Improved maximum parsimony models for phylogenetic networks

van Iersel, Leo; Jones, Mark; Scornavacca, Celine

DOI
10.1093/sysbio/syx094

Publication date
2018

Document Version
Accepted author manuscript

Published in
Systematic Biology

Citation (APA)
van Iersel, L., Jones, M., & Scornavacca, C. (2018). Improved maximum parsimony models for phylogenetic networks. *Systematic Biology*, *67*(3), 518-542. [syx094]. https://doi.org/10.1093/sysbio/syx094

Improved maximum parsimony models for phylogenetic networks*

Leo van Iersel† Mark Jones† Celine Scornavacca‡

December 20, 2017

Abstract

Phylogenetic networks are well suited to represent evolutionary histories comprising reticulate evolution. Several methods aiming at reconstructing explicit phylogenetic networks have been developed in the last two decades. In this paper, we propose a new definition of maximum parsimony for phylogenetic networks that permits to model biological scenarios that cannot be modeled by the definitions currently present in the literature (namely, the "hardwired" and "softwired" parsimony). Building on this new definition, we provide several algorithmic results that lay the foundations for new parsimony-based methods for phylogenetic network reconstruction.

Phylogenetic networks are used to represent evolutionary relationships when the history of a set of taxa of interest accommodate events causing inheritance from multiple ancestors [9], a phenomenon called *reticulate evolution*. Examples of such reticulate events include hybrid speciation or hybridization [28, 1], horizontal gene transfer [3, 42] and recombination [34, 38]. In its broadest sense a phylogenetic network can simply be thought of as a graph (directed or undirected) with its leaves labeled by taxa. For an introduction to phylogenetic networks, see [21, 29].

1 Introduction

Separate lines of research have developed combinatorial methods to reconstruct explicit[1] phylogenetic networks. They all share the same underlying approach: First, combinatorial objects such as phylogenetic trees, clusters or trinets (networks on three leaves) are constructed from the data of the species under study; second, these combinatorial objects are combined into an explicit phylogenetic network. The way they are combined and the parameter to optimize give a large range of different problems. A review of this kind of approach can be found in [21].

In addition to these combinatorial approaches, in the last ten years a number of likelihood-based methods have appeared. They can be roughly categorized in two classes: those that have sequence alignments as input [22, 31]–analogous to classic maximum-likelihood methods for tree reconstruction – and those that instead use the predictions of population-genetics models – namely the multi-species coalescent [8]–to seek a phylogenetic network that matches the observed frequencies of (inferred) input trees [26, 37]. The latter approach is relevant to small evolutionary scales where the observed input trees may not match any of the trees "displayed" by the network. These methods are potentially more accurate than the combinatorial ones, but they require more complex computations, and thus they are generally much slower.

Finally[2], some work has been done toward the generalization of parsimony-based methods [13], to phylogenetic network reconstruction [24, 25, 12]. These kinds of methods, as in the case of tree reconstruction, are not model-based and thus they are less powerful than likelihood-based ones; for example, they are not statistically consistent [11], that is, the probability to obtain the correct tree does not converge to one as more and more data are analyzed. Still, these methods have an important part to play in network reconstruction. For instance, they can be used in combination with likelihood-based methods to compute the network with which to start the maximum-likelihood search, or to design fast local-search techniques. More importantly, they will be extremely useful in cases when likelihood-based approaches cannot scale up to the input data.

In this paper, we focus on the parsimony framework, providing several algorithmic results that lay the foundations of new parsimony-based methods for phylogenetic network reconstruction.

The main hypothesis of parsimony-based methods is that character changes are not frequent and thus the phylogenies that best explain the data are those requiring the fewest evolutionary changes. In a parsimony approach, each character can be analyzed independently from the others. Correspondingly, given a phylogenetic tree T, once the parsimony score $PS(c_j|T)$ is calculated for each character c_j, the parsimony score of the alignment A of length m is given by the (weighted) sum of the parsimony score of each character, i.e. $PS(A|T) = \sum_{j=1}^{m}(w_j \cdot PS(c_j|T))$

*To appear in *Systematic Biology*.

†Delft Institute of Applied Mathematics, Delft University of Technology, Postbus 5031,2600 GA Delft, The Netherlands, l.j.j.v.iersel@gmail.com, markelliotlloyd@gmail.com.

‡Institut des Sciences de l'Evolution, Université de Montpellier, CNRS, IRD, EPHE, Institut de Biologie Computationnelle (IBC), Place Eugène Bataillon, Montpellier, France, celine.scornavacca@umontpellier.fr.

[1]Phylogenetic networks can be used in two different ways : either to represent conflicting signals in the data – in which case we speak of *abstract* or *data-display* phylogenetic networks – or to represent putative evolutionary histories involving reticulate events – in which case the network is called *explicit*.

[2]For completeness, we should also mention the related approaches of reconstructing Ancestral Recombination Graphs (ARGs) [16] and Admixture graphs [33].

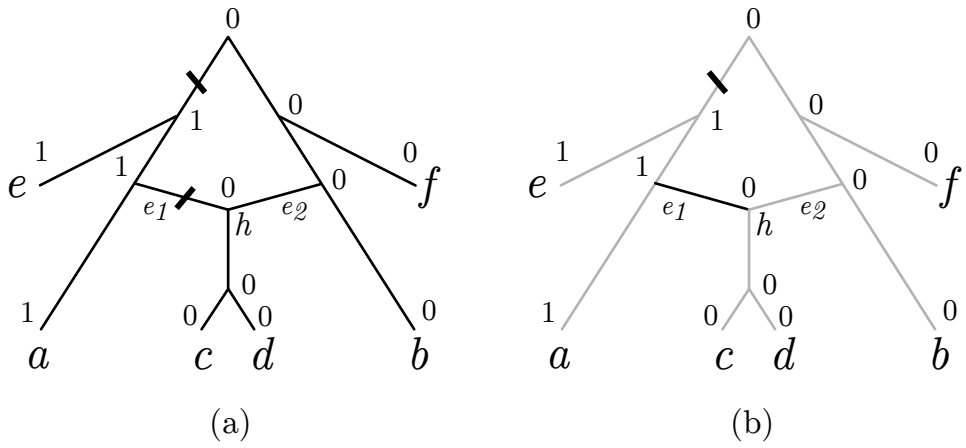

Figure 1: The hardwired parsimony score of the depicted network is 2 (c.f. (a)), while the softwired parsimony score is 1 (one of the possible trees displayed by the network is depicted in gray in (b)). Changes are depicted by thick lines crossing the branches.

where w_j is a user-defined weight for the character c_j, which can be used, for example, to model the confidence in the character. Assuming that the sequences of the internal nodes of the tree are known, one can easily determine the number of substitutions necessary to explain different states for c_j at the two extremities of a branch e. Denoting this value by $PS(c_j|e)$, $PS(c_j|T)$ is simply the sum of $PS(c_j|e)$ over all branches e of T, weighted by the substitution cost[3] (i.e. the cost of changing state for c_j). Since only terminal sequences are known, we need to find the combination of internal sequences that minimizes $PS(c_j|T)$. An $O(nm)$-time algorithm to calculate $PS(A|T)$ was proposed by Fitch [13]. However, finding the Maximum Parsimony (MP) tree, i.e. the tree T that gives the minimum value of $PS(A|T)$, is a difficult (i.e. NP-hard) problem [7].

When moving from phylogenetic trees to phylogenetic networks, we find two different definitions of maximum parsimony in the literature.

The first definition (the "hardwired" parsimony) is just the natural extension of the Fitch parsimony (recalled above) to networks: One aims at finding the assignment of states to internal nodes of the network such that the total number of branches that connect nodes in different states is minimized [25]. Note that this definition counts a state-change if a reticulation node has the same state as one of its ancestral nodes and the other ancestral node has a different state, see for example the reticulation h in Figure 1(a). Hence, hardwired parsimony counts more state-changes than necessary in a parsimony framework since h could very well have inherited its state from the ancestral node having the same state.

Put differently, although the evolution of the genome of h is best described by a network, the evolution of each "atomic" part still follows a tree. This is why in the second definition (the "softwired" parsimony) the parsimony score of a character on a network is defined as the score of the best tree *displayed by the network* [31, 23], see Figure 1(b) for an example and the paragraph *Displayed Trees* on page 5 for a formal definition.

The softwired parsimony is, in our opinion, more biologically relevant than the hardwired one, but it has a glitch: The definition of tree displayed by the network given in the literature forces the tree to take sides between the two parental species: the gene tree inside the network can only "utilize" one of the branches entering each hybrid node (e.g., either e_1 or e_2 in Figure 1(b)). However, this is not always the case, for example in the presence of allopolyploidy (see Fig. 2(a)), or of Incomplete Lineage Sorting (ILS) at the time of hybrid speciations, see Figure 2(b).

In this paper, we introduce a new variant of parsimony for phylogenetic networks, which improves on the previous definitions by permitting to model these processes while staying computationally feasible for reasonably tree-like networks. We define the *parental parsimony* score of a character on a network as the score of the best "parental" tree of the network. Intuitively speaking, a tree is a *parental tree* of a network if it can be drawn inside the network in such a way that the internal nodes of the tree correspond to branching nodes of the network. Importantly, different branches of the tree are allowed to travel through the same network branch, which is not allowed for displayed trees. The different tree-branches inside a network branch can, for example, model different copies of a gene present in the genome, or different variants of a gene present in a population. Consequently, parental trees are able to model situations as the ones depicted in Figure 2, which cannot be modeled using displayed trees (see the paragraph *Parental Trees* on page 5 for a formal definition).

While a network with one reticulation has only two displayed trees, it can have exponentially many parental trees, and computing the parental parsimony score is a completely different challenge than computing the softwired score. Fortunately, we shall show that, to compute the parental parsimony score it is not necessary to find all parental trees: parental parsimony can instead be elegantly characterized using "lineage functions". The main idea here is to not assign single states to the internal network-vertices – as is done for hardwired and softwired parsimony – but to assign sets, or even multisets (which may contain duplicates) of states. Given such an assignment, the parental parsimony score can be computed, see for an example Figure 3.

In Section "NP-hardness", we show that computing the parental parsimony score of a given network is compu-

[3]In the following, we will consider substitution costs to be all equal to 1 to facilitate notations, but all the results presented in this paper hold for any substitution cost scheme where substitution costs are positive.

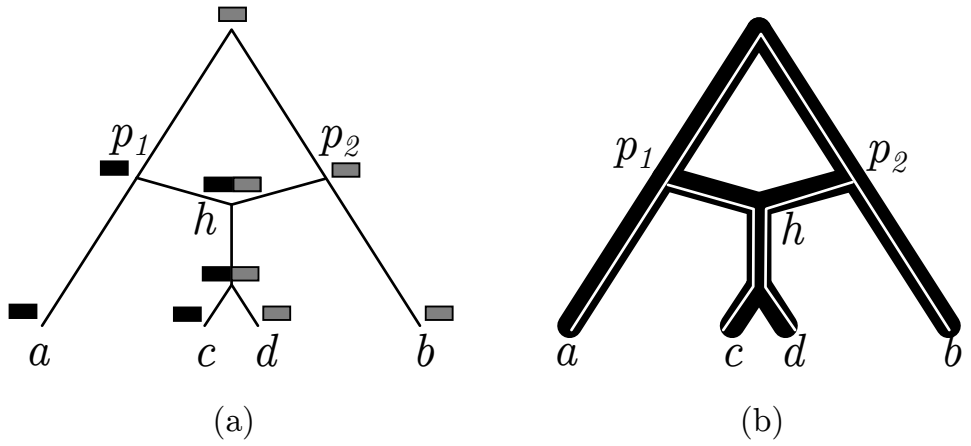

Figure 2: (a) Because of allopolyploid hybridization, the species h contains two copies of a gene, one inherited from the ancestral species p_1 and the other from the ancestral species p_2. (b) Because of ILS, the gene in a and the one in c coalesce in the ancestral species p_1, while the gene in b and the one in d coalesce in the ancestral species p_2. In both cases, the true gene tree is not displayed by the network.

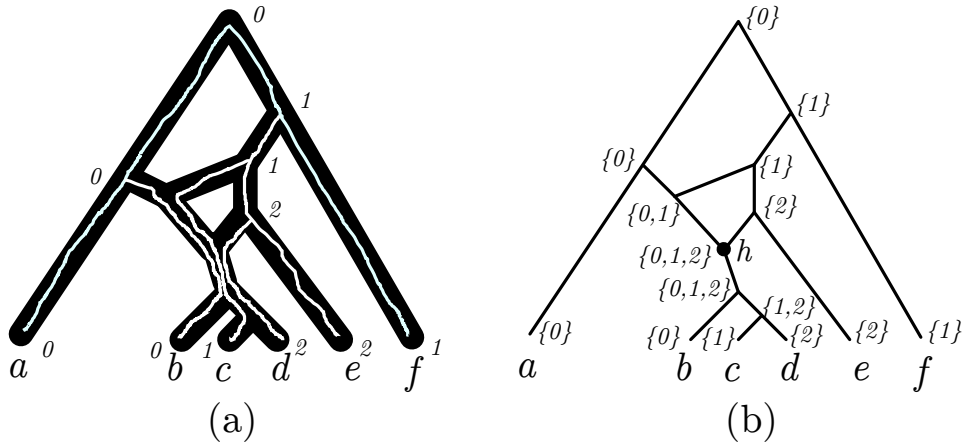

Figure 3: Example of a lineage function, which is represented by the labelling of the nodes of the network on the right. On the left you see the same network with a parental tree drawn inside it and a character state for each node of the tree. Intuitively, the label of each network node, described by the lineage function, contains all states that are present on lineages of the parental tree that pass through this network node. From this lineage function, the smallest parsimony score of a parental tree can be computed. For example, consider the reticulate node h highlighted by a dot. It has label $\{0, 1, 2\}$ and parents with labels $\{0, 1\}$ and $\{2\}$, so no state-changes are needed: states 0 and 1 can be inherited via lineages coming from the first parent, while state 2 can be inherited via a lineage coming from the other parent, as is the case in the parental tree drawn inside the network on the left.

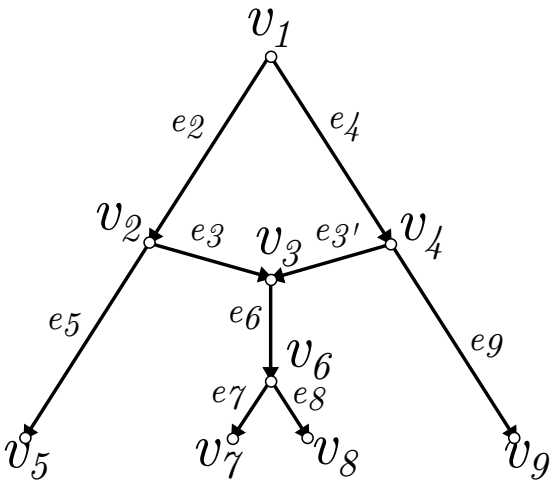

Figure 4: A directed acyclic graph N. We have that $E(N) = \{e_2, e_3, e_{3'}, e_4, e_5, e_6, e_7, e_8, e_9\}$ and $V(N) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$. Also, since v_1 has in-degree 0, it is the root of N, while $\{v_5, v_7, v_8, v_9\}$ all have out-degree 0 and they thus compose the set $L(N)$. Note that N is rooted and binary.

tationally hard (NP-hard). Moreover, this is the case already when the network is binary and extremely tree-like in the sense that no reticulation node is a descendant from another reticulation node, and each internal node has at least one non-reticulate descendant branch. Hence, computing the parental parsimony score is indeed extremely challenging.

Nevertheless, we have developed a dynamic programming algorithm for computing the parental parsimony score, which runs efficiently if the number of reticulations of the network is small. This algorithm is described in the section "Fixed-Parameter Tractability with respect to reticulation number".

Moreover, in the case that the total number of reticulations is large but the level of the network is small, the parental parsimony score can still be computed efficiently, where the "level" of a network, intuitively the number of reticulate events per reticulated component, is a measure for how tree-like a network is. In the section "Parameterizing by level" we describe how we have extended our dynamic programming algorithm to handle such situations. The developed algorithm computes the parental parsimony score of any network and character exactly. Its running time is only exponential in the level of the network and in the number of possible states, while the running time depends only linearly on the size of the network. Hence the algorithm scales very well to large data sets, as long as the network is reasonably tree-like.

One appealing property of our new model is that it is extremely flexible and is also especially convenient for developing mathematical proofs and algorithms.

In the section "Extensions", we show that it can be used in a much more general framework, for example allowing extant species to have multiple homologous genes, and to take gene duplication and gene loss into account. In addition, it can even be applied to the other variants of network parsimony, i.e. the hardwired and softwired ones. When ILS is present in the data (and when it is not only due to hybrid speciations), all trees can have a non-zero probability to explain the data and one should search the best tree among *all possible trees* on X (see for example [35, 4], where the authors adopt a similar approach to score species trees using biallelic genetic markers). Interestingly, our framework can also be used to model general ILS (see section "Modeling ILS").

2 Methods

In order to prove the results presented in the previous section, we need to introduce a theoretical framework for parsimony-based inference of explicit phylogenetic networks.

2.1 Preliminaries

2.1.1 Phylogenetic Networks

In graph theory, a directed acyclic graph (DAG) N is a graph where edges are directed and in which directed cycles are not permitted. We will write $N = (V, E)$ to denote the fact that N is a DAG with node set V and edge set E. We also write $V(N)$ to denote the nodes of N and $E(N)$ to denote its edges. The out-degree (in-degree) of a node is the number of edges starting (ending) in the node. The nodes of out-degree 0 are called the *leaves* of N and are also denoted by $L(N)$. The nodes of in-degree 0 are called the *roots* of N. We say N is *rooted* if N has only one root, and in such cases the root is denoted ρ_N. A DAG is *binary* if the total degree of each node is at most 3, with no node having in-degree 3 or out-degree 3. See Figure 4 for an example of several concepts introduced in this paragraph.

Let X be a finite set of taxa. A *(phylogenetic) network on X* is a connected DAG in which the nodes of out-degree 0 are bijectively labelled by the elements of X (bijectively means that each taxon in X labels exactly one

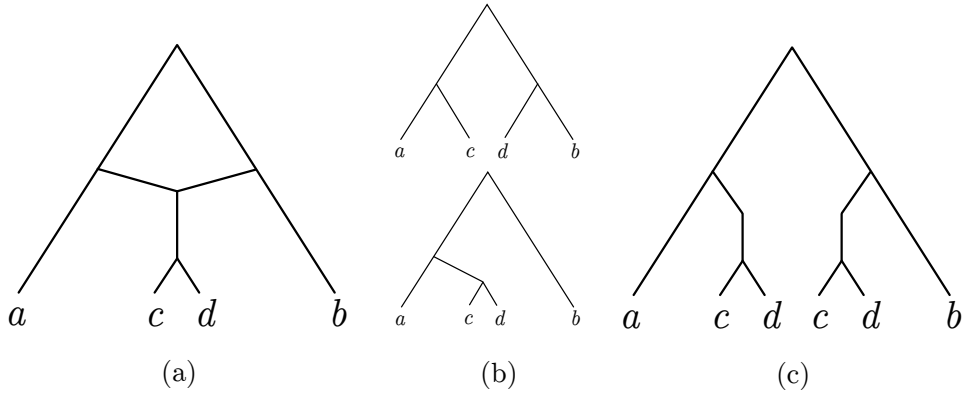

Figure 5: (a) A network N on $X = \{a, b, c, d\}$ and (b) two trees on X. The upper tree is a parental tree of N but it is not displayed by N, while the lower tree is displayed by N and (thus) also a parental tree of N. (c) The tree $U^*(N)$ used in the formal definition of parental trees.

node in $L(N)$). Throughout this paper, we will assume we are working with rooted binary networks[4] on a set X. An example of a rooted binary network on a set $X = \{a, b, c, d\}$ is given in Figure 5(a). Note that in this drawing, and in all other drawings of this paper except for Figure 4, edges are not drawn as directed to simplify the figures; still, they are to be considered as directed away from the root, which is the uppermost node in each drawing.

Given a binary rooted phylogenetic network $N = (V, E)$, the *reticulation nodes* of N are the nodes $u \in V$ with in-degree 2. We say that N is a *tree* if N contains no reticulation nodes. Given two nodes $u, v \in V(N)$, we say that u is a *parent* of v and v is a *child* of u if $E(N)$ contains a directed edge from u to v. For any $v \in V$, we let $par(v)$ denote the set of parents of v in N. We say that u is an *ancestor* of v and v is a *descendant* of u, if there is a path from u to v in N. We say u is a *trivial* ancestor of v (and v is a *trivial* descendant of u) if $u = v$. A network N is *tree-child* if every non-leaf node has at least one child that is not a reticulation node. The *reticulation depth* of N is the maximum number of reticulation nodes on any directed path in N.

2.1.2 Displayed Trees

A displayed tree of a network is a possible evolution of a character down a network, if the character's genealogy is only allowed to branch when the network branches, and no two lineages of the character's genealogy are allowed to evolve down the same network branch; see Figure 5(b-bottom) for an example. More formally, a tree T is *displayed* by a network N if T can be obtained from a subgraph of N by suppressing nodes of in-degree and out-degree 1. Given a rooted phylogenetic network N on X, let $\mathcal{T}(N)$ denote the set of all phylogenetic trees on X that are displayed by N.

2.1.3 Parental Trees

Informally, a phylogenetic tree T on X is a *parental tree* of N if the nodes of T can be mapped to nodes of N, and the edges in T mapped to directed paths of positive length in N joining the corresponding nodes, in such a way that the leaf of T labelled with x is mapped to the leaf of N labelled x, for each $x \in X$. Note that while every tree displayed by N is parental, not every parental tree is displayed by N, as the paths in N corresponding to different edges in T may overlap, see Figures 5(b) and 6(a). In other words, a parental tree is one that describes a possible evolution of a character down a network, if the character's genealogy is still only allowed to branch whenever the network branches, but different lineages of the character's genealogy are allowed to evolve down the same network branch.

In order to define parental trees more formally, we first define the tree $U^*(N)$ derived from N. Here $U^*(N)$ is a *multi-labelled tree* or *MUL tree* [20], i.e. one in which a taxon in X can label more than one leaf. Our definitions in this subsection are based on those in [19], where parental trees are called *weakly displayed trees*.

Let $U^*(N)$ be the tree whose nodes are the directed paths in N starting at ρ_N. For each pair of paths π, π' in N, there is an edge in $U^*(N)$ from π to π' if and only if $\pi' = \pi e$ for some edge e in N. In addition, each node in $U^*(N)$ corresponding to a path in N that starts at ρ_N and ends at $x \in X$ is labelled by x. See Figure 5(c) for an example.

We say that a phylogenetic tree T on X is a *parental tree* of N if it is displayed by $U^*(N)$. Let $\mathcal{PT}(N)$ denote the set of all phylogenetic trees on X that are parental trees of N. We can easily verify that a tree T is displayed by $U^*(N)$ by labelling each node in T with the corresponding path in N. Still, since the size of $U^*(N)$ can be exponential in the size of N, in the following we do not attempt to construct $U^*(N)$ explicitly.

The definition of parental trees can be easily extended to cases with multiple individuals per species, as done for example in [43]: it suffices to allow parental trees to be multi-labelled trees. An example is shown in Figure 6. See the section "Extensions" for a more formal discussion. Note that [44] gave a slightly different definition of parental

[4]Some definitions of a network additionally require that there are no nodes of in-degree and out-degree 1. We do not make this stipulation here, because it will occasionally be useful to consider subgraphs of a network, which can end up having such nodes, and we would still like to categorize these subgraphs as networks. This does not alter the considered problems.

Figure 6: On the left, a network N on $X = \{a, b, c, d\}$ with a parental tree (in white) drawn inside it. In this example, we have two individuals for the species b and one individual for all other species. On the right, the MUL tree $U^*(N)$ with the same parental tree drawn in it.

trees – taking into account branch lengths and inheritance probabilities – which is well suited for probabilistic models.

2.2 Parsimony scores

Throughout this paper, we will make the common hypothesis of site independence. Under this hypothesis, the parsimony score of an alignment is the (weighted) sum of the parsimony scores of the characters composing the alignment. In the following, we will thus consider a single character at a time.

Given a set U and $p \in \mathbb{N}$, a *p-state character* α on U is a function from U to $\{1, \ldots, p\}$. If $p = 2$, we say that α is *binary*.

2.2.1 Parsimony on trees

Let α be a p-state character on X and T a rooted phylogenetic tree on X. Then a p-state character τ on $V(T)$ is an *extension* of α if $\tau(x) = \alpha(x)$ for all $x \in X$. Given a p-state character τ on $V(T)$ and an edge $uv \in E(T)$, the *change* $c_\tau(uv)$ on uv w.r.t. τ is defined to be 0 if $\tau(u) = \tau(v)$, and 1 otherwise. Given a tree T on X and a p-state character α on X, the *parsimony score* of T and α can be defined [13] as

$$PS(T, \alpha) = \min_{\tau} \sum_{uv \in E(T)} c_\tau(uv),$$

where the minimum is taken over all extensions τ of α to $V(T)$ (we call an extension attaining the minimum, an *optimal* one).

2.2.2 Hardwired and softwired parsimony

Given a network N on X and a p-state character α on X, the *hardwired parsimony score* of N and α can be defined [25] exactly as the parsimony score of a tree:

$$PS_{hw}(N, \alpha) = \min_{\tau} \sum_{uv \in E(N)} c_\tau(uv),$$

where the minimum is taken over all extensions τ of α to $V(N)$. We note here for the record that if N is derived from a network N' by suppressing nodes of in-degree and out-degree 1, then $PS_{hw}(N, \alpha) = PS_{hw}(N', \alpha)$.

The *softwired parsimony score* [24] of N and α is the minimum parsimony score of any tree on X displayed by N, that is

$$PS_{sw}(N, \alpha) = \min_{T \in \mathcal{T}(N)} \min_{\tau} \sum_{uv \in E(T)} c_\tau(uv).$$

2.2.3 Parental parsimony

The parental parsimony score of a network and character is the minimum number of state-changes necessary in any evolution of the character down the network, if the character's genealogy is only allowed to branch when the network branches, while multiple lineages of the character's genealogy are allowed to evolve down the same network branch. This is the case, for example, in the presence of allopolyploidy or of ILS at the time of hybrid speciations. More formally, the *parental parsimony score* of N and α is the minimum parsimony score of any parental tree of N, that is

$$PS_{pt}(N, \alpha) = \min_{T \in \mathcal{PT}(N)} \min_{\tau} \sum_{uv \in E(T)} c_\tau(uv).$$

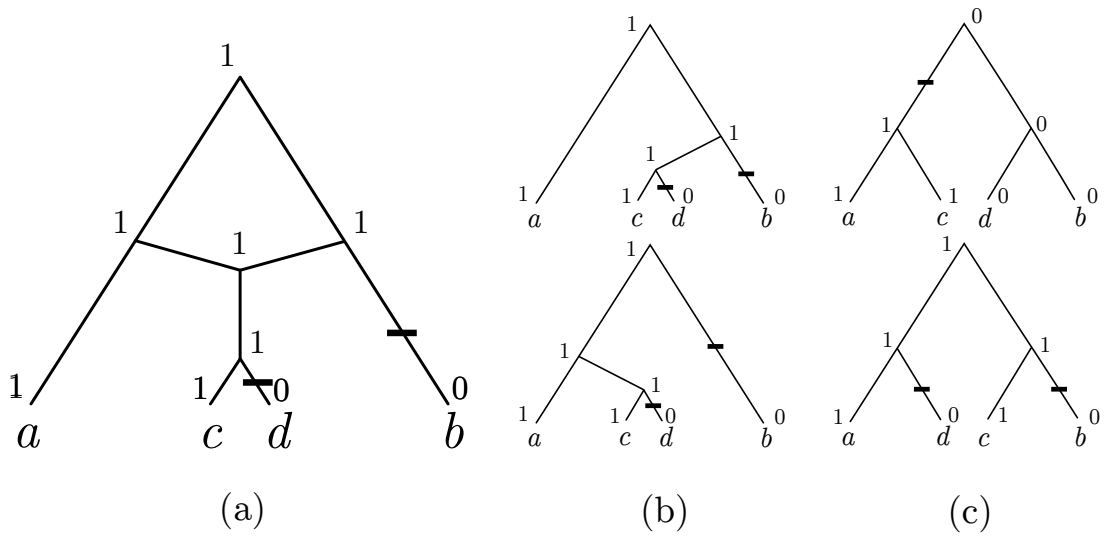

Figure 7: The different parsimony scores of the network N given in Figure 5 are 2, 2 and 1 for the hardwired, softwired and parental parsimony, respectively: (a) An optimal extension for the hardwired PS; (b) Optimal extensions for the two trees displayed by N for the softwired PS; (c) Optimal extensions for the two other parental trees of N (the ones that are not depicted in (b)) for the parental PS. Changes are depicted by a thick line.

An example of the different parsimony scores can be found in Figure 7. Note that parental parsimony is a better model than hardwired parsimony because the latter model counts more state-changes than necessary at the reticulate events. Note also that parental parsimony can model allopolyploidy and ILS at the time of hybrid speciations better than softwired parsimony.

We are now ready to formally introduce the problem tackled in this paper:

Parental Parsimony Problem (PPP)
Input: A network N on X; a p-state character α on X.
Output: $PS_{pt}(N, \alpha)$.

In the main part of this paper, we will focus on solving PPP. However, the techniques used in this paper can also be used to solve a number of other parsimony-related problems. Furthermore, our results generalize to the case where each leaf may be assigned a set or even a multiset of states. This is of interest, for example, when we have several individuals per each species and the individuals do not all agree on the value to associate to the character under study. We discuss the extension of our results in more detail in the sections "Extensions" and "Modeling ILS".

2.3 Characterising Parental Parsimony with Lineage Functions

In this section, we introduce the notion of a lineage function[5], in which every node in a network is mapped to a set of states. Informally, this is a way of tracking how many branches of a parental tree travel through each node of the network, and what states are assigned to each of those branches. A lineage function does not fully characterize a parental tree, but it does characterize the parsimony score of the tree, and it is easy to find a tree with minimal parsimony score corresponding to a given lineage function. Moreover, lineage functions provide a useful tool for dynamic programming algorithms to solve PPP.

Given a set U, we denote by $\mathcal{P}(U)$ the set of all subsets of U. Then we define a lineage function as follows.

Definition 1. *Given a rooted phylogenetic network N on X and an integer p, a* (p-width) lineage function on N *is a function $f : V(N) \to \mathcal{P}(\{1, \ldots, p\})$.*
We say that f is rooted *if $|f(\rho_N)| = 1$.*
Given a p-state character α on X, we say that f is α-consistent *if $f(x) = \{\alpha(x)\}$ for all $x \in X$.*

Consider as an example the parental tree depicted Figure 3(b). In this example, the network node h has three branches of the parental tree passing through it. These branches have states $0, 1$ and 2 respectively. Thus, the corresponding lineage function (in Fig. 3(a)) assigns this node the value $\{0, 1, 2\}$.

Given a lineage function f, we wish to define the *weight* of f, that gives the minimum parsimony score of a parental tree corresponding to f. The key observation behind our methods is that this score can be determined by comparing, for each $v \in V(N)$, the set $f(v)$ with the sets $f(u)$ of each parent u of v. To that end, we first define the weight of a node v with respect to f:

Definition 2. *Let f be a lineage function on N.*

[5]We note that lineage functions have a similar flavor as "ancestral configurations", which are used for reconciling gene trees to species networks [41].

Given a node v of N, the weight *of v with respect to f, denoted $w_f(v)$, is defined as*

$$w_f(v) = \begin{cases} 0 \text{ if } v = \rho_N \\ \infty \text{ if } v \neq \rho_N \text{ and } |f(v)| > \sum_{u \in par(v)} |f(u)| \\ |f(v) \setminus (\bigcup_{u \in par(v)} f(u))| \text{ otherwise} \end{cases}$$

The total weight *of f is defined as*

$$w(f) = \sum_{v \in V(N)} w_f(v)$$

Consider again the lineage function depicted in Figure 3. The network node with assigned set $\{0, 1\}$ has parents with assigned sets $\{0\}$ and $\{1\}$ respectively, and therefore the cost on this node is 0. This reflects the fact that each branch of the parental tree passing through this node comes from a branch of one of the parents having a matching state. On the other hand, the node with assigned set $\{2\}$ – and whose only parent is assigned $\{1\}$ – has a cost of 1, as the only branch passing through this node had to change state from 1 to 2.

A weight of value ∞ denotes that there is no parental tree corresponding to the given lineage function. This only happens when there are not enough branches traveling through the parent node(s) to cover the number of branches required in a child node. For example, if u is the only parent of v in a network, there can be no parental tree with one branch going through u and two branches passing through v.

We remind that, although we will focus on solving PPP, lineage functions can be used to calculate optimum scores for a number of measures, including softwired and hardwired parsimony scores, gene loss and duplication minimization, and arbitrary combinations of these scores. We discuss this further in the section "Extensions".

We summarize the relation between lineage functions and parental parsimony in the following theorem:

Theorem 1. *For any binary network N on X and p-state character α on X, $PS_{pt}(N, \alpha) = \min\{w(f) : f$ is a rooted α-consistent lineage function on $N\}$.*

The proof of this result and all other missing proofs can be found in the Appendix.

Thanks to this result, in the remainder of this paper we may view PPP as the problem of finding a minimum weight rooted α-consistent lineage function.

We now prove another property of lineage functions, namely that we may assume that any lineage function assigns each node to a non-empty subset of $\{1, \ldots, p\}$.

Lemma 1. *Let N be a phylogenetic network on X and α a p-state character on X. Then, for any rooted lineage function f on N, there exists a rooted lineage function f' on N such that $w(f') \leq w(f)$, and $f(v) \neq \emptyset$ for all $v \in V(N)$.*

Proof. Suppose w.l.o.g. that $w(f) < \infty$. Now consider a highest $v \in V$ such that $f(v) = \emptyset$. As f is rooted, v has a parent u in N and $f(u) \neq \emptyset$. Then let f' be the lineage function that is identical to f, except that $f'(v) = \{i\}$ for some arbitrary $i \in f(u)$.

Observe that $w_{f'}(v') \leq w_f(v')$ for any child v' of v. For v itself, as $f'(v) \subseteq f'(u)$ we have that $w_{f'}(v) = 0 \leq w_f(v)$. For each other $u' \in V(N)$, we have $w_{f'}(u') = w_f(u')$. Thus $w(f') \leq w(f)$. By repeating this process exhaustively, we end up with a lineage function f' such that $w(f') \leq w(f)$ and $f(v) \neq \emptyset$ for all $v \in V(N)$, as required. □

We also prove a bound on the size of $f(v)$ in terms of the reticulation ancestors of v, for any lineage function f on N and $v \in V(N)$.

Lemma 2. *Let $v \in V(N)$ be an arbitrary node of N and k' the number of reticulation nodes that are ancestors of v in N (including v itself). Then $|f(v)| \leq 2^{k'}$ for any rooted lineage function on N with $w(f) < \infty$.*

In particular, $|f(v)| \leq 2^k$ for all $v \in V(N)$, where k is the number of reticulation nodes in N.

Proof. We prove the claim by induction on the depth of v (that is, the length of a longest path from ρ_N to v). If $v = \rho_N$, then $k' = 0$, and as f is rooted, $|f(v)| = 1 = 2^{k'}$.

So now assume that $v \neq \rho_N$. If v is not a reticulation node, then v has a single parent u. As $w(f) < \infty$, $|f(v)| \leq |f(u)|$. By the inductive hypothesis (and the fact that every reticulation ancestor of u is an ancestor of v), $|f(u)| \leq 2^{k'}$. Thus, $|f(v)| \leq 2^{k'}$.

If v is a reticulation node, then v has two parents u and u', each of which has fewer reticulation ancestors than u. Thus, as $w(f) < \infty$ and by the inductive hypothesis, $|f(v)| \leq |f(u)| + |f(u')| \leq 2 \cdot 2^{k'-1} = 2^{k'}$. □

To summarize, in this section we have shown that to compute the parental parsimony score of a network and character, we do not need to determine how exactly the character evolved down the network. It suffices to determine the number of lineages of the character's genealogy corresponding to each network node, and the state of each such lineage. Moreover, we have shown that, if a network node has a limited number of reticulate ancestors, there can only be a limited number of lineages of the character genealogy that evolve down this network node. In addition, there always exists an optimal assignment where each network node corresponds to at least one lineage of the character genealogy.

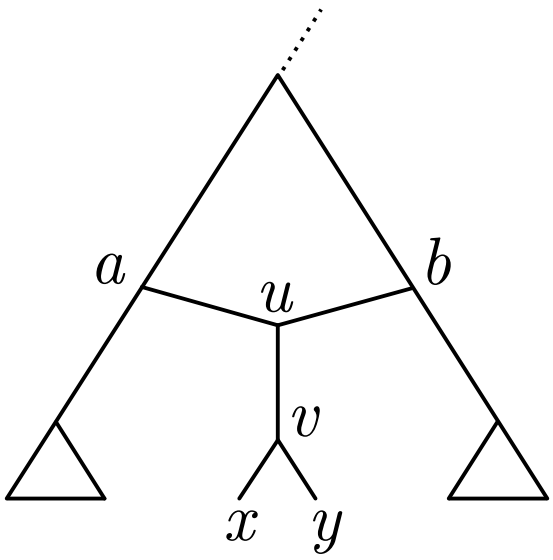

Figure 8: The gadget used in the NP-hardness proof.

2.4 NP-hardness

In this section, we show that PPP is NP-hard, even when the input character is binary and the input network is tree-child and has reticulation depth at most 1.

Lemma 1 implies that when $p = 2$, we may assume a lineage function assigns each node to one of three possible sets: $\{1\}$, $\{2\}$ or $\{1, 2\}$.

Our hardness reduction is based on the following observation: suppose that u is a reticulation node with parents a and b, that u has a single child v, and that v has leaf children x and y, with $\alpha(x) = \{1\}, \alpha(y) = \{2\}$. Suppose furthermore that a and b are non-reticulation nodes with no reticulation ancestors, and therefore $|f(a)| = |f(b)| = 1$ for any rooted lineage function f with $w(f) < \infty$ (by Lemma 2). If $f(a) \neq f(b)$, then we may set $f(u) = f(v) = \{1, 2\}$ and ensure that $w_f(u) = w_f(v) = w_f(x) = w_f(y) = 0$. On the other hand if $f(a) = f(b) = \{1\}$, then for some $v' \in \{u, v, y\}$ we must have that $w_f(u) \geq 1$, and similarly when $f(a) = f(b) = \{2\}$.

Thus, the subgraph on a, b, u, v, x, y can be viewed as a gadget that imposes a cost of 1 for setting $f(a) = f(b)$. Such gadgets (one is depicted in Fig. 8) can be used to create a reduction to PPP from the NP-hard problem MAX-CUT [2].

Theorem 2. *PPP is NP-hard, even when the character α is binary, and the network N is tree-child and has reticulation depth at most* 1*.*

To summarize this section, we have shown that computing the parental parsimony score is computationally a hard problem even for extremely tree-like networks. In more detail, this is even the case when no reticulate node is an ancestor of another reticulate node, and no network node has two reticulate descendant branches.

2.5 Fixed-Parameter Tractability with respect to reticulation number.

In this section, we give an algorithm to solve the PPP that is exponential only w.r.t. the reticulation number of the network k and the number of possible states p, and polynomial w.r.t. the number of leaves of the network.[6]

In light of Theorem 1, we will view PPP as the problem of finding a minimum cost (rooted, α-consistent) lineage function on N. In the remainder of this paper, we let $\mathcal{Y}$ denote the set $\mathcal{P}(\{1, \ldots, p\}) \setminus \{\emptyset\}$, i.e. $\mathcal{Y}$ is the set of all possible sets that may be assigned to a node by a lineage function. (Note that, when we come to generalize our results in the next sections, $\mathcal{Y}$ may be replaced with another set.) We let μ denote $|\mathcal{Y}|$. Thus when $p = 2$, we have $\mathcal{Y} = \{\{1\}, \{2\}, \{1, 2\}\}$ and $\mu = 3$.

Let P be a set of nodes in N constructed by taking one parent of each reticulation node in N. Then let F be the network derived from N by deleting all out-edges of nodes in P. It can be seen that (a) F is a forest, and (b) the leaves of the F are exactly $X \cup P$.

Our next step is to guess the values assigned by the lineage function to P. We denote this guess by a function $f' : P \to \mathcal{Y}$, and we repeat the remaining steps for each possible f', keeping the option that gives us the minimum cost lineage function. The number of possible functions f' is $|P|^{|\mathcal{Y}|} = O(k^{2^p-1})$. This is where the exponential part of our algorithm occurs.

Given P and f', we now have a forest F where the assignment of a lineage function is fixed on each of the leaves. That is, we know that for any lineage function f on $V(N)$, we must have that $f(x) = \{\alpha(x)\}$ for each $x \in X$, and $f(v) = f'(v)$ for each $v \in P$. We now calculate the optimum assignment on the remaining nodes, using standard dynamic programming techniques.

For each internal node u of a tree T in F (starting with the lowest nodes), and each $S \in \mathcal{Y}$, we calculate and store a value $\chi(u, S)$ representing the total cost of an optimal assignment to the descendants of u in T, under the

[6]We thus have a *fixed-parameter algorithm* with respect to k and p. We refer to [10, 14, 15, 32] for an introduction to fixed parameter tractability.

assumption that v is assigned the value S. If we have already calculated the values $\chi(v, S')$ for each child v of u and $S' \in \mathcal{Y}$, then it is easy to calculate the value of $\chi(u, S)$. For each child v, we simply choose the assignment S' to v that minimizes the value of $\chi(v, S')$ together with the costs imposed on v, under the assumption that u is assigned S and v is assigned S'. Doing this for each child of u gives us the optimum cost with respect to the choice (u, S). Calculating the costs on v can be complicated by the fact that v may have parents other than u - namely, a parent u' in P whose out-edge was deleted in the construction of F. However, as the assignment on u' is fixed in the "guessing phase" of the algorithm, we can account for this parent without increasing the complexity of the algorithm.

Once the values $\chi(u, S)$ have been calculated for each root u of a tree in F and each $S \in \mathcal{Y}$, we can find the optimal cost of a lineage function on $V(N)$ by combining the optimal lineage functions for each tree. In Algorithm 1, we give a pseudocode for the algorithm outlined in this section.[7] The algorithm calls as a sub-method the algorithm $\textsc{Cost}(v, S, S', f')$. This algorithm returns the value $w_f(v)$ achieved by any lineage function f that extends f' and assigns value S' to v, and value S to the only parent u of v not in P. If every parent of v is in P, we replace S with the placeholder value $\emptyset$, and $\textsc{Cost}(v, \emptyset, S', f')$ returns the value $w_f(v)$ achieved by any lineage function f that extends f' and assigns value S' to v. Thanks to Algorithm 1, we have the following:

Theorem 3. *PPP is fixed-parameter tractable with respect to k and p.*

We work through an example application of the algorithm in the Appendix.

Data: Binary network N on X; character $\alpha : X \to \{1, \ldots, p\}$
Result: $PS_{pt}(N, \alpha)$
Let $\mathcal{Y} = \mathcal{P}(\{1, \ldots, p\}) \setminus \{\emptyset\}$;
Let P be a set containing one parent of each reticulation node;
Let F be the forest obtained from N by deleting the out-edges of P;
Let $T_1, \ldots, T_r$ be trees of F, with roots $\rho_1 = \rho_N, \ldots, \rho_r$;
for *each function* $f' : P \to \mathcal{Y}$ *with* $|f'(\rho_N)| = 1$ **do**
  for $i = 1, \ldots, r$ **do**
    for *each vertex* u *of* $V(T_i)$ *(in reverse topological ordering) and each* $S \in \mathcal{Y}$ **do**
      if u is a leaf in X,

$$H[u, S] := \begin{cases} 0 & \text{if } S = \{\alpha(u)\} \\ \infty & \text{otherwise} \end{cases}$$

      if $u \in P$,

$$H[u, S] := \begin{cases} 0 & \text{if } S = f'(u) \\ \infty & \text{otherwise} \end{cases}$$

      if u has *one* child v in T_i, set $H[u, S]$ to

$$\min_{S' \in \mathcal{Y}} (H[v, S'] + \textsc{Cost}(v, S, S', f'))$$

      if u has *two* children v_1, v_2 in T_i, set $H[u, S]$ to

$$\min_{S_1 \in \mathcal{Y}} (H[v_1, S_1] + \textsc{Cost}(v_1, S, S_1, f')) + \min_{S_2 \in \mathcal{Y}} (H[v_2, S_2] + \textsc{Cost}(v_2, S, S_2, f'))$$

    end
  end

$$opt_{f'} := \min_{j \in \{1, \ldots, p\}} H[\rho_1, \{j\}] + \sum_{i=2}^{r} \min_{S \in \mathcal{Y}} \left[\textsc{Cost}(\rho_i, \emptyset, S, f') + H[\rho_i, S]\right]$$

end
return the smallest value of $opt_{f'}$ over all f'

Algorithm 1: Computing the parental parsimony score when the parameter is the number of reticulation nodes.

Data: Node v in N with at most one parent u not in P; set S in $\mathcal{Y} \cup \emptyset$; set S' in $\mathcal{Y}$; assignment $f' : P \to \mathcal{Y}$
Result: Cost on edges entering v for any lineage function f that extends f' and assigns $f(u) = S$ (if u exists) and $f(v) = S'$.
if $v = \rho_N$, **return** 0;
Let P_v contain the parents of v that are in P;
if $|S'| > |S| + \sum_{u \in P_v} |f'(u)|$ **then**
  return ∞
else
  return $|S' \setminus (S \cup \bigcup_{u \in P_v} f'(u))|$
end

Algorithm 2: Algorithm Cost, a subroutine of Algorithm 1 for computing the local cost at a node v.

To recapitulate, in this section we have shown how the parental parsimony score of a phylogenetic network and a given character can be computed. We have proven that the algorithm scales well for large numbers of taxa, as long as the number of reticulate nodes in the network, and the number of possible states per character, are small.

[7] The pseudocode contains some simplifications compared to the method described in the appendix, as that description has a few complications to allow for the extension to network level.

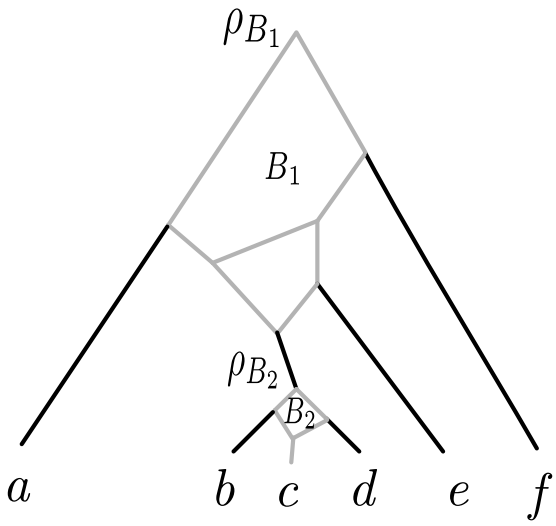

Figure 9: The network in the figure contains two non-trivial blobs B_1 and B_2, depicted in gray.

2.6 Parameterizing by level

We now show how to extend our fixed-parameter tractability result to the *level* of a network. Our approach in this section builds on the dynamic programming technique used in the previous section. Given a phylogenetic network N, a *blob* of N is a maximal subgraph of N for which the underlying undirected graph is biconnected. Call a blob a *trivial* blob if it consists of two nodes joined by a single edge. It is easy to show that the blobs of N partition the edges of N, that every node of N is in at most 1 non-trivial blob (using the fact that N is binary), and that every blob in N has exactly one root. Given a blob B in N, let r_B denote the number of reticulation nodes in B. The *level* l of N is the maximum value of r_B over all blobs B in N. For example, the network in Figure 9 contains two blobs and its level is 2 since $r_{B_1} = 2$ and $r_{B_2} = 1$.

Consider a lowest blob B in N, with ρ_B the root of B. Then B has reticulation number at most l. Therefore using the same approach as the previous section, we can calculate for each $S \in \mathcal{Y}$ the minimum cost of an α-consistent lineage function that assigns ρ_B the value S. We apply this idea recursively. For each blob B of N (starting with the lowest), we use our algorithm to calculate the minimum cost $\chi(\rho_B, S)$ of an α-consistent lineage function f on B and its descendants, for which $f_(\rho_B) = S$. Here we treat the roots of the 'child blobs' B' of B as leaves of B. The only difference from our previous algorithm is that, rather than being assigned a particular value by α, these leaves can be assigned different values, and impose a cost of $\chi(\rho_{B'}, S')$ when assigned the value S'. This difference only requires a small change to our previous algorithm, and implies the following:

Theorem 4. *PPP is fixed-parameter tractable with respect to l and p.*

Less formally, in this section we have shown how the parental parsimony score of a phylogenetic network and a given character can be computed even when the numbers of taxa and reticulate nodes in the network are large, as long as the reticulate nodes are spread out over different reticulated components containing only a small number of reticulate nodes each, and the characters have a small number of possible states.

3 Extensions

In the previous sections, our proofs did not depend on the exact definition of a lineage function, a character, or the weight of a lineage function. In fact, the proofs depended only on the following properties:

- A *lineage function* is a function from the nodes of a network N to a set $\mathcal{Y}$ of size μ;
- A *character* is a function from the leaves of N to a subset of $\mathcal{Y}$, and a lineage function is *α-consistent* if it extends α;
- The *weight* $w(f)$ of a lineage function f on a network N is defined to be $\sum_{v \in V(N)} w_f(v)$, where $w_f(\rho_N) = 0$, and for any $v \in V(N) \setminus \{\rho_N\}$ the value $w_f(v)$ depends only on the values assigned by f to u and each of its parents;
- A lineage function f on N is *rooted* if $f(\rho_n)$ is within some specified subset of $\mathcal{Y}$.

If the above properties hold, then the previous result (c.f. proof of Theorem 4 in appendix) implies the following:

Theorem 5. *For any rooted binary phylogenetic network N with level l, and for character α on N, an α-consistent rooted lineage function on N of minimum weight can be found in time $O(\mu^{l+3}|V(N)|)$.*

This implies that, for example for binary characters ($\mu = 3$), we should certainly be able to deal with networks with $l = 4$, i.e. at most 4 reticulations per blob.

We can therefore extend the results of our paper to other measures besides parental parsimony, in cases where the function to optimize can be expressed as a minimum weight lineage function with respect to some weight measure. In the remainder of this section, we sketch how to do this for some biologically relevant examples.

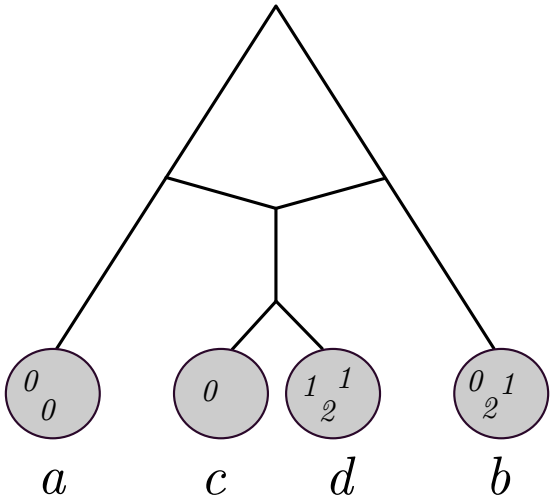

Figure 10: An example of application of the extension to multisets of states. For the four species in the network, we have the following samples: two individuals for species a, both with state 0, one individual for species c with state 0, three individuals for species d with state 1, 1 and 2 and, finally, three individuals for species b, with states 0, 1 and 2.

3.1 Extending characters to sets of states

For some applications, we may want to sample multiple individuals from the same species, and potentially obtain different states for each of them (see for example [43]). In such cases, we would like to model parental trees as MUL trees, in which the number of leaves labelled with taxon $x \in X$ is equal to the number of different states recorded for individuals in x. We then seek a character on the parental tree in which the set of states assigned to leaves labelled with x is equal to the set of states observed in individuals in x.

Our methods extend naturally to this scenario - we simply allow the input character α to be a function $L(N) \to \mathcal{P}(\{1,\ldots,p\})$ rather than $L(N) \to \{1,\ldots,p\}$. See for example species b in Figure 10. That is, we now assume that each leaf may be assigned a set of states rather than a single state. We believe this natural extension has other good biological motivations: in addition to the application to SNPs in a population given above, the different states can also model different characters for homologous copies of a gene present in the genome. However, this easy extension here only works when we have *sets* of states, i.e., when the recorded states of a taxon are all different. In the next subsection, we will show how to get rid of this restriction.

3.2 Extending characters to multisets of states

It makes also sense to consider the problem where a character assigns each leaf not just to a set of states, but a multiset of states, see for example species a and d in Figure 10. This requires a bit more work than extending characters to sets, but we may proceed as follows:

- The set $\mathcal{Y}$ now becomes a set of multisets over $\{1,\ldots,p\}$. In order to bound $|\mathcal{Y}|$, it can be shown that each state is assigned to a single node no more times than the maximum number of times it is assigned to a single leaf. Thus, we get a bound on $|\mathcal{Y}|$ in terms of the sizes of the multisets assigned by α.
- Given a lineage function f and non-root node v, the weight $w_f(v)$ can still be written as $|f(v)\backslash(\bigcup_{u\in par(v)} f(u))|$, except that the union now has to be taken as a multiset union - i.e. if a single state appears once in $f(u_1)$ and twice in $f(u_2)$ for the two parents u_1, u_2 of v, then it appears three times in $\bigcup_{u\in par(v)} f(u)$, and if it furthermore appears five times in $f(v)$, it would appear two times in $f(v) \setminus (\bigcup_{u\in par(v)} f(u))$.

Biologically, this means that we can handle situations where multiple individuals are sampled per species, or homologous copies of a gene are present in a genome.

3.3 Softwired Parsimony

We can represent softwired parsimony with the following adjustments:

- The set $\mathcal{Y}$ is the set of all sets containing a single state, together with the empty set.
- The weight $w_f(v)$ is defined as for parental parsimony: it is ∞ if $|f(v)|$ is bigger than $\sum_{u\in par(v)} |f(u)|$, and $|f(v) \setminus (\bigcup_{u\in par(v)} f(u))|$ otherwise. Thus, any node v is taken to be part of the chosen displayed tree if $|f(v)| = 1$. Also, if v is not the root of the network, $w_f(v) = \infty$ if there is no $u \in par(v)$ with $|f(u)| = 1$ (since any node in the displayed tree except for ρ_N must have a parent); otherwise $w_f(v) = 0$ if there is an available parent node assigned the same state (which is then taken as the parent of v in the displayed tree tree), and $w_f(v) = 1$ if there is not.

This means that, for data where no ILS or allopolyploidy is present/expected, a parental parsimony-based method can easily be set to use the softwired parsimony score instead of the parental parsimony score. In the next subsection, we show that even the hardwired parsimony score can be computed using our model. However, this is

less interesting biologically since hardwired parsimony does not seem to be a reasonable model for any data that is not purely tree-like.

3.4 Hardwired Parsimony

We can represent hardwired parsimony with the following adjustments:

- The set $\mathcal{Y}$ is the set of all sets containing a single state. (Note we do not allow the empty set – hardwired parsimony requires that every node is assigned a state).
- The weight $w_f(v)$ is defined as follows: $w_f(v) = |u \in par(v) : f(u) \neq f(v)|$.

3.5 Allowing for gene duplication

Returning to parental parsimony: Our current definition of w_f sets $w_f(v) = \infty$ if f assigns v a larger set than the combined sizes of sets assigned to its parents. This reflects the requirement that each split in a parental tree must correspond to a split in the network - when the parental tree splits, its out-edges must go down different edges of the network, and thus a node u cannot contribute more than $f(u)$ branches towards one of its children v. We can relax this requirement, and allow for duplication events in parental trees - thus, a parental tree can split with both child edges continuing down the same network edge, and a given network node can end up containing more 'branches' than all of its parents combined.

To express this, we simply drop the requirement that $w_f(v) = \infty$ if $|f(v)| > \sum_{u \in par(v)} |f(u)|$, and just have $|f(v) \setminus (\bigcup_{u \in par(v)} f(u))|$ for all non-root nodes v.

The bound on $\mathcal{Y}$ remains bounded by a function of the size of the (multi)sets assigned to leaves by α.

If we are interested in minimizing duplications instead of parsimony score, we can instead set $w_f(v) = \max\{0, |f(v)| - \sum_{u \in par(v)} |f(u)|\}$.

If we are interested in minimizing both parsimony score and duplications, we can simply add the two version of w_f together (or any weighted combination of them, if one is considered more important than the other).

3.6 Minimizing gene loss

We can express the number of gene losses from a parsimony tree by setting $w_f(v) = \max\{0, (\sum_{u \in par(v)} |f(u)|) - |f(v)|\}$ - this is the number of branches that passed through parents of v and were not passed on to v. In the case of an LGT network, it may be the case that some edge uv is a secondary arc [5], in which not all genes are expected to be transferred, and therefore we do not care about gene losses on this arc. In such cases we do not count u towards the set $par(v)$ for the purposes of $w_f(v)$.

As with gene duplications, we can combine this score with other scores like parsimony and gene duplication, if we are interested in multiple measures.

To recapitulate the last two subsections, we are able to deal with gene duplication events and we are able to take such events, as well as the number of gene losses, into account in the computed score.

4 Modeling ILS

We now show how lineage functions may be applied in the context of Incomplete Lineage Sorting (ILS), which is one of the processes by which a gene tree and a species tree may come to differ. The MDC (Minimize Deep Coalescences) criterion has been suggested to quantify the amount of ILS[8] involved in the evolution of a given gene [27]. The MDC criterion has been recently extended to species networks [40]: the gene tree and species network are assumed to be given, and the task is to find an embedding of the gene tree within the species tree in a way that minimizes the ILS score (see below for formal definitions). This approach is not directly applicable to our problem because here we are not given as input a gene tree. However, it can be still of use: given a binary network and a character on X, we can try and find a combination of gene tree and embedding that minimizes (some weighted combination of) the parsimony score *and* the ILS score. Thus in a sense our approach "cuts out the middle man" by not requiring a separate method to construct candidate gene trees.

Definitions in this section are based on those in [40] but we generalize the definitions in that paper to also handle phylogenetic networks, rather than just species trees.

Definition 3. *Given a species network N on X, let $Path(N)$ denote the set of directed paths in N (including paths of length 0). Given a tree T on X, a* coalescent history *is a function $h : V(T) \cup E(T) \to V(N) \cup Path(N)$ such that*

- *for each $x \in V(T)$ and $e \in E(T)$, $h(x) \in V(N)$ and $h(e) \in Path(N)$;*
- *$h(x) = x$ for all $x \in X$;*
- *for each edge $e = xy \in E(T)$, $h(e)$ is a path in N from $h(x)$ to $h(y)$.*

Definition 4. *Given a network N on X, a tree T on X, and a coalescent history $h : V(T) \cup E(T) \to V(N) \cup Path(N)$, we say an edge $xy \in E(T)$* passes through *an edge $uv \in E(N)$ if uv is part of the path $h(xy)$. For any edge $uv \in E(N)$, the* number of lineages *in uv is the number of branches of T passing through the edge uv. The* number of extra lineages *$XL_{T,h}(uv)$ in uv is the number of lineages in uv minus 1. The* deep coalescence *$XL_{T,h}(N)$ is the sum of $XL_{T,h}(uv)$ over all edge uv in N.*

[8] Note that other models to quantify ILS taking into account also gene duplications and gene losses [39] and even horizontal gene transfers [6] exist.

We note that if $h(x) = h(y)$ for some edge $xy \in E(T)$, then xy does not pass through any edge of N. Intuitively, this is because x and y are considered to occur at relatively close times, and therefore the edge between x and y should not count as an "extra lineage" in the context of ILS.

We will now define the *combined ILS score* of a species N on X together with a character $\alpha : X \to [p]$. Recall that softwired and hardwired parsimony are both defined in terms of the minimum *hardwired* parsimony of a set of gene trees on X satisfying certain conditions. In the case of softwired parsimony, the trees are required to be displayed by N, whereas for parental parsimony, the trees have to be parentally displayed by N. The notion of combined ILS score generalizes this further - now we allow any tree on X, but we impose an additional cost based on the minimum deep coalescence of the tree with respect to N. More formally:

Definition 5. *Fix two positive integers A, B. Given a network N on X and a character $\alpha : X \to [p]$, the* combined ILS score *of (N, α), denoted $ILS(N, \alpha)$ is the minimum value of*

$$A \cdot PS_{hw}(T, \alpha) + B \cdot XL_{T,h}(N)$$

over all gene trees T on X and coalescent histories h for T:

We now show that we can model the combined ILS score with lineage functions.

Definition 6. *Given a network N, a lineage function $f : V(N) \to \mathcal{P}([p])$, and a node $v \in V(N)$, the* ILS weight *of v with respect to f, denoted $w_f(v)$, is defined as follows. If $v = \rho_N$ then $w_f(v) = A \cdot \max(|f(v)| - 1, 0)$. If $v \neq \rho_N$ and $f(u) = \emptyset$ for each parent u of v but $f(v) \neq \emptyset$, then $w_f(v) = \infty$. Otherwise, $w_f(v)$ is defined to be the minimum value of*

$$A \cdot |f(v) \setminus (\bigcup_{u \in par(v)} S_u)| + B \cdot \sum_{u \in par(v)} \max(|S_u| - 1, 0)$$

where the minimum is taken over all possible sets $\{S_u : u \in par(v)\}$ such that for each $u \in par(v)$, S_u is a subset of $f(v) \cap f(u)$.

The total ILS weight *of f is defined as*

$$w(f) = \sum_{v \in V(N)} w_f(v)$$

The rough idea behind this definition is that S_u corresponds to the states of branches "passing through" the edge uv. Thus $\min(|S_u| - 1, 0)$ is the number of extra lineages in uv, while $|f(v) \setminus (\bigcup_{u \in par(v)} S_u)|$ is the number of state changes that had to occur (either because a given state in v was not present in any parent of v, or because the branch with that state did not pass through the edge leading to v).

For the purposes of this section, we say a lineage function $f : V(N) \to \mathcal{P}([p])$ is *rooted* if $f(\rho_N) \neq \emptyset$ (note that we may have $|f(\rho_N)| > 1$).

Lemma 3. *Given a species network N on X and a character $\alpha : X \to [p]$, the combined ILS score of (N, α) is the minimum total ILS weight of a rooted α-consistent lineage function f on N.*

Theorem 5 implies that a lineage function of minimum weight can be found in time $O(\mu^{l+3}|V(N)|)$, where l is the level of the network and μ is the size of the set $\mathcal{Y}$ for which f is the function $f : V(N) \to \mathcal{Y}$. In this case we have that $\mathcal{Y} = \mathcal{P}([p])$, and so $\mu = 2^p$. By Lemma 3, we have that the combined ILS score of (N, α) can be found in time $O(\mu^{l+3}|V(N)|) = O(2^{(l+3)p}|V(N)|)$. Note that this implies the following result:

Corollary 1. *Given a species tree T on X and a binary character $\alpha : X \to [p]$, the combined ILS score of (T, α) can be found in polynomial time.*

It is also interesting to notice that, [40] gave an algorithm to find the minimum $XL_{T,h}(N)$ over all possible coalescent histories for a given gene tree T having a complexity of $O(\sum_{x \in L(N)} h(x) \cdot 2^{\sum_{x \in L(N)} h(x)a(x)})$, where $a(x)$ is the number of alleles sampled from x, and $h(x)$ is the maximum number of reticulations on a path from the root of N to x. Thus, their algorithm is exponential in X.

To summarize this section, we have shown that our model and algorithm can also be used when ILS is expected to have occurred and is not only due to hybrid speciations. In such situations, the evolution of a character down a network can branch anywhere in the network, not only in branching nodes but also in-between them. To make sure that we do not postulate more ILS than necessary, we look at the number of *additional lineages*, i.e., the number of lineages of the character genealogy that evolve down a network branch in addition to the single lineage that one would expect without ILS. We have show that, using the same algorithm as before, we can minimize a combination of the parsimony score and the ILS score.

Note that we can combine all extensions presented above in a "blending" of ILS modeling, duplication and losses minimization with sets/multisets of states, and deal with it via lineage functions.

5 Discussion

Parental parsimony avoids several shortcomings of the previous versions of parsimony on networks: hardwired and softwired. In particular, it can be used in the case of allopolyploid hybridization and can easily handle data with multiple individuals per species. Moreover, it can even be used in a different framework including gene duplication, gene loss and incomplete lineage sorting.

However, the improved modeling power of parental parsimony comes at a cost of higher computational demands. Indeed, we have shown that computing the parental parsimony score of a given network is already NP-hard for

relatively simple networks (no reticulate branch has a reticulate ancestor and each internal node has at least one non-reticulate descendant branch). Nevertheless, our dynamic programming algorithm can compute this score efficiently for networks that are reasonably tree-like, in the sense that the "level" of the network is not too large. For networks with large level, our algorithm will be too slow to be practical. Therefore, an important open question is whether it is possible to reduce the amount of guessing in the algorithm, to improve its applicability to high-level networks.

It is also important to note that another shortcoming of hardwired and softwired parsimony has not been overcome yet: When computing a parsimony score of a phylogenetic tree or network, we always consider each character separately, using the assumption that different characters evolve independently. While this is a reasonable assumption for phylogenetic trees, it is not always safe to assume this for phylogenetic networks. Indeed, characters that are close together in a sequence could be more likely to follow the same parental tree inside the network. However, taking this dependence into account when computing the parental parsimony score is problematic. One option would be to compute breakpoints and to force characters within the same block (consisting of all characters between two breakpoints) to choose the same parental tree in a similar flavour to what done in [17, 18, 30]. However, extending our algorithm to handle this would make the running time exponential in the number of characters considered simultaneously. Hence, such an approach would not be useful in practice.

At the same time, it is worth considering whether the improved modeling power of parental parsimony is worth the extra computational effort. For example, it has been shown recently that the effects of incomplete lineage sorting are not always relevant [36].

Therefore, a logical next step is to implement the proposed algorithm and to do extensive simulations and tests on practical data in order to find out how different the softwired and parental parsimony scores are, and which score is most relevant to be used for analyzing different kinds of biological data sets.

Funding

Leo van Iersel was partly supported by the Netherlands Organization for Scientific Research (NWO), including Vidi grant 639.072.602, and partially by the 4TU Applied Mathematics Institute. Mark Jones was supported by Vidi grant 639.072.602 from NWO. Celine Scornavacca was partially supported by the French Agence Nationale de la Recherche Investissements d'Avenir/ Bioinformatique (ANR-10-BINF-01-02, Ancestrome).

Acknowledgements

The authors wish to thank Fabio Pardi for useful discussions and the editor and reviewers for useful suggestions and comments.

References

[1] Richard Abbott, Dirk Albach, Stephen Ansell, Jan W Arntzen, Stuart JE Baird, Nicolas Bierne, J Boughman, Alan Brelsford, C Alex Buerkle, Richard Buggs, et al. Hybridization and speciation. *Journal of Evolutionary Biology*, 26(2):229–246, 2013.

[2] Paola Alimonti and Viggo Kann. Hardness of approximating problems on cubic graphs. *Algorithms and Complexity*, pages 288–298, 1997.

[3] Luis Boto. Horizontal gene transfer in evolution: facts and challenges. *Proceedings of the Royal Society of London B: Biological Sciences*, 277(1683):819–827, 2010.

[4] David Bryant, Remco Bouckaert, Joseph Felsenstein, Noah A Rosenberg, and Arindam RoyChoudhury. Inferring species trees directly from biallelic genetic markers: bypassing gene trees in a full coalescent analysis. *Molecular biology and evolution*, 29(8):1917–1932, 2012.

[5] Gabriel Cardona, Joan Carles Pons, and Francesc Rosselló. A reconstruction problem for a class of phylogenetic networks with lateral gene transfers. *Algorithms for Molecular Biology*, 10(1):28, 2015.

[6] Yao-ban Chan, Vincent Ranwez, and Celine Scornavacca. Inferring incomplete lineage sorting, duplications, transfers and losseswith reconciliations. *To appear in Journal of Theoretical Biology*, 2017.

[7] William HE Day, David S Johnson, and David Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical biosciences*, 81(1):33–42, 1986.

[8] James H Degnan and Noah A Rosenberg. Gene tree discordance, phylogenetic inference and the multispecies coalescent. *Trends in ecology & evolution*, 24(6):332–340, 2009.

[9] W Ford Doolittle. Phylogenetic classification and the universal tree. *Science*, 284(5423):2124–2128, 1999.

[10] Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

[11] Joseph Felsenstein. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Zoology*, pages 401–410, 1978.

[12] Mareike Fischer, Leo Van Iersel, Steven Kelk, and Celine Scornavacca. On computing the maximum parsimony score of a phylogenetic network. *SIAM Journal on Discrete Mathematics*, 29(1):559–585, 2015.

[13] Walter M Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

[14] Jörg Flum and Martin Grohe. Parameterized complexity theory, volume xiv of texts in theoretical computer science. an eatcs series, 2006.

[15] Jens Gramm, Arfst Nickelsen, and Till Tantau. Fixed-parameter algorithms in phylogenetics. *The Computer Journal*, 51(1):79–101, 2008.

[16] Dan Gusfield. *ReCombinatorics: the algorithmics of ancestral recombination graphs and explicit phylogenetic networks*. MIT Press, 2014.

[17] Jotun Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical biosciences*, 98(2):185–200, 1990.

[18] Jotun Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36(4):396–405, 1993.

[19] Katharina T Huber, Vincent Moulton, Mike Steel, and Taoyang Wu. Folding and unfolding phylogenetic trees and networks. *Journal of Mathematical Biology*, 73(6-7):1761–1780, 2016.

[20] K.T. Huber and V. Moulton. Phylogenetic networks from multi-labelled trees. *Journal of Mathematical Biology*, 52(5):613–632, 2006.

[21] Daniel H Huson, Regula Rupp, and Celine Scornavacca. *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press, 2010.

[22] Guohua Jin, Luay Nakhleh, Sagi Snir, and Tamir Tuller. Maximum likelihood of phylogenetic networks. *Bioinformatics*, 22(21):2604–2611, 2006.

[23] Guohua Jin, Luay Nakhleh, Sagi Snir, and Tamir Tuller. Maximum likelihood of phylogenetic networks. *Bioinformatics*, 22(21):2604–2611, 2006.

[24] Guohua Jin, Luay Nakhleh, Sagi Snir, and Tamir Tuller. Efficient parsimony-based methods for phylogenetic network reconstruction. *Bioinformatics*, 23(2):e123–e128, 2007.

[25] Lavanya Kannan and Ward C Wheeler. Maximum parsimony on phylogenetic networks. *Algorithms for Molecular Biology*, 7(1):9, 2012.

[26] Liang Liu, Lili Yu, and Scott V Edwards. A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. *BMC evolutionary biology*, 10(1):302, 2010.

[27] Wayne P Maddison. Gene trees in species trees. *Systematic biology*, 46(3):523–536, 1997.

[28] James Mallet. Hybrid speciation. *Nature*, 446(7133):279, 2007.

[29] DA Morrison. *Introduction to Phylogenetic Networks*. RJR Productions, 2011.

[30] C Thach Nguyen, Nguyen Bao Nguyen, Wing-Kin Sung, and Louxin Zhang. Reconstructing recombination network from sequence data: the small parsimony problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 4(3):394–402, 2007.

[31] Quan Nguyen and Teemu Roos. Likelihood-based inference of phylogenetic networks from sequence data by phylodag. In *International Conference on Algorithms for Computational Biology*, pages 126–140. Springer, 2015.

[32] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[33] Joseph K Pickrell and Jonathan K Pritchard. Inference of population splits and mixtures from genome-wide allele frequency data. *PLoS Genet*, 8(11):e1002967, 2012.

[34] David Posada, Keith A Crandall, and Edward C Holmes. Recombination in evolutionary genomics. *Annual Review of Genetics*, 36(1):75–97, 2002.

[35] Arindam RoyChoudhury, Joseph Felsenstein, and Elizabeth A Thompson. A two-stage pruning algorithm for likelihood computation for a population tree. *Genetics*, 180(2):1095–1105, 2008.

[36] Celine Scornavacca and Nicolas Galtier. Incomplete lineage sorting in mammalian phylogenomics. *Systematic Biology*, 66(1):112–120, 2017.

[37] Claudia Solís-Lemus and Cécile Ané. Inferring phylogenetic networks with maximum pseudolikelihood under incomplete lineage sorting. *PLoS Genet*, 12(3):e1005896, 2016.

[38] Séverine Vuilleumier and Sebastian Bonhoeffer. Contribution of recombination to the evolutionary history of hiv. *Current Opinion in HIV and AIDS*, 10(2):84–89, 2015.

[39] Yi-Chieh Wu, Matthew D Rasmussen, Mukul S Bansal, and Manolis Kellis. Most parsimonious reconciliation in the presence of gene duplication, loss, and deep coalescence using labeled coalescent trees. *Genome research*, 24(3):475–486, 2014.

[40] Yun Yu, R Matthew Barnett, and Luay Nakhleh. Parsimonious inference of hybridization in the presence of incomplete lineage sorting. *Systematic biology*, 62(5):738–751, 2013.

[41] Yun Yu, Nikola Ristic, and Luay Nakhleh. Fast algorithms and heuristics for phylogenomics under ils and hybridization. *BMC bioinformatics*, 14(15):S6, 2013.

[42] Olga Zhaxybayeva and W Ford Doolittle. Lateral gene transfer. *Current Biology*, 21(7):R242–R246, 2011.

[43] Jiafan Zhu, Yun Yu, and Luay Nakhleh. In the light of deep coalescence: revisiting trees within networks. *BMC Bioinformatics*, 17(14):271, 2016.

[44] Sha Zhu and James H Degnan. Displayed trees do not determine distinguishability under the network multispecies coalescent. *Systematic Biology*, 66(2):283–298, 2016.

A Appendix

A.1 Equivalence of lineage functions and PPP

The purpose of this section is to prove Theorem 1:

Theorem 1. *For any binary network N on X and p-state character α on X, $PS_{pt}(N,\alpha) = \min\{w(f) : f$ is a rooted α-consistent lineage function on $N\}$.*

Recall the definition of $U^*(N)$. For each $v \in V(N)$, let Π_v denote the set of all nodes π in $U^*(N)$ for which π is a path in N from ρ_N to v. Before we can show the equivalence of PPP with lineage functions, we need the following technical lemma. Informally, it says that each state is assigned to at most one $\pi \in \Pi_v$ for each $v \in V(N)$.

Lemma 4. *For any binary network N on X and p-state character α on X, there exists a tree $T \in \mathcal{PT}(N)$ and p-state character τ on $V(T)$ extending α, with $\sum_{uv\in E(T)} c_\tau(uv) = PS_{pt}(N,\alpha)$ such that $\tau(\pi) \neq \tau(\pi')$, for any $v \in V(N)$ and distinct $\pi, \pi' \in \Pi_v$.*

Proof. Recalling that $PS_{hw}(T,\alpha) = PS_{hw}(T',\alpha)$ whenever T' is derived from T by suppressing nodes of in-degree and out-degree 1, we may treat a parental tree $T \in \mathcal{PT}(N)$ as a tree on X that is isomorphic to a subtree of $U^*(N)$. So let T be such a tree, and τ a p-state character $V(T)$, such that $\sum_{uv\in E(T)} c_\tau(uv) = PS_{pt}(N,\alpha)$. For each $u \in V(N)$, let W_u be the set of nodes in T that are mapped to a node in Π_v by the isomorphism.

If $\tau(x) \neq \tau(y)$, for any $u \in V(N)$ and distinct $x, y \in W_u$, then (T,τ) satisfies the claim. So now assume this is not the case. We will adjust T and τ to produce a pair (T',τ') for which the claim holds.

Choose a lowest u in N for which there exist distinct $x, y \in W_u$ with $\tau(x) = \tau(y)$. We have that u is not a leaf of N, as $|W_u| = 1$ for such v. Our next to step is show that we may assume $\tau(z) \neq \tau(x)$, for any child z of x, and that furthermore $\tau(z) \neq \tau(y')$, for any child z of x and $y' \in W_u \setminus \{x\}$.

If there exists a child z of x and $y' \in W_u \setminus \{x\}$ such that $\tau(z) = \tau(y')$, adjust T and τ as follows. Let v be the child of u in N for which $z \in W_v$, and note that z is the only child of x in W_v. If y' has a child in W_v then denote this child by z'. Note that, since z and z' are both in W_v and z is lower than x, we have that $\tau(z) \neq \tau(z')$. Now delete the edges xz, $y'z'$ and add the edges xz', $y'z$ (or just delete xz and add $y'z$ if z' does not exist). Let T' denote the resulting tree.

Before continuing, we need to show that T' is still a parental tree of N. To do this, let $\mu : V(T) \to V(U^*(N))$ denote the isomorphism from T to a subtree of $U^*(N)$. Then $\mu(x)$ is a node in $U^*(N)$ corresponding to a path in N from ρ_N to u. Let π_x denote this path, and similarly let $\pi_{y'}$ denote the path from ρ_N to u corresponding to $\mu(y')$. Note that for every descendant w of x in W_v, $\mu(w)$ corresponds to a path that begins with π_x. Similarly, for every descendant w' of y' in W_v, $\mu(w')$ corresponds to a path that begins with $\pi_{y'}$. To define an isomorphism from T' to a subtree of $U^*(N)$, we can therefore do the following: If $w \in V(T')$ is not a descendant of z or z' then set $\mu'(w) = \mu(w)$. For each descendant w of z, let $\mu'(w)$ be $\mu(w)$ with the path segment corresponding to π_x replaced with $\pi_{y'}$. For each descendant w of z', let $\mu'(w)$ be $\mu(w)$ with the path segment corresponding to $\pi_{y'}$ replaced with π_x. Then μ' is an isomorphism from T' to a subtree of $U^*(N)$, and so T' is a parental tree of N.

So if there exists a child z of x and $y' \in W_u \setminus \{x\}$ with $\tau(y') = \tau(z)$, the tree T' as described above is a parental tree of N. Furthermore consider $\sum_{uv\in E(T')} c_\tau(uv)$, where τ is the same character as before (in particular, $\tau(z)$ and $\tau(z')$ are unchanged even though they have different parents in T'). This value is equal to $\sum_{uv\in E(T)} c_\tau(uv) + (c_\tau(y'z) - c_\tau(xz)) + (c_\tau(xz') - c_\tau(y'z'))$. If $\tau(y') \neq \tau(x)$, then $c_\tau(y'z) = 0 < 1 = c_\tau(xz)$, and $c_\tau(xz') \leq 1 = c_\tau(y'z')$ (since $\tau(z') \neq \tau(z) = \tau(y')$). Thus $\sum_{uv\in E(T')} c_\tau(uv) < \sum_{uv\in E(T)} c_\tau(uv) = PS_{pt}(N,\alpha)$, a contradiction. So we must have that $\tau(y') = \tau(x)$, and therefore $\sum_{uv\in E(T')} c_\tau(uv) = \sum_{uv\in E(T)} c_\tau(uv) = PS_{pt}(N,\alpha)$.

Thus we now have that for any child z of x, there is no $y' \in W_u \setminus \{x\}$ with $\tau(y') = \tau(z)$ unless $\tau(x) = \tau(y') = \tau(z)$, and in such a case we may produce a parental tree of the same parsimony score such that z is no longer a child of x (and every node except x and y' has the same children as before). Repeating this process for any child z of x with $\tau(z) = \tau(x)$, and recalling that there exists $y \in W_u \setminus \{x\}$ with $\tau(y) = \tau(x)$, we may now assume that for any child z of x, $\tau(z) \neq \tau(y')$ for any $y' \in W_u$ (including x).

Now if x has no children, we can delete x from T. The resulting tree T' is still isomorphic to a subtree of $U^*(N)$, and there exists a character τ' on $V(T')$ such that $\sum_{uv\in E(T')} c_\tau(uv) \leq \sum_{uv\in E(T)} c_\tau(uv) = PS_{pt}(N,\alpha)$ (we let τ' be τ restricted to T'). Otherwise, we can define τ' to be the character on T such that $\tau'(x) = \tau(z)$ for some child z of x, and $\tau'(w) = \tau(w)$ for all other w. Then $\sum_{uv\in E(T)} c_{\tau'}(uv) \leq \sum_{uv\in E(T)} c_\tau(uv) - 1 + 1 = PS_{pt}(N,\alpha)$. Furthermore $\tau'(x) \neq \tau(y')$ for any $y' \in W_u \setminus \{x\}$, as $\tau(z) \neq \tau(y')$.

As we either reduce the number of nodes in T, or the number of pairs $x, y \in W_u$ with $\tau(x) = \tau(y)$, this process cannot be repeated indefinitely. It follows that by applying it exhaustively, we eventually reach a tree $T' \in \mathcal{PT}(N)$ and p-state character τ' on $V(T')$ that satisfy the claim.

□

Given Lemma 4, we can now prove one direction of Theorem 1:

Lemma 5. *Given a phylogenetic network N on X and a p-state character α on X, there exists a rooted α-consistent lineage function f on N such that $w(f) \leq PS_{pt}(N,\alpha)$.*

Proof. Let T be a parental tree in N, and τ an extension of α to $V(T)$ such that $\sum_{uv\in E(T)} c_\tau(uv) = PS_{pt}(N,\alpha)$. Let T^* be the subtree of $U^*(N)$ such that T can be obtained from T^* by suppressing nodes of in-degree and out-degree 1. Let τ^* be an extension of α to $V(T^*)$ such that $\sum_{\pi\pi'\in E(T^*)} c_{\tau^*}(\pi\pi') = \sum_{uv\in E(T)} c_\tau(uv)$. (Such a τ^* exists, as $PS_{hw}(T^*,\alpha) = PS_{hw}(T,\alpha)$.)

Then we may define a lineage function f on N as follows: For all $v \in V(N)$, set $f(v) = \{\tau^*(\pi) : \pi \in \Pi_v\}$. That is, $f(v)$ is the set of states assigned to at least one node of T^* corresponding to a path in N ending at v. By Lemma 4, we may assume that $|f(v)| = |\Pi_v|$.

We now show that f is a rooted, α-consistent lineage function on N. To see this, observe that T^* has exactly one node in Π_{ρ_N}, and exactly one node π_x in Π_x for each $x \in X$, and that for each $x \in X$, $f(x) = \{\tau^*(\pi_x)\} = \{\alpha(x)\}$.

Finally we show that $w(f) \leq PS_{pt}(N, \alpha)$. Observe first that for any $v \in V(N) \setminus \{\rho_N\}$, each node in Π_v has a parent in Π_u for some $u \in par(v)$. Thus $|f(v)| = |\Pi_v| \leq \sum_{u \in par(v)} |\Pi_u| = \sum_{u \in par(v)} |f(u)|$, and so $w_f(v) \neq \infty$.

Observe that for any $v \neq \rho_N$, if there exists π' in Π_v with parent π and $\tau^*(\pi') \in f(v) \setminus (\bigcup_{u \in par(v)} f(u))$, it must be the case that $\tau^*(\pi') \neq \tau^*(\pi)$, and so $c_{\tau^*}(\pi\pi') = 1$. Therefore $\sum\{c_{\tau^*}(\pi\pi') : \pi\pi' \in E(T^*), \pi' \in \Pi_v\} \geq |f(v) \setminus (\bigcup_{u \in par(v)} f(u))| = w_f(v)$.

Adding up over all $v \in V(N) \setminus \rho_N$, we have

$$\begin{aligned} w(f) &= \sum_{v \in V(N)} w_f(v) = \sum_{v \in V(N) \setminus \rho_N} w_f(v) \\ &\leq \sum_{v \in V(N) \setminus \rho_N} \sum \{c_{\tau^*}(\pi\pi') : \pi\pi' \in E(T^*), \pi' \in \Pi_v\} \\ &= \sum_{\pi\pi' \in E(T^*)} c_{\tau^*}(\pi\pi') = \sum_{uv \in E(T)} c_\tau(uv) = PS_{pt}(N, \alpha) \end{aligned}$$

as required. □

We now prove the other direction of Theorem 1:

Lemma 6. *Given a phylogenetic network N on X and a p-state character α on X, for any rooted α-consistent lineage function f on N it holds that $w(f) \geq PS_{pt}(N, \alpha)$.*

Proof. Let f be a rooted α-consistent lineage function on N, and assume w.l.o.g. that $w(f) < \infty$. We will construct a tree T^* on X which is a subtree of $U^*(N)$, and a character τ^* on $V(T^*)$ that extends α, such that $\sum_{\pi\pi' \in E(T^*)} c_{\tau^*}(\pi\pi') \leq w(f)$. Then by suppressing nodes of in-degree and out-degree 1 in T^*, we get a tree $T \in \mathcal{PT}(N)$ such that $PS_{hw}(T, \alpha) = PS_{hw}(T^*, \alpha) \leq \sum_{\pi\pi' \in E(T^*)} c_{\tau^*}(\pi\pi') \leq w(f)$. The pair (T^*, τ^*) will be such that for each $v \in V(N)$, if T^*_v is the set of nodes in T^* corresponding to a path ending in v, then $|T^*_v| = |f(v)|$ and $\{\tau^*(\pi) : \pi \in T^*_v\} = f(v)$.

We construct T^* and τ^* in a top-down manner. Taking a topological ordering of $V(N)$, for each $v \in V(N)$ in turn we construct the vertices of T^* corresponding to paths ending at v.

Initially, we let T^* be the single node π_{ρ_N} in $U^*(N)$ corresponding to ρ_N (i.e. the single-vertex path consisting only of ρ_N). As f is rooted, $f(\rho_N)$ contains a single element of $\{1, \ldots, p\}$; we let $\tau^*(\pi_{\rho_N})$ be this single element.

Now consider some $v \in V(N) \setminus \{\rho_N\}$, and assume we have already constructed the set of nodes in T^* corresponding to paths ending in u for any $u \in par(v)$. Let $T^*_{par(v)}$ be this set of nodes. We mark a subset of nodes of $T^*_{par(v)}$ and add children to them, as follows:

For each $i \in f(v) \cap (\bigcup_{u \in par(v)} f(u))$ in turn, choose an element π of $T^*_u, u \in par(v)$, for which $\tau^*(\pi) = i$. For the chosen π, mark π and add the path $\pi' = \pi\{uv\}$ as a child of π. Set $\tau^*(\pi') = i$. Observe that $c_{\tau^*}(\pi\pi') = 0$.

After this, for each $i \in f(v) \setminus (\bigcup_{u \in par(v)} f(u))$ in turn, choose a currently-unmarked element π of $T^*_{par(v)}$. (Such an element must exist, as otherwise $|f(v)| > \sum_{u \in par(v)} |T^*_u| = \sum_{u \in par(v)} |f(u)|$ and $w_f(v) = \infty$, a contradiction.) For the chosen π, mark π and add the path $\pi' = \pi\{uv\}$ as a child of π, where $u \in par(v)$ is such that $\pi \in T^*_u$. Set $\tau^*(\pi') = i$. Observe that $c_{\tau^*}(\pi\pi') = 1$ (as $\tau^*(\pi) \neq i$).

This completes the construction of T^*_v. Observe now that $|T^*_v| = |f(v)|$ and $\{\tau^*(\pi) : \pi \in \Pi_v\} = f(v)$, as required. Furthermore observe that $\sum\{c_{\tau^*}(\pi\pi') : \pi\pi' \in E(T^*), \pi' \in \Pi_v\} = |f(v) \setminus (\bigcup_{u \in par(v)} f(u))| = w_f(v)$.

By construction, the resulting tree T^* is a subgraph of $U^*(N)$. As f is α-consistent, T^* will have exactly one node corresponding to a path ending in x, for each $x \in X$. Thus the tree T, derived from T^* by suppressing nodes of in-degree and out-degree 1, is a parental tree of N, and so $\sum_{\pi\pi' \in E(T^*)} c_{\tau^*}(\pi\pi') \geq PS_{pt}(N, \alpha)$. Finally, we observe that

$$\begin{aligned} w(f) &= \sum_{v \in V(N)} w_f(v) = \sum_{v \in V(N) \setminus \rho_N} w_f(v) \\ &= \sum_{v \in V(N) \setminus \rho_N} \sum \{c_{\tau^*}(\pi\pi') : \pi\pi' \in E(T^*), \pi' \in \Pi_v\} \\ &= \sum_{\pi\pi' \in E(T^*)} c_{\tau^*}(\pi\pi') \geq PS_{pt}(N, \alpha). \end{aligned}$$

□

Combining the previous two lemmas, we get Theorem 1.

A.2 NP-hardness – missing proofs

In this section we prove Theorem 2.

Theorem 2. *PPP is NP-hard, even when the character α is binary, and the network N is tree-child and has reticulation depth at most* 1.

To prove Theorem 2, we will first give a reduction from the following problem:

TREE-INFLUENCED BIPARTITION (TIB)
Input: A graph $G = (V, E = E_= \uplus E_\neq)$, such that $T = (V, E_=)$ is a tree spanning V and every leaf of T has degree 1 in G; an assignment $g' : L(T) \to \{1, 2\}$ and an integer w.
Output: An assignment $g : V \to \{1, 2\}$ extending g' with at most w *unsatisfied* edges, if one exists, where an edge $uv \in E_=$ is *satisfied* if $g(u) = g(v)$, and an edge $uv \in E_\neq$ is *satisfied* if $g(u) \neq g(v)$.

Lemma 7. *Given an instance (G, g', w) of TIB, we can in polynomial time construct an instance (N, α) of PPP, such that (G, g', w) is a* YES*-instance of TIB if and only if $PS_{pt}(N, \alpha) \leq w$, and such that N is tree-child and has reticulation depth at most one, and α is binary.*

Proof. Let $(G = (V, E = E_= \uplus E_\neq), \alpha, w)$ be an instance of TIB.

We construct the network N as follows. Let $T = (V, E_=)$, the tree formed by the edges in $E_=$. Choose an arbitrary non-leaf vertex of T as the root, and orient all edges in $E_=$ away from the root. For each edge e in $E_\neq$, let u_e, v_e, x_e, y_e be new nodes. Then let $V' = V \cup \{u_e, v_e, x_e, y_e : e \in E_\neq\}$, and let $E' = E_= \cup \{au_e, bu_e, u_ev_e, v_ex_e, v_ey_e : e = ab \in E_\neq\}$. Let $N = (V', E')$. Figure 8 shows the structure on $u = u_e, v = v_e, x = x_e, y = y_e$ for some $e = ab \in E_\neq$.

Observe that the leaves of N are $L(T)$ together with x_e and y_e for each $e \in E_\neq$, and that $\{u_e : e \in E_\neq\}$ are the reticulation nodes of N. We therefore have that N is tree-child - indeed, every parent of u_e has another child in T which is a tree node (the leaves of T cannot be incident to $e \in E_\neq$ by definition of a TIB instance), and every other node either has a tree node child or is a leaf. As there is no path from u_e to $u_{e'}$ for any $e \neq e'$, we have that there is at most one reticulation node on any path.

Now we define the binary character $\alpha : L(N) \to \mathcal{P}(\{1, 2\})$ as follows. If $x \in L(T)$ then $\alpha(x) = \{g'(x)\}$. If $x \in L(N) \setminus L(T)$ then x is either x_e or y_e for some $e \in E_\neq$. If $x = x_e$ then $\alpha(x) = 1$, and if $x = y_e$ then $\alpha(x) = 2$.

We now show that if (G, g', w) is a YES-instance of TIB then $PS_{pt}(N, \alpha) \leq w$.

Indeed, let $g : V \to \{1, 2\}$ be an assignment extending f' such that g has at most w unsatisfied edges. Then define the lineage function $f : V' \to \mathcal{P}(\{1, 2\})$ as follows. If $v \in V$ then $f(v) = \{g(v)\}$. For each $e \in E_\neq$, set $f(u_e) = f(v_e) = \{1, 2\}, f(x_e) = 1, f(y_e) = 2$. Observe that f is rooted and α-consistent. For any non-root $v \in V$ with parent u, $w_f(v) = 0$ if and only if $g(u) = g(v)$, and 1 otherwise. For any $e \in E_\neq$, we have $w_f(v_e) = w_f(x_e) = w_f(y_e) = 0$. Finally for each $e = ab \in E_\neq$, we have that $w_f(u_e) = 1$ if and only if $g(a) = g(b)$, and 0 otherwise. We therefore have that $w(f) = \sum_{v \in V' \setminus \rho_N} w_f(v) = \sum_{uv \in E'} w_f(v) = |ab \in E_= : g(a) \neq g(b)| + |ab \in E_\neq : g(a) = g(b)| \leq w$. Therefore by Theorem 1, $PS_{pt}(N, \alpha) \leq w$.

Now for the converse, suppose that $PS_{pt}(N, \alpha) \leq w$. Therefore by Theorem 1, there exists a rooted α-consistent lineage function f on N, such that $w(f) \leq w$. For every $v \in V$, v has no reticulation ancestors, and therefore by Lemma 2 we may assume that $|f(v)| = 1$. Then define the assignment $g : V \to \{1, 2\}$ by setting $g(v)$ to be the unique element of $f(v)$, for all $v \in V$. Observe that g extends g', as for all $x \in L(T)$, x is in $L(N)$ and therefore $\{g(x)\} = f(x) = \alpha(x) = \{g'(x)\}$.

It remains to show that the number of unsatisfied edges is at most $w(f)$. Let $F_= = \{uv \in E_= : g(u) \neq g(v)\}$, and $F_\neq = \{uv \in E_= : g(u) = g(v)\}$. Thus $|F_=| + |F_\neq|$ is the number of unsatisfied edges with respect to g. As previously argued, for any edge uv in $E_=$ we have that $w_f(v) = 1$ if $g(u) \neq g(v)$, and 0 otherwise. Therefore $\sum_{v \in V \setminus \{\rho_N\}} w_f(v) = |F_=|$. Observe that for any $e \in F_\neq$, if $w_f(x_e) = w_f(y_e) = 0$ then $f(v_e) = \{1, 2\}$, in which case either $w_f(v_e) > 0$ (if $f(u_e) \neq \{1, 2\}$) or $w_f(u_e) > 0$ (otherwise). Then $w_f(u_e) + w_f(v_e) + w_f(x_e) + w_f(y_e) \geq 1$ for each $e \in F_\neq$. Thus $w(f) = \sum_{v \in V' \setminus \rho_N} w_f(v) \geq |F_=| + |F_\neq|$, and so g has at most w unsatisfied edges. □

We now need to prove that TIB is NP-hard. We do this by reduction from the problem MAX-CUT on cubic graphs.

MAX-CUT
Input: An undirected graph $H = (U, F)$
Output: An assignment $f : U \to \{1, 2\}$ such that $|\{uv \in E : f(u) \neq f(v)\}|$ is maximized.

It is shown in [2] that MAX-CUT on cubic graphs is APX-hard, from which the following theorem follows.

Theorem 6. MAX-CUT *is NP-hard, even when every vertex in has degree* 3 *in H.*

Lemma 8. *TIB is NP-hard.*

Proof. Given an instance $H = (U, F)$ of MAX-CUT, we construct an instance of TIB as follows.

For each $x \in U$, we will let x_1, x_2, x_3 be three specific nodes in our constructed graph G. Furthermore, we associate each of x_1, x_2, x_3 with a different edge incident to x. Thus, from now on, we will say an edge $e = xy$ is associated with the pair (x_i, y_j) for some $i, j \in [3]$, and then x_i will not appear in a pair associated with any other edge.

For each $x \in U$, define the gadget G_x as follows. Let V_x contain the nodes ρ_x, x_1, x_2, x_3, a_x, b_x, c_x, d_x, $r_{x,1}$, $r_{x,2}$, $l_{x,1}$, $l_{x,2}$. Let $E_{=,x}$ contain the edges $\rho_x a_x$, $a_x x_1$, $x_1 x_2$, $x_2 x_3$, $x_3 d_x$, $d_x l_{x,1}$, $d_x l_{x,2}$, $\rho_x b_x$, $b_x c_x$, $c_x r_{x,1}$, $c_x r_{x,2}$, and let $E_{\neq,x}$ contain the edge $a_x b_x$. Let $G_x = (V_x, E_{=,x} \cup E_{\neq,x})$. (See Fig. A11.)

Now we join the gadgets G_x by constructing an arbitrary binary tree, with internal nodes V_0, whose leaves are $\{\rho_x : x \in U\}$. In addition, for each edge $e = xy$ associated with the pair (x_i, y_j), we add the edge $x_i y_j$ to $E_\neq$.

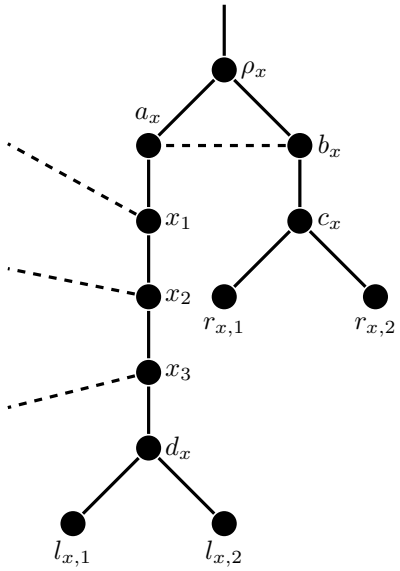

FIGURE A11: Example gadget G_x. Solid edges are in $E_=$, dashed edges are in $E_{\neq}$. Nodes x_1, x_2, x_3 represent three copies of the vertex x, with incident dashed edges going to nodes corresponding to the three neighbors of x. Leaves are assigned as follows: $g'(r_{x,1}) = g'(l_{x,1}) = 1$, $g'(r_{x,2}) = g'(r_{x,2}) = 2$. The gadget enforces that x_1, x_2, x_3 must all be assigned the same value.

Observe that the leaves of G are $r_{x,1}, r_{x,2}, l_{x,1}, l_{x,2}$ for each $x \in U$. For each $x \in U$, set $g'(r_{x,1}) = g'(l_{x,1}) = \{1\}$, and $g'(r_{x,2}) = g'(r_{x,2}) = \{2\}$.

This concludes the construction of our TIB instance.

Claim 1. *Given any assignment $f : U \to \{1, 2\}$, there exists an assignment $g : V \to \{1, 2\}$ extending g' with at most $3|U| + |E| - |\{uv \in E : f(u) \neq f(v)\}|$ unsatisfied edges.*

Proof. Define g as follows. For any $v \in L(T)$ set $g(v) = g'(v)$. Set $g(\rho_x) = \{1\}$ for all $x \in U$ and $g(u) = \{1\}$ for all $u \in V_0$. For each $x \in U$, set $g(a_x) = g(x_1) = g(x_2) = g(x_3) = g(d_x) = \{f(x)\}$, and $g(b_x) = g(c_x) = \{3 - f(x)\}$.

Observe that within each G_x, exactly one of the edges $d_x l_{x,1}, d_x l_{x,2}$ is unsatisfied, exactly one of $c_x r_{x,1}, c_x r_{x,2}$ is unsatisfied, and exactly one of $\rho_x a_x, \rho_x b_x$ is unsatisfied. All other edges within G_x are satisfied. All edges incident to V_0 are satisfied. Finally, for each $e = xy \in E$ associated with (x_i, y_j), the edge $x_i y_j$ is satisfied if and only if $f(x) \neq f(y)$.

Thus overall, the number of unsatisfied edges in G is $3|U| + |E| - |\{uv \in E : f(u) \neq f(v)\}|$. □

Claim 2. *Given any assignment $g : V \to \{1, 2\}$ extending g', there exists an assignment $f : U \to \{1, 2\}$ such that g has at least $3|U| + |E| - |\{uv \in E : f(u) \neq f(v)\}|$ unsatisfied edges.*

Proof. We first show that there exists an assignment $g'' : V \to \{1, 2\}$ extending g' with no more unsatisfied edges than g, such that $g''(x_1) = g''(x_2) = g''(x_3)$ for all $x \in U$. If $g(x_1) = g(x_2) = g(x_3)$ for all $x \in U$ then there is nothing to prove. So assume that, without loss of generality, two elements of $\{x_1, x_2, x_3\}$ are assigned value 1, and one of them is assigned value 2. (The case when two elements are assigned 2 and one element is assigned 1 is dealt with symmetrically.) Let $g'' : V \to \{1, 2\}$ be the assignment such that $g''(a_x) = g''(x_1) = g''(x_2) = g''(x_3) = g''(d_x) = \{1\}$, and $g''(b_x) = g''(c_x) = \{2\}$, and $g''(v) = g(v)$ for all other v. We now show that $w(g'') \leq w(g)$.

Clearly, it is enough to show that of the edges incident to $a_x, x_1, x_2, x_3, d_x, b_x, c_x$, g has at least as many unsatisfied edges as g''. To this end, let $x_i \in \{1, 2, 3\}$ be the unique index for which $g(x_i) = 2$. In g'', there is at most one unsatisfied edge incident to x_i (that being the edge $x_i y_j \in E_{\neq}$ leaving G_x). But in g there is at least one unsatisfied edge incident to x_i (as x_i is a neighbour of at least one x_j, $j \neq i$). Of the remaining edges, exactly one of the edges $d_x l_{x,1}, d_x l_{x,2}$ is unsatisfied in g'', but this is also the case in g. Exactly one of the edges $c_x r_{x,1}, c_x r_{x,2}$ is unsatisfied in g'', but again this is also the case in g. Exactly one of the edges $\rho_x a_x, \rho_x b_x, a_x b_x$ is unsatisfied in g'', but at least these many are unsatisfied in g. Of the edges leaving G_x leaving x_j for $j \neq i$, the same edges are unsatisfied in g and g'' (as the assignment on x_j does not change). The remaining edges ($b_x c_x$, and the edges on the path from a_x to d_x that are not incident to x_i) are satisfied in g''. Therefore, g'' has at most as many unsatisfied edges as g.

By repeating this process on g'', we eventually get an assignment that assigns the same value to x_1, x_2, x_3, for each $x \in U$. Therefore we may assume in what follows that $g(x_1) = g(x_2) = g(x_3)$ for all $x \in U$.

So now define the function f as follows. For each $x \in U$, let $f(x) = 1$ if $g(x_1) = g(x_2) = g(x_3) = \{1\}$, and $f(x) = \{2\}$ if $g(x_1) = g(x_2) = g(x_3) = \{2\}$.

As noted above, there at least 3 unsatisfied edges within G_x for each $x \in U$. For any $e = xy \in E$ associated with (x_i, y_j), $f(x) \neq f(y)$ if and only if $x_i y_j$ is satisfied in g. Therefore, g has at least $3|U| + |E| - |\{uv \in E : f(u) \neq f(v)\}|$ unsatisfied edges.

□

Putting the claims together, H has a cut of size at least s if and only if $(G, N, 3|U| + |E| - s)$ is a YES-instance of TIB. □

Given Lemma 7 and Theorem 8, we have Theorem 2.

In addition to being tree-child with reticulation depth 1, we note that N can be made time-consistent by adding a gadget to each edge above a reticulation node, as in the proof of Theorem 4.3 in [12]. (A network is *time consistent* if each node v can be assigned an integer value $t(v)$, such that for any edge uv, $t(u) = t(v)$ if v is a reticulation node, and $t(u) < t(v)$ otherwise.)

A.3 Fixed-Parameter Tractability with respect to reticulation number – missing proofs

In this section we show that PPP is fixed-parameter tractable with respect to the number of reticulation nodes. In the next section, we extend this result to network level.

In this section and the next, we will not use any particular properties of the weight $w(f)$ of a lineage function f, except that $w(f)$ is the sum of $w_f(v)$ over all $v \in V(N)$, and that the value $w_f(v)$ depends only on the set assigned by f to v and each of its parents. The fact that we only use these properties will make it easier to extend our results to other measures, as discussed in the section "Extensions to other measures".

We approach the problem as one of finding a rooted α-consistent lineage function of minimum weight. We will find this by guessing the value of the lineage function on a small number of nodes, such that when these nodes are removed the network becomes a forest. We can then find the optimal lineage function on each tree in this forest using dynamic programming techniques.

The next lemma is stated in a slightly more general way than we need for this section. We do this so that we can make use of this lemma when we come to take network level as a parameter.

Lemma 9. *Let N be a binary network with a single root ρ_N and leaf set X, with k reticulation nodes, and suppose that for each $x \in X$ we are given a cost function $c^*_x : \mathcal{Y} \to \mathbb{N}_0 \cup \{\infty\}$. Then in $O(\mu^{k+2} \cdot |V(N)|)$ time we can construct a table I with entries indexed by $\mathcal{Y}$, such that for each $S \in \mathcal{Y}$, $I[S]$ gives the minimum value of $w^*(f) = w(f) + \sum_{x \in X} c^*_x(f(x))$, over all lineage functions $f : V(N) \to \mathcal{Y}$ such that $f(\rho_N) = S$.*

Observe that by setting $c^*_x(S') = 0$ if $S' = \alpha(x)$, and $c^*_x(S') = \infty$ otherwise, Lemma 9 gives us a way of calculating the minimum value of $w(f)$ for all α-consistent lineage functions f on N such that $f(\rho_N) = S$. By trying $S = \{i\}$ for each $i \in \{1, \ldots, p\}$, this gives us a $O(\mu^{k+2} \cdot |V(N)| + p) = O(\mu^{k+2} \cdot |V(N)|)$ time algorithm for calculating $PS_{pt}(N, \alpha)$.

Proof of Lemma 9. Construct the set $P \subseteq V(N)$ by choosing, for each reticulation node v in N, an arbitrary parent u of v, and adding u to P. As N is a binary phylogenetic network with k reticulation nodes, we have that $|P| \leq k$.

Now consider any function $f' : P \to \mathcal{Y}$. There are at most μ^k such functions. We now fix f', and assume in what follows that any reticulation function f must be an extension of f'. Therefore we will use the constant σ_u to denote $f'(u)$, for any $u \in P$.

For any set $V' \subseteq V(N)$, let $\mathcal{G}(V')$ denote the set of all functions $g : V' \cup P \to \mathcal{Y}$ such that $g(u) = \sigma_u$ for all $u \in P$. Thus, we seek, for each $S \in \mathcal{Y}$, a function $f \in \mathcal{G}(V(N))$ that assigns $f(\rho_N) = S$ and minimizes $w^*(f)$.

We now construct a forest covering $V(N)$, as follows: Let F be the network derived from N by deleting every out-edge of each node in P. Observe that F is an out-forest (indeed, every node has in-degree at most 1 as any reticulation node in N has at least one of its parents in P). Let $T_1, \ldots, T_r$ be the trees of this forest cover, and let $\rho_1, \ldots, \rho_r$ be the respective roots of these trees. (Note that some trees may consist of a single node, if a node in P has all its parents in P.) Without loss of generality, assume that T_1 is the tree containing ρ_N, and thus $\rho_1 = \rho_N$. Observe that for any $v \in V(T_i)$, any parent of v is in $V(T_i) \cup P$, for each $i \in [r]$. Therefore for any function $g \in \mathcal{G}(V(T_i))$, $w_g(v)$ is well-defined (that is, $w_f(v)$ as defined in Definition 2 remains well-defined even if we replace f with g). Observe that all the leaves of T_i are in $X \cup P$.

The remainder of the proof is split up into a number of claims. The purpose of the first claim is to show that we may consider each tree T_i independently. To this end, we define the value $w^*_i(g)$ for any $i \in [r]$ and any function $g \in \mathcal{G}(V(T_i))$, as follows:

$$w^*_i(g) = \sum_{v \in V(T_i)} w_g(v) + \sum_{v \in V(T_i) \cap X} c^*_v(g(v))$$

Claim 3.

$$\min_{\substack{f \in \mathcal{G}(V(N)) \\ f(\rho_N) = S}} w^*(f) =$$

$$\min_{\substack{g \in \mathcal{G}(V(T_1)) \\ g(\rho_1) = S}} w^*_1(g) + \sum_{i \in [r] \setminus \{1\}} \left(\min_{g \in \mathcal{G}(V(T_i))} w^*_i(g) \right).$$

Proof. For each $i \in [r]$ and any $f \in \mathcal{G}(V(N))$, let f_i be f restricted to $V(T_i) \cup P$. Then f_i is in $\mathcal{G}(V(T_i))$, and for any $v \in V(T_i))$ we have $w_{f_i}(v) = w_f(v)$. Therefore:

$$
\begin{aligned}
w^*(f) &= w(f) + \sum_{x \in X} c^*_x(f(x)) \\
&= \sum_{v \in V(N)} w_f(v) + \sum_{v \in X} c^*_v(f(v)) \\
&= \sum_{i \in [r]} \Big(\sum_{v \in V(T_i)} w_f(v) + \sum_{v \in V(T_i) \cap X} c^*_v(f(v)) \Big) \\
&= \sum_{i \in [r]} \Big(\sum_{v \in V(T_i)} w_{f_i}(v) + \sum_{v \in V(T_i) \cap X} c^*_v(f_i(v)) \Big) \\
&= \sum_{i \in [r]} (w^*_i(f_i)))
\end{aligned}
$$

In particular, when f' is the lineage function in $\mathcal{G}(V(N))$ with $f'(\rho_N) = S$ minimizing $w^*(f')$, we have

$$
\begin{aligned}
&\min_{\substack{f \in \mathcal{G}(V(N)) \\ f(\rho_N) = S}} w^*(f) = w^*(f') = \sum_{i \in [r]} (w^*_i(f'_i))) \\
&\geq \min_{\substack{g \in \mathcal{G}(V(T_1)) \\ g(\rho_1) = S}} w^*_1(g) + \sum_{i \in [r] \setminus \{1\}} \Big(\min_{g \in \mathcal{G}(V(T_i))} w^*_i(g) \Big).
\end{aligned}
$$

On the other hand, for each $i \in [r]$ let f_i be the lineage function $g \in \mathcal{G}(V(T_i))$ minimizing $w^*_i(g)$, with the additional requirement that $f_1(\rho_1) = S$. Then for any $i \neq j$, any u in the domain of both f_i and f_j is in P, and therefore $f_i(u) = \sigma_u = f_j(u)$. Therefore there exists a function $f' \in \mathcal{G}(V(N))$ which is the union of all f_i, that is, f_i is f' restricted to $V(T_i) \cup P$ for each $i \in [r]$. Thus

$$
\begin{aligned}
&\min_{\substack{g \in \mathcal{G}(V(T_1)) \\ g(\rho_1) = S}} w^*_1(g) + \sum_{i \in [r] \setminus \{1\}} \Big(\min_{g \in \mathcal{G}(V(T_i))} w^*_i(g) \Big) \\
&= \sum_{i \in [r]} (w^*_i(f_i))) = w^*(f') \geq \min_{\substack{f \in \mathcal{G}(V(N)) \\ f(\rho_N) = S}} w^*(f).
\end{aligned}
$$

□

We may now turn our attention to finding the function $g \in \mathcal{G}(V(T_i))$ minimizing $w^*_i(g)$, for each $i \in [r]$.

For any edge uv in T_i, and any $g \in \mathcal{G}(V')$ where $\{u, v\} \subseteq V'$, the value of $w_g(v)$ depends only on the values $g(u)$ and $g(v)$ (since any other parent of v is in P and therefore the value assigned to it by g is already fixed). Therefore, for any $v \in V(T_i) \setminus \rho_i$ with parent u and any $S, S' \in \mathcal{Y}$, we may define the constant $\beta_{(v,S,S')}$ to be the value of $w_g(v)$ for any $g \in \mathcal{G}(V')$ and $\{u, v\} \subseteq V'$ such that $g(u) = S$ and $g(v) = S'$. Similarly, the value of $w_g(\rho_i)$ depends only on the value $g(\rho_i)$, and we define $\beta_{(\rho_i,S,)}$ to be the value of $w_g(\rho_i)$ for any $g \in \mathcal{G}(V')$ and $\rho_i \in V'$ such that $g(\rho_i) = S$. These constants will be of use in the analysis to follow.

For any $u \in V(T_i)$, let $D(u)$ denote the set of descendants of u in T_i (which includes u itself), and set $D'(u) = D(u) \setminus \{u\}$. Now for any $u \in V(T_i)$ and any $S \in \mathcal{Y}$, we define the value $\chi(u, S)$ as as follows:

$$
\chi(u, S) = \min_{\substack{g \in \mathcal{G}(D(u)) \\ g(u) = S}} \Big(\sum_{v \in D'(u)} w_g(v) + \sum_{v \in D(u) \cap X} c^*_v(g(v)) \Big)
$$

where the minimum value is taken to be ∞ if there is no $g \in \mathcal{G}(D(u))$ such that $g(u) = S$.

The value $\chi(u, S)$ can be calculated recursively, as we will soon show. We first show that $\chi(u, S)$ can be used to find a $g \in \mathcal{G}(V(T_i))$ minimizing $w^*_i(g)$.

Claim 4.

$$
\min_{\substack{g \in \mathcal{G}(V(T_i)) \\ g(\rho_i) = S}} w^*_i(g) = \beta_{(\rho_i,S)} + \chi(\rho_i, S)
$$

Proof.

$$
\begin{aligned}
&\min_{\substack{g \in \mathcal{G}(V(T_i)) \\ g(\rho_i) = S}} w^*_i(g) \\
&= \min_{\substack{g \in \mathcal{G}(V(T_i)) \\ g(\rho_i) = S}} \Big(\sum_{v \in V(T_i)} w_g(v) + \sum_{v \in V(T_i) \cap X} c^*_v(g(v)) \Big) \\
&= \min_{\substack{g \in \mathcal{G}(V(T_i)) \\ g(\rho_i) = S}} \Big(w_g(\rho_i) + \sum_{v \in D'(\rho_i)} w_g(v) + \sum_{v \in D(\rho_i) \cap X} c^*_v(g(v)) \Big) \\
&= \beta_{(\rho_i,S)} + \min_{\substack{g \in \mathcal{G}(D(\rho_i)) \\ g(\rho_i) = S}} \Big(\sum_{v \in D'(\rho_i)} w_g(v) + \sum_{v \in D(\rho_i) \cap X} c^*_v(g(v)) \Big) \\
&= \beta_{(\rho_i,S)} + \chi(\rho_i, S)
\end{aligned}
$$

□

The next claim immediately follows from Claim 4:

Claim 5.

$$\min_{g\in\mathcal{G}(V(T_i))} w_i^*(g) = \min_{S\in\mathcal{Y}}(\beta_{(\rho_i,S)} + \chi(\rho_i, S))$$

The next three claims give us a recursive structure for calculating $\chi(u, S)$. The first one follows from the definitions and needs no proof.

Claim 6. *If $v \in L(T_i)$, then $\chi(v, S) = c_v^*(S)$ if $v \in X$, and if $v \in P$ then $\chi(v, S) = 0$ if $S = \sigma_v$ and $\chi(v, S) = \infty$ otherwise.*

Claim 7. *If $u \in T_i$ has a single child v in T_i, then*

$$\chi(u, S) = \min_{S'\in\mathcal{Y}}(\beta_{(v,S,S')} + \chi(v, S')).$$

Proof.

$$\begin{aligned}
&\chi(u, S)\\
&= \min_{\substack{g\in\mathcal{G}(D(u))\\ g(u)=S}}(\sum_{v'\in D'(u)} w_g(v') + \sum_{v'\in D(u)\cap X} c_{v'}^*(g(v')))\\
&= \min_{\substack{S'\in\mathcal{Y}\\ g\in\mathcal{G}(D(u))\\ g(u)=S\\ g(v)=S'}} (w_g(v) + \sum_{v'\in D'(v)} w_g(v') + \sum_{v'\in D(v)\cap X} c_{v'}^*(g(v')))\\
&= \min_{S'\in\mathcal{Y}}(\beta_{(v,S,S')} +\\
&\min_{\substack{g'\in\mathcal{G}(D(v))\\ g'(v)=S'}}(\sum_{v'\in D'(v)} w_{g'}(v') + \sum_{v'\in D(v)\cap X} c_{v'}^*(g'(v'))))\\
&= \min_{S'\in\mathcal{Y}}(\beta_{(v,S,S')} + \chi(v, S'))
\end{aligned}$$

□

Claim 8. *If $u \in T_i$ has two children v_1, v_2 in T_i, then $\chi(u, S) = \min_{S_1\in\mathcal{Y}}(\beta_{(v_1,S,S_1)}+\chi(v_1, S_1))+\min_{S_2\in\mathcal{Y}}(\beta_{(v_2,S,S_2)}+\chi(v_2, S_2))$.*

Proof.

$$\begin{aligned}
&\chi(u, S)\\
&= \min_{\substack{g\in\mathcal{G}(D(u))\\ g(u)=S}}(\sum_{v\in D'(u)} w_g(v) + \sum_{v\in D(u)\cap X} c_v^*(g(v)))\\
&= \min_{S_1,S_2\in\mathcal{Y}}(\min_{\substack{g\in\mathcal{G}(D(u))\\ g(u)=S\\ g(v_1)=S_1\\ g(v_2)=S_2}}(w_g(v_1) + \sum_{v\in D'(v_1)} w_g(v)\\
&+ \sum_{v\in D(v_1)\cap X} c_v^*(g(v)) + w_g(v_2) + \sum_{v\in D'(v_2)} w_g(v)\\
&+ \sum_{v\in D(v_2)\cap X} c_v^*(g(v))))\\
&= \min_{S_1,S_2\in\mathcal{Y}}(\beta_{(v_1,S,S_1)} + \min_{\substack{g\in\mathcal{G}(D(v_1))\\ g(v_1)=S_1}}(\sum_{v\in D'(v_1)} w_g(v)+\\
&\sum_{v\in D(v_1)\cap X} c_v^*(g(v))) + \beta_{(v_2,S,S_2)}+\\
&\min_{\substack{g\in\mathcal{G}(D(v_2))\\ g(v_2)=S_2}}(\sum_{v\in D'(v_2)} w_g(v) + \sum_{v\in D(v_2)\cap X} c_v^*(g(v))))\\
&= \min_{S_1,S_2\in\mathcal{Y}}(\beta_{(v_1,S,S_1)} + \chi(v_1, S_1) + \beta_{(v_2,S,S_2)} + \chi(v_2, S_2))\\
&= \min_{S_1\in\mathcal{Y}}(\beta_{(v_1,S,S_1)} + \chi(v_1, S_1))\\
&+ \min_{S_2\in\mathcal{Y}}(\beta_{(v_2,S,S_2)} + \chi(v_2, S_2))
\end{aligned}$$

□

Claim 9. *In $O(\mu^2|V(T_i)|)$ time, we can construct a table H with entries $H[u, S]$ for each $u \in V(T_i), S \in \mathcal{Y}$, such that $H[u, S] = \chi(u, S)$ for each choice of u and S.*

Proof. We calculate the entries of H in a bottom-up order. That is, we only calculate entries of the form $H[u, S]$ for some $u \in V(T_i)$ after we have calculated all entries of the form $H[v, S']$ for each child v of u.

If u is a leaf, then following Claim 6, we may set $H[u, S] = c^*_u(S)$ if $u \in X$, and if $u \in P_1$ then we may set $H[u, S] = 0$ if $S = \sigma_v$ and $H[u, S] = \infty$ otherwise. This takes $O(1)$ time for each choice of S.

If u has a single child v, then following Claim 7, we may set $H[u, S] = \min_{S' \in \mathcal{Y}}(\beta_{(v,S,S')} + H[v, S'])$. Assuming all entries of the form $H[v, S]$ have been calculated, this takes $O(|\mathcal{Y}|) = O(\mu)$ time.

If u has two children v_1, v_2, then following Claim 8, we may set $H[u, S] = \min_{S_1 \in \mathcal{Y}}(\beta_{(v_1,S,S_1)} + H[v_1, S_1]) + \min_{S_2 \in \mathcal{Y}}(\beta_{(v_2,S,S_2)} + H[v_2, S_2])$. Assuming all entries of the form $H[v_1, S_1]$ and $H[v_2, S_2]$ have been calculated, this again takes $O(\mu)$ time.

Thus, the construction of each individual entry in H takes $O(\mu)$ time. As there are $|V(T_i)| \cdot |\mathcal{Y}|$ such entries, the construction of H takes $O(\mu^2|V(T_i)|)$ time in total.

□

Once the table H has been constructed, then by Claim 4 we can find $\min\{w^*_1(g) : g \in \mathcal{G}(V(T_1)), g(\rho_1) = S\}$ by calculating $\beta_{(\rho_1,S)} + \chi(\rho_1, S) = \beta_{(\rho_1,S)} + H[\rho_1, S]$. Similarly by Claim 5, for each $i \in [r] \setminus \{1\}$ we can find $\min\{w^*_i(g) : g \in \mathcal{G}(V(T_i))\}$ in $O(\mu)$ time by calculating $\beta_{(\rho_i,S)} + \chi(\rho_i, S) = \beta_{(\rho_i,S)} + H[\rho_i, S]$ for each $S \in \mathcal{Y}$. It follows from Claim 3 that $\min_{f \in \mathcal{G}(V(N)) f(\rho_N)=S} w^*(f)$ can be calculated in time $O(\sum_{i\in[r]} \mu^2|V(T_i)|) = O(\mu^2|V(N)|)$. The algorithm can be made constructive using standard backtracking techniques.

Finally, recall that the class $\mathcal{G}(V(N))$, and therefore the function χ and table H, were defined relative to a guessed assignment f' on P, and that there were $O(\mu^k)$ possible guesses for this assignment. So let $H_{f'}$ denote the table H constructed for a particular f'. To calculate $I[S]$, we need to take the minimum value of $H_{f'}[\rho_N, S]$ over all assignments f' on P. Therefore, the total time taken to construct I is $O(\mu^k \cdot \mu^2|V(N)| + \mu \cdot \mu^k) = O(\mu^{k+2}|V(N)|)$. □

As observed above, by setting $c^*_x(S) = 0$ if $S = \alpha(x)$, and $c^*_x(S) = \infty$ otherwise, and by trying every value of S in $\mathcal{Y}$ with $|S| = 1$, Lemma 9 gives us a $O(\mu^{k+2} \cdot |V(N)| + p) = O((2^p)^{k+2} \cdot |V(N)|) = O(2^{(k+2)p} \cdot |V(N)|)$ time algorithm for calculating $PS_{pt}(N, \alpha)$, thus proving Theorem 3.

A.4 Fixed-parameter tractability with respect to level – missing proofs

Recall the definition of a *blob* of a network N (a maximal subgraph for which the underlying undirected graph is biconnected), and that l denotes the *level* of N, i.e. the maximum number of reticulation nodes in any blob of N. In this section, we prove Theorem 4:

Theorem 4. *PPP is fixed-parameter tractable with respect to l and p.*

For each blob B of N, let ρ_B denote the unique root of B. Let R denote the set of nodes in N containing ρ_N together with ρ_B for every non-trivial blob B in N.

For each $\rho \in R$, let $\gamma(\rho)$ denote the subset of $V(N)$ consisting of ρ and all its descendants. Let $\beta(\rho)$ denote the set of nodes whose lowest non-trivial ancestor in R is ρ, together with ρ itself. Thus, the leafs of $N[\beta(\rho)]$ are in $X \cup R$, and the root of $N[\beta(\rho)]$ is ρ. Furthermore, $\beta(\rho)$ contains the non-root nodes of at most 1 non-trivial blob, and therefore at most l reticulation nodes. Let $L(\beta(\rho))$ be shorthand for the leaves of $N[\beta(\rho)]$.

For any set $U \subseteq V(N)$, let $\mathcal{F}(U)$ be the set of α-consistent lineage functions on U. Now for any $u \in X \cup T$, we may define the function $c^*_u : \mathcal{Y} \to \mathbb{N}_0 \cup \{\infty\}$ as follows:

$$c^*_u(S) = \min_{\substack{f \in \mathcal{F}(\gamma(u)) \\ f(u)=S}} \Big(\sum_{v \in \gamma(u) \setminus \{u\}} w_f(v) \Big)$$

where $c^*_u(S) = \infty$ if there is no f satisfying the conditions of the minimum.

We will show how to recursively calculate c^*_u shortly. We first note that $c^*_{\rho_N}$ can be used to find a minimum weight rooted α-consistent lineage function on N. The following claim follows from the definitions and needs no proof.

Claim 10. *For any $S \in \mathcal{Y}$,*

$$\min_{\substack{f \in \mathcal{F}(V(N)) \\ f(\rho_N)=S}} w(f) = c^*_{\rho_N}(S).$$

The next two claims show how to calculate c^*_u recursively. The first claim again follows from the definitions and needs no proof.

Claim 11. *For any $x \in X$ and $S \in \mathcal{Y}$, $c^*_x(S) = 0$ if $S = \alpha(x)$, and $c^*_x(S) = \infty$ otherwise.*

Claim 12. *For any $\rho \in R$ and $S \in \mathcal{Y}$,*

$$c^*_\rho(S) = \min_{\substack{f \in \mathcal{F}(\beta(\rho)) \\ f(\rho)=S}} \Big(\sum_{v \in \beta(\rho) \setminus \{\rho\}} w_f(v) + \sum_{v \in L(\beta(\rho))} c^*_v(f(v)) \Big).$$

Proof. Let $\rho_1, \ldots \rho_r$ be an arbitrary ordering of $L(\beta(\rho))$. Observe that $\gamma(\rho) = \beta(\rho) \cup \bigcup_{i\in[r]}(\gamma(\rho_i) \setminus \{\rho_i\})$, and that this is a disjoint union. (In the cases where $\rho_i \in X$, the set $\gamma(\rho_i) \setminus \{\rho_i\}$ is empty; we really only care about ρ_i when $\rho_i \in R$, but for the purposes of our proofs it is simpler to consider all elements of $L(\beta(\rho))$ together.)

Then we have the following (where as usual, if there is no function satisfying some set of conditions then the minimum value of functions satisfying those conditions is taken to be ∞):

$$
\begin{aligned}
c^*_\rho(S) &= \min_{\substack{f\in\mathcal{F}(\gamma(\rho))\\ f(\rho)=S}} \Big(\sum_{v\in\gamma(\rho)\setminus\{\rho\}} w_f(v)\Big)\\
&= \min_{\substack{S_1,\dots S_r\in\mathcal{Y}\\ f\in\mathcal{F}(\gamma(\rho))\\ f(\rho)=S\\ f(\rho_1)=S_1\\ \dots\\ f(\rho_r)=S_r}} \Big(\sum_{v\in\beta(\rho)\setminus\{\rho\}} w_f(v) + \sum_{i\in[r]}\sum_{v\in\gamma(\rho_i)\setminus\{\rho_i\}} w_f(v)\Big)\\
&= \min_{S_1,\dots S_r\in\mathcal{Y}} \Big(\min_{\substack{f\in\mathcal{F}(\beta(\rho))\\ f(\rho)=S\\ f(\rho_1)=S_1\\ \dots\\ f(\rho_r)=S_r}} \Big(\sum_{v\in\beta(\rho)\setminus\{\rho\}} w_f(v)\Big)\\
&\quad + \sum_{i\in[r]} \min_{\substack{f\in\mathcal{F}(\gamma(\rho_i))\\ f(\rho_i)=S_i}} \Big(\sum_{v\in\gamma(\rho_i)\setminus\{\rho_i\}} w_f(v)\Big)\Big)\\
&= \min_{S_1,\dots S_r\in\mathcal{Y}} \Big(\min_{\substack{f\in\mathcal{F}(\beta(\rho))\\ f(\rho)=S\\ f(\rho_1)=S_1\\ \dots\\ f(\rho_r)=S_r}} \Big(\sum_{v\in\beta(\rho)\setminus\{\rho\}} w_f(v)\Big) + \sum_{i\in[r]} c^*_{\rho_i}(S_i)\Big)\\
&= \min_{\substack{f\in\mathcal{F}(\beta(\rho))\\ f(\rho)=S}} \Big(\sum_{v\in\beta(\rho)\setminus\{\rho\}} w_f(v) + \sum_{v\in L(\beta(\rho))} c^*_v(f(v))\Big).
\end{aligned}
$$

□

Claim 13. *In $O(\mu^{l+3}|V(N)|)$ time, we can construct a table H with entries $H[u,S]$ for each $u \in R\cup X, S\in\mathcal{Y}$, such that $H[u,S] = c^*_u(S)$ for each choice of u and S.*

Proof. We calculate the entries of H in a bottom-up order. That is, we only calculate entries of the form $H[u,S]$ for some $u \in R\cup X$ after we have calculated all entries of the form $H[v,S']$ for each descendant v of u in $R\cup X$.

If u is a leaf of N, then $u\in X$, and following Claim 11, we may set $H[u,S]=0$ if $S=\alpha(u)$, and $H[u,S]=\infty$ otherwise.

If $u\in R$, we may assume that we already know the functions c^*_v for $v\in L(\beta(\rho))$, as we have calculated $H[v,S]$ for each $S\in\mathcal{Y}$. Then we can apply Lemma 9 on the network $N[\beta(\rho)]$ to calculate, in $O(\mu^{l+2}\cdot|\beta(u)|)$ time, a table I such that $I[S]$ gives the minimum value of $w(f)+\sum_{x\in X} c^*_x(f(x))$ over all lineage functions f on $N[\beta(u)]$ with $f(u)=S$, for each $S\in\mathcal{Y}$. By Claim 12, this is exactly $c^*_u(S)$. Then we may set $H[u,S]=I[S]$.

Thus, the construction of each individual entry $H[u,S]$ takes $O(\mu^{l+2}\cdot|\beta(u)|)$ time. In total, the construction of H therefore takes $O(\mu^{l+2}|\mathcal{Y}|\sum_{i\in[r]}|\beta(u)|) = O(\mu^{l+3}|V(N)|)$ time. □

Once the table H has been constructed, then by Claim 10 we can find $\min_{f\in\mathcal{F}(V(N))} w(f)$ by calculating $\min_{S\in\mathcal{Y}} H[\rho_N,S]$, in time $O(|\mathcal{Y}|$. Thus, the total time to find a minimum weight α-consistent lineage function on N is $O(\mu^{l+3}|V(N)|+|\mathcal{Y}|) = O(\mu^{l+3}|V(N)|) = O(2^{p(l+3)}|V(N)|)$.

A.5 Modeling ILS – missing proofs

The purpose of this section is to prove Lemma 3.

We first show that the minimum total ILS weight of a rooted α-consistent lineage function f on N is *at most* the combined ILS score of (N,α). Consider a gene tree T and coalescent history $h: V(T)\cup E(T)\to V(N)\cup Path(N)$ for which $A\cdot PS_{hw}(T,\alpha)+B\cdot XL_{T,h}(N)$ is minimized. Furthermore let $\tau: V(T)\to[p]$ be a character extending α for which $\sum_{xy\in E(T)} c_\tau(xy) = PS_{hw}(T,\alpha)$.

We first note that we may make the following assumptions about T,h,τ:

- $h(\rho_T)=\rho_N$. (Indeed, if T instead has a root r with $h(r)\neq\rho_N$, we may simply add ρ_T as the new root with child r, set $h(\rho_T)=\rho_N$, $\tau(\rho_T)=\tau(r)$, and let $h(\rho_T r)$ be any path in N from ρ_N to $h(r)$. This does not increase the combined ILS score.)
- The path $h(e)$ has length at most 1 for any edge $e\in E(T)$; in particular, an edge $xy\in E(T)$ passes through $uv\in E(N)$ if and only if $h(x)=u, h(y)=v$. (Indeed, if a path $h(xy)$ is of the form $h(x)=u_0,u_1,\dots u_l=h(y)$, $l>1$, then we may subdivide the edge xy with a series of nodes $x_1,\dots x_{l-1}$, letting $x_0=x, x_l=y$, and then set $g(x_i)=u_i$, $h(x_{i-1}x_i)=u_{i-1}u_i$ for each $i\in[l]$, and $\tau(x_i)=\tau(y)$ for each $i\in[l-1]$.)
- For all non-leaf $x\in V(T)$, there is a child y of x for which $\tau(x)=\tau(y)$. (Indeed, if this is not the case for some x, then we may choose a child y of x arbitrarily and set $\tau(x)$ to be $\tau(y)$ without affecting the combined ILS score.)

Given the above, we may define the lineage function $f: V(N)\to\mathcal{P}([p])$ as follows: $f(v)=\{\tau(x): h(x)=v\}$ for all $x\in V(T)$. We note that as $h(\rho_T)=\rho_N$, f is rooted. We note that for any $l\in X$, $\tau(x)=l$ for any x such

that $h(x) = l$ (using the fact that x is either the leaf l in T or has a child y such that $\tau(x) = \tau(y)$, and such a y also has $h(y) = l$). It follows that f is α-consistent.

We now show that $w_f \leq ILS(N, \alpha)$.

First consider the root ρ_N of N. We claim that $\max(|f(\rho_N)| - 1, 0) \leq \sum_{xy \in E(T), h(y) = \rho_N} c_\tau(xy)$. Indeed, let $i = \tau(\rho_T)$. Then $i \in f(\rho_N)$, and for any $j \in f(\rho_N) \backslash \{i\}$, there exists an edge $xy \in E(T)$ such that $h(y) = \rho_N, \tau(y) = j$ and $\tau(x) \neq j$, and hence $c_\tau(xy) = 1$. As this y must be different for each j, we have that $|f(\rho_N)| - 1 = |f(\rho_N) \backslash \{i\}| \leq \sum_{xy \in E(T), h(y) = \rho_N} c_\tau(xy)$, as required.

Now consider a non-root node $v \in V(N)$, and construct the set $\{S_u : u \in par(v)\}$ as follows: for each parent u of v, set $S_u = \{\tau(y) : xy \in E(T)$ passes through uv and $\tau(x) = \tau(y)\}$. Observe that $S_u \subseteq f(v) \cap f(u)$.

We make two claims about $\{S_u : u \in par(v)\}$. Firstly, we claim that for each $u \in par(v)$, $\max(|S_u| - 1, 0) \leq XL_{T,h}(uv)$. Indeed, if $|S_u| \leq 1$ then there is nothing to prove. Otherwise, by construction of S_u there are at least $|S_u|$ edges passing through uv and therefore $XL_{T,h}(uv) \geq |S_u| - 1$. This proves the first claim.

Our second claim is that $|f(v) \setminus (\bigcup_{u \in par(v)} S_u)| \leq \sum_{xy \in E(T), h(y) = v} c_\tau(xy)$. Indeed, for each $i \in f(v) \setminus (\bigcup_{u \in par(v)} S_u)$, there must exist $y \in V(T)$ such that $h(y) = v$ and $\tau(y) = i$. Choosing the highest such y and letting x be its parent, we note that if $h(x) = v$ then by definition $\tau(x) \neq i$ and so $c_\tau(xy) = 1$. If $h(x) \neq v$, then it must be that $h(x) = u$ for some parent u of v. But in this case xy passes through uv, and so if $\tau(x) = i$ then $i \in S_u$, a contradiction. Thus in either case we have $c_\tau(xy) = 1$ for some y such that $h(y) = v$ and $\tau(y) = i$. As this y must be different for each $i \in f(v) \setminus (\bigcup_{u \in par(v)} S_u)$, we have $|f(v) \setminus (\bigcup_{u \in par(v)} S_u)| \leq \sum_{xy \in E(T), h(y) = v} c_\tau(xy)$, as claimed.

To put everything together, for any non-root $v \in V(N)$, we have by construction that if $f(v) \neq \emptyset$ then $f(u) \neq \emptyset$ for some parent u of v. It follows that for every non-root node $v \in V(N)$, $w_f(v) \leq A \cdot |f(v) \setminus (\bigcup_{u \in par(v)} S_u)| + B \cdot \sum_{u \in par(v)} \max(|S_u| - 1, 0) \leq A \cdot \sum_{xy \in E(T), h(y) = v} c_\tau(xy) + B \cdot \sum_{u \in par(v)} XL_{T,h}(uv)$, where $\{S_u : u \in par(v)\}$ is constructed as above. On the other hand if $v = \rho_N$ then $w_f(v) = A \cdot \max(|f(v)| - 1, 0) \leq A \cdot \sum_{xy \in E(T), h(y) = \rho_N} c_\tau(xy)$.

Therefore we have

$$
\begin{aligned}
w_f &= \sum_{v \in V(N)} w_f(v) \\
&\leq A \cdot \sum_{xy \in E(T), h(y) = \rho_N} c_\tau(xy) \\
&+ \sum_{v \in V(N) \backslash \rho_N} A \cdot \sum_{xy \in E(T), h(y) = v} c_\tau(xy) \\
&+ \sum_{v \in V(N) \backslash \rho_N} B \cdot \sum_{u \in par(v)} XL_{T,h}(uv) \\
&= A \cdot \sum_{xy \in E(T)} c_\tau(xy) + B \cdot \sum_{uv \in E(N)} XL_{T,h}(uv) \\
&= A \cdot PS_{hw}(T, \alpha) + B \cdot XL_{T,h}(N) \\
&= ILS(N, \alpha).
\end{aligned}
$$

This concludes the proof that the minimum total ILS weight of a rooted α-consistent lineage function f on N is at most the combined ILS score of (N, α). We now show that the combined ILS score of (N, α) is at most the minimum total ILS weight of a rooted α-consistent lineage function f on N.

Let f be a rooted α-consistent lineage function on N with minimum total ILS weight, and assume without loss of generality that $w_f < \infty$. We construct a gene tree T, coalescent history $h : V(T) \cup E(T) \to V(N) \cup Path(N)$ and character $\tau : V(T) \to [p]$ as follows.

For each $v \in V(N)$ and $i \in f(v)$, add a node $x_{v,i}$ to $V(T)$ with $h(x_{v,i}) = v$ and $\tau(x_{v,i}) = i$. To add the remaining nodes and edges to the tree T under construction, we will process each non-root node of N one at a time.

First consider $v = \rho_N$. If $|f(v)| = 1$, then we add no edges or nodes and the unique node $x_{\rho_N, i}$ will be the root of T. Otherwise, we "add to T" a random binary tree T' whose leaves are the nodes $x_{\rho_N, i}$ for each $i \in f(\rho_N)$. For each internal node x of this tree (including the root), set $h(x) = \rho_N$ and $\tau(x) = j$ for the minimum $j \in f(v)$. Observe that $\sum_{xy \in E(T')} \{c_\tau(xy) : h(y) = \rho_N\} = \max(|f(\rho_N)| - 1, 0)$. As ρ_N has no parents in N, it follows that $A \cdot \sum_{xy \in E(T')} \{c_\tau(xy) : h(y) = \rho_N\} + B \cdot \sum_{u \in par(\rho_N)} XL_{T,h}(u\rho_N) = A \cdot \max(|f(\rho_N)| - 1, 0) = w_f(\rho_N)$.

Now for a non-root node v in N with $f(v) \neq \emptyset$, let $\{S_u : u \in par(v)\}$ be the set such that $S(u) \subseteq f(v) \cap f(u)$ for all $u \in par(v)$, minimizing

$$A \cdot |f(v) \setminus (\bigcup_{u \in par(v)} S_u)| + B \cdot \sum_{u \in par(v)} \max(|S_u| - 1, 0).$$

If $S_u = \emptyset$ for all $u \in par(v)$ (which only happens if $f(v) \cap f(u) = \emptyset$ for all $u \in par(v)$), then choose an arbitrary $u \in par(v)$ for which $f(u) \neq \emptyset$ (which must exist as $w_f < \infty$). Choose an arbitrary $i \in f(u)$. Now add a tree T' whose root (of out-degree 1) is $x_{u,i}$ and whose leaves are all nodes of the form $x_{v,j}$ for $j \in f(v)$, with all internal nodes x having $\tau(x) = j$ for the minimum $j \in f(v)$. For the edge e between $x_{u,i}$ and its child, set $h(e) = uv$, and for all other edges e in this tree let $h(e)$ be the trival path consisting of v. Observe that in this case $\sum_{xy \in E(T')} \{c_\tau(xy) : h(y) = v\} = |f(v)|$, and $XL_{T,h}(uv) = 0$ for all $u \in par(v)$. It follows that $A \cdot \sum_{xy \in E(T')} \{c_\tau(xy) : h(y) = v\} + B \cdot \sum_{u \in par(v)} XL_{T,h}(uv) = A \cdot |f(v)| = A \cdot |f(v) \setminus (\bigcup_{u \in par(v)} S_u)| + B \cdot \sum_{u \in par(v)} \max(|S_u| - 1, 0) = w_f(v)$.

So now suppose that $S_u \neq \emptyset$ for some $u \in par(v)$. Then, do the following for all $S_u \neq \emptyset$, $u \in par(v)$.

Choose an arbitrary $i \in f(u) \cap (\bigcup_{u \in par(v)} S_u)$, and an arbitrary $u \in par(v)$ such that $i \in f(u)$. Then add a tree T' whose root (of out-degree 1) is $x_{u,i}$ and whose leaves are $x_{v,i}$ together with all nodes of the form $x_{v,j}$ for

$j \in f(v) \setminus (\bigcup_{u \in par(v)} S_u)$. Set $h(x) = v$ and $\tau(x) = i$ for each internal node x of this tree. For the edge e between $x_{u,i}$ and its child, set $h(e) = uv$, and for all other edges e in this tree let $h(e)$ be the trival path consisting of v. For the remaining $i' \in f(u) \cap (\bigcup_{u \in par(v)} S_u)$, choose an arbitrary $u \in par(v)$ such that $i' \in f(u)$. Add an edge e between $x_{u,i'}$ and $x_{v,i'}$, setting $h(e) = uv$.

Observe that in this case $\sum_{xy \in E(T')} \{c_\tau(xy) : h(y) = v\} = |f(v) \setminus (\bigcup_{u \in par(v)} S_u)|$, and that $XL_{T,h}(uv) = \max(|S_u| - 1, 0)$ for all $u \in par(v)$ (as there are exacty $|S_u|$ edges passing through uv). It follows that $A \cdot \sum_{xy \in E(T')} \{c_\tau(xy) : h(y) = v\} + B \cdot \sum_{u \in par(v)} XL_{T,h}(uv) = A \cdot |f(v) \setminus (\bigcup_{u \in par(v)} S_u)| + B \cdot \sum_{u \in par(v)} \max(|S_u| - 1, 0) = w_f(v)$.

It follows that

$$\begin{aligned} ILS(N, \alpha) &= A \cdot PS_{hw}(T, \alpha) + B \cdot XL_{T,h}(N) \\ &= A \cdot \sum_{xy \in E(T)} c_\tau(xy) + B \cdot \sum_{uv \in E(N)} XL_{T,h}(uv) \\ &= \sum_{v \in V(N)} (A \cdot \sum \{c_\tau(xy) : h(y) = v\} \\ &+ B \cdot \sum_{u \in par(v)} XL_{T,h}(uv)) \\ &= \sum_{v \in V(N)} w_f(v) = w_f. \end{aligned}$$

This completes the proof that the combined ILS score of (N, α) is at most the minimum total ILS weight of a rooted α-consistent lineage function f on N.

A.6 Example

Consider the network N in Figure 4, and suppose we are given a character α on the leaf set of N such that $\alpha(v_5) = \{1\}, \alpha(v_7) = \{1\}, \alpha(v_8) = \{2\}, \alpha(v_9) = \{2\}$. We will show how to apply the algorithm of Section "Fixed-Parameter Tractability with respect to reticulation number" on the example (N, α). We follow the pseudocode in Algorithm 1.

As α assigns all leaves one of two values, we have that $p = 2$, and therefore set $\mathcal{Y} = \mathcal{P}([2]) \setminus \{\emptyset\} = \{\{1\}, \{2\}, \{1, 2\}\}$.

There is only one reticulation node in N, the node v_3. Thus P consists of one parent of v_3. Arbitrarily, let us say $P = \{v_4\}$.

Now consider the forest F derived from N by deleting all out-edges of P. Then F contains two trees, denoted T_1, T_2. T_1 consists of the nodes $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$ and the edges $e_2, e_3, e_4, e_5, e_6, e_7, e_8$. T_2 consists of the single node v_9. Thus we have $\rho_1 = v_1, \rho_2 = v_9$.

Now we guess a lineage assignment f' on P. There are three possibilities: $f'(v_4) = \{1\}$, $f'(v_4) = \{2\}$, or $f'(v_4) = \{1, 2\}$. We will now handle the case where $f'(v_4) = \{1\}$ in detail (other cases will be handled in a similar way).

We process each tree in F separately, beginning with T_1. We will calculate $\chi(u, S)$ for each $u \in V(T_1)$ and $S \in \mathcal{S}$, processing the nodes of $V(T_1)$ in a reverse topological order. Recall that $\chi(u, S)$ calculates the minimum cost of a lineage function f that assigns $f(u) = S$, where the cost is added up over all descendants in T_1 of u *not including u itself*. The reason for this is that we cannot calculate the cost on u until we also know the assignment on the parents of u. Moreover, the function f is required to be an extension of α and f'. We will record the values $\chi(u, S)$ in Figure A12.

For each of the leaves u in $\{v_4, v_5, v_7, v_8\}$, it easy to calculate $\chi(u, S)$ - the assignment on u is already determined (either by f' in the case of v_4, or by α in the case of v_5, v_7, v_8). Thus we may simply set $\chi(u, S) = 0$ if S is the required value, and ∞ otherwise.

The node v_6 is simple to process, as we already know the optimal assignment to each of its children, regardless of the assignment to v_6 - as discussed above, we must have $f(v_7) = \{1\}$ and $f(v_8) = \{2\}$. It follows that if $f(v_6) = \{1\}$ or $f(v_6) = \{2\}$, then the cost imposed will be 1 on one of its children, and 0 on the other, for a total of 1. On the other hand, if $f(v_6) = \{1, 2\}$ then each of its children will have an imposed cost of 0. thus we may set $\chi(v_6, \{1\}) = \chi(v_6, \{2\}) = 1$ and $\chi(v_6, \{1, 2\}) = 0$.

The node v_3 has v_6 as its only child and v_6 has no other parents, so to determine $\chi(v_3, S)$ it is enough to consider the values $\chi(v_6, S')$ for each $S' \in \mathcal{Y}$ together with the cost imposed on v_6 by assigning S to v_3 and S' to v_6. If $f(v_3) = \{1\}$, then by assigning $f(v_6) = \{1\}$ we would get a cost of 0 on v_6 and (referring to $\chi(v_6, \{1\})$) a cost of 1 on the descendants of v_6, for a total of 1; $f(v_6) = \{2\}$ would impose a total cost of 2 (1 on v_6, and 1 on its descendants); on the other hand setting $f(v_6) = \{1, 2\}$ would impose a cost of 0 on the descendants on v_6 but ∞ on v_6 itself (as we would have $|f(v_6)| > |f(v_3)|$). Thus we get that the optimum cost is 1 if $f(v_3) = \{1\}$, and set $\chi(v_3, \{1\}) = 1$. By similar arguments we get $\chi(v_3, \{2\}) = 1$, but $\chi(v_3, \{1, 2\}) = 0$ (in the last case, the optimum assignment assigns $f(v_6) = \{1, 2\}$ as well).

The node v_2 is the most complicated one to process, as its child v_3 has a parent in N other than v_2, namely the node v_4. However as $v_4 \in P$ we already know that v_4 must be assigned the value $\{1\}$. Thus if $f(v_2) = \{1\}$ the optimal cost imposed on v_3 and its descendants will be 1 - either by setting $f(v_3) = \{1\}$ for a cost of 0 on v_3 and 1 on its descendants, or by setting $f(v_3) = \{1, 2\}$ for a cost of 1 on v_3 (as $|f(v_3) \setminus (f(v_2) \cup f(v_4))| = 1$) and 0 on its descendants. On the other hand if $f(v_2) = \{2\}$ or $f(v_2) = \{1, 2\}$ then we can set $f(v_3) = \{1, 2\}$ for a total cost of 0 on v_3 and its descendants. Turning now to the child v_5, we have that v_5 must be assigned the value $\{1\}$ due to

v_4	0	∞	∞
v_5	0	∞	∞
v_7	0	∞	∞
v_8	∞	0	∞
v_6	1	1	0
v_3	1	1	0
v_2	1	1	0
v_1	1	1	0
$B_{(v_1,S)} + \chi(v_1, S)$	**1**	**1**	0

v_9	∞	0	∞
$B_{(v_9,S)} + \chi(v_9, S)$	∞	**1**	∞

FIGURE A12: Values constructed during the example application of the algorithm to the network in Figure 4 with character α such that $\alpha(v_5) = \alpha(v_7) = 1, \alpha(v_8) = \alpha(v_9) = 2$, for the case $f'(v_4) = \{1\}$. The last row in each table gives the values $B_{(\rho_i,S)} + H[\rho_i, S]$ for each tree root ρ_i and $S \in \mathcal{Y}$; the minimum of these (in bold) is the optimal score over tree T_i. Each other row represents the values $H[v, S]$ for each $v \in V(N)$ and $S \in \mathcal{Y}$ Note that v_1 may not be assigned the value $\{1, 2\}$.

v_4	∞	0	∞
v_5	0	∞	∞
v_7	0	∞	∞
v_8	∞	0	∞
v_6	1	1	0
v_3	1	1	0
v_2	0	2	0
v_1	1	1	0
$B_{(v_1,S)} + \chi(v_1, S)$	**1**	**1**	0

v_9	∞	0	∞
$B_{(v_9,S)} + \chi(v_9, S)$	∞	**0**	∞

FIGURE A13: Same as Figure A12 but for $f'(v_4) = \{2\}$.

α, and that therefore the cost imposed on v_5 is 0 if $f(v_2) = \{1\}$, 1 if $f(v_2) = \{2\}$, and 0 if $f(v_2) = \{1, 2\}$. Adding these values together, we have that the optimal cost on the descendants of v_2 is 1 if $f(v_2) = \{1\}$, 1 if $f(v_2) = \{2\}$, and 0 if $f(v_2) = \{1, 2\}$, and we may set $\chi(v_2, S)$ accordingly.

Finally, the node v_1 has two children v_2 and v_4, each of which has no other parents. We can observe that if $f(v_1) = \{1\}$, the optimal cost on v_2 and its descendants is 1 (setting $f(v_2) = 1$). If $f(v_1) = \{2\}$, the optimal cost on v_2 and its descendants is also 1, and if $f(v_1) = \{1, 2\}$, the optimal cost on v_2 and its descendants is 0. The optimal cost on v_4 is 0 if $f(v_1) = \{1\}$, 1 if $f(v_1) = \{2\}$, and 0 if $f(v_1) = \{1, 2\}$. Adding these together, we get $\chi(v_1, \{1\}) = 1, \chi(v_1, \{2\}) = 1, \chi(v_1, \{1, 2\}) = 0$.

Now that we have processed T_1, we can see that the optimal assignment is one that assigns $f(v_1) = \{1, 2\}$. However we are not allowed to assign v_1 this value, as it is the root of the network and therefore must be assigned a value S with $|S| = 1$. Thus the optimal assignment may assign either $f(v_1) = \{1\}$ or $f(v_1) = \{2\}$, both for a cost on T_1 of 1.

We still need to process T_2. However this case is much easier as there is only one node, v_9. As v_9 is a leaf of the network, we already know the value that must be assigned to it, namely $\{2\}$. thus $\chi(v_9, \{1\}) = \infty, \chi(v_9, \{2\}) = 0, \chi(v_9, \{1, 2\}) = \infty$. Our processing of T_2 is not complete, however, as we also need to count the costs imposed on v_9 by the fact that its parent v_4 has assigned value $\{1\}$. This imposes an additional cost of 1 on v_9 (as $|f(v_9) \setminus f(v_4)| = 1$), and so the total cost on T_2 is 1.

Thus, the total cost when $f'(v_4) = \{1\}$ is $1 + 1 = 2$.

In Figures 13 and 14, we process the costs with respect to the other values of f', i.e. when $f'(v_4) = \{2\}$ or $f'(v_4) = \{1, 2\}$. We see that for $f'(v_4) = \{1, 2\}$ the total cost is ∞, as the only non-infinite costs are returned when $f(v_1) = \{1, 2\}$, which is not allowed. The infinite cost essentially comes from the fact that we cannot have $|f(v_4)| = 2$ when $|f(v_1)|$ must be 1 and v_1 is the only parent in N of v_4. For the case $f'(v_4) = \{2\}$, however, we have a total cost of 1. (Such a cost occurs, for example, when $f(v_1) = f(v_2) = f(v_5) = f(v_7) = \{1\}, f(v_4) = f(v_8) = f(v_9) = \{2\}$, and $f(v_3) = f(v_6) = \{1, 2\}$). Since this is the minimum total cost over all partial assignments f', we get that the parental parsimony score of (N, α) is 1.

v_4	∞	∞	0
v_5	0	∞	∞
v_7	0	∞	∞
v_8	∞	0	∞
v_6	1	1	0
v_3	1	1	0
v_2	0	1	0
v_1	∞	∞	0
$B_{(v_1,S)} + \chi(v_1, S)$	∞	∞	0

v_9	∞	0	∞
$B_{(v_9,S)} + \chi(v_9, S)$	∞	**0**	∞

FIGURE A14: Same as Figure A12 $f'(v_4) = \{1, 2\}$.