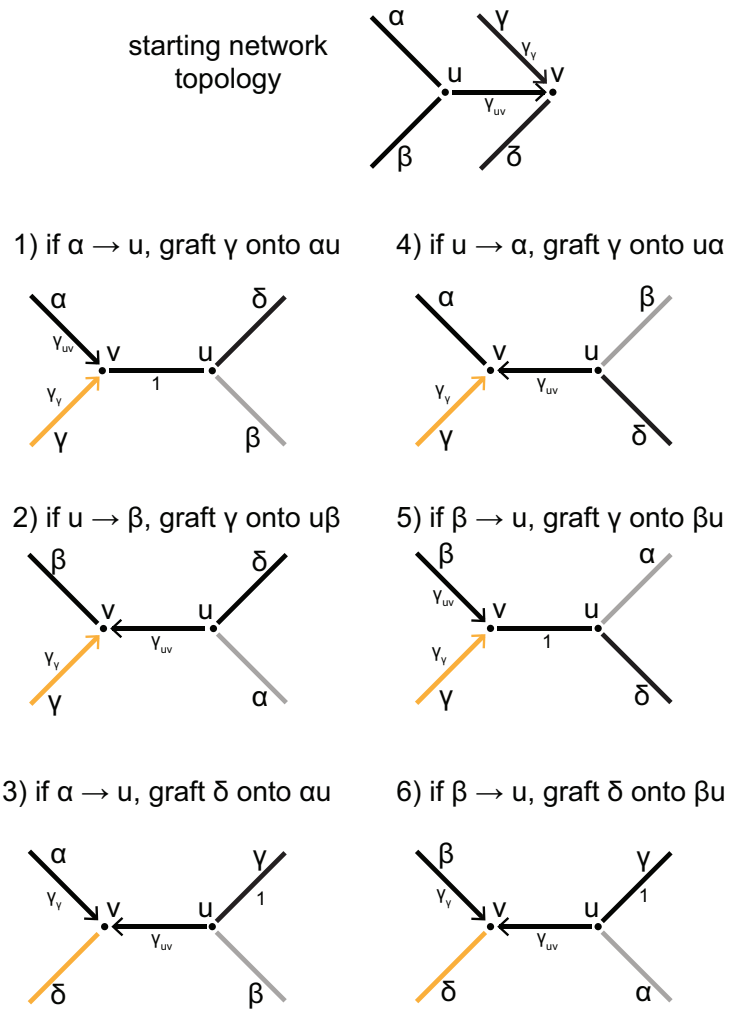

Figure 4.10: BR unrooted moves on an edge $e = (u, v)$ that may contain the root. For instance, u may serve as the root. Backbone edges are shown in black. Edges are annotated with inheritance weights, for edges that may change status. Cases 1-3 are identical to the BR rooted moves in Figure 4.9.

As in the BR rooted case, the rest of the network affects the validity of these moves. If there is a directed path from node β to node γ, then moves 1-3 are invalid. If there is a directed path from node α to node γ, then moves 4-6 are invalid.

4.2.5 Implementation of sNNI Moves

For the BR moves (Figure 4.10), our implementation does not attempt to re-root the network at either α or β. Instead, we use the current direction of edges (under the current root position) as fixed, restricting the sNNI options. To make up for this restriction in sNNIs for the BR unrooted case, we implemented a move that simply re-positions the root.

Our implementation allows for easy undoing of these moves. Because of this, we can implement the move, check if the network's fit to the data has improved and quickly undo it if needed. This removes the need to copy the network before doing an NNI move. To check if the fit has improved after the NNI, we first optimize the lengths and inheritance weights of the five edges affected by the move: the focus edge e and four edges adjacent to it. The code to implement these moves is publicly available as part of the open-source PhyloNetworks Julia package.

4.3 Strategies to Handle Constraints

In this section, we provide details on strategies to avoid and resolve non-identifiable structures so that the search is constrained to visit only canonical networks. We also introduce strategies to accommodate user-defined constraints to group multiple individuals within a population or populations within a clade.

4.3.1 Avoiding Non-Identifiable Structures

We now describe how PhyLiNC meets the constraint of visiting canonical networks only by avoiding non-identifiable structures or by removing them when they arise.

Practical Recommendations in Shrinking Two- and Three-Cycles in Semi-Directed Phylogenetic Networks

As discussed in Chapter Three, two-cycles and three-cycles are non- or nearly non-identifiable in semi-directed phylogenetic networks with concatenated sequence data. Under reasonable time constraints, two-cycles are fully non-identifiable. We do not allow them in the visited candidate networks (though we do not impose any time-consistency constraint due to their near non-idenfiability in general). This is in

line with the field, since most other authors define phylogenetic networks as *not* having any two-cycles in the first place. Three-cycles are also nearly non-identifiable so, by default, we do not allow them in the network.

Two- and three-cycles may arise in three ways during optimization. First, analysts may provide input networks containing two- and three-cycles. We handle this by shrinking these cycles before beginning optimization (e.g. Figure 4.11). Second, sNNI moves may convert a four-cycle into a three-cycle structure or a three-cycle into a two-cycle structure. We avoid this problem by forbidding sNNI moves that would create two-cycles. We similarly prevent sNNI moves that would create three-cycle by default, though analysts can override this choice. Lastly, smaller cycles may be created by the deletion of a hybrid edge. If the network is not tree-child, removing a hybrid edge can create a two- or three-cycle. To eliminate these cycles, we use the shrinking techniques and pairwise distance-based parameter adjustments introduced next, which are based on theory from Chapter Three.

SHRINKING TWO-CYCLES We aim to shrink two-cycles with minimal change to the network's likelihood. When shrinking a two-cycle, four branches are replaced with one branch (Figure 4.11). This new branch has length $\tilde{t}$ equal to the sum of the parent and child branches and the weighted average of the lengths from the two branches creating the two-cycle:

$$\tilde{t} = t_a + \gamma_1 t_1 + (1-\gamma_1) t_2 + t_b$$

This new branch has inheritance equal to the original γ for the branch of length t_b. If $t_1 \neq t_2$, the network likelihood may change.

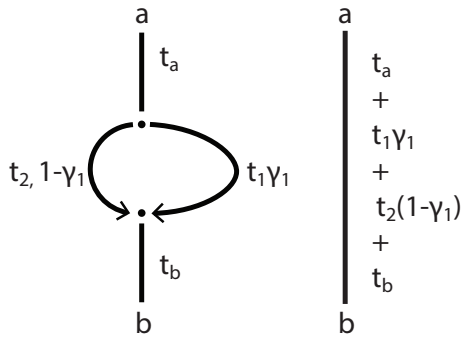

Figure 4.11: Two-Cycle Before and After Shrinking

SHRINKING THREE-CYCLES When shrinking three-cycles, we aim to change the likelihood as little as possible. There are two types of three-cycles, which we introduced in chapter three as type 1 and type 2. The first has two hybrid nodes and contains three displayed trees. The second has one hybrid node and contains two displayed trees. For both types of three-cycles, we assign branch lengths and inheritance values to the new structure to match the previous structure's likelihood as closely as possible, according to the formulas in chapter three's Theorem 3.3.3. The new branch lengths represent the average pairwise distances over the three-cycle's displayed trees.

AVOIDING TWO- AND THREE-CYCLE STRUCTURES DURING SNNI MOVES To avoid two- and three-cycle, we must also take care to prevent them from being created during the topology search, specifically during sNNI moves. The creation of a three-cycle by an sNNI would require the presence of a four-cycle before the move, assuming that the current network is free of two- and three-cycles. Figure 4.12 shows how a three-cycle could be created by grafting β onto $v\delta$ during an sNNI around edge $e = (u, v)$ or (v, u). To avoid three-cycles, we check that α and γ are not adjacent, and that β and δ are not adjacent. These adjacencies are shown with dashed edges in Figure 4.12. If either edge exists, then grafting β onto $v\delta$ would create a three-cycle, and the sNNI is not valid.

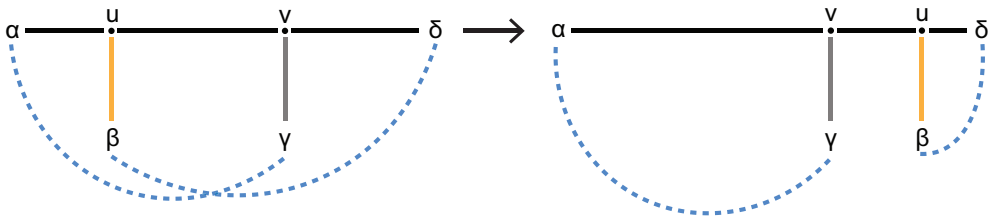

Figure 4.12: Avoiding Three Cycles in NNIs. To prevent three-cycles during the generic grafting of β onto $v\delta$, we check for the lack of edges shown by dashed lines (in blue). Edge directions and edge status (tree/hybrid) are not shown, since they differ between the various types of sNNIs. Only adjacencies matter when checking for three-cycles.

UNZIPPED RETICULATIONS

To respect the identifiability finding from our zipper theorem (Theorem 3.2.1), our algorithm unzips every hybridization by setting the length of edges below all hybrid node to zero and extending the lengths of parent edges accordingly.

We implement this constraint in PhyLiNC as follows. First, any existing reticulations within the starting

network are unzipped. Second, we never optimize branch lengths below hybrid nodes: these are fixed at zero. Third, we bring this restriction through in our NNI moves and hybrid additions: all sNNI moves ensure that the hybridization remains unzipped; new hybridizations are added with child branch lengths unzipped. Finally, when deleting a hybrid, the edge formerly below a hybrid is released, meaning that this branch length can now be optimized as normal.

Hybrid Ladders

To reduce identifiability issues arising from hybrid ladders (Theorem 3.3.4), we do not allow hybrid ladders by default. To do so, we forbid hybrid ladders in starting networks (unless analysts specify otherwise) and avoid them during network topology rearrangements. Three of our topology moves could create hybrid ladders: sNNI rearrangements, hybrid additions, and moves that flip a hybrid edge. In all three cases, we exclude any move that would create a hybrid ladder as invalid.

Avoiding hybrid ladders is challenging and, despite best efforts, our method does not avoid them completely. In rare cases, removing an existing hybrid edge can create a hybrid ladder. This occurs when a reticulation exists directly upstream of a W structure (as defined in Chapter Three). We did not seek to check for these rare cases. Therefore, the final estimated network may contain a hybrid ladder, even when using the default option to forbid hybrid ladders.

Ghost Hybrid Edges

After every topology move, we optimize inheritance weights of hybrid edges. If any edge has its inheritance weight optimized at zero, we call it a *ghost* hybrid edge. To implement the move that deletes a hybrid, we artificially turn the edge to be deleted into a ghost edge, such that the hybrid deletion is easily reversible.

When a network has hybrid ladders or W structures, ghost edges can lead to challenges in our likelihood-based numerical optimization before they are removed. Challenges occur when the removal of the ghost edge causes the removal of additional edges upstream of the ghost edge. In Figure 4.13, pruning the red edge from the network would lead to pruning its two parent edges (in blue) and possibly other edges further upstream. So if the red edge becomes a ghost edge (with an inheritance weight of zero) and neither of the two blue edges' parameters affect the likelihood. Attempting to optimize the lengths or weights of

the blue edges would lead to numerical issues.

Our algorithm checks for these ghost edges, skips the unnecessary optimization of upstream edges that do not affect the likelihood, and remove ghost edges as needed.

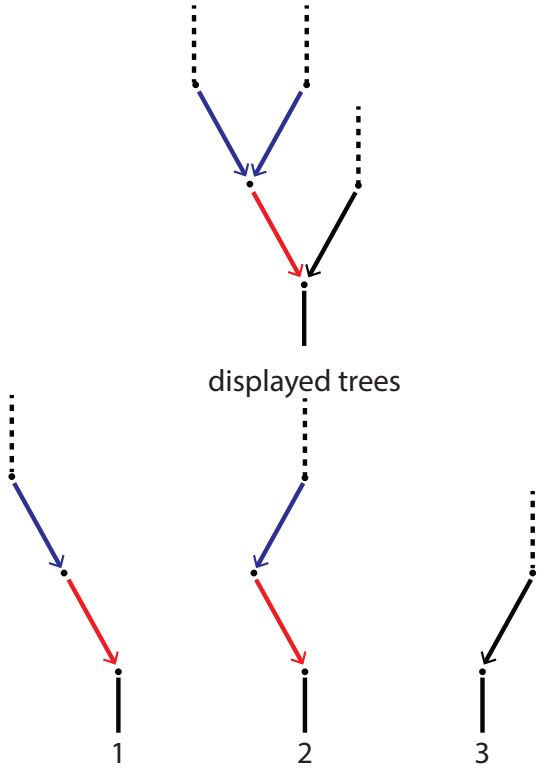

Figure 4.13: If the red edge is a *ghost* edge (with inheritance weight of zero) then none of the DNA sites have been inherited through this edge, so it can be pruned from the network. Subsequently, both blue edges can also be pruned from the network, as they do not contribute any genetic material to any observed leaf. Technically, if the red edge is a ghost, then all of the displayed trees with blue edges have weight zero and do not contribute to the likelihood: the blue edges' parameters do not affect the likelihood.

4.3.2 DAG Constraints during Hybrid Additions and Hybrid Flips

In addition to avoiding non-identifiable structures, we also must consider how each topology move may affect the directed acyclic graph (DAG) status of our network. As discussed in Chapter Two, a phylogenetic network must be a DAG in order to be admissible. Three moves in our topology search can alter the DAG status of our network: BR type sNNIs, add hybrid edge moves, and flip hybrid edge moves. The sNNI case is described in the section 4.2.

Before adding a hybrid edge from edge e_1 to e_2, we carefully ensure that no directed cycles would be created. We organize this check into two cases based on whether edge e_2's direction is changed. When

adding the new hybrid edge follows e_2's existing direction, we check two conditions. We confirm, first, that e_2's child node, c_2, is not equal to e_1's parent node, p_1, (meaning that e_2 is not directly above e_1) and, second, that c_2 is not an ancestor of p_1. If either of these conditions is true, making this addition would create a directed cycle.

When the add hybrid move reverses e_2's direction, we explore two situations. First, if e_2 cannot contain the root or is already a hybrid edge, then e_2's direction cannot be changed and this move is not possible. Second, if p_1 is the parent node of both edges e_2 and e_1 or if p_1 is the undirected descendant of e_2's parent node starting from edge e_2, then the hybrid move would create a directed cycle and is prohibited.

Before flipping a hybrid edge, we check that the network would not gain any directed cycles. A hybrid flip move reverses the direction of edge e, converting e's parent node into the new hybrid node, $\tilde{h}$, and replacing e's partner edge, e_p, with a new partner edge, $\tilde{e}_p$. To confirm that this move would not create a directed cycle, we carefully examine paths from $\tilde{h}$ to p_p, e_p's parent node. If there is more than one semi-directed path from $\tilde{h}$ through an adjacent edge to node p_p, then flipping e's direction would create a directed cycle from node p_p, through edge e, node $\tilde{h}$ and others, back to node p_p. This flip hybrid move is prohibited. If $\tilde{h}$ has only one adjacent edge that creates a semi-directed path from $\tilde{h}$ to p_p, then we assign this adjacent edge as the new $\tilde{e}_p$, ensuring that a directed cycle is not created.

4.3.3 Multiple Individuals in a Species

Our method allows analysts to group data from multiple individuals within a species or within a population. We use the term “species" and “population" interchangeably for the description of the method. In practice, the appropriate grouping level should be decided by domain experts, so we prefer to use the more generic term “population". Multiple sequences from the same population could be derived from multiple phased alleles within one individual, or from multiple individuals known to be sampled from the same population. We implement this external information about population membership by maintaining a leaf for each individual, but constraining all the leaves from the same population to be siblings within the network. In this section, we describe how PhyLiNC handles this constraint.

To allow for multiple individuals within one population, analysts must provide a file linking individual samples to their population. Their starting network must include one leaf for each population. The linkage file should have one sample per row, with two columns. The first column indicates the population,

while the second gives the individual's identifier. For a hypothetical data set across rodents, containing multiple samples from the species *Mus. musculus*, one row of this file might read, "Mus, Mus1". Analysts can provide multiple individuals for more than one population within their sample.

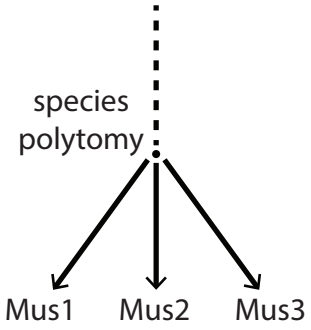

Figure 4.14: Three individuals from one population

Our method constrains the network topology by holding individuals within their population, creating a population node with all individuals downstream (Figure 4.14). This population node could be a polytomy, with more than 2 daughter nodes, if there are data from 3 or more individuals in this population.

Our algorithm then prevents sNNI, hybridization, and root moves that would disturb any population polytomy. NNIs centered on an edge above a population polytomy are considered as invalid. New hybrid proposals never propose adding a hybrid edge into or out of a population polytomy group. Because hybrid edges are never allowed to enter a group of individuals in a population polytomy, deleting hybrid edges will never violate this grouping. Finally, a network is never re-rooted below this population polytomy (as networks are never re-rooted at leaves).

We chose this constraint for two reasons. First, our primary goal is to infer relationships between populations, not relationships between individuals within a population. Second, discordance between gene trees is expected to be high within populations due to incomplete lineage sorting. It could be difficult to estimate a single tree within a population, without modelling the coalescent process. A polytomy is expected to be a parsimonious and useful approximation at the population level. In summary, our method allows analysts to include data for multiple individuals within each population, without an increase in the complexity of the search space.

4.3.4 Clade Constraints

Analysts may want to group populations (or species) based on previous knowledge. We refer to these groupings as *clade* constraints. On a rooted tree, a clade is a group of species that more closely related to each other than to any other leaf in the tree. In other words, they form the full set of descendants of some internal node. The most common example of prior knowledge that can be expressed as a clade is the identity of outgroup species. If multiple species (or populations) are included in the data to root the network, constraining the ingroup species to form a clade would ensure that the network reflects this prior information. Clade constraints are not yet implemented, but the software was written to accommodate the addition of this new type of constraint. To our knowledge, there does not exist any network estimation method that allows analysts to define clade constraints. We describe the challenges here.

Because the concept of clades is defined on rooted trees, the first challenge is to extend its definition to networks. We may consider two options on a network. The simplest option constrains the clade to exist in the network's major tree: the tree obtained by deleting any hybrid edge with $\gamma < 0.5$. For the clade to remain intact, any hybrid edge going into this clade from outside the clade would need to have $\gamma < 0.5$. This constraint could be sensitive to small variations in γ estimates close to 0.5. A second option would require that the clade of interest be displayed in at least one of the displayed trees. This option would be symmetric in the parent edges at each hybrid node and would be robust to errors in γ estimates. However, this second option seems difficult to interpret and less relevant to prior knowledge on species relationships.

Clade constraints would require careful attention to be implemented efficiently. Ideally, sNNI and hybrid moves would check for the constraint to be maintained without having to traverse all the displayed trees. The optimization of inheritance weights should also be constrained so as to not disturb a constrained clade on the major tree (under the first option). This is left for future work.

4.4 Simulation Experiments

We tested the efficacy of PhyLiNC in simulation experiments. In the first, we explore the accuracy of the method on data generated under incomplete lineage sorting, compared with SNaQ (Solís-Lemus and Ané, 2016). In the second, we compare the topology search with and without a hybrid flip move. In the last,

we explore the accuracy of PhyLiNC's estimation with independent and non-independent site data. In each of the simulations, we ran PhyLiNC using the HKY85 model without rate variation and a limit of 35 consecutive rejections before completion.

MEASURING NETWORK ESTIMATION ACCURACY For all three experiments, we measure network estimation accuracy by comparing the estimated network with the known true network using a dissimilarity measure called unrooted hardwired cluster "distance". This is an extension to Robinson-Foulds (RF) distance on trees (Robinson and Foulds, 1981). On a tree, the RF distance counts the number of edges present in one tree but not in the other, if we equate "edge" with "clade" (which we can do on a rooted tree). On a network, one extension of "clade" is the concept of "hardwired cluster" (HC): the full set of leaves that inherited *some* of its genetic material from a given internal node. The hardwired cluster distance between two networks counts the number of hardwired clusters that are present in one network but not in the other (Huson *et al.*, 2010). The hardwired cluster distance is technically a dissimilarity measure because there exist distinct networks for which the hardwired cluster dissimilarity is zero. (However, it has been shown to be a true metric on tree-child, weakly time-consistent phylogenetic networks (Cardona *et al.*, 2008).)

We extended this dissimilarity measure to semi-directed network as follows. We define the *unrooted hardwired cluster distance* between semi-directed networks N_1 and N_2 as the minimum HC distance between the rooted versions of N_1 and N_2, where the minimum is taken over all admissible placements of the root in N_1 and in N_2.

We report the unrooted HC distance divided by two, as is customary. On binary trees, this rescaling amounts to counting the number of edges in the first network that are not in the second. The number of edges in the second tree that are not in the first is necessarily identical. When comparing the estimated and true topologies, this is also the number of edges incorrectly estimated as true, so this is called the "false positive rate". On networks, this interpretation can be helpful also, although would not apply if the networks had different numbers of reticulations.

4.4.1 Comparison with SNaQ on 6-taxon Networks

To compare PhyLiNC and SNaQ on speed and accuracy, we carry out simulations on two networks with six taxa and one or two hybridizations. The true networks are shown in Figure 4.15. This simulation provides an opportunity to test the robustness of our method to violations of two assumptions: incomplete lineage sorting and grouped data. First, our method assumes no incomplete lineage sorting. To test how our method fares when this assumption is violated, we use data simulated under the coalescent process. Second, PhyLiNC assumes sites in the data are independent. In this simulation, sites were grouped in genes of 500 sites. Each gene had a single gene tree, inducing dependence between the 500 sites within a gene. Exploring our method's performance under these conditions will clarify its robustness in the face of these violations.

Methods

For each of these networks, Solís-Lemus and Ané (2016) simulated gene trees under the coalescent model using the `ms` software with varying numbers of genes (10, 30, 100 and 300) (Hudson, 2002). For each of the gene trees, they simulated the evolution of 500-site DNA sequences under the HKY85 substitution model and coalescent process using Seq-Gen (Rambaut and Grassly, 1997) (as in Yu *et al.* (2014)). For each network and each number of genes, 30 replicates were generated (for a total of 120 data sets per network).

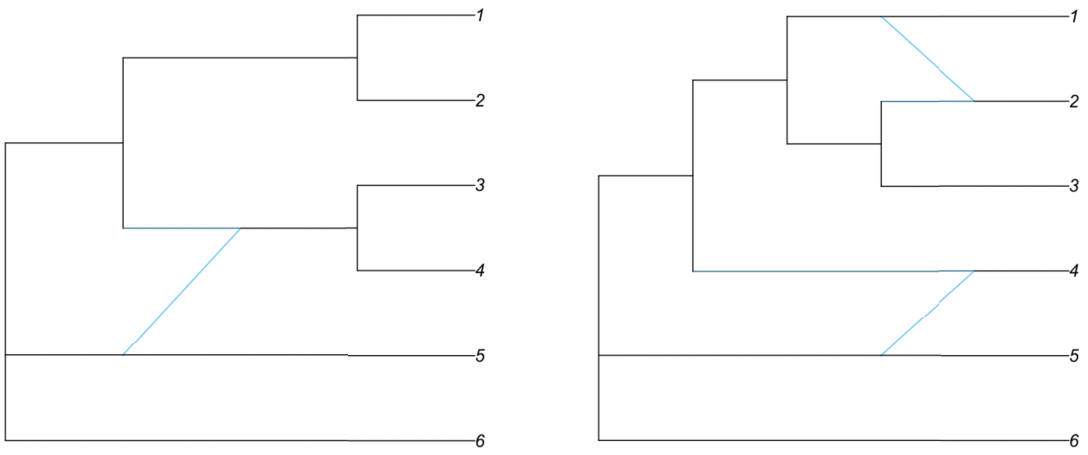

Figure 4.15: Left: network 1 with six taxa and one hybridization. Right: network 2 with six taxa and two hybridizations.

We used these simulated data to estimate networks using our method. For each replicate, we concatenated the sequences from all genes into one alignment and inferred a starting tree using IQ-TREE (Nguyen *et al.*, 2015). With this starting tree and the concatenated alignment, we estimate a network with our method, PhyLiNC, on 1.8 - 2.9 GHz processors. In this experiment, we used a version of PhyLiNC that did not include the hybrid flip move. Each call to PhyLiNC includes ten runs, by default, which were run in parallel. The comparison method, SNaQ, was run on 2.7 - 3.5 GHz processors by Solís-Lemus and Ané (2016). We set a maximum of 1 (resp. 2) reticulation(s) for analyzing the data simulated under the first (resp. second) network.

To calculate accuracy, we measured the distance between the estimated network and the true network, as well as the unrooted HC distance between the estimated and true major tree. We recorded the clock time required for each estimation.

RESULTS

As expected, PhyLiNC estimated the major tree more accurately than the full network (Figure 4.17). The accuracy of estimation of the major tree was stronger with SNaQ for both networks (Figure 4.16). For the network with one hybridization, the probability of estimating the true network was also higher with SNaQ, which recovered the true network 100% of the time for the network with one hybrid when using 100 or more genes. In contrast, PhyLiNC recovered the correct network only 10% of the time with 300 genes (Figure 4.18). For the network with two hybridizations, the estimation problem was much harder, but SNaQ still recovered the network about 44% of the time, while PhyLiNC never recovered the correct network.

For SNaQ, computational time averaged less than 40 minutes for both networks and all numbers of genes. For PhyLiNC, computational time was longer, ranging from an average of 43 to 3604 minutes (Figure 4.19).

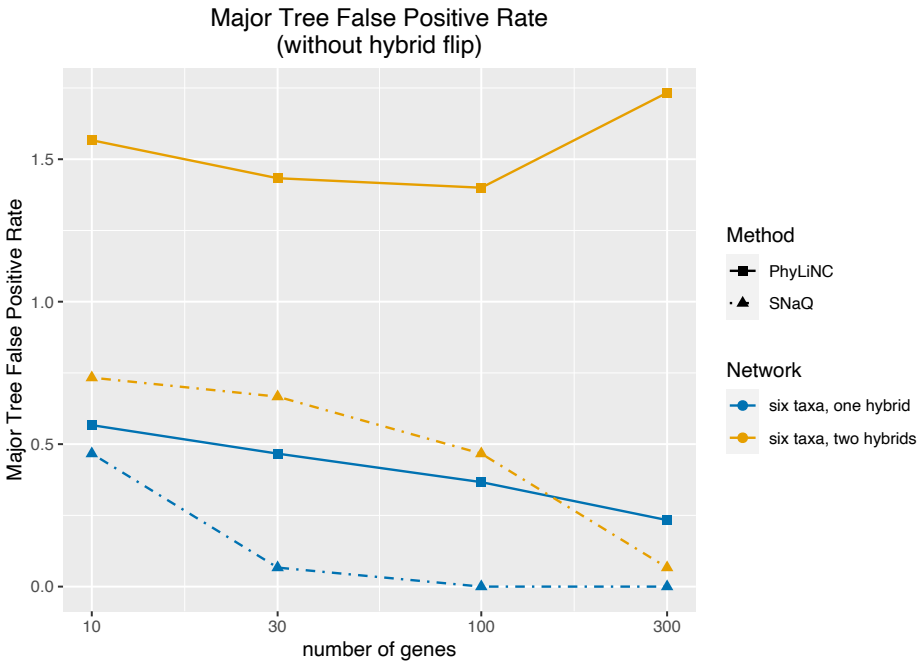

Figure 4.16: Comparing the Accuracy of PhyLiNC and SNaQ to recover the major tree. The major tree is obtained by suppressing all minor hybrid edges ($\gamma < 0.5$) to capture the major vertical inheritance pattern. Accuracy is measured as half the unrooted hardwired cluster distance between the true and estimated topologies, which equals the Robinson-Foulds distance when both topologies are trees.

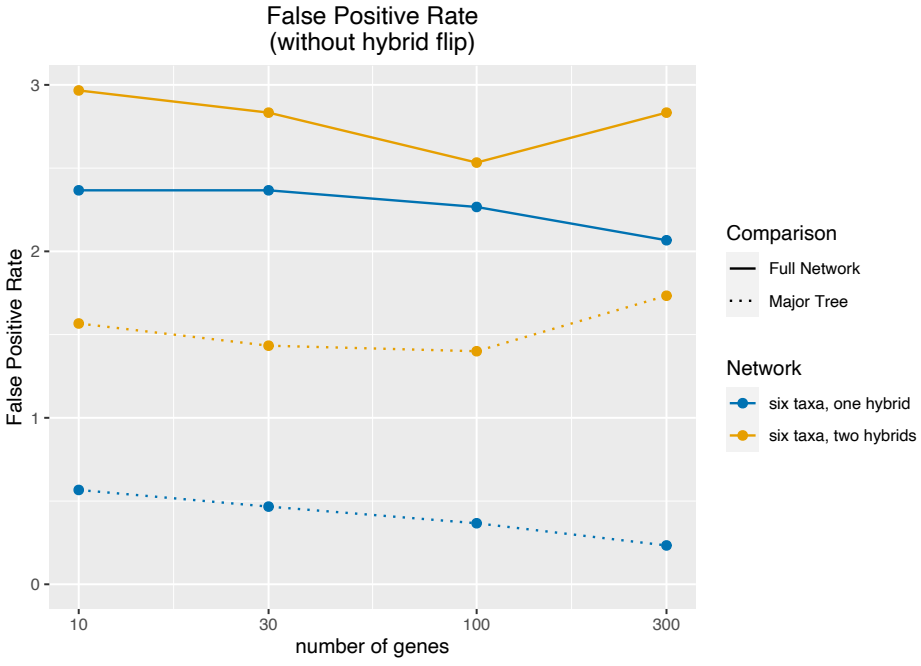

Figure 4.17: Accuracy of PhyLiNC to recover the full network and major tree from sequence alignments. Accuracy is measured as half the unrooted hardwired cluster distance between the true and estimated topologies, equal to the Robinson-Foulds distance when both topologies are trees.

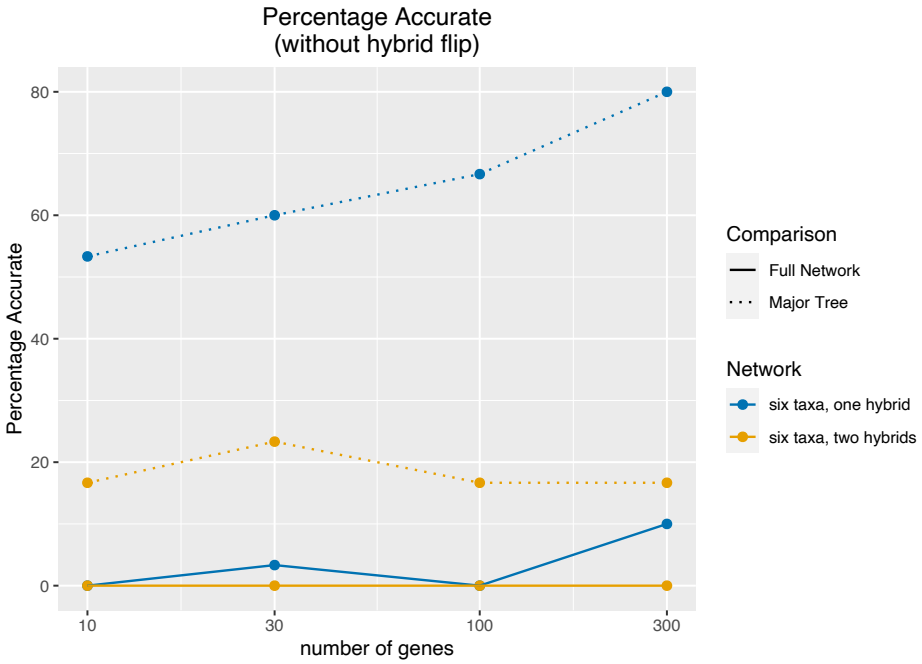

Figure 4.18: Percentage of networks and major trees estimated perfectly by PhyLiNC from sequence alignments. The major tree is obtained by suppressing all minor hybrid edges ($\gamma < 0.5$) to capture the major vertical inheritance pattern.

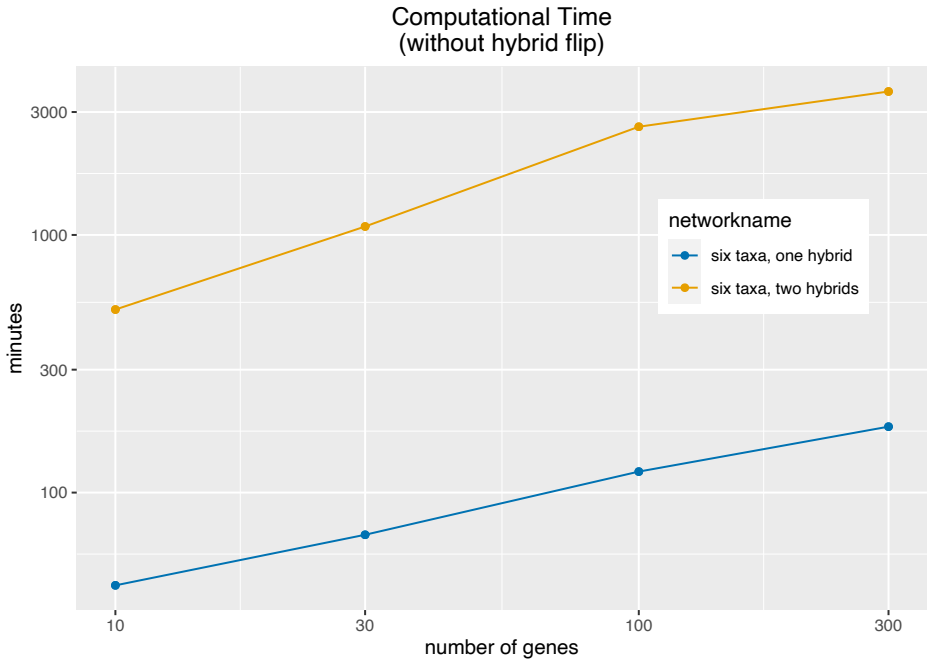

Figure 4.19: Time to estimate networks by the number of genes. Time and number of genes are both displayed on the log scale.

4.4.2 Exploring the Efficacy of Topology Search with Hybrid Flip Move

In section 4.4.1, we examined how well the topology search works without a hybrid flip move. Given the low accuracy shown in that simulation experiment, we were motivated to see if we could search the space of networks more effectively. To do this, we implemented the hybrid flip move, which reverses the direction of a minor hybrid edge within the network. The space of networks is connected without this move (Gambette *et al.*, 2017). However, some effective network reconstruction methods do include a move to flip hybrid edges (Solís-Lemus and Ané, 2016; Wen *et al.*, 2016). We hypothesized that adding a hybrid flip move would enlarge the neighborhood of each network and help our topology search escape local minima.

In section 4.4.1, the topology search performed sNNI, add hybrid, delete hybrid, and rerooting moves at rates of 60%, 25%, 5%, and 10%, respectively. We repeated the experiment in exactly the same way as in section 4.4.1, except that sNNI, add hybrid, delete hybrid, flip hybrid, and rerooting moves were proposed at rates of 60%, 20%, 5%, 5%, and 10%, respectively. As in section 4.4.1, we estimate a network on 1.8 - 2.9 GHz processors and each call to PhyLiNC includes ten runs, by default, which were run in parallel.

Results

Contrary to our hypothesis, PhyLiNC's estimation accuracy did not improve with the addition of a hybrid flip to the topology search. The average number of false positive edges estimated in the network and major tree increased in most cases (Figure 4.20). When estimating the network with one hybridization from 10 genes, the average distance between the estimated and true major tree increased from 0.6 without hybrid flips to 1.0 with hybrid flips. When estimating the network with two hybridizations from 300 genes, the average distance between the estimated and true network increased from 2.8 without hybrid flips to 3.2 with hybrid flips.

The percentage accurate estimation was also lower with this new addition (Figure 4.21). With the hybrid flip, the full network was never accurately estimated. The major tree was estimated more often without hybrid flips (18% - 80%) than with hybrid flips (10% - 50%). With the addition of the hybrid flip move, estimation was significantly faster (Figure 4.22), taking an average of between 4.5 and 88.6 minutes.

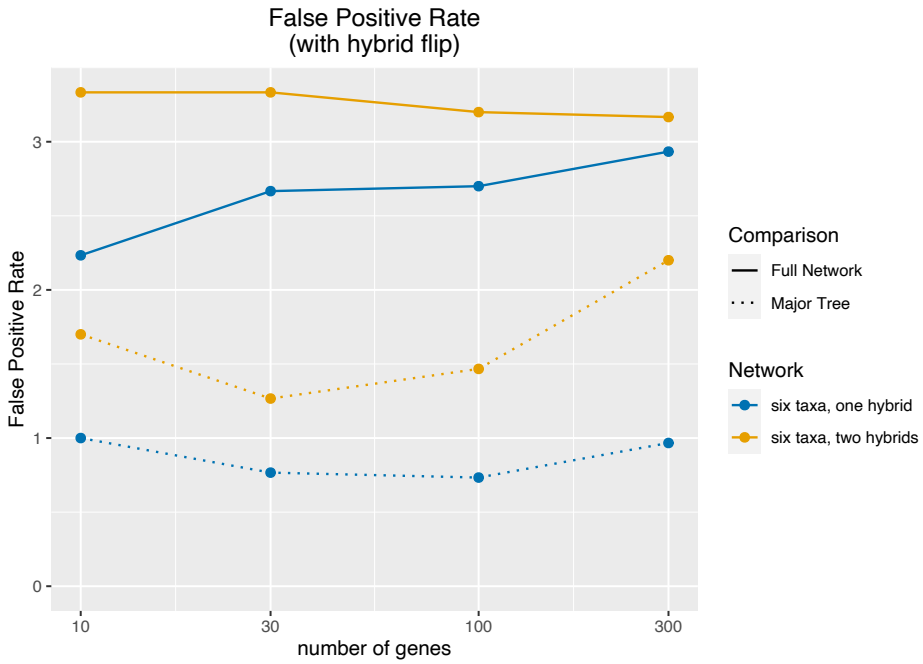

Figure 4.20: Accuracy of PhyLiNC with hybrid flip move to recover the full network and major tree from sequence alignments. Accuracy is measured as half the unrooted hardwired cluster distance between the true and estimated topologies

.

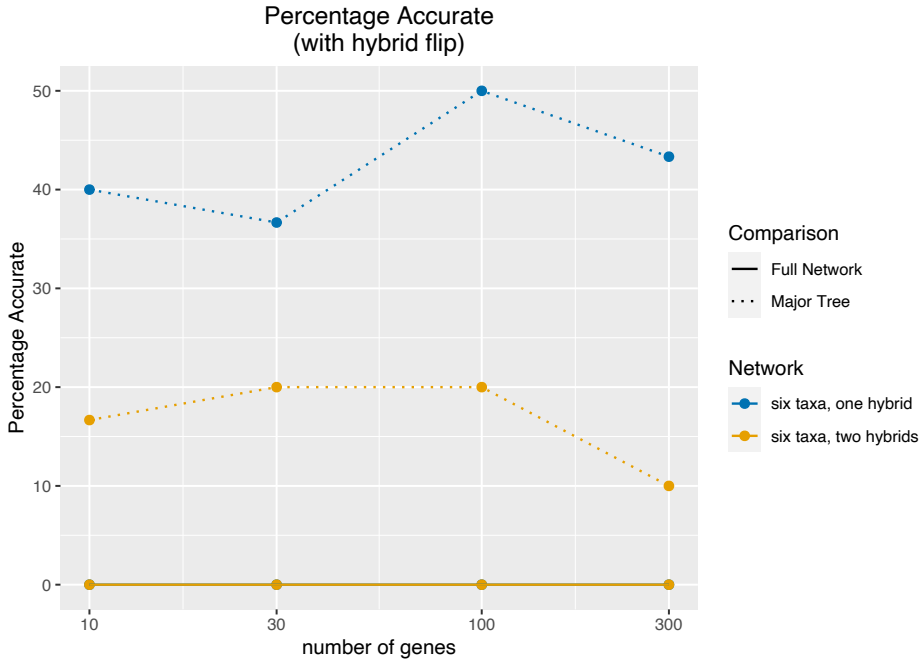

Figure 4.21: Percentage of networks and major trees estimated perfectly by PhyLiNC with hybrid flip move from sequence alignments. The major tree is obtained by suppressing all minor hybrid edges ($\gamma < 0.5$) to capture the major vertical inheritance pattern.

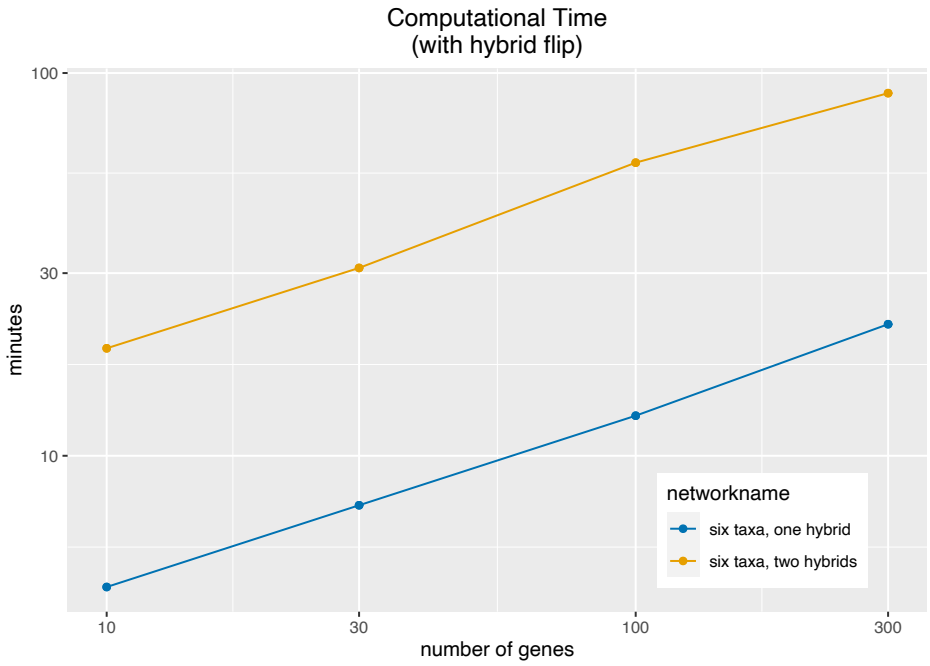

Figure 4.22: Computational time to estimate networks with PhyLiNC with hybrid flips. Time and number of genes are both displayed on the log scale.

4.4.3 Robustness to a Lack of Independence Between Sites

We further explored the robustness of PhyLiNC's inference to a violation of the assumption of independence between sites using a realistic network developed from real data. The original network was estimated from 12 Myotis mouse-eared bat taxa by Morales and Carstens (2018) using ASTRAL (Mirarab *et al.*, 2014). From this network, we selected six taxa, added realistic hybrid edges to allow us to fully explore PhyLiNC's ability to find hybridizations at different levels of phylogenetic history (Figure 4.23). We used branch lengths in coalescent units from figures in Morales and Carstens (2018).

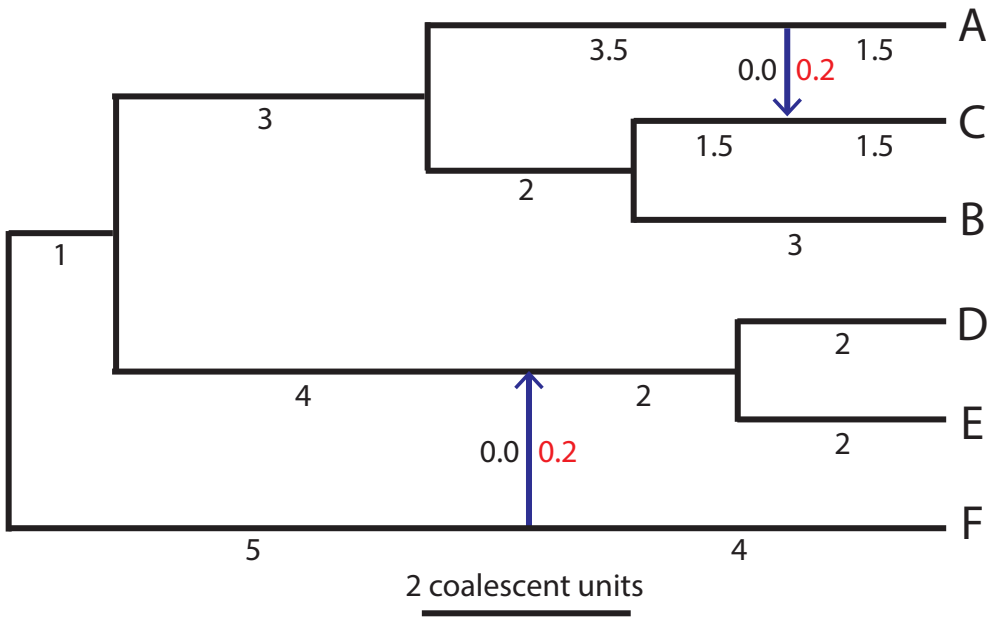

Figure 4.23: The ultrametric myotis-derived network used to simulate data. Branch lengths are in coalescent units. Realistic horizontal edges were added for the simulation.

Starting from this as our true network, we explored how PhyLiNC performs on sequence data with a range of dependence. We simulated two types of data: grouped sites (genes or loci) and independent SNP-like sites. We simulated 4000 (for independent sites) or 40 (for grouped sites) gene trees with Hybrid Lambda software under the Kingman coalescent model with population size of 10,000 and mutation rate per locus of 0.00005 (Zhu *et al.*, 2015). For each of these gene trees, we converted the coalescent-unit branch lengths to substitutions-per-site-unit branch lengths by multiplying the coalescent-unit branch lengths by 0.02. (We chose the constant 0.02 for biologically-plausible branch lengths between 0.01 and 0.1.) For each of these adjusted gene trees, we simulated the evolution of either 1-site (for independent sites) or 100-site (for grouped sites) DNA sequences under the HKY85 substitution model with $\kappa = 2$ and base frequencies

= [0.300414, 0.191363, 0.196748, 0.311475] without variation in rates across sites using Seq-Gen software (Rambaut and Grassly, 1997).

After concatenating these sequences, we estimated starting trees with IQ-TREE, then estimated a network with PhyLiNC (Nguyen *et al.*, 2015). We ran the simulation with and without the hybrid flip move. We generated ten replicates for each version of PhyLiNC (with and without the hybrid flip move) and each data type (independent and grouped sites). In each case, PhyLiNC was run with a maximum of two hybridizations. Each call to PhyLiNC included ten runs, by default, which were run in parallel. All simulations were run on 3.0 GHz processors.

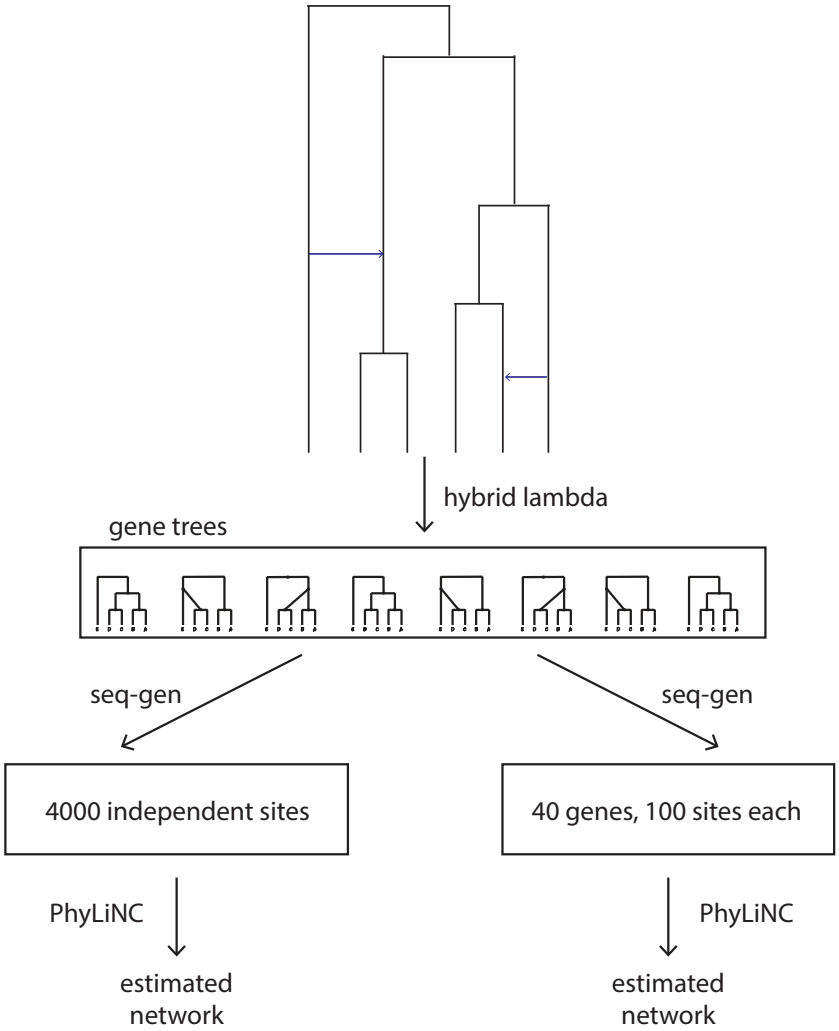

Figure 4.24: Simulation Three. We simulated two types of sequence data: concatenated independent sites and concatenated grouped sites. We generated ten replicates for each version of PhyLiNC (with and without the hybrid flip move) and each data type (independent and grouped sites).

RESULTS

PhyLiNC estimated a network from independent sites about as accurately as from grouped sites. It recovered the major tree significantly more closely from independent site data compared with grouped site data ($p = 0.036$). The average number of false positive edges in the network was 3.4 (without hybrid flip) and 3.6 (with hybrid flip) for independent sites, compared with 3.1 (without hybrid flip) and 3.9 (with hybrid flip) for grouped sites (Figure 4.25). The major tree was estimated more accurately from independent site data, with an average of 0.5 and 0.6 false positive edges (without, with hybrid flip), compared with 0.9 and 1.2 (without, with hybrid flip) for the grouped site case.

PhyLiNC was not able to recover the full network perfectly. However, it was able to recover the major tree more often from independent site data, in between 40-60% of simulations (Figure 4.26). From grouped site data, the percent of perfectly-estimated major trees was lower, at 20-40%. The time costs of estimating a network from independent and grouped site data were about equivalent when the hybrid flip move was included, with an average of 75 minutes for the independent site case and 66 minutes for the grouped site case. Without the hybrid flip move, estimating a network from independent sites took significantly longer on average, 1010 minutes compared with 432 minutes for grouped sites.

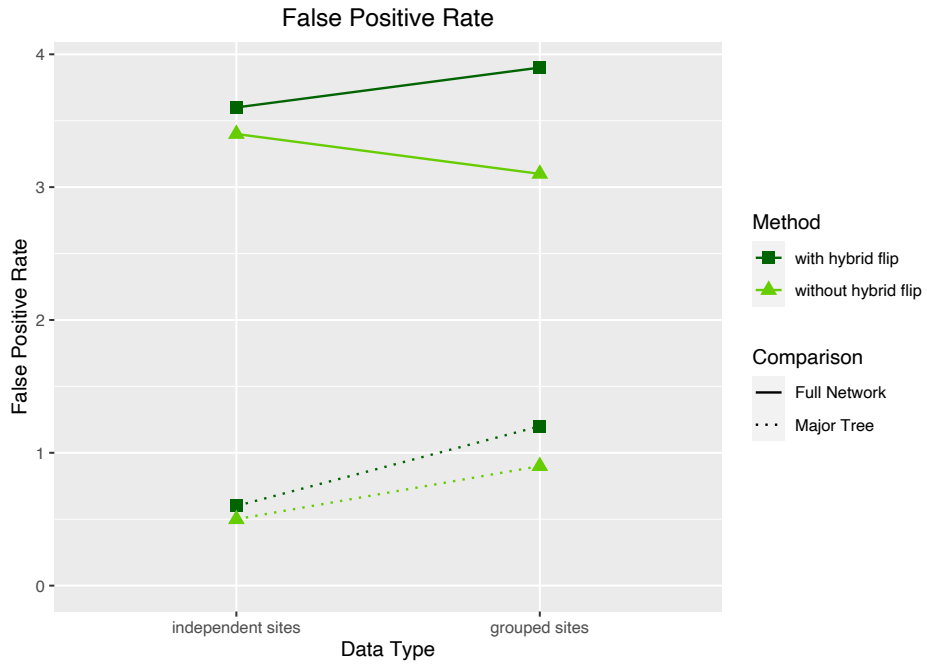

Figure 4.25: Accuracy of PhyLiNC move to recover the full network and major tree from sequence alignments. Independent site data came from loci with only one site, while grouped site data came from loci with 100 sites each.

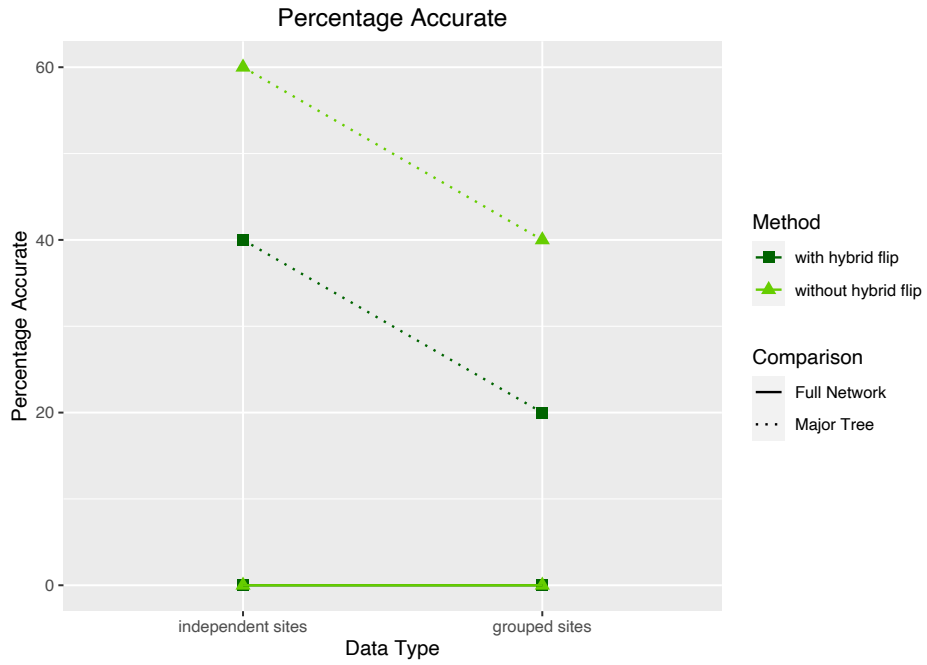

Figure 4.26: Percentage of networks and major trees estimated perfectly by PhyLiNC move from sequence alignments. Independent site data came from loci with only one site, while grouped site data came from loci with 100 sites each.

4.5 Concluding Remarks

In this chapter, we presented our method for estimating parameterized phylogenetic networks from DNA sequence data. We focused particularly on the topology search and presented our adaptation of nearest neighbor moves to parameterized semi-directed networks. We presented results from three simulation experiments to explore the method's accuracy and robustness to common violations of assumptions. While our method's time performance was good, its accuracy was not as high as we had hoped. Current results suggest that the addition of a new topology search move, a hybrid flip move, did not improve the accuracy of network estimation.

The complex biological processes underlying evolution make estimating phylogenies challenging. Hybridization, incomplete lineage sorting, recombination, duplication, loss, and unexpectedly low or high mutation rates all add complexity and make network reconstruction more difficult. Ignoring hybridizations can lead to misidentifying the major tree lineage (Leache *et al.*, 2014; Solís-Lemus *et al.*, 2016), but high rates of incomplete lineage sorting, duplication, or loss could make these hybridizations and the overall topology challenging to infer (Degnan, 2018; Zhu and Degnan, 2017). In addition, a low mutation rate leads to uniformity among samples providing too little signal, while a high mutation rate leads to high noise, again making the network challenging to estimate (Degnan, 2018).

With our method, we handle hybridization and variation in mutation rates by inferring hybridization events, estimating evolutionary rates, and allowing for variation in rates across sites. However, some possible sources of error do remain. We make several important assumptions that, if violated, could lead to errors in estimation. First, we do not consider incomplete lineage sorting or duplication and loss in our model, meaning that we must assume they do not play a significant role in our data. Second, we assume that sites are independent. Because of this assumption, our model is more suited to data like SNPs and highly invariant data than to clustered gene or locus data.

4.5.1 Future Work

Next, I discuss three areas for future work to extend the methodologies presented in this thesis.

Identifiability In Chapter Three, we present identifiability results for semi-directed phylogenetic networks. We show that root placement, small cycle structures, some branch lengths, and hybrid ladder

node ordering are not or nearly not identifiable. From these results, we define and estimate canonical networks under our model.

While we have extended the understanding of identifiability of phylogenetic networks under our model, we cannot ensure that this canonical network is fully identifiable from sequence data. Future work to prove that this canonical network is fully identifiable would give us confirmation that our method does not attempt to distinguish between indistinguishable networks (Pardi and Scornavacca, 2015).

BRANCH LENGTH AND INHERITANCE WEIGHT OPTIMIZATION Accurate network estimation relies on the accurate network parameters, particularly branch lengths and inheritance weights. Future work on the optimal sampling of branches for local optimization could lead to improvements in computational efficiency. Currently, our method samples branch lengths randomly with replacement for local optimization between topology moves. Instead of sampling with replacement for local optimization, we could sample branches without replacement, in sets called “epochs". After one epoch is complete, we would repeat this sampling without replacement until topology optimization is complete. This would ensure that all branch lengths are optimized at least once per epoch. We hope that this could improve the efficiency of branch length and inheritance weight optimization during topology optimization, leading to better overall network estimation.

ANALYST-GUIDED CLADE GROUPING As part of our method implementation, we present an analyst-led option to group individuals within a population (presented in detail in section 4.3.4). We also presented challenges behind two additional ways to group species known to form a clade. Adding these additional clade grouping tools could be a useful way for analysts to contribute more of their field domain knowledge and improve the accuracy of network inference.

* * *

In this dissertation, we introduce a new likelihood-based method to estimate parameterized semi-directed phylogenetic networks from concatenated sequence data. We present a model for extending current efficient likelihood calculations to a network paradigm, as well as new identifiability results on phylogenetic networks under this model. We also extend nearest neighbor moves to parameterized semi-directed networks. Finally, we present a user-friendly implementation of our method. We hope that these contributions move the field of phylogenetic analysis forward and allow biologists to estimate phylogenetic

histories that more accurately reflect their biological systems.

Acknowledgements. This work was supported by National Science Foundation awards DMS-1902892 and DMS-2023239.

CHAPTER 5

PHYLINC USER MANUAL

Abstract This chapter provides a user-friendly manual for estimating phylogenetic networks using our method. It provides examples and step-by-step instructions to guide analysts through the first steps of estimating a network, selecting a model, grouping individuals within a species, and running our method in parallel.

Keywords network reconstruction; Markov models; evolution; parameter estimation; speciation; gene flow; hybridization; phylogenetic networks

Contribution I led computational work for this method and wrote the user tutorial.

5.1 Introduction

In this user manual, we describe how to estimate phylogenetic networks using PhyLiNC (phylogenetic Likelihood Network from Concatenated data), a network estimation method in the PhyloNetworks Julia package. We provide examples and step-by-step instructions to guide analysts through the first steps of estimating a network, selecting a model, grouping individuals within a species, and running our method in parallel.

Assumptions on the phylogenetic network We make several assumptions about the networks PhyLiNC estimates. First, we assume a model without incomplete lineage sorting. Second, we assume that sites within the data are independent. Third, we assume that the network does not include parallel edges, also known as two-cycles. These are not or nearly not-identifiable under our model, so we do not attempt to infer them. Fourth, the length of the edge below a reticulation is not identifiable, so we infer a network with reticulations unzipped: edges below reticulations are set to zero and hybrid edges (parental lineages) lengths are increased accordingly. Finally, we assume that a hybrid node can have only two parent edges, meaning that each hybridization event can only occur between two species. However, we do allow for multiple hybridization events in succession (with keyword argument nohybridladder=false) for more complex hybridization events.

5.2 Estimate a Network from Concatenated DNA Sequences using PhyLiNC

Starting at a given starting tree or network, PhyLiNC uses a hill-climbing algorithm to search for the network that best fits the given DNA sequences. The search starts at or within a Geometric-distributed number of NNIs of the starting tree or network. Then, it proceeds with a hill-climbing search made up of nearest neighbor interchange moves, root change moves, add hybrid, and remove hybrid proposals. Throughout topology optimization, it optimizes evolutionary rates, rate variation across sites, branch lengths and inheritance inheritance weights. This search strategy is run ten times, by default, and the best network is returned.

To estimate a phylogenetic network with PhyLiNC, need three pieces of information: concatenated DNA

sequence data, a starting network, and a substitution model.

DATA The DNA sequences should be in fasta format. We can find a small example called "h1.fasta" in the PhyloNetworks package in the examples folder. There is no need to read the data file into Julia ahead of time. To estimate a network, we simply need a path to the data file.

```
using PhyloNetworks
fastafile = joinpath(pkgdir(PhyloNetworks), "examples","h1_net.fasta")
```

STARTING TREE OR NETWORK A starting network can be obtained using the analyst's knowledge of the field or with a tree-building tool like IQTree (Nguyen *et al.*, 2015). It should be in Newick string format, as shown in the example "h1.tree" in the PhyloNetworks package in the examples folder. The starting tree or network cannot have more reticulations than our final estimated network: it should have at most maxhybrid reticulations. (If maxhybrid is not provided, then the starting network can have at most one reticulation.) The starting network can be of any level.

Read the starting tree or network into the Julia session with PhyloNetwork's ReadTopology() function.

```
startingtreefile = joinpath(pkgdir(PhyloNetworks), "examples","h1_net.treefile")
startingtree = readTopology(startingtreefile)
```

If any branch lengths are missing in the input network, phyLiNC estimates the starting branch lengths using pairwise distances. Otherwise, it uses the input branch lengths as starting branch lengths, only unzipping all reticulations, as described in the assumptions paragraph above.

ESTIMATE A NETWORK Now that we have data and a starting tree or network, we can estimate a phylogenetic network with the following call, where the first argument is the starting tree, the second is the fasta file path, and the third is a choice of substitution model, either :HKY85 or :JC69 (explained in detail in section 5.4).

```
using PhyloNetworks
fastafile = joinpath(pkgdir(PhyloNetworks), "examples","h1_net.fasta")
startingtreefile = joinpath(pkgdir(PhyloNetworks), "examples","h1_net.treefile")
startingtree = readTopology(startingtreefile)
netobject = PhyloNetworks.phyLiNC(startingtree, fastafile, :HKY85)
```

Note: Because Julia is a JIT (just-in-time) compiled programming language, the first use of a function takes longer than subsequent calls.

5.3 Beyond Defaults: Customize the Network's Optimization with Optional Arguments

Only three arguments are required to estimate a network with PhyLiNC: a starting tree or network, a fasta file path, and a substitution model symbol. However, in many cases, analysts may want to move beyond this simple example for a more customized estimation. We use two types of optional arguments: positional and keyword arguments. Positional arguments appear before the semi-colon in the argument call and are identified by their position. Keyword arguments are placed after the semi-colon and must be appear with their name and an equal sign: "maxhybrid = 1".

Positional Arguments (default value in parentheses):

- symbol for the model of rate variation across sites (:noRV for no rate variation). Use :G or :Gamma for Gamma-distributed rates, :I or :Inv for a proportion of invariable sites, or :GI or :GammaInv for a combination.
- integer (4) for the number of categories to use in estimating evolutionary rates using a discretized gamma model. When allowing for rate variation, four categories is standard. With 1 category, no rate variation is assumed.

Optional keyword arguments for topology estimation and logging (default value in parentheses):

- **speciesfile** (""): path to a csv file with samples in rows and two columns: species (column 1), individual (column 2). Include this file to group individuals by species.
- **filename** ("phyLiNC"): root name for the output files (.out, .err). If empty (""), files are **not** created, progress log goes to the screen only (standard out).
- **maxhybrid** (1): the maximum number of hybridizations to include in the estimated network (allows one hybridization by default). Note: the starting network should have maxhybrid or fewer reticulations.

- **nruns** (10): number of independent starting points for the search
- **no3cycle** (true): prevents 3-cycles, which are (almost) not identifiable
- **nohybridladder** (true): prevents hybrid ladder in network. If true, the input network must not have hybrid ladders. Warning: Setting this to true will avoid most hybrid ladders, but some can still occur in some cases when deleting hybrid edges. (This might be replaced with an option to avoid all non-tree-child networks in the future.)

Optional keyword arguments to control the search and output (default value in parentheses):

- **seed** (0 for randomly-chosen seed): seed to replicate a given search
- **nreject** (75): Controls when to stop proposing new network topologies and, ultimately, the precision of the estimation. This gives the maximum number of times that new topologies are proposed and rejected in a row. Lower values of nreject result in a less thorough but faster search. A higher value will lead to more precise estimation and higher computing times. When running test cases, use a small nreject such as 5.
- **probST** (0.5): probability to use net as the starting topology for each given run. If probST < 1, the starting topology is k NNI moves away from net, where k is drawn from a geometric distribution: $p(1-p)^k$, with success probability p = probST.
- **maxmoves** (100): maximum number of topology moves before branch lengths, hybrid weight values, evolutionary rates, and rate variation parameters are reestimated.
- **verbose** (true): set to false to turn off screen output
- **alphamin** (0.02): minimum value for shape parameter alpha in rate variation across sites model.
- **alphamax** (50.0): maximum value for shape parameter alpha in rate variation across sites model.

Optional keyword arguments to control the optimization of branch lengths and inheritance weights (default value in parentheses):

- **ftolRel** (1e-6) and **ftolAbs** (1e-6): relative and absolute differences of the network score between the current and proposed parameters

- **xtolRel** (1e-5) and **xtolAbs** (1e-5): relative and absolute differences between the current and proposed parameters. Greater values will result in a less thorough but faster search. These parameters are used when evaluating candidate networks only. Regardless of these arguments, once a final topology is chosen, branch lengths are optimized using stricter tolerances (1e-10, 1e-12, 1e-10, 1e-10) for better estimates.

The following code block shows an example using both positional and keyword optional arguments.

```
obj_custom = PhyloNetworks.phyLiNC(startingtree, fastafile, :JC69, :G, 2;
    maxhybrid=3, nreject=5, verbose=true, seed=123)
```

5.4 Modeling Evolutionary Rates

PhyLiNC allows analysts to choose from two Markov models for evolution and four ways to model rate variation. Together, these can be combined for eight ways to model evolutionary rates.

5.4.1 Markov Model Substitution Models

There are two nucleic acid substitution models to choose from: JC69 (Jukes and Cantor, 1969) and HKY85 (Hasegawa *et al.*, 1985). For each model, the transition matrix Q is normalized to have an average of 1 mutation per unit of time.

- JC69: The Jukes & Cantor (1969) nucleic acid substitution model assumes a single evolutionary rate for all transitions between the characters A, C, G, and T.
- HKY85: The HKY85 nucleic acid substitution model lets transitions occur at the different rate than transversions. The model is made up of two components: a transition/transversion ratio, which we call κ or rate, and a vector of probabilities, called π. The rate presents the transition/transversion ratio: $\kappa = \alpha/\beta$. The vector π shows the base frequencies of each character state (A, C, G, T) in the data. These are estimated from the data and must sum to 1.0.

PhyLiNC optimizes the rate parameters in the model throughout network estimation. After estimation is complete, it present the transition rate matrix Q for the final network. The model is stored in the object that the function phyLiNC returns, which we have called netobj here. To explore the model further, use the following commands.

```
model = netobject.model
nstates(model)
getlabels(netobject.model)
```

5.4.2 Rate Variation and Invariable Sites

By default, the model does not allow rates to vary across sites. However, we can allow for more complexity by including an extra symbol argument in the function call. This turns either of the substitution models above into the "+G" or "+I" version: e.g. HKY85+G or JC69+I. To allow rates to vary across sites, we can add a discrete Gamma model (+G), developed by Yang (1994). To account for invariable sites, we can use the +I model developed by Hasegawa *et al.* (1985). We can combine the two types of rate variation (+G+I, Gu *et al.* (1995)) but this is discouraged (Jia *et al.*, 2014).

Using rate variation increases the number of parameters by one (+G or +I) or by two (+G+I). Because the mean of the desired distribution or rates is 1, we use a Gamma distribution with shape parameter α and scale parameter $\theta = 1/\alpha$ (so $\beta = \alpha$).

For example, to allow for variable rates across sites using the discrete Gamma model, we add the :G symbol to our function call. We can also specify the number of categories for our discrete Gamma model by including the number after the rate variation model symbol. (Four categories is standard. If one category is specified, then rates are assumed not to vary.)

```
obj_G = PhyloNetworks.phyLiNC(startingtree, fastafile, :JC69, :G, 4)
```

Our model is well-suited to data with a high percentages of invariable sites. To model invariant sites, adding +I to the Markov model, run phyLiNC with the :I symbol.

```
obj_I = PhyloNetworks.phyLiNC(startingtree, fastafile, :JC69, :I)
```

After estimating the network, phyLiNC returns the rate variation model, in addition to the Markov model. We can explore the details of our network's model in this way.

```
ratemodel = obj_G.ratemodel
nparams(ratemodel)
```

5.5 Group Multiple Alleles or Individuals within a Species

We allow analysts to fix alleles or individuals within a species group. This constrains the output network to connect this set of individuals to one polytomy node representing the base of the species.

To group individuals within a species in this way, we include a species mapping file path as a keyword argument. This file groups individuals by species and should have the following format: a csv file with samples in rows and two columns named "species" and "individual". An example can be found in the examples folder ("mappingIndividuals.csv").

With a population-level starting network or tree, a fasta file path with a line for each individual, and a species mapping file path, we can estimate a network with grouped individuals with this call:

```
populationtree = joinpath(pkgdir(PhyloNetworks), "examples", "population_level.
    treefile")
individualfasta = joinpath(pkgdir(PhyloNetworks), "examples", "individuals.fasta")
mappingfile = joinpath(pkgdir(PhyloNetworks), "examples", "mappingIndividuals.csv"
    )
obj_multipleindiv = PhyloNetworks.phyLiNC(populationtree, individualfasta, :JC69;
    maxhybrid=2, speciesfile=mappingfile)
```

5.6 Estimate Networks in Parallel

By default, phyLiNC estimates ten networks, then returns the network with the highest likelihood. To speed this process, we can compute these ten runs in parallel. For example, if our machine has multiple cores, we can tell Julia to add ten worker processes by starting Julia with the command

```
julia -p 10
```

This starts Julia with eleven processes: one main and ten workers. Alternatively, we can start Julia as usual, then add processes. If we are unsure how many we have added, we can check using nworkers().

```
using Distributed
addprocs(10)
nworkers()
```

This allows PhyLiNC to estimate a network more quickly, completing the ten runs in parallel using these ten worker processes.

Acknowledgements. This work was supported by National Science Foundation awards DMS-1902892 and DMS-2023239.

BIBLIOGRAPHY

Aitkin, M. and Longford, N. (1986). Statistical Modelling Issues in School Effectiveness Studies. *Journal of the Royal Statistical Society. Series A (General)*, **149**(1), 1–43.

Allman, E. S. and Rhodes, J. A. (2006). The identifiability of tree topology for phylogenetic models, including covarion and mixture models. *Journal of Computational Biology*, **13**(5), 1101–13.

Allman, E. S., Ané, C., and Rhodes, J. A. (2008). Identifiability of a Markovian model of molecular evolution with gamma-distributed rates. *Advances in Applied Probability*, **40**(1), 229–249.

Allman, E. S., Petrović, S., Rhodes, J. A., and Sullivant, S. (2011). Identifiability of two-tree mixtures for group-based models. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **8**, 710–722.

Athens, J. K., Catlin, B. B., Remington, P. L., and Gangnon, R. E. (2013). Using empirical bayes methods to rank counties on population health measures. *Preventing Chronic Disease*, **10**, E129.

Athens, J. K., Remington, P. L., and Gangnon, R. E. (2015). Improving the rank precision of population health measures for small areas with longitudinal and joint outcome models. *PLoS ONE*, **10**(6), e0130027.

Blischak, P. D., Chifman, J., Wolfe, A. D., and Kubatko, L. S. (2018). HyDe: a Python Package for Genome-Scale Hybridization Detection. *Systematic Biology*, **67**(5), 821–829.

Boni, M. F., Lemey, P., Jiang, X., Lam, T. T. Y., Perry, B. W., Castoe, T. A., Rambaut, A., and Robertson, D. L. (2020). Evolutionary origins of the SARS-CoV-2 sarbecovirus lineage responsible for the COVID-19 pandemic. *Nature Microbiology*, **5**, 1408–1417.

Bordewich, M. and Semple, C. (2016). Determining phylogenetic networks from inter-taxa distances. *Journal of Mathematical Biology*, **73**(2), 283–303.

Bordewich, M., Linz, S., and Semple, C. (2017). Lost in space? Generalising subtree prune and regraft to spaces of phylogenetic networks. *Journal of Theoretical Biology*, **423**, 1–12.

Bordewich, M., Semple, C., and Tokac, N. (2018). Constructing Tree-Child Networks from Distance Matrices. *Algorithmica*, **80**, 2240–2259.

Bouckaert, R., Vaughan, T. G., Barido-Sottani, J., Duchêne, S., Fourment, M., Gavryushkina, A., Heled, J., Jones, G., Kühnert, D., De Maio, N., Matschiner, M., Mendes, F. K., Müller, N. F., Ogilvie, H. A., du Plessis, L., Popinga, A., Rambaut, A., Rasmussen, D., Siveroni, I., Suchard, M. A., Wu, C. H., Xie, D., Zhang, C., Stadler, T., and Drummond, A. J. (2019). BEAST 2.5: An advanced software platform for Bayesian evolutionary analysis. *PLoS computational biology*, **15**(4), e1006650.

Bryant, D. and Hahn, M. W. (2020). The Concatenation Question. In C. Scornavacca, F. Delsuc, and N. Galtier, editors, *Phylogenetics in the Genomic Era*, chapter 3.4, pages 3.4:1–3.4:23. No commercial publisher.

Burbrink, F. T. and Gehara, M. (2018). The Biogeography of Deep Time Phylogenetic Reticulation. *Systematic biology*, **67**(5), 743–744.

Burmeister, A. R. (2015). Horizontal Gene Transfer. *Evolution, Medicine and Public Health*, **1**, 193–194.

Candido, D. S., Claro, I. M., de Jesus, J. G., Souza, W. M., Moreira, F. R., Dellicour, S., Mellan, T. A., du Plessis, L., Pereira, R. H., Sales, F. C., Manuli, E. R., Thézé, J., Almeida, L., Menezes, M. T., Voloch, C. M., Fumagalli, M. J., Coletti, T. M., da Silva, C. A., Ramundo, M. S., Amorim, M. R., Hoeltgebaum, H. H., Mishra, S., Gill, M. S., Carvalho, L. M., Buss, L. F., Prete, C. A., Ashworth, J., Nakaya, H. I., Peixoto, P. S., Brady, O. J., Nicholls, S. M., Tanuri, A., Rossi, Á. D., Braga, C. K., Gerber, A. L., de Guimarães, A. P. C., Gaburo, N., Alencar, C. S., Ferreira, A. C., Lima, C. X., Levi, J. E., Granato, C., Ferreira, G. M., Francisco, R. S., Granja, F., Garcia, M. T., Moretti, M. L., Perroud, M. W., Castiñeiras, T. M., Lazari, C. S., Hill, S. C., de Souza Santos, A. A., Simeoni, C. L., Forato, J., Sposito, A. C., Schreiber, A. Z., Santos, M. N., de Sá, C. Z., Souza, R. P., Resende-Moreira, L. C., Teixeira, M. M., Hubner, J., Leme, P. A., Moreira, R. G., Nogueira, M. L., Ferguson, N. M., Costa, S. F., Proenca-Modena, J. L., Vasconcelos, A. T. R., Bhatt, S., Lemey, P., Wu, C. H.,

Rambaut, A., Loman, N. J., Aguiar, R. S., Pybus, O. G., Sabino, E. C., and Faria, N. R. (2020). Evolution and epidemic spread of SARS-CoV-2 in Brazil. *Science*, **369**(6508), 1255–1260.

Cao, Z., Liu, X., Ogilvie, H. A., Yan, Z., and Nakhleh, L. (2019). Practical Aspects of Phylogenetic Network Analysis Using PhyloNet. *bioRxiv*.

Cardona, G., Rosselló, F., and Valiente, G. (2008). Tripartitions do not always discriminate phylogenetic networks. *Mathematical Biosciences*, **211**(2), 356–70.

Chang, J. T. (1996). Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency. *Mathematical Biosciences*, **137**, 51–73.

Chen, X., Lemmon, A. R., Lemmon, E. M., Pyron, R. A., and Burbrink, F. T. (2017). Using phylogenomics to understand the link between biogeographic origins and regional diversification in ratsnakes. *Molecular phylogenetics and evolution*, **111**, 206–218.

Chifman, J. and Kubatko, L. (2014). Quartet inference from SNP data under the coalescent model. *Bioinformatics*, **30**(23), 3317–24.

Chung, Y. and Dunson, D. B. (2009). Nonparametric bayes conditional distribution modeling with variable selection. *Journal of the American Statistical Association*, **104**(488), 1646–1660.

Cui, R., Schumer, M., Kruesi, K., Walter, R., Andolfatto, P., and Rosenthal, G. G. (2013). Phylogenomics reveals extensive reticulate evolution in Xiphophorus fishes. *Evolution; international journal of organic evolution*, **67**(8), 2166–2179.

Degnan, J. H. (2018). Modeling hybridization under the network multispecies coalescent. *Systematic Biology*, **67**(5), 786–799.

Elworth, R. A. L., Ogilvie, H. A., Zhu, J., and Nakhleh, L. (2019). Advances in Computational Methods for Phylogenetic Networks in the Presence of Hybridization. In T. Warnow, editor, *Bioinformatics and Phylogenetics*, pages 317–360. Springer, Cham.

Erdős, P. L., Francis, A., and Mezei, T. R. (2020). Rooted NNI moves on tree-based phylogenetic networks. *arXiv*.

Felsenstein, J. (1973). Maximum Likelihood and Minimum-Steps Methods for Estimating Evolutionary Trees from Data on Discrete Characters. *Systematic Zoology*, **22**(3), 240–249.

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of molecular evolution*, **17**(6), 368–376.

Felsenstein, J. (2004). *Inferring Phylogenies*. Sinauer Associates, Sunderland, Mass., 2nd edition.

Fontaine, M. C., Pease, J. B., Steele, A., Waterhouse, R. M., Neafsey, D. E., Sharakhov, I. V., Jiang, X., Hall, A. B., Catteruccia, F., Kakani, E., Mitchell, S. N., Wu, Y.-C., Smith, H. A., Love, R. R., Lawniczak, M. K., Slotman, M. A., Emrich, S. J., Hahn, M. W., and Besansky, N. J. (2015). Mosquito genomics. Extensive introgression in a malaria vector species complex revealed by phylogenomics. *Science (New York, N.Y.)*, **347**(6217), 1258524.

Forster, P., Forster, L., Renfrew, C., and Forster, M. (2020). Phylogenetic network analysis of SARS-CoV-2 genomes. *Proceedings of the National Academy of Sciences of the United States of America*, **117**(17), 9241–9243.

Francis, A. and Moulton, V. (2018). Identifiability of tree-child phylogenetic networks under a probabilistic recombination-mutation model of evolution. *Journal of Theoretical Biology*.

Francis, A., Huber, K. T., Moulton, V., and Wu, T. (2018). Bounds for phylogenetic network space metrics. *Journal of mathematical biology*, **76**(5), 1229–1248.

Gadagkar, S. R., Rosenberg, M. S., and Kumar, S. (2005). Inferring species phylogenies from multiple genes: Concatenated sequence tree versus consensus gene tree. *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, **304B**(1), 64–74.

Gambette, P. and Huber, K. T. (2012). On encodings of phylogenetic networks of bounded level. *Journal of Mathematical Biology*, **65**, 157–180.

Gambette, P., van Iersel, L., Jones, M., Lafond, M., Pardi, F., and Scornavacca, C. (2017). Rearrangement moves on rooted phylogenetic networks. *PLoS computational biology*, **13**(8), e1005611.

Gelfand, A. E., Kottas, A., and Maceachern, S. N. (2005). Bayesian nonparametric spatial modeling with dirichlet process mixing. *Journal of the American Statistical Association*, **100**(471), 1021–1035.

Gibbons, J. D., Olkin, I., and Sobel, M. (1979). An introduction to ranking and selection. *American Statistician*, **33**(4), 185–195.

Goldstein, H. and Spiegelhalter, D. J. (1996). League tables and their limitations: Statistical issues in comparisons of institutional performance. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, **159**(3), 385–409.

Green, R. E., Krause, J., Briggs, A. W., Maricic, T., Stenzel, U., Kircher, M., Patterson, N., Li, H., Zhai, W., Fritz, M. H. Y., Hansen, N. F., Durand, E. Y., Malaspinas, A. S., Jensen, J. D., Marques-Bonet, T., Alkan, C., Prüfer, K., Meyer, M., Burbano, H. A., Good, J. M., Schultz, R., Aximu-Petri, A., Butthof, A., Höber, B., Höffner, B., Siegemund, M., Weihmann, A., Nusbaum, C., Lander, E. S., Russ, C., Novod, N., Affourtit, J., Egholm, M., Verna, C., Rudan, P., Brajkovic, D., Kucan, Ž., Gušic, I., Doronichev, V. B., Golovanova, L. V., Lalueza-Fox, C., De La Rasilla, M., Fortea, J., Rosas, A., Schmitz, R. W., Johnson, P. L., Eichler, E. E., Falush, D., Birney, E., Mullikin, J. C., Slatkin, M., Nielsen, R., Kelso, J., Lachmann, M., Reich, D., and Pääbo, S. (2010). A draft sequence of the neandertal genome. *Science*, **328**(5979), 710–722.

Gross, E. and Long, C. (2018). Distinguishing Phylogenetic Networks. *SIAM JOURNAL ON APPLIED ALGEBRA AND GEOMETRY*, **2**(1), 72–93.

Gross, E., Van Iersel, L., Janssen, R., Jones, M., Long, C., and Murakami, Y. (2020). Distinguishing level-1 phylogenetic networks on the basis of data generated by Markov processes. *arXiv*.

Gu, X., Fu, Y. X., and Li, W. H. (1995). Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. *Molecular Biology and Evolution*, **12**(4), 546–557.

Hasegawa, M., Kishino, H., and Yano, T.-a. (1985). Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution*, **22**(2), 160–174.

Hejase, H. A. and Liu, K. J. (2016). A scalability study of phylogenetic network inference methods using empirical datasets and simulations involving a single reticulation. *BMC Bioinformatics*, **17**(422).

Henderson, N. C. and Newton, M. A. (2016). Making the cut: improved ranking and selection for large-scale inference. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **78**(4), 781–804.

Hoang, D. T., Chernomor, O., von Haeseler, A., Minh, B. Q., and Vinh, L. S. (2018). UFBoot2: Improving the Ultrafast Bootstrap Approximation. *Molecular biology and evolution*, **35**(2), 518–522.

Hornik, K. (2005). A CLUE for CLUster Ensembles. *Journal of Statistical Software*, **14**(12).

Hornik, K. (2019). clue: Cluster ensembles. R package version 0.3-57.

Huber, K. T. and Moulton, V. (2013). Encoding and Constructing 1-Nested Phylogenetic Networks with Trinets. *Algorithmica*, **66**, 714–738.

Huber, K. T., Linz, S., Moulton, V., and Wu, T. (2016a). Spaces of phylogenetic networks from generalized nearest-neighbor interchange operations. *Journal of Mathematical Biology*, **72**(3), 699–725.

Huber, K. T., Moulton, V., and Wu, T. (2016b). Transforming phylogenetic networks: Moving beyond tree space. *Journal of theoretical biology*, **404**, 30–39.

Hudson, R. R. (2002). Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, **18**(2), 337–8.

Huelsenbeck, J. P., Larget, B., Miller, R. E., and Ronquist, F. (2002). Potential applications and pitfalls of Bayesian inference of phylogeny. *Systematic Biology*, **51**(5), 673–88.

Huson, D. H. and Scornavacca, C. (2011). A survey of combinatorial methods for phylogenetic networks. *Genome Biology and Evolution*, **3**, 23–35.

Huson, D. H., Rupp, R., and Scornavacca, C. (2010). *Phylogenetic networks: Concepts, algorithms and applications*. Cambridge University Press, Cambridge.

Isabel, S., Graña-Miraglia, L., Gutierrez, J. M., Bundalovic-Torma, C., Groves, H. E., Isabel, M. R., Eshaghi, A. R., Patel, S. N., Gubbay, J. B., Poutanen, T., Guttman, D. S., and Poutanen, S. M. (2020). Evolutionary and structural analyses of SARS-CoV-2 D614G spike protein mutation now documented worldwide. *Scientific Reports*, **10**(1), 14031.

Jacques, J., Grimonprez, Q., and Biernacki, C. (2014). Rankcluster: An R package for clustering multivariate partial rankings. *R Journal*, **6**(1), 101–110.

Jewett, P. I., Zhu, L., Huang, B., Feuer, E. J., and Gangnon, R. E. (2019). Optimal Bayesian point estimates and credible intervals for ranking with application to county health indices. *Statistical Methods in Medical Research*, **28**(9), 2876–2891.

Jia, F., Lo, N., and Ho, S. Y. (2014). The impact of modelling rate heterogeneity among sites on phylogenetic estimates of intraspecific evolutionary rates and timescales. *PLoS ONE*, **9**(5).

Jin, G., Nakhleh, L., Snir, S., and Tuller, T. (2006). Maximum likelihood of phylogenetic networks. *Bioinformatics*, **22**(21), 2604–2611.

Johnson, S. G. (2020). The NLopt nonlinear-optimization package.

Jukes, T. H. and Cantor, C. R. (1969). Evolution of protein molecules. In H. Munro, editor, *Mammalian protein metabolism*, chapter 24, pages 21–132. Academic Press, New York.

Kamneva, O. K. and Rosenberg, N. A. (2017). Simulation-based evaluation of hybridization network reconstruction methods in the presence of incomplete lineage sorting. *Evolutionary Bioinformatics*, **13**.

Kaplan, N. and Langley, C. H. (1979). A new estimate of sequence divergence of mitochondrial DNA using restriction endonuclease mappings. *Journal Of Molecular Evolution*, **13**(4), 295–304.

Keeling, P. J. and Palmer, J. D. (2008). Horizontal gene transfer in eukaryotic evolution. *Nature Reviews Genetics*, **9**, 605.

Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, **16**(2), 111–120.

Klawitter, J. (2020). *Spaces of phylogenetic networks*. Ph.D. thesis, The University of Auckland.

Kolaczkowski, B. and Thornton, J. W. (2004). Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. *Nature*, **431**(7011), 980–984.

Kozlov, A. M., Darriba, D., Flouri, T., Morel, B., and Stamatakis, A. (2019). RAxML-NG: A fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics*, **35**(21), 4453–4455.

Kraft, D. (1988). A software package for sequential quadratic programming. *Forschungsbericht Deutsche Forschungs und Versuchsanstalt für Luft und Raumfahrt*.

Kubatko, L. S., Carstens, B. C., and Knowles, L. L. (2009). STEM: Species tree estimation using maximum likelihood for gene trees under coalescence. *Bioinformatics*, **25**(7), 971–973.

Kuhn, H. W. (1955). The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, **2.1-2**, 83–97.

Laird, N. (1978). Nonparametric Maximum Likelihood Estimation of a Mixing Distribution. *Journal of the American Statistical Association*, **73**(364), 805–811.

Laird, N. M. and Louis, T. A. (1989). Empirical Bayes Ranking Methods. *Journal of Educational Statistics*, **14**(1), 29–46.

Larget, B. and Simon, D. L. (1999). Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. *Molecular Biology and Evolution*, **16**(6), 750–759.

Leache, A. D., Harris, R. B., Rannala, B., and Yang, Z. (2014). The influence of gene flow on species tree estimation: a simulation study. *Systematic biology*, **63**(1), 17–30.

Lee, Y. and Kim, H. (2020). Bayesian Nonparametric Joint Mixture Model for Clustering Spatially Correlated Time Series. *Technometrics*, **62**(3), 313–329.

Louis, T. A. and Shen, W. (1999). Innovations in bayes and empirical bayes methods: estimating parameters, populations and ranks. *Statistics in medicine*, **18**(17-18), 2493–2505.

Maddison, D. and Maddison, W. (1996). The Tree of Life Web Project.

Mallet, J. (2007). Hybrid speciation. *Nature*, **446**, 279–283.

Mallet, J., Besansky, N., and Hahn, M. W. (2016). How reticulated are species? *BioEssays*, **38**(2), 140–149.

Meng, C. and Kubatko, L. S. (2009). Detecting hybrid speciation in the presence of incomplete lineage sorting using gene tree incongruence: a model. *Theoretical Population Biology*, **75**(1), 35–45.

Mindell, D. P. (2013). The Tree of Life: Metaphor, Model, and Heuristic Device. *Systematic Biology*, **62**(3), 479–489.

Mirarab, S., Reaz, R., Bayzid, M. S., Zimmermann, T., S. Swenson, M., and Warnow, T. (2014). ASTRAL: Genome-scale coalescent-based species tree estimation. In *Bioinformatics*. Oxford University Press.

Morales, A. E. and Carstens, B. C. (2018). Evidence that Myotis lucifugus "Subspecies" are five nonsister species, despite gene flow. *Systematic Biology*, **67**(5), 756–769.

Moret, B. M., Nakhleh, L., Warnow, T., Linder, C. R., Tholse, A., Padolina, A., Sun, J., and Timme, R. (2004). Phylogenetic networks: Modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **1**(1), 13–23.

Mueller, N. F., Ogilvie, H., Zhang, C., Drummond, A., and Stadler, T. (2018). Inference of species histories in the presence of gene flow. *bioRxiv*.

Nakhleh, L. (2010). Evolutionary Phylogenetic Networks: Models and Issues. In *Problem Solving Handbook in Computational Biology and Bioinformatics*. SPRINGER, Boston, MA.

Nguyen, L. T., Schmidt, H. A., Von Haeseler, A., and Minh, B. Q. (2015). IQ-TREE: A fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Molecular Biology and Evolution*, **32**(1), 268–274.

Ochman, H., Lawrence, J. G., and Groisman, E. A. (2000). Lateral gene transfer and the nature of bacterial innovation. *Nature*, **405**(6784), 299–304.

Paddock, S. M. and Louis, T. A. (2011). Percentile-based Empirical Distribution Function Estimates for Performance Evaluation of Healthcare Providers. *Journal of the Royal Statistical Society. Series C, Applied statistics*, **60**(4), 575–589.

Pamilo, P. and Nei, M. (1988). Relationships between gene trees and species trees. *Molecular Biology and Evolution*, **5**(5), 568–583.

Pardi, F. and Scornavacca, C. (2015). Reconstructible Phylogenetic Networks: Do Not Distinguish the Indistinguishable. *PLoS Computational Biology*, **11**(4).

Pearl, J. (1982). Reverend bayes on inference engines: a distributed hierarchical approach. In *AAAI-82 Proceedings*, pages 133–6.

Pereson, M. J., Mojsiejczuk, L., Martínez, A. P., Flichman, D. M., Garcia, G. H., and Di Lello, F. A. (2020). Phylogenetic analysis of SARS-CoV-2 in the first few months since its emergence. *Journal of Medical Virology*, page 10.1002/jmv.26545.

Pickrell, J. K. and Pritchard, J. K. (2012). Inference of Population Splits and Mixtures from Genome-Wide Allele Frequency Data. *PLOS Genetics*, **8**(11), e1002967.

Powell, M. J. D. (1994). A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In S. Gomez and J. Hennart, editors, *Advances in Optimization and Numerical Analysis. Mathematics and Its Applications, vol 275*. Springer, Dordrecht.

Rabier, C.-E., Berry, V., Glaszmann, J.-C., Pardi, F., and Celine, S. (2020). On the inference of complex phylogenetic networks by Markov Chain Monte-Carlo. *bioRxiv*.

Rambaut, A. and Grassly, N. C. (1997). Seq-gen: An application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Bioinformatics*, **13**(3), 235–8.

Robinson, D. F. and Foulds, L. R. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, **53**(1), 131–147.

Shen, W. and Louis, T. A. (1998). Triple-goal estimates in two-stage hierarchical models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **60**(2), 455.

Solís-Lemus, C. and Ané, C. (2016). Inferring Phylogenetic Networks with Maximum Pseudolikelihood under Incomplete Lineage Sorting. *PLoS Genetics*, **12**(3), 1–21.

Solís-Lemus, C., Yang, M., and Ané, C. (2016). Inconsistency of Species Tree Methods under Gene Flow. *Systematic Biology*, **65**(5), 843–851.

Solís-Lemus, C., Bastide, P., and Ane, C. (2017). PhyloNetworks: A Package for Phylogenetic Networks. *Molecular biology and evolution*, **34**(12), 3292–3298.

Solis-Lemus, C., Coen, A., and Ané, C. (2020). On the Identifiability of Phylogenetic Networks under a Pseudolikelihood model. *arXiv*.

Stamatakis, A. and Kozlov, A. M. (2020). Efficient Maximum Likelihood Tree Building Methods. In C. Scornavacca, F. Delsuc, and N. Galtier, editors, *Phylogenetics in the Genomic Era*, chapter 1.2, pages 1.2:1–1.2:18. No commercial publisher.

Steel, M. and Penny, D. (2000). Parsimony, Likelihood, and the Role of Models in Molecular Phylogenetics. *Molecular Biology and Evolution*, **17**(6), 839–850.

Strimmer, K. and Moulton, V. (2000). Likelihood analysis of phylogenetic networks using directed graphical models. *Molecular Biology and Evolution*, **17**(6), 875–81.

Strimmer, K., Wiuf, C., and Moulton, V. (2001). Recombination Analysis Using Directed Graphical Models. *Molecular Biology and Evolution*, **18**(1), 97–99.

Syvanen, M. (1994). Horizontal Gene Transfer: Evidence and Possible Consequences. *Annual Review of Genetics*.

Tavaré, S. (1986). Some probabilistic and statistical problems in the analysis of DNA sequences.

Thatte, B. D. (2013). Reconstructing pedigrees: Some identifiability questions for a recombination-mutation model. *Journal of Mathematical Biology*, **66**(1-2), 37–74.

Thiergart, T., Landan, G., and Martin, W. F. (2014). Concatenated alignments and the case of the disappearing tree. *BMC Evolutionary Biology*, **14**(1), 266.

Tonini J, Moore A, Stern D, Shcheglovitova M, O. G. (2015). Concatenation and Species Tree Methods Exhibit Range of Simulated Conditions Sci-Hub. *PLOS Currents Tree of Life*, pages 1–14.

van Iersel, L. and Moulton, V. (2014). Trinets encode tree-child and level-2 phylogenetic networks. *Journal of Mathematical Biology*, **68**, 1707–1729.

Wang, L., Didelot, X., Yang, J., Wong, G., Shi, Y., Liu, W., Gao, G. F., and Bi, Y. (2020). Inference of person-to-person transmission of COVID-19 reveals hidden super-spreading events during the early outbreak phase. *Nature Communications*, **11**, 5006.

Wen, D. and Nakhleh, L. (2017). Coestimating Reticulate Phylogenies and Gene Trees from Multilocus Sequence Data. *Systematic Biology*, **67**(April), 439–457.

Wen, D., Yu, Y., and Nakhleh, L. (2016). Bayesian Inference of Reticulate Phylogenies under the Multispecies Network Coalescent. *PLOS Genetics*, **12**(5), 1–17.

Wen, D., Yu, Y., Zhu, J., and Nakhleh, L. (2018). Inferring phylogenetic networks using PhyloNet. *Systematic Biology*.

Yang, Z. (1994). Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *Journal of Molecular Evolution*, **39**(3), 306–314.

Yu, Y. and Nakhleh, L. (2015). A maximum pseudo-likelihood approach for phylogenetic networks. *BMC Genomics*.

Yu, Y., Degnan, J. H., and Nakhleh, L. (2012). The probability of a gene tree topology within a phylogenetic network with applications to hybridization detection. *PLoS Genetics*, **8**(4).

Yu, Y., Dong, J., Liu, K. J., and Nakhleh, L. (2014). Maximum likelihood inference of reticulate evolutionary histories. *Proceedings of the National Academy of Sciences*, **111**(46), 16448–16453.

Zhang, C., Ogilvie, H. A., Drummond, A. J., and Stadler, T. (2018). Bayesian Inference of Species Networks from Multilocus Sequence Data. *Molecular Biology and Evolution*, **35**(2), 504–517.

Zhu, S. and Degnan, J. H. (2017). Displayed trees do not determine distinguishability under the network multispecies coalescent. *Systematic Biology*, **66**(2), 283–298.

Zhu, S., Degnan, J. H., Goldstien, S. J., and Eldon, B. (2015). Hybrid-Lambda: Simulation of multiple merger and Kingman gene genealogies in species networks and species trees. *BMC Bioinformatics*, **16**(1).