Amalgamating Source Trees with Different Taxonomic Levels

Vincent Berry, Olaf Bininda-Emonds, Charles Semple

Amalgamating Source Trees with Different Taxonomic Levels

Vincent Berry[1], Olaf R. P. Bininda-Emonds[2], and Charles Semple[3]

22 October 2012

[1]*Méthodes et Algorithmes pour la Bioinformatique (MAB) team, Université Montpellier 2, L.I.R.M.M. - C.N.R.S., 161 rue Ada, 34095 Montpellier Cedex 5, France*
vberry@lirmm.fr

[2]*Systematics and Evolutionary Biology, Institute for Biology and Environmental Sciences (IBU), Carl von Ossietzky University Oldenburg, Carl von Ossietzky Str. 9–11, Postbox 2503, 26111 Oldenburg, Germany*
olaf.bininda@uni-oldenburg.de

[3]*Biomathematics Research Centre, Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch 8140, New Zealand*
charles.semple@canterbury.ac.nz

Keywords. supertree methods, nested taxa, Phocidae phylogeny, phylogenetics.

Abstract

Supertree methods combine a collection of source trees into a single parent tree or supertree. For almost all such methods, the terminal taxa across the source trees have to be non-nested for the output supertree to make sense. Motivated by Page, the first supertree method for combining rooted source trees where the taxa can be hierarchically nested is called AncestralBuild. In addition to taxa labeling the

leaves, this method allows the rooted source trees to have taxa labeling some of the interior nodes at a higher taxonomic level than their descendants (for example, genera versus species). However, the utility of AncestralBuild is somewhat restricted as it is mostly intended to decide if a collection of rooted source trees is compatible. If the initial collection is not compatible, then no tree is returned. To overcome this restriction, we introduce here the MultiLevelSupertree (MLS) supertree method whose input is the same as that for AncestralBuild, but which accommodates incompatibilities amongst rooted source trees using a MinCut-like procedure. We show that MLS has several desirable properties including the preservation of common subtrees amongst the source trees, the preservation of ancestral relationships whenever they are compatible, as well as running in polynomial time. Furthermore, application to a small test data set (the mammalian carnivore family Phocidae) indicates that the method correctly places nested taxa at different taxonomic levels (reflecting vertical signal), even in cases where the input trees harbor a significant level of conflict between their clades (i.e. in their horizontal signal).

Introduction

Supertrees were first described formally by Gordon (1986). However, it was not until Purvis (1995) that the full potential for supertrees to yield large, comprehensive phylogenetic hypotheses was realized. Purvis (1995) used the then recently described supertree technique Matrix Representation with Parsimony (MRP) (Baum, 1992; Ragan, 1992) to combine partial estimates of primate phylogeny from 112 papers drawn from the literature to derive one of the first, complete species-level phylogenies for the order. Since then, this literature-based application of supertree construction has been used increasingly for a wide variety of taxonomic groups (for a now outdated list, see Bininda-Emonds (2004)).

Such "traditional" supertree analyses often confront problems related to taxonomic differences between published papers. Differences can arise either because of the use of different names for the same entity (typically different species synonyms) or because the trees include terminal taxa at different taxonomic levels (e.g., trees with families versus species as terminal taxa). The former problem is comparatively trivial, with an effective solution being to standardize all taxon names according to an explicit synonymy list (see Bininda-Emonds et al. (2004)).

By contrast, combining source trees with taxa at different taxonomic levels (possibly with taxa labeling internal nodes such as those given in Figure 1) is more problematic—most existing supertree techniques are unable to deal with hierarchically-nested terminal taxa in the complete taxon set drawn across all source trees. For example, most supertree methods have no option but to place the taxa *Canis lupus*, *Canis*, and Mammalia as terminal taxa such that these three nested taxa could end up as sister taxa within a clade or, possibly, not even closely related to one another. Both solutions are nonsensical in light of taxonomic information.

The best solution to this problem to date has been to analogously standardize the taxon names to remove any instances of nested terminal taxa. This seems to be acceptable when one is standardizing names to the highest taxonomic level. Based on explicit taxonomic information, *Canis lupus* can, for instance, be easily assigned to *Canis*, which, in turn, can be assigned to Mammalia. Standardizing names to the lowest taxonomic level, as has usually been the case to derive the most inclusive supertrees possible, is

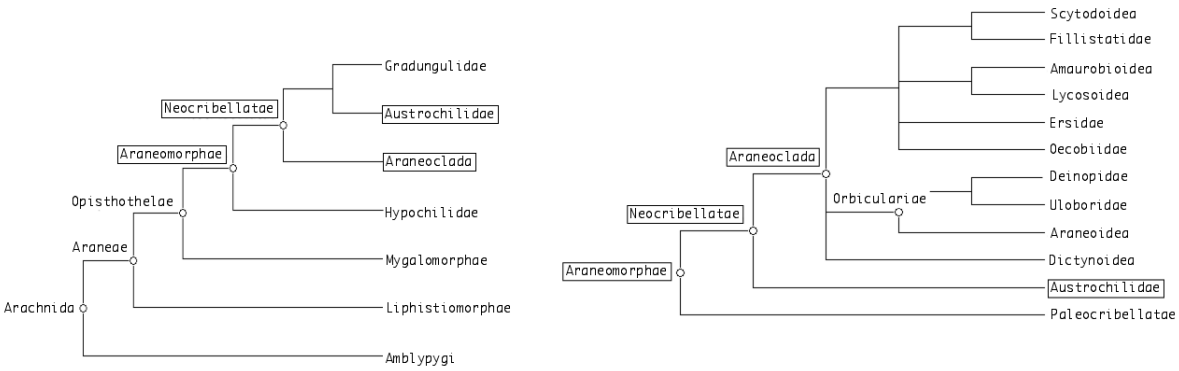

Figure 1: Two trees for spiders and related taxa. Internal nodes labeled by higher-level taxa (in comparison to the labels at the tips) are indicated with a circle, while taxa common to both trees are enclosed in boxes. These two trees, taken from Page (2004), were originally obtained from study S1x6x97c14c42c30 in TreeBASE (`http://www.treebase.org`).

more problematic. Which taxon best represents Mammalia, especially if several different mammal species are present among all source trees? Two solutions have been used, both of which have inherent limitations. The first is to represent the higher-level taxon by having all its constituent taxa form an extra, unresolved node. This solution, however, strongly presupposes the monophyly of the higher-level taxon and also includes information not potentially found in the source work (Bininda-Emonds et al., 2004). An alternative solution has been to use a single taxon in the form of the "type taxon" of the higher-level taxon (as advocated by Bininda-Emonds et al. (2004)). For instance, *Canis lupus* is the type species of the genus *Canis* and so could be used to represent it. Similarly, *Canis* is the type genus of Canidae. But, because no type taxa exist for taxa beyond the genus and family levels in botany and zoology, respectively (e.g., no type or even nominal taxon exists for Mammalia), this solution only works at the lowest taxonomic levels unless subjective decisions are made.

Inspired by problems posed by Page (2004), the supertree method AncestralBuild (Daniel and Semple, 2004; Berry and Semple, 2006) offers a more appealing solution to the problem of nested terminal taxa. AncestralBuild is unique among supertree methods because it can incorporate hierarchically-nested information in the form of internal node labels on the rooted source trees to derive the supertree. As such, the nestings are resolved based purely on information already present in the source trees and not on assumptions of the investigator. Like the Build algorithm (Aho et al., 1981) on which it is based, AncestralBuild runs in polynomial time, but can only combine (ancestrally) compatible sets of rooted source trees (i.e., ones for which a supertree exists that preserves all groupings of taxa and all ancestral relationships in the set). In the case of incompatibility amongst the source trees, AncestralBuild returns the answer "not ancestrally compatible".

In this paper, we generalize AncestralBuild to a supertree method whose input is the same as that for AncestralBuild, but which allows for incompatibilities amongst the rooted source trees. Called MultiLevelSupertree (MLS), this generalization retains many desirable and provable properties. These properties include the preservation of relationships common to all source trees, producing a supertree that is consistent with all of the source trees if the source trees are compatible, and running in polynomial time. Moreover, based on a simple empirical data set as a proof-of-concept, we show that it works at least as well as the most commonly-used supertree

method (MRP), producing trees with reasonable clades.

For the reader familiar with Build and its generalization MinCutSupertree (Semple and Steel, 2000; Page, 2002), the way in which MLS resolves topological conflicts in the source trees is reminiscent of the way in which MinCutSupertree resolves such conflicts. Thereby, unlike for most supertree methods (see Wilkinson et al. (2004)), it is also possible to document a number of desirable properties for MLS. However, in general, both MinCutSupertree and MLS will produce different supertrees as the computation used for this resolution is performed on different graphs and MLS can also potentially make use of more information. For more discussion about MinCutSupertree, see Page (2002).

The paper is organized as follows. The next section contains a high-level description of MLS and its properties, while the two sections after that together with the appendix, formally presents MLS and establishes these properties. The paper can be read independently of these latter sections and so a reader may prefer to skip these sections on a first read. The next two sections discuss the possibility of using an additional source tree to provide a taxonomic framework, and detail the implementation of MLS, which is freely available at

`http://www.atgc-montpellier.fr/supertree/mls/`.

These details include various options that are available to the end-user. The paper ends with an analysis of the application of MLS to a data set of the phocid seals, including comparisons with previous studies, and with a brief discussion.

High-Level Description of MultilevelSupertree and its Properties

The purpose of this section is to provide a high-level description of MultiLevelSupertree and its properties. The formal details including verification of these properties is given in the subsequent two sections and the appendix.

The input to MultiLevelSupertree (MLS) is a collection of rooted

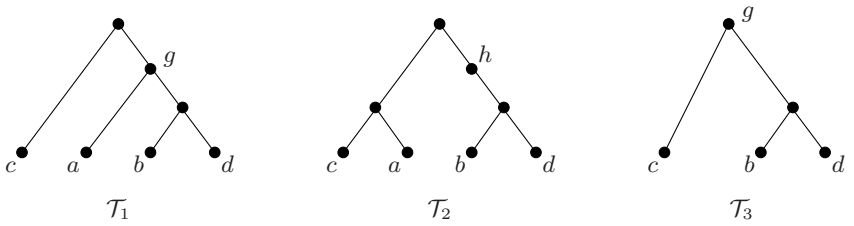

Figure 2: A collection $\mathcal{P}$ of rooted semi-labeled trees. For the purposes of a later more-detailed example, each tree has been assigned weight 1.

source trees with overlapping, but not necessarily identical taxon sets. Apart from one exception, the output is a supertree. The source trees need not be compatible (i.e., a supertree that simultaneously infers all of the ancestral relationships described by each of the source trees need not exist). Moreover, unlike traditional supertree methods, MLS allows the rooted source trees to have taxa labeling some of the interior nodes, thereby incorporating vertical (hierarchical) as well as horizontal taxon overlap. Here, a taxon labeling an interior node is at a higher taxonomic level than its descendants. To illustrate, the two rooted trees shown in Figure 1 are allowable source trees to MLS. Like the rooted source trees, the supertree returned by MLS may have some of its taxa labeling interior nodes. The one exception where a supertree is not returned by MLS is when the vertical relationships of the rooted source trees imply that there is a pair of taxa each of which is an ancestor and descendant of the other ("cyclic-descendancy").

We next give a high-level description of MLS with the help of a "toy" example. Suppose that the input to MLS is the collection of rooted source trees $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ shown in Figure 2. In an initial, preprocessing stage, MLS assigns distinct new labels to each unlabeled node in each of the source trees. In our example, the three rooted trees $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$ in Figure 3 have been obtained from $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$, respectively, through such assignments. Intuitively, these new labels act as "ancestral placeholders" and allow for the construction of a single "descendancy graph" that encodes all of the taxonomic relationships and is the next step in the preprocessing stage.

Rather than describe the descendancy graph in general, we describe its construction for our example. The nodes of the descendancy graph, which we call "label" nodes, consist of the taxa and new labels of $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$. The descendancy graph uses edges and arcs (directed edges) to represent

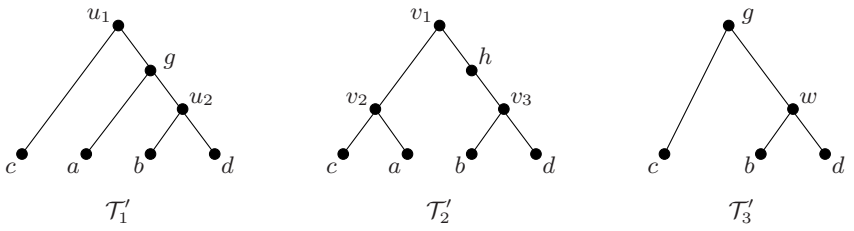

Figure 3: A collection $\mathcal{P}'$ of rooted fully-labeled trees obtained from the collection $\mathcal{P}$ shown in Figure 2 by adding distinct new labels.

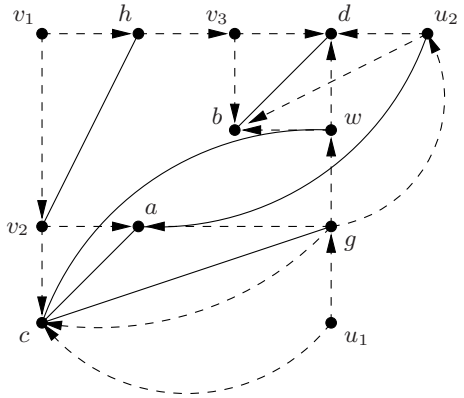

Figure 4: The descendancy graph of the collection $\mathcal{P}'$ shown in Figure 3. Arcs are shown as dashed lines with an arrow showing the direction of the arc, while edges are shown as solid lines.

horizontal and vertical taxonomic relationships, respectively. An edge joins two nodes precisely if the nodes are siblings in at least one of $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$, whereas an arc joins two nodes precisely if the tail node of the arc is the parent of the head node of the arc. Because of the placeholders, the resulting graph displays all the taxonomic relationships of $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$, and therefore of $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$. The descendancy graph for our example is shown in Figure 4. It is at this step that MLS checks for any cyclic descendancies in the form of any directed cycles in the descendancy graph. The descendancy graph in Figure 4 has no such cycles.

To complete the preprocessing stage, MLS "weights" the descendancy graph, which merely encodes topological relationships and makes no distinction whether or not these relationships are supported by one, some, or

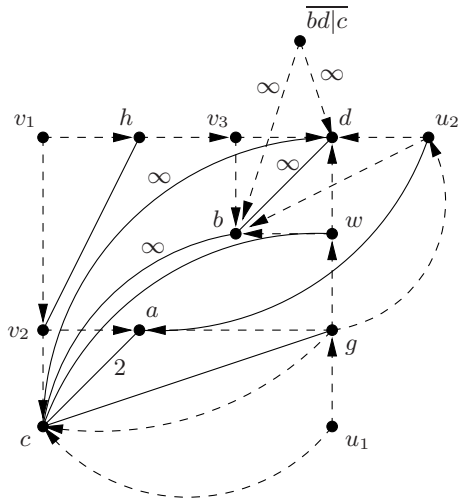

Figure 5: The weighted-descendancy graph of the collection $\mathcal{P}'$ shown in Figure 3. The node $\overline{bd|c}$ is a triple node; together with its incident arcs, it represents the relationship $bd|c$ amongst the taxa b, d, and c supported by each of the trees in $\mathcal{P}'$. Except for the arcs $(\overline{bd|c}, b)$ and $(\overline{bd|c}, d)$, and the edges $\{b, d\}$, $\{b, c\}$, $\{c, d\}$, and $\{a, c\}$, all edges and arcs have weight 1. For simplicity, these latter weights are omitted from the weighted-descendancy graph. Again, arcs are shown as dashed lines with an arrow showing the direction of the arc and edges are shown as solid lines.

all source trees. To this end, MLS weights each edge and arc with the number of trees amongst $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$ that support the non-descendant or descendant relationship represented by the edge. As an illustration, the non-descendant relationship between taxa a and c in $\mathcal{T}_2'$ is also supported by $\mathcal{T}_1'$, but not by $\mathcal{T}_3'$ where taxon a is missing. Thus, the edge joining nodes a and c in the descendancy graph is given weight 2. If all trees display a descendant relationship between two taxa, or a non-descendant relationship between two or amongst three taxa, then these relationships are given weight infinity such that they must hold in the supertree returned by MLS. The resulting "weighted-descendancy graph" of $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$ in our example is shown in Figure 5. For the reader familiar with "rooted triples", note that a new node (called a "triple" node) has been adjoined to the original descendancy graph. This node and its two incident arcs represent the fact that the relationship $bd|c$ amongst taxa b, d, and c is supported by each of the trees $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$.

Although it is not implemented in the current version of MLS, the use of the weighted-descendancy graph means that MLS can easily be extended to account for differential support between and within trees, something that has been demonstrated to be beneficial for MRP-based supertree analysis (Bininda-Emonds and Sanderson, 2001). Currently, MLS assumes that all trees (with one exception; see next section) and all nodes within those trees are equally well supported.

Once the preprocessing stage is complete, MLS calls its one and only subroutine FREE where essentially all the computation is done. Taking the weighted-descendancy graph of $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$ as input, FREE outputs a supertree that attempts, if possible, to display all the topological relationships inferred by $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$. In constructing this supertree, FREE begins at the root and recursively works its way towards the tips of the supertree. Guiding this process and paralleling this recursion, FREE recursively dismantles the inputted weighted-descendancy graph. At each step, the algorithm finds a node in the graph with no arcs directed towards it and no incident edges called a "free" node, which corresponds to the generation of a new node in the supertree. When this node is removed from the graph, the disconnected parts of the graph are analyzed separately; each one giving rise to a subtree connected to the above mentioned node in the supertree. If there are no topological conflicts amongst $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$, then FREE returns a supertree that displays all the topological relationships amongst $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$. By contrast, FREE resolves any conflict amongst the input trees using the information encoded in the weighted-descendancy graph. Such conflicts arise when the algorithm can not find a free node at some step when decomposing the graph. The process for resolving these conflicts involves finding a solution to an optimization problem (in particular, a minimum-weight cut in a graph). This process is referred to as "freeing a node" and the idea is to contradict as few as possible inter-taxa relationships as given by $\mathcal{T}_1'$, $\mathcal{T}_2'$, and $\mathcal{T}_3'$ when producing the supertree for them, in which case the resulting supertree will not display all the topological relationships among these trees. In either case, once the supertree is returned by FREE, it is stripped of its new labels, and the resulting tree is returned by MLS.

Having given a high-level description of MLS, we end this section with a high-level description of some of its properties (proofs can be found in the Appendix).

(i) If there are no topological conflicts amongst the initial collection of

rooted source trees, then MLS returns a supertree whose inter-taxa relationships are consistent with each of the input trees (i.e. no inter-taxa relationship inferred by an input tree conflicts with any relationship inferred by the supertree).

(ii) If there is a subset of taxa common to each of the input trees and this common subset induces the same inter-taxa relationships in each of the input trees, then MLS applied to this input returns a supertree that preserves these particular inter-taxa relationships.

(iii) MLS runs in time that is polynomial in the number of input trees and the total number of taxa amongst the input trees.

Formal Description of MultiLevelSupertree

In this section, we formally present MultiLevelSupertree (MLS), while the next section and the Appendix formally describes and verifies its properties. Together with the Appendix, this and the next section may be skipped on a first reading if the reader is satisfied with the high-level description of MLS given in the previous section and prefers to read the implementation and application to data set sections.

Much of the notation and terminology replicates that which can be found in Berry and Semple (2006) or Semple and Steel (2003). To avoid repetition, we will assume that the reader is familiar with standard graph-theoretic and phylogenetic notation and terminology.

Semi-labeled Trees

Extending the notion of a rooted phylogenetic tree, a *rooted semi-labeled tree* $\mathcal{T}$ on a taxa set X is an ordered pair $(T; \phi)$ consisting of a rooted tree T with root node ρ, and a map ϕ from X into the node set V of T such that

(i) for all non-root nodes v of degree at most two, ϕ assigns v an element of X, and

(ii) if ρ has degree zero or one, then ϕ also assigns ρ an element of X.

The taxa set X is the *label set* of $\mathcal{T}$ and is often denoted $\mathcal{L}(\mathcal{T})$. We say that $\mathcal{T}$ is *singularly labeled* if each node of T is assigned at most one taxa in X. Furthermore, $\mathcal{T}$ is *fully-labeled* if each node is assigned an element of X. To illustrate, each of $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ in Figure 2 is a rooted semi-labeled tree. Moreover, $\mathcal{L}(\mathcal{T}_1) = \{a, b, c, d, g\}$. In the context of this paper, all rooted semi-labeled trees except possibly the supertree returned by MLS are singularly labeled. The *label set* of a collection $\mathcal{P}$ of rooted semi-labeled trees is the union of the label sets of the trees in $\mathcal{P}$ and is denoted by $\mathcal{L}(\mathcal{P})$.

For a collection $\mathcal{P}$ of source trees, it is frequently the case that each of the trees in $\mathcal{P}$ are assigned a specific weight. This weighting allows one to account for some trait of the primary data such as dependence amongst data sets or to rate the source trees on the basis of some optimality score. It also allows to represent some data sets by a set of equally optimal trees instead of a single tree as happens regularly in maximum parsimony analyses. To this end, $\mathcal{P}$ is said to be *weighted* if each tree $\mathcal{T}$ in $\mathcal{P}$ has been assigned a real-valued weight $w(\mathcal{T})$.

Descendancy and Compatibility

Let $\mathcal{T}$ be a rooted semi-labeled tree, and let v be a non-root node of $\mathcal{T}$ of degree 1 or 2. The tree that is obtained from $\mathcal{T}$ by *contracting* v is the tree resulting from contracting the edge incident with v if v has degree 1 or replacing v and its two incident edges with a single edge if v has degree 2.

Let $\mathcal{T} = (T; \phi)$ be a rooted semi-labeled tree on X, and let $a, b \in X$. We say that a is a *descendant* of b (or, alternatively, b is an *ancestor* of a) if the path from $\phi(a)$ to the root of $\mathcal{T}$ includes $\phi(b)$. Symbolically, we denote this relationship by $b \leq_{\mathcal{T}} a$. Note that a is both a descendant and an ancestor of itself. If, in addition, $\phi(a) \neq \phi(b)$, then we denote the relationship by $b <_{\mathcal{T}} a$. If a is a descendant of b in $\mathcal{T}$ and $\{\phi(a), \phi(b)\}$ is an edge in T, then a is a *child* of b (or, alternatively, b is the *parent* of a). Furthermore, we say that a and b are *not comparable* if neither a is a descendant of b nor b is a descendant of a, in which case we denote this by $a||_{\mathcal{T}} b$. In the case that a and b are not comparable, the node of T that is the last common node on the paths from the root of T to $\phi(a)$ and from the root of T to $\phi(b)$ is called the *most recent common ancestor* of a and b, and is denoted by $\mathrm{mrca}_{\mathcal{T}}(a, b)$. If a is not comparable to b in $\mathcal{T}$, and a and b have the same parent, then a and b are *siblings*.

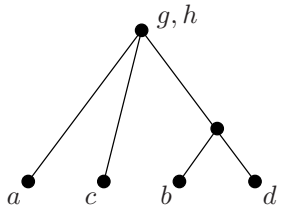

Figure 6: The rooted semi-labeled tree returned by MLS in the running example when applied to the collection $\mathcal{P}$ of trees shown in Figure 2.

Let $\mathcal{T}$ be a rooted semi-labeled tree on X and let $\mathcal{T}'$ be a rooted semi-labeled tree on X', where X is a subset of X'. We say that $\mathcal{T}'$ *ancestrally displays* $\mathcal{T}$ if, up to contracting non-root nodes of degree 2, the minimal rooted subtree of $\mathcal{T}'$ connecting the nodes assigned elements in X is a refinement of $\mathcal{T}$ and, for all $a, b \in X$, whenever a is a descendant of b in $\mathcal{T}$, it is still a descendant in the resulting subtree. Note that *refinement* means that $\mathcal{T}$ can be obtained from the minimal rooted subtree of $\mathcal{T}'$ connecting the nodes assigned elements in X by contracting edges. A collection $\mathcal{P}$ of rooted semi-labeled trees is *ancestrally compatible* if there is a rooted semi-labeled tree that ancestrally displays each of the trees in $\mathcal{P}$, in which case we say that this tree *ancestrally displays* $\mathcal{P}$. To illustrate, the rooted semi-labeled tree shown in Figure 6 ancestrally displays the tree $\mathcal{T}_3$ in Figure 2.

Mixed Graphs.

A *mixed graph* is a graph that contains both edges and arcs. For convenience, we sometimes refer to the edges and arcs as *links* when there is no need to make a distinction. The (connected) *arc components* of a mixed graph G are the maximal sub-graphs obtained from G by masking the edges of G and whereby nodes u and v are in the same component if, ignoring the directions of the arcs, there is a path from u to v. Ignoring the edges incident with u, the *in-degree* of a node u in G is the number of arcs directed into u. Let u and v be two nodes in G. We say that u and v are *edge-adjacent* (resp. *arc-adjacent*) if there exists an edge (resp. arc) joining u and v. Ignoring the direction of the arcs, a path from u to v consisting of arcs is called an *arc-path*. Additionally, if we are always moving with the direction of the arcs when traversing the path, then the arc-path is a *directed path* from u to v. A *directed cycle* is a directed path in the which the first and last nodes

are the same.

Let D be a mixed graph with node set V, arc set A, and edge set E. Let V', A', and E' be subsets of V, A, and E, respectively. The mixed graph obtained from D by deleting each of the arcs and edges in $A' \cup E'$ is denoted by $D \backslash (A' \cup E')$. Note that A' or E' could be empty. The mixed graph obtained from D by deleting each of the nodes in V' together with their incident arcs and edges is denoted by $D \backslash V'$. Furthermore, the subgraph of D whose node set is V', and whose arc and edge sets are

$$\{(c, a) : a, c \in V' \text{ and } (c, a) \in A\}$$

and

$$\{\{a, b\} : a, b \in V' \text{ and } \{a, b\} \in E\}$$

is denoted by $D|V'$.

Weighted-descendancy graph

Central to MLS is the weighted-descendancy graph, a mixed graph which, as mentioned earlier in the paper, encodes all of the relevant information given by the initial collection of source trees. Let $\mathcal{P}$ be a collection of weighted rooted semi-labeled trees with $\mathcal{L}(\mathcal{P}) = X$. We say that $\mathcal{P}'$ has been obtained from $\mathcal{P}$ by *adding distinct new labels* if we replace each tree $\mathcal{T} = (T; \phi)$ in $\mathcal{P}$ with a rooted fully-labeled tree $\mathcal{T}'$ obtained by assigning an arbitrary label not in X to each node of T not assigned a label under ϕ so that, across all trees in $\mathcal{P}$, no two added labels are the same. For example, recalling our "toy" example from earlier in the paper, the collection $\mathcal{P}' = \{\mathcal{T}_1', \mathcal{T}_2', \mathcal{T}_3'\}$ shown in Figure 3 has been obtained from the collection $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$ of rooted semi-labeled trees shown in Figure 2 by adding distinct new labels. We will continue with this example as a way of illustrating MLS.

Let $X' = \mathcal{L}(\mathcal{P}')$ and note that $X \subseteq X'$. The *descendancy graph* $D(\mathcal{P}')$ of $\mathcal{P}'$ is the mixed graph whose node set is X', and whose arc and edge sets are

$$\{(b, a) : a \text{ is a child of } b \text{ in some } \mathcal{T} \in \mathcal{P}'\}$$

and

$$\{\{a, b\} : a \text{ and } b \text{ are siblings in some } \mathcal{T} \in \mathcal{P}'\},$$

respectively. Figure 4 shows the descendancy graph corresponding to the collection $\mathcal{P}'$ of rooted fully-labeled trees shown in Figure 3. Note that the definition of the descendancy graph given here differs from that given by Berry and Semple (2006) and Daniel and Semple (2004), but coincides with the so-called *restricted descendancy graph* in Berry and Semple (2006). If $D(\mathcal{P}')$ contains a directed cycle, then, as the added new labels only appear once, the label set of $\mathcal{P}$ contains a sequence of nested taxa that is cyclic. Such cycles are due to vertical conflicts between taxa in source trees and we refer to them as *cyclic-descendancies*.

Now weight the trees in $\mathcal{P}'$ with weight function w so that the weight of each tree is the same as that of its counterpart in $\mathcal{P}$. The *weighted-descendancy graph* of $\mathcal{P}'$, denoted $D_w(\mathcal{P}')$, is the graph that is obtained from $D(\mathcal{P}')$ by assigning the weight

$$\sum_{\mathcal{T} \in \mathcal{P}';\, b <_{\mathcal{T}} a} w(\mathcal{T})$$

to each arc (b, a) and the weight

$$\sum_{\mathcal{T} \in \mathcal{P}';\, b||_{\mathcal{T}} a} w(\mathcal{T})$$

to each edge $\{a, b\}$, and then making the following modifications:

(i) If there are labels $a, b \in X$ such that $a, b \in \mathcal{L}(\mathcal{T})$ and $b <_{\mathcal{T}} a$ for all $\mathcal{T} \in \mathcal{P}$, then replace the weight of the arc (b, a) with weight ∞ if (b, a) is an arc in $D(\mathcal{P}')$; otherwise add the new arc (b, a) with weight ∞.

(ii) If there are labels $a, b \in X$ such that $a, b \in \mathcal{L}(\mathcal{T})$ and $b||_{\mathcal{T}} a$ for all $\mathcal{T} \in \mathcal{P}$, then replace the weight of the edge $\{a, b\}$ with weight ∞ if $\{a, b\}$ is an edge in $D(\mathcal{P}')$; otherwise add the new edge $\{a, b\}$ with weight ∞.

(iii) If there are labels $a, b, c \in X$ such that the rooted triple $ab|c$ is ancestrally displayed by every tree in $\mathcal{P}$, then add the new node $\overline{ab|c}$, and the new arcs $(\overline{ab|c}, a)$ and $(\overline{ab|c}, b)$ each with weight ∞.

Continuing our running example, suppose that each of the trees shown in Figure 2 has weight 1. Then the weighted-descendancy graph of the collection $\mathcal{P}'$ shown in Figure 3 is the mixed graph shown in Figure 5.

Freeing nodes

The weighted-descendancy graph illustrates relationships amongst the taxa. While MLS works on this graph, what it is effectively doing is starting at the roots of the trees in the initial collection and working towards their leaves, at the same time building the supertree from its root to its leaves. As part of this process, it wants to continually recognize nodes of the weighted-descendancy graph (or one of its subgraphs) that, in a certain sense, are not constrained by the other nodes. We call such nodes "free". At the first iteration, free nodes will correspond to the root of the supertree and, in subsequent iterations, to roots of subtrees of the supertree as the top-down process continues. Ultimately, free nodes will correspond to the leaves of the supertree.

Continuing with the notation of the previous subsection, we call a node of $D_w(\mathcal{P}')$ a *label* node if it is an element of X'; otherwise, we call it a *triple* node. A label node x of $D_w(\mathcal{P}')$ is said to be *free* if it has in-degree zero and no incident edges. For example, in Figure 5, $\overline{bd|c}$ is a triple node and, furthermore, both u_1 and v_1 are label nodes that are free.

Let A and E denote the arc and edge sets of $D_w(\mathcal{P}')$, respectively. Let A' and E' be (possibly empty) subsets of A and E, respectively. We say that $A' \cup E'$ has *finite weight* if

$$\sum_{a \in A'} w(a) + \sum_{e \in E'} w(e)$$

is finite. A label node x of in-degree zero is said to be *freed* by $A' \cup E'$ if $A' \cup E'$ has finite weight and the mixed graph obtained from $D_w(\mathcal{P}')$ by deleting each of the arcs and edges in $A' \cup E'$ has the property that each of the remaining edges incident with x joins two distinct arc components. Furthermore, a triple node $\overline{ab|c}$ of $D_w(\mathcal{P}')$ is said to be *freed* by A' if A' has finite weight and the mixed graph obtained from $D_w(\mathcal{P}')$ by deleting each of the arcs in A' has the property that c is not in the same arc component as a and b. Because of the requirement that A' has finite weight, neither of the arcs incident with $\overline{ab|c}$ in $D_w(\mathcal{P}')$ are in A' and so if A' frees $\overline{ab|c}$, then a and b are in the same arc component of $D_w(\mathcal{P}')\backslash A'$. In the upcoming description of MLS, we refer to a minimum-weight subset of $A \cup E$ that frees either a label node or a triple node as a *minimum-weight cut*. As we shall see in the next section, the reason for this definition is that the task of selecting such subsets can be viewed as finding a minimum-weight cut in a certain graph.

For simplicity, the above definitions are in terms of the weighted-descendancy graph of $\mathcal{P}'$. However, in the description of MLS, we are also interested in subgraphs of this graph. The above definitions extend to such graphs in the obvious way.

MultiLevelSupertree

We now give a formal description of MLS.

Algorithm: MLS($\mathcal{P}$)
Input: A collection $\mathcal{P}$ of weighted rooted semi-labeled trees with $\mathcal{L}(\mathcal{P}) = X$.
Output: A rooted semi-labeled tree $\mathcal{T}$ with label set X or the statement $\mathcal{P}$ *contains cyclic-descendancies*.

1. Construct a collection $\mathcal{P}'$ of weighted rooted fully-labeled trees from $\mathcal{P}$ by adding distinct new labels.

2. Construct the descendancy graph $D(\mathcal{P}')$ of $\mathcal{P}'$. If $D(\mathcal{P}')$ contains a directed cycle, then halt and return $\mathcal{P}$ *contains cyclic-descendancies*

3. Construct the weighted-descendancy graph $D_w(\mathcal{P}')$ of $\mathcal{P}'$.

4. Call the subroutine Free($D_w(\mathcal{P}')$).

5. Return the semi-labeled tree that is returned by Free($D_w(\mathcal{P}')$) with the added labels removed, and unlabeled vertices of degree 1 and unlabeled non-root vertices of degree 2 contracted.

Algorithm: Free(G_w)
Input: A subgraph G_w of $D_w(\mathcal{P}')$.
Output: A rooted fully-labeled tree $\mathcal{T}'$ with root node v'.

1. **(a)** Let $\mathcal{Q}$ denote the set of triple nodes $\overline{ab|c}$ in G_w, where a and b are nodes in G_w, but c is not a node in G_w. Reset G_w to be the graph $G_w \backslash \mathcal{Q}$.

(b) Find the node sets, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ say, of the arc components of G_w.

(c) If $k = 1$, then go to Step 2. Otherwise, for each $i \in \{1, 2, \ldots, k\}$, call Free($G_w|\mathcal{S}_i$). Return the tree whose root node is unlabeled and which has $\mathcal{T}_1', \mathcal{T}_2', \ldots, \mathcal{T}_k'$ (the trees returned by the recursive calls) as child subtrees.

2. **(a)** Let $\mathcal{S}_0$ denote the set of free nodes of G_w. If $\mathcal{S}_0$ is empty, then go to Step 3. If $\mathcal{S}_0$ comprises exactly one node labeled ℓ with out-degree zero, then return the tree composed of just one leaf labeled ℓ. Otherwise, go to Step 2b

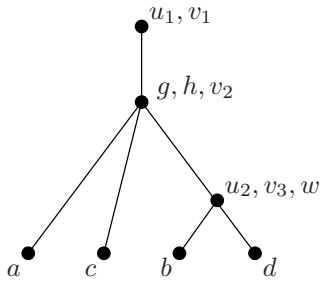

Figure 7: The rooted semi-labeled tree returned by FREE in the running example when applied to the weighted-descendancy graph shown in Figure 5.

(b) Reset G_w to be the graph $G_w \backslash \mathcal{S}_0$.

(c) Find the node sets, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ say, of the arc components of G_w.

(d) For each $i \in \{1, 2, \ldots, k\}$, call FREE$(G_w|\mathcal{S}_i)$. Return the tree whose root node is labeled by $\mathcal{S}_0$ and which has $\mathcal{T}'_1, \mathcal{T}'_2, \ldots, \mathcal{T}'_k$ (the trees returned by the recursive calls) as child subtrees.

3. (a) Let $\mathcal{S}_0$ denote the set of label nodes of G_w that can be freed with a minimum-weight cut. If $\mathcal{S}_0$ is empty, then go to Step 4. Otherwise go to Step 3b.

(b) Reset G_w to be the graph obtained from itself by deleting, for each element x in $\mathcal{S}_0$, a minimum-weight set of edges and arcs that frees x.

(c) Reset G_w to be the graph $G_w \backslash \mathcal{S}_0$. Go to Step 5.

4. (a) Let $\mathcal{S}_0$ denote the set of rooted triple nodes of G_w that can be freed with a minimum-weight cut. Select one element $\overline{ab|c}$ in $\mathcal{S}_0$.

(b) Reset G_w to be the graph obtained from itself by deleting a minimum-weight set of arcs that frees $\overline{ab|c}$. Go to Step 5.

5. (a) Find the node sets, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ say, of the arc components of G_w.

(b) For each $i \in \{1, 2, \ldots, k\}$, call FREE$(G_w|\mathcal{S}_i)$. Return the tree whose root node is labeled by $\mathcal{S}_0$ and which has $\mathcal{T}'_1, \mathcal{T}'_2, \ldots, \mathcal{T}'_k$ (the trees returned by the recursive calls) as child subtrees.

□

Before detailing some formal remarks, we illustrate MLS by applying it to the collection $\mathcal{P}$ of rooted semi-labeled trees shown in Figure 2, where each tree has weight 1. Suppose that Step 1 constructs the collection $\mathcal{P}'$ of rooted fully-labeled trees shown in Figure 3. Then Step 2 constructs the

descendancy graph $D(\mathcal{P}')$ as shown in Figure 4. As $D(\mathcal{P}')$ has no cyclic-descendancies, MLS constructs the weighted-descendancy graph $D_w(\mathcal{P}')$ as instructed in the next step and as shown in Figure 5. In the first iteration of FREE, $D_w(\mathcal{P}')$ is unchanged at the end of Step 1. At Step 2a, u_1 and v_1 are identified as the only free nodes. Deleting these nodes results in one arc component and FREE is called in Step 2d just for the mixed graph, G_w say, obtained from $D_w(\mathcal{P}')$ by deleting u_1 and v_1. In the second iteration, G_w is unchanged at the end of Steps 1 and 2. At Step 3, h, v_2, and g are identified as the only nodes that can be freed with a minimum-weight cut. Here the weight of such a cut is 1. For each of h and v_2, the edge $\{h, v_2\}$ is a minimum-weight cut, while for g the edge $\{g, c\}$ is a minimum-weight cut. The subroutine FREE now deletes the edges and arcs of a minimum-weight cut for each of h, v_2, and g. For the purposes of the illustration, we have chosen to delete the edges $\{h, v_2\}$ and $\{g, c\}$. This deletion together with the deletion of h, v_2, and g results in the creation of three arc components in Step 5. These components have node sets $\{a\}$, $\{c\}$, and $\{u_2, v_3, w, b, d, \overline{bd|c}\}$. Recursive calls to FREE investigate these three components separately. The subtrees returned by the three recursive calls are used as child subtrees of the root node labeled by the labels g, h, and v_2 at Step 5 in the second iteration of FREE. The tree eventually returned by FREE is shown in Figure 7, while the tree returned by MLS is shown in Figure 6.

Remarks

1. For convenience, we have implicitly assumed in the description of the algorithm that the mixed graph $D_w(\mathcal{P}')$ initially inputted to the subroutine FREE is arc connected. Allowing for $D_w(\mathcal{P}')$ to not be arc connected can be easily accommodated by calling FREE on each arc component of $D_w(\mathcal{P}')$ and then returning the tree whose maximal proper subtrees are the trees returned by each of these calls to FREE at the beginning of Step 5 in MLS.

2. To determine whether $D(\mathcal{P}')$ has a cyclic-dependancy, one has to determine whether $D(\mathcal{P}')$ has any directed cycles. It is well-known that this can be done by continually finding nodes of in-degree zero and deleting the resulting nodes. If at some stage before the null graph is reach, there is no such node, then $D(\mathcal{P}')$ has a directed cycle. On the other hand, if one can always find such a node, then there is no directed cycles and so no cyclic-descendancies.

3. MLS is well-defined, that is, it either returns a rooted semi-labeled tree

$\mathcal{T}$ with label set X or the statement $\mathcal{P}$ *contains cyclic-descendancies*. This fact is not immediately clear, as it relies the property that there is always a non-empty set of free nodes, or label or triple nodes that can be freed at each iteration of FREE, and will be established in the next section (and Appendix).

4. MLS extends the algorithm ANCESTRALBUILD. Recall from the introduction that ANCESTRALBUILD determines in polynomial-time whether or not a collection of rooted semi-labeled trees is ancestrally compatible. Ignoring the weights and rooted triple nodes of the weighted-descendancy graph, ANCESTRALBUILD can be obtained from MLS by removing the initial check for cyclic-descendancies, replacing FREE with the subroutine FREE$'$, and changing Step 5 of MLS appropriately depending on the outcome of FREE$'$:

Algorithm: FREE$'(G)$
Input: A subgraph G of $D(\mathcal{P}')$.
Output: A rooted fully-labeled tree $\mathcal{T}'$ with root node v' or the statement $\mathcal{P}$ *is not ancestrally compatible*.

1. Let $\mathcal{S}_0$ denote the set of free nodes of G. If $\mathcal{S}_0$ is empty, then halt and return $\mathcal{P}$ *is not ancestrally compatible*. If $\mathcal{S}_0$ comprises exactly one node labeled ℓ with out-degree zero, then return the tree composed of just one leaf labeled ℓ.

2. Reset G to be the graph $G \backslash \mathcal{S}_0$.

3. Find the node sets, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ say, of the arc components of G.

4. For each $i \in \{1, 2, \ldots, k\}$, call FREE$'(G|\mathcal{S}_i)$. Return the tree whose root node is labeled by $\mathcal{S}_0$ and which has $\mathcal{T}_1', \mathcal{T}_2', \ldots, \mathcal{T}_k'$ (the trees returned by the recursive calls) as child subtrees.

Note that FREE$'$ is simply FREE with Steps 1, 3, 4, and 5 removed and Step 2a modified in the case $\mathcal{S}_0$ is empty. The check for cyclic-descendancies is implicitly included in finding nodes of in-degree zero. Observe that it is the simple check of finding at least one free node at each iteration that decides whether or not $\mathcal{P}$ is ancestrally compatible.

5. Lastly, an alternative weighting scheme to quantify the weight of a cut is to count the (weighted) number of trees in $\mathcal{P}'$ whose topological signal is conflicted if one were to delete the links of the cut. The idea is to resolve a conflict situation by contradicting the smallest possible (weighted) number of source trees. However, optimizing such a cut is an NP-hard problem. For completeness, a proof of this hardness is included in the appendix.

Properties of MLS

This section establishes some theoretical properties of MLS. The first result says that MLS is well-defined; its proof is in the appendix.

Proposition 1 *Let $\mathcal{P}$ be a collection of rooted semi-labeled trees with $\mathcal{L}(\mathcal{P}) = X$. Then* MLS *applied to $\mathcal{P}$ either returns a rooted semi-labeled tree with label set X or the statement $\mathcal{P}$* contains cyclic-descendancies.

The next consideration is whether the running time of MLS is polynomial in the size of the initial collection $\mathcal{P}$ of rooted semi-labeled trees. It follows from the second remark after the description of MLS that one can decide in polynomial time whether the descendancy graph has cyclic-descendancies. Thus, as the weighted-descendancy graph is being reduced at each iteration of the subroutine FREE, the only other part of the algorithm that needs to be considered is the time it takes to find a minimum-weight cut to free a label or triple node in Steps 3 and 4 of FREE. The next two lemmas show that finding such cuts is equivalent to finding a minimum-weight cut in an associated network. Since finding the latter is well-known to be polynomial time (e.g., Hao and Orlin (1994)), it follows that MLS is polynomial time.

Let $\mathcal{P}'$ be a collection of weighted rooted fully-labeled trees, and let G_w be a subgraph of the weighted-descendancy graph of $\mathcal{P}'$ with node, arc, and edge sets V, A, and E, respectively. Since the weighted-descendancy graph initially inputed to FREE contains no directed cycles, we may assume that G_w contains no directed cycles.

We first show the above-mentioned equivalence for freeing label nodes. The equivalence for freeing triple nodes is simpler and given afterwards. Let x be a label node of G_w with in-degree zero that is not free. Thus, in G_w, there is at least one edge incident with x. Let $\{x, y_1\}, \{x, y_2\}, \ldots, \{x, y_k\}$ denote the edges of G_w incident with x, where $k \geq 1$. Let N denote the graph obtained from G_w by deleting the edges of G_w, replacing each arc (a, b) with the edge $\{a, b\}$, adding a new node x' and, for each edge $\{x, y_i\}$ in G_w, adding a new edge $\{x', y_i\}$ with weight $w(\{x, y_i\})$. Finding a minimum-weight subset of $A \cup E$ that frees x in G_w is equivalent to finding a minimum-weight cut in N that separates x and x', that is, a subset of edges in N whose

removal puts x and x' into two separate components and, amongst all such subsets, the sum of the weights of the edges is minimized. In particular, identifying each edge of N with its counterpart in G_w (either an arc or an edge depending on how it was derived) and using this set-up, we have the following lemma.

Lemma 2 *Let S be a subset of $A \cup E$. Then S is a minimum-weight subset of $A \cup E$ that frees x if and only if S is a minimum-weight cut in N that separates x and x'.*

Proof. First suppose that S is a minimum-weight subset of $A \cup E$ that frees x in G_w. Then, for each edge $\{x, y_i\}$, either

(i) x is not edge-adjacent to y_i in $G_w \backslash S$, or

(ii) x and y_i are in separate arc components in $G_w \backslash S$.

Consider $N \backslash S$. If (i) holds, then there is no edge joining x' and y_i in $N \backslash S$, while if (ii) holds, then there is no path in $N \backslash S$ from x to y_i using only edges derived from A. Combining these two implications for all y_i, we deduce that in $N \backslash S$ there is no path from x to x'. Thus the weight of a minimum cut in N that separates x and x' is at most the weight of S.

Now suppose that S is a minimum-weight cut in N that separates x and x'. Then, for each y_i, either

(I) there is no edge joining x' and y_i in $N \backslash S$, or

(II) there is no path from x to y_i in $N \backslash S$.

By combining (I) and (II) for all y_i, it follows that S is a subset of $A \cup E$ that frees x in G_w. Thus the minimum weight of such a subset is at most the weight of S. The lemma now follows. □

The equivalence for freeing triple nodes is really just a straight translation of the problem in terms of minimum-weight cuts. Let $\overline{ab|c}$ be a triple node of G_w that is not free. Now let N denote the graph obtained from G_w by deleting the edges of G_w and replacing each arc (a, b) with the edge

$\{a, b\}$. Finding a minimum-weight subset of A that frees $\overline{ab|c}$ is equivalent to finding a minimum-weight cut in N that separates $\overline{ab|c}$ and c. In particular, with the set-up as above, we have the following lemma whose proof is omitted.

Lemma 3 *Let S be a subset of A. Then S is a minimum-weight subset of A that frees $\overline{ab|c}$ if and only if S is a minimum-weight cut in N that separates $\overline{ab|c}$ and c.*

Combining the last two lemmas with the second remark after the description of MLS, we deduce that the running time of MLS applied to $\mathcal{P}$ is polynomial in $|\mathcal{P}| + |\mathcal{L}(\mathcal{P})|$. The next result makes this more precise.

Theorem 4 *Let $\mathcal{P}$ be a collection of weighted rooted semi-labeled trees. Then the running time of* MLS *applied to $\mathcal{P}$ is polynomial in $|\mathcal{P}| + |\mathcal{L}(\mathcal{P})|$.*

Proof. The most time-consuming operations depend on the size of the weighted-descendancy graph $D_w(\mathcal{P}')$ of $\mathcal{P}'$ or one of its subgraphs. Let $n = |\mathcal{L}(\mathcal{P}')|$ and note that n is polynomial in the size of $|\mathcal{P}| + |\mathcal{L}(\mathcal{P})|$. Let n and m be the number of nodes, and number of arcs and edges in $D_w(\mathcal{P}')$, respectively. Then n is the number n of label nodes plus the number of rooted triple nodes (which can be cubic in n), while $\mathsf{m} = O(n^3)$ in the worst case, as each rooted triple node contributes a constant number of arcs. Hence the size of $D_w(\mathcal{P}')$ and the time to construct $D_w(\mathcal{P}')$ is polynomial in $|\mathcal{P}| + |\mathcal{L}(\mathcal{P})|$.

Let G_w be $D_w(\mathcal{P}')$ or one of its subgraphs. Steps 3 and 4 of the FREE routine require to compute minimum-weight cuts to find label and triple nodes to free, respectively. Each running of FREE applied to G_w frees at least one node, that is then removed, so this routine runs at most $O(\mathsf{n})$ times. Each such running requires in the worst case examining each of the $O(\mathsf{n})$ nodes of G_w, and determining whether it can be freed at minimum cost. Lemma 2 states that this can be achieved by the computation of a minimum-weight cut separating two fixed nodes x and x' in a network N whose size is proportional to that of G_w. A well-known result in combinatorial optimization states that it is possible to do such a computation in the same time at that required to compute a maximum flow between x and x'. In particular, we can resort to the $O(\mathsf{m} \cdot min\{\mathsf{n}^{1/2}, \mathsf{m}^{2/3}\})$ with additional

log factors time algorithm of Goldberg and Rao (1998). Thus, the various runnings of the FREE routine require $O(\mathsf{n}^2 \cdot \mathsf{m} \cdot min\{\mathsf{n}^{1/2}, \mathsf{m}^{2/3}\})$ with additional log factors. Except for possibly the construction of $D_w(\mathcal{P}')$, this is clearly the most time consuming part of MLS. It now follows that MLS applied to $\mathcal{P}$ runs in time polynomial in $|\mathcal{P}| + |\mathcal{L}(\mathcal{P})|$. □

The last two results in this section describe properties of the tree returned by MLS. Earlier, we showed how ANCESTRALBUILD can be obtained from MLS. The next result formalizes this connection; its proof is in the appendix. One particular outcome of this result is that if a collection $\mathcal{P}$ of rooted source trees are compatible, then the supertree returned by MLS when applied to $\mathcal{P}$ is consistent with each of the trees in $\mathcal{P}$. Of course, one would always like any reconstruction method to have such a consistency property, however, there is no guarantee that this is the case.

Theorem 5 *Let $\mathcal{P}$ be a collection of rooted semi-labeled trees, and suppose that $\mathcal{P}$ is ancestrally compatible. Then* MLS *applied to $\mathcal{P}$ returns the same rooted semi-labeled tree as* ANCESTRALBUILD *applied to $\mathcal{P}$. In particular,* MLS *returns a rooted semi-labeled tree that ancestrally displays $\mathcal{P}$.*

The last result in this section requires some additional preliminaries. The analogue of this result for rooted phylogenetic trees and MINCUTSUPERTREE is established in Semple and Steel (2000). A rooted semi-labeled tree $\mathcal{T}$ is *binary* if $\mathcal{T}$ is singularly labeled and the degree of any node is at most three. Let $\mathcal{T} = (T; \phi)$ be a binary rooted semi-labeled tree and let $\{a, b, c\}$ be a subset of the label set of $\mathcal{T}$. Suppose that $a||_{\mathcal{T}}b$ and let v denote the most recent common ancestor of $\phi(a)$ and $\phi(b)$. Furthermore, suppose that $\phi^{-1}(v)$ is empty and $\phi(c) = u$, where u is the parent of v in T and has degree 2. Then the rooted semi-labeled tree that is obtained from $\mathcal{T}$ by contracting the edge $\{u, v\}$ and assigning c to the new identified node is said to be obtained by a *local contraction*.

Theorem 6 *Let $\mathcal{P}$ be a collection of rooted semi-labeled trees and let $\mathcal{T}$ be a binary rooted semi-labeled tree that is ancestrally displayed by each of the trees in $\mathcal{P}$. Then, up to local contractions,* MLS *applied to $\mathcal{P}$ returns a rooted semi-labeled tree that ancestrally displays $\mathcal{T}$.*

Proof. Let $\mathcal{T}'$ be the rooted semi-labeled tree returned by an application of MLS to $\mathcal{P}$. By a straightforward modification of the last part of the

proof of Proposition 4.3 in Daniel and Semple (2005), to prove the theorem, it suffices to show that, for all $a, b, c \in \mathcal{L}(\mathcal{T})$, the following properties are satisfied:

(i) if $c <_{\mathcal{T}} a$, then $c <_{\mathcal{T}'} a$;

(ii) if $a||_{\mathcal{T}} b$, then $a||_{\mathcal{T}'} b$; and

(iii) if $ab|c$ is a rooted triple of $\mathcal{T}$, then $ab|c$ is a rooted triple of $\mathcal{T}'$.

Because of the addition of arcs and edges with weight ∞ joining label nodes in the construction of the weighted-descendancy graph, it is clear that $\mathcal{T}'$ satisfies (i) and (ii). Furthermore, if $ab|c$ is a rooted triple of $\mathcal{T}$, then the triple node $\overline{ab|c}$ and two incident arcs with weight ∞ are added to the weighted-descendancy graph. As a result, a and b remain in the same arc component until at least one iteration beyond that in which c is in a separate arc component. This guarantee that $\mathcal{T}'$ also satisfies (iii). Thus, up to local contractions, $\mathcal{T}'$ ancestrally displays $\mathcal{T}$, completing the proof of the theorem. □

Employing a taxonomic framework

The supertree framework specifically allows the input trees to have different input taxon sets. However, when source trees share increasingly fewer taxa, most supertree methods output increasingly unresolved supertrees, reflecting the high number of possibilities according to which the taxa from the individual source trees can be interleaved. The problems associated with insufficient overlap on taxa sets of source trees is a well-known phenomenon in the supertree literature (Bininda-Emonds, 2004).

Given that MLS is a supertree method, it is not immune to overlap problems. However, because the method can deal with taxa at different taxonomic levels, two different kinds of overlap become relevant: horizontal overlap between terminal taxa as in traditional supertree studies, and vertical overlap between taxa at different taxonomic levels. Although MLS presents an appealing solution to the problem of combining source trees with hierarchically-nested taxa because it uses only the information present in the

source trees themselves (rather than synonymizing taxon names), phylogenies in the literature generally lack internal node labels. In other words, sufficient vertical overlap is often missing from real-life data sets, which, as for horizontal overlap, can similarly lead to meaningless or artifactual results. In the absence of such necessary vertical information, MLS, like other supertree methods, will be unable to decipher the hierarchical relationships of the various taxa and can place otherwise nested taxa, such as *Canis* and Mammalia, as sister taxa.

To increase the horizontal overlap in a supertree study, it often suffices to add source trees that make a bridge between taxon sets of different source trees (e.g., a complete, but highly down-weighted, seed tree; (Bininda-Emonds and Sanderson, 2001)). Similarly, the lack of vertical overlap in a multi-level analysis can be filled by adding additional source trees containing taxa at different levels, thus making the nested relationships that exist between several taxa explicit. Therefore, as suggested by Berry and Semple (2006), it will often be advisable to include a reference taxonomy—or *backbone* tree—specifying the nesting information among all the taxa and labels in the set of source trees as an additional source tree. This reference taxonomy will ideally be composed mainly of poorly resolved clades and comparatively down-weighted in the analysis so as to merely guide it, rather than influence it unduly. In this way, it fulfills the same role as the seed tree advocated by Bininda-Emonds and Sanderson (2001) for conventional supertree analyses. Alternatively, the user can complement the set of source trees with several smaller trees having taxa at both internal nodes and leaves. In fact, such trees can be as small as the tree allowed by the Newick format, namely containing just two taxa one on top of the other (see the implementation documentation for details).

Implementation

The algorithm MLS has been implemented in Java using part of the source code of the SplitsTree v4.6 package (Huson and Bryant, 2006)—the latter is not required to run MLS—using the Mascopt library to compute minimum cuts (Lalande et al., 2004). The implementation is freely available at `www.atgc-montpellier.fr/supertree/mls`. In this section, we discuss several aspects of this implementation. Throughout the section, $\mathcal{P}$ refers to the initial collection of weighted source trees, whereas $\mathcal{P}'$ is a collection of

weighted rooted fully-labeled trees obtained from $\mathcal{P}$ by adding distinct new labels. Furthermore, we collectively refer to arcs and edges as "links" and we often identify the nodes of the trees in $\mathcal{P}'$ with their label.

Relative importance of descendancy and sibling links

Each source tree gives rise to arcs and edges in the weighted-descendancy graph $D_w(\mathcal{P}')$ of $\mathcal{P}'$ to express topological constraints it induces on its taxa. However, it might be preferable in some data sets to give more weight to arcs, which express node descendancies in source trees, than to edges, which express non-comparability of sibling nodes in source trees. This differential weighting only becomes relevant methodologically when $D_w(\mathcal{P}')$ or one of its "subgraphs" has no free nodes due to conflicts amongst source trees. In such cases where a minimum-weight cut in the graph has to be made, one might prefer favoring the removal of edges (horizontal signal) over arcs (vertical signal). This is particularly relevant when the vertical relationships are either held to be more accurate or more important than the potentially conflicting horizontal signals. The program has a specific option allowing more weight to be given to the arcs contributed by an individual source tree to $D_w(\mathcal{P}')$ than to the edges contributed by the same source tree; otherwise, they both receive the same weight by default.

Using transitive arcs

The vertical relationships of the trees in $\mathcal{P}'$ can be encoded in the descendancy graph $D(\mathcal{P}')$ of $\mathcal{P}'$ either by encoding only direct arcs between nodes or, alternatively, by encoding both direct and indirect arcs. For instance, if, in some source tree, c is the parent of b and b is the parent of a, then the two arcs (c, b) and (b, a) are added in the construction of $D(\mathcal{P}')$. However, since c is an ancestor of a, one might wonder why the "transitive" arc (c, a) is not added to $D(\mathcal{P}')$.

We believe that encoding only direct arcs is preferable for two reasons. First, this impedes large trees from exerting a greater influence on the resulting supertree. Indeed, if n is the number of taxa in a tree in $\mathcal{P}'$, there are only $O(n)$ direct arcs, but up to $O(n^2)$ indirect ones. Secondly, and more practically, the running time is proportional to the number of links in

the graph. Thus, not encoding transitive arcs also lowers the running time. Nevertheless, we include a parameter in the implementation that switches the addition of transitive arcs on and off in constructing the descendancy graph $D(\mathcal{P}')$.

A preprocessing step to deduce internal labels

Like some supertree methods such as MinCutSupertree and its variant, modified MinCutSupertree (Semple and Steel, 2000; Page, 2002), MLS is sometimes prone towards producing comb-like trees. This is due to the iterative approach of the algorithm, which removes successive minimum-weight cuts to free labels and triple nodes in subgraphs of $D_w(\mathcal{P}')$. The significance of this phenomenon depends chiefly on the number of labels present in exactly one tree in $\mathcal{P}'$. In particular, freeing such uniquely labeled nodes usually requires the removal of only a single edge, and thereby costs little in comparison to freeing a node whose label is shared by several trees in $\mathcal{P}'$. As a result, reducing the number of additional labels in the construction of $\mathcal{P}'$ as much as possible beforehand is highly desirable.

The inclusion of a seed taxonomy as described in the last section can play an important role here because it facilitates the preprocessing of the regular source trees to deduce taxon names for some of their internal nodes, thereby reducing the number of additional labels required to construct $\mathcal{P}'$ from $\mathcal{P}$. The preprocessing considers each taxon t_i present at an internal node in the taxonomy in turn. For each t_i, it computes the set S_i of descendant taxa. Then each source tree not containing taxon t_i is examined. If there exists an unlabeled node in such trees whose set of descendant taxa is exactly S_i, then this node is assigned label t_i. Note that this preprocessing step is optional and, when selected, the user must also remember to include a taxonomy in the source-tree file as the *last* tree of the file.

Freeing nodes sequentially

If, at some step, there are no free nodes, then the subroutine Free simultaneously frees all label nodes that can be freed with a minimum-weight cut, so as not to favour any one node with respect to the others. However, when dealing with data sets containing intricate topological conflicts, this process

can result in highly unresolved nodes in the supertree. To avoid this, the user can opt for an alternative behaviour where nodes are freed sequentially one after the other. This option can, however, result in more comb-like trees.

Choosing meaningful minimum-weight cuts

When selecting a minimum-weight cut for freeing either a label or a triple node, we are free to choose any such cut. One could simply find an arbitrary minimum-weight cut in polynomial time and choose this cut or one could examine the set of all minimum-weight cuts and select that one(s) that best fit(s) some predefined criteria. However, the later selection process is problematic because there may be exponentially many such cuts to consider. Nevertheless, despite this possibility, we include the possibility for the user to ask for the best minimum-weight cut subject to the following ordered criteria as this usually provides more meaningful supertrees and, for commonly-sized data sets, still leads to acceptable running times:

1. The sum of the weights of the trees in $\mathcal{P}'$ that induce a link in the cut is minimized. This has the effect of preferring cuts whose links are supported by the fewest individual trees.
2. The number of nodes that are simultaneously freed is maximized.
3. The penalty score for non-respecting parts of source trees is minimized, where the *penalty score* of a cut is the sum of the weights of the trees in $\mathcal{P}'$ supporting the links in the cut minus the sum of weights of the trees in $\mathcal{P}'$ contradicting a link in the cut. Thus, cuts are preferentially made to those links showing the greatest conflict amongst the source trees.

If there is only one minimum-weight cut meeting the first criterion, then this cut is selected. Otherwise, all those that satisfy this criterion are compared via the second criterion and so on. If several minimum-weight cuts remain after the last criterion, then one of these cuts is chosen arbitrarily.

Shifting internal taxa to their usual place

Due to conflicts amongst source trees, MLS sometimes outputs a supertree with internal nodes having a single child. When such a node is unlabeled, it can safely be suppressed without altering the phylogenetic meaning of the supertree. On the other hand, if a single-child node has, say, the label l, then, in phylogenetic terms, all taxa in the subtree rooted at this node are representative of the taxonomic group l. Typically, this situation is usually depicted by l labeling a node with two or more children. To reach this situation from the supertree initially computed, MLS provides an option called `PhyloTree` that moves an internal label such as l towards the tips to the closest unlabeled node. This move may still result in the node labeled by l having a single child but, if it cannot be moved any further towards the tips, this implies that this single child also has a label, say l', with l' forming a subgroup of taxa within l. As we assume the program will be used mainly in a phylogenetic context, the `PhyloTree` option is switched on by default.

Application to a Data Set

In this section, we apply MLS to an empirical data set of the mammalian seal family Phocidae and compare this application with an MRP analysis. This data set is straightforward in the sense that there are relatively few conflicting signals and one expects the output to be well resolved. As such, it provides an appropriate proof-of-concept for MLS. For the analysis, a taxonomic seed tree was included and the default parameters for MLS were used (i.e., without use of transitive arcs and with all nodes freed simultaneously).

The Phocidae data set comprises a subset of the literature source trees used to build the Phocidae subtree of the carnivore supertree (Bininda-Emonds et al., 1999). The data set spans 43 taxa (20 terminal taxa and 23 higher-level taxa) collectively belonging to eight different taxonomic levels from family to subspecies. As explicit "links" between higher-level taxa are not deducible from the source trees, we also included a minimal taxonomic tree derived in part from Wozencraft (1993). Except for the taxonomic tree, all trees were weighted equally. The weighted-descendancy graph computed by MLS contains 88 nodes linked by 96 edges and 164 arcs. For the MRP analyses, all terminal taxa were synonymized to the species level using the

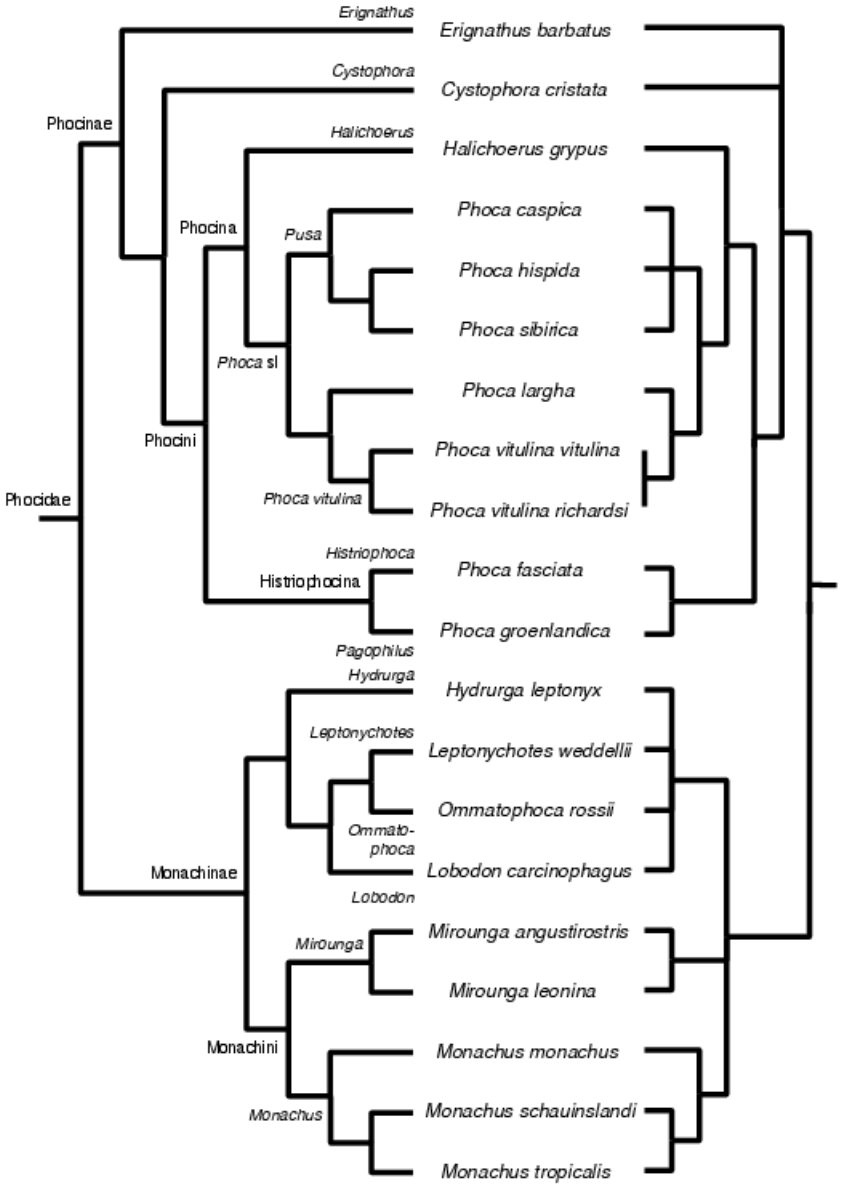

Figure 8: The resulting MLS (left) and MRP (right) supertrees for the Phocidae data set obtained from a subset of the source trees used to build the phocid supertree of the carnivore supertree (Bininda-Emonds et al., 1999)

methods outlined in the Introduction (e.g., use of type species) and an appropriate minimal taxonomic tree was also included. The MRP supertree was taken to be the strict consensus of all (16) equally most parsimonious solutions.

The resulting MLS supertree is shown in Figure 8 and is congruent with the MRP supertree obtained from the same data set. Importantly, however, the MLS supertree is both more resolved than the MRP supertree and was also obtained without having to perform any taxonomic substitutions to obtain a common taxon set. For example, the taxon *Monachus* could be entered as a terminal taxon rather than being synonymized with its type species *Monachus monachus* as was necessary for the MRP analysis. Similarly, the MLS supertree contains two subspecies of *Phoca vitulina* that were synonymized away in the MRP analysis. In so doing, the MLS supertree is able to test the hypothesis that these two subspecies do indeed form a clade, something that is not possible in the MRP supertree, where their monophyly was necessarily assumed *a priori*. Finally, the MLS supertree also helpfully retains and presents the hierarchical taxonomic information found among the set of source trees, presenting them as internal node labels.

In summary, MLS obtains a supertree for this test case that is both reasonable and also accurately reflects the relationships produced by the standard MRP supertree method. Moreover, it did so making fewer strong and occasionally subjective taxonomic assumptions while simultaneously providing more resolution and information in the end supertree. Although the data set is generally well behaved, conflict within it is still present as witnessed by the 16 equally most parsimonious solutions in the MRP analysis as well as MLS having to perform seven minimum-cut computations. The congruence between the MLS and MRP supertrees, as well as the fact that both trees reflect current opinion regarding relationships within Phocidae, would indicate that MLS is resolving these conflicts in a reasonable way. Indeed, all resolutions in the MLS supertree are found at least implicitly among the source trees and the MLS supertree is actually identical with the 50% majority-rule consensus tree for the MRP analyses. This latter fact reflects both the general sensitivity of parsimony to conflict as well as the potentially more decisive nature of MLS in cases of conflict because of its unique ability to incorporate additional information in the form of vertical taxonomic signal.

Discussion

Handling taxonomic differences between different studies, particularly that of taxa at different taxonomic levels, has long been recognized as problematic in supertree analyses. Page (2004) made explicit mention of this problem and suggested possible solutions. The first automated and practical way of dealing with it was the supertree method AncestralBuild (Berry and Semple, 2006; Daniel and Semple, 2004). However, this method returns a supertree only if the source trees are ancestrally compatible, a requirement that is frequently violated by real-world data sets. MLS overcomes this restriction on compatibility by resolving conflicting signals amongst the source trees in an optimal way using minimum-weight cuts and thus presents the first practical supertree method to tackle the important problem of heterogeneity of taxonomic levels amongst the taxa in the source trees. Moreover, MLS has several desirable properties including the preservation of common binary subtrees amongst the source trees and returning a supertree that whose inter-taxa relationships are consistent with each of the source trees if they are no topological conflicts amongst the source trees. Importantly, our analysis of a real-world data set shows that it can produce supertrees with meaningful clades.

Looking forward, MLS not only avoids tedious and subjective preprocessing tasks involving taxonomic differences amongst the source trees, but it might also be a method of choice for assembling very large trees such as those considered in 'Tree of Life' projects. Here, a large set of source trees spanning numerous taxa could be processed with a divide-and-conquer approach, in a similar but slightly different way to that proposed by Bininda-Emonds and Stamatakis (2006) as follows. First, source trees would be augmented with internal taxon labels in an automated way such as that proposed by the PhyloExplorer tool (Ranwez et al., 2009). Here, the genus, family, and other higher-level taxonomic taxa to which the internal nodes correspond would be inferred from the leaves of the trees. These nested-taxa trees would then initially be used to resolve the lower levels of the 'Tree of Life'. In particular, these trees or parts thereof would be clustered in groups with highly overlapping taxa spanning the lower taxonomic levels. From each such cluster, MLS would propose a nested-taxa supertree. Once the lower levels have been resolved, the process would sequentially resolve the overlaying taxonomic levels in turn. Doing so would require identifying those original source trees spanning more than one of the previous clusters.

However, the resolution of these upper levels would not be done using the full trees. Instead, to minimize the computational burden, those parts of the trees concerning the already resolved lower levels would be replaced by a short summary of the unified consensus obtained by MLS. Once the trees are reduced, the taxonomic level for which they were meant could be resolved by inputting them into MLS. The procedure would continue climbing the levels of the 'Tree of Life', iteratively dealing with a series of higher taxonomic levels until finally reaching the universal common ancestor level.

Funding

The authors thank the Université of Montpellier and University of Canterbury for funding a visit of VB to CS. The first author was supported by the *Phylariane* ANR-08-EMER-011-01 project (see `http://www.lirmm.fr/phylariane`). The third author was supported by the Allan Wilson Centre for Molecular Ecology and Evolution and the New Zealand Marsden Fund.

Acknowledgements

We thanks the two anonymous referees for their constructive comments, particularly those relating to the reorganization of the paper.

References

Aho, A. V., Y. Sagiv, T. G. Szymanski, and J. D. Ullman. 1981. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. 10:405–421.

Baum, B. R. 1992. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. Taxon 41:3–10.

Berry, V. and C. Semple. 2006. Fast computation of supertrees for compatible phylogenies with nested taxa. Syst. Biol. 55:270–288.

Berta, A. and A. R. Wyss. 1994. Pinniped phylogeny. Pages 33–56 *in* Contributions in marine mammal paleontology honoring Frank C. Whitmore, Jr. (A. Berta and T. A. Deméré, eds.). Proceedings of the San Diego Society of Natural History, San Diego.

Bininda-Emonds, O. R. P. 2004. The evolution of supertrees. Trends Ecol. Evol. 19:315–322.

Bininda-Emonds, O. R. P., J. L. Gittleman, and A. Purvis. 1999. Building large trees by combining phylogenetic information: a complete phylogeny of the extant carnivora (mammalia). Biol. Rev. 74:143–175.

Bininda-Emonds, O. R. P., K. E. Jones, S. A. Price, M. Cardillo, R. Grenyer, and A. Purvis. 2004. Garbage in, garbage out: data issues in supertree construction. Pages 267–280 *in* Phylogenetic supertrees: combining information to reveal the Tree of Life (O. R. P. Bininda-Emonds, ed.). Kluwer, Dordrecht.

Bininda-Emonds, O. R. P. and A. P. Russell. 1996. A morphological perspective on the phylogenetic relationships of the extant phocid seals (mammalia: Carnivora: Phocidae). Bonner zoologische Monographien 41:1–256.

Bininda-Emonds, O. R. P. and M. J. Sanderson. 2001. Assessment of the accuracy of matrix representation with parsimony supertree construction. Syst. Biol. 50:565–579.

Bininda-Emonds, O. R. P. and A. Stamatakis. 2006. Taxon sampling versus computational complexity and their impact on obtaining the tree of life. Pages 77–95 *in* Reconstructing the Tree of Life: taxonomy and systematics of species rich taxa (T. R. Hodkinson and J. A. N. Parnell, eds.). CRC Press, New York.

Bogdanov, L. V. and V. D. Pastukhov. 1982. New data on the taxonomic position of the baikal seal phoca (pusa) sibirica gmel. Pages 7–12 *in* Morphophysiological and ecological studies of the Baikal seal (V. D. Pastukhov, ed.). Nauka, Novosibirsk.

Burns, J. J. and F. H. Fay. 1970. Comparative morphology of the skull of the ribbon seal, histriophoca fasciata, with remarks on the systematics of phocidae. J. Zool. 161:363–394.

Chapskii, K. K. 1955. An attempt at revision of the systematics and diagnostics of seals of the subfamily phocinae. Trudy Zoologicheskovo Instituta Akademii Nauk SSSR 17:160–199.

Daniel, P. and C. Semple. 2004. Supertree algorithms for nested taxa. Pages 151–171 *in* Phylogenetic supertrees: combining information to reveal the Tree of Life (O. R. P. Bininda-Emonds, ed.). Kluwer, Dordrecht.

Daniel, P. and C. Semple. 2005. A class of general supertree methods for nested taxa. SIAM J. Discrete Math. 19:463–480.

Goldberg, A. V. and S. Rao. 1998. Beyond the flow decomposition barrier. J. ACM 45:783–797.

Gordon, A. D. 1986. Consensus supertrees: the synthesis of rooted trees containing overlapping sets of labeled leaves. J. Classif. 3:31–39.

Hao, J. and J. B. Orlin. 1994. A faster algorithm for finding the minimum cut in a directed graph. J. Algorithm 17:424–446.

Hendey, Q. B. 1972. The evolution and dispersal of the monachinae (mammalia: Pinnipedia). Annals of the South African Museum 59:99–113.

Huson, D. H. and D. Bryant. 2006. Application of phylogenetic networks in evolutionary studies. Mol. Biol. Evol. 23:254–267.

Lalande, J. F., M. Syska, and Y. Verhoeven. 2004. Mascopt - a network optimization library: graph manipulation. Tech. rep. RT-0293, INRIA.

Lento, G. M., R. E. Hickson, G. K. Chambers, and D. Penny. 1995. Use of spectral analysis to test hypotheses on the origin of pinnipeds. Mol. Biol. Evol. 12:28–52.

Ling, J. K. 1978. Pelage characteristics and systematic relationships in the pinnipedia. Mammalia 42:305–313.

Mouchaty, S., J. A. Cook, and G. F. Shields. 1995. Phylogenetic analysis of northern hair seals based on nucleotide sequences of the mitochondrial cytochrome b gene. J. Mammal. 76:1178–1185.

Nojima, T. 1990. A morphological consideration of the relationships of pinnipedia to other carnivorans based on the bony tentorium and bony falx. Mar. Mammal Sci 6:54–74.

Page, R. D. M. 2002. Modified mincut supertrees. Pages 537–552 *in* Second International Workshop on Algorithms in Bioinformatics (R. Guig and D. Gusfield, eds.) Springer, New York.

Page, R. D. M. 2004. Taxonomy, supertrees, and the tree of life. Pages 247–265 *in* Phylogenetic supertrees: combining information to reveal the Tree of Life (O. R. P. Bininda-Emonds, ed.). Kluwer, Dordrecht.

Perry, E. A., S. M. Carr, S. E. Bartlett, and W. S. Davidson. 1995. A phylogenetic perspective on the evolution of reproductive behavior in pagophilic seals of the northwest atlantic as indicated by mitochondrial dna sequences. J. Mammal. 76:22–31.

Purvis, A. 1995. A composite estimate of primate phylogeny. Philos. T. Roy. Soc. B 348:405–421.

Ragan, M. A. 1992. Phylogenetic inference based on matrix representation of trees. Mol. Phylogenet. Evol. 1:53–58.

Ranwez, V., N. Clairon, F. Delsuc, S. Pourali, N. Auberval, S. Diser, and V. Berry. 2009. Phyloexplorer: a web server to validate, explore and query phylogenetic trees. BMC Evol. Biol. 9:108.

Sarich, V. M. 1976. Transferrin. Transactions of the Zoological Society of London 33:165–171.

Semple, C. and M. Steel. 2000. A supertree method for rooted trees. Discrete Appl. Math. 105:147–158.

Semple, C. and M. Steel. 2003. Phylogenetics. Oxford University Press, Oxford.

Slade, R. W., C. Moritz, and A. Heideman. 1994. Multiple nuclear-gene phylogenies: application to pinnipeds and comparison with a mitochondrial dna gene phylogeny. Mol. Biol. Evol. 11:341–356.

Wilkinson, M., J. L. Thorley, D. Pisani, F.-J. Lapointe, and J. O. McInerney. 2004. Some desiderata for liberal supertrees. Pages 227–246 *in* Phylogenetic supertrees: combining information to reveal the Tree of Life (O. R. P. Bininda-Emonds, ed.). Kluwer, Dordrecht.

Wozencraft, W. C. 1993. Order carnivora. Pages 279–348 *in* Mammal species of the world: a taxonomic and geographic reference (D. E. Wilson and D. M. Reeder, eds.). Smithsonian Institution Press, Washington.

Appendix

The appendix consists of four parts. The first two parts consist of the proofs of Proposition 1 and Theorem 5. The third part shows that the alternative weighting scheme described in the last remark following the description of MLS in Section leads to an NP-hard problem, while the fourth part references the source trees of the Phocidae data set.

Proof of Proposition 1

For the proof, notation is consistent with the description of MLS given in Section . Suppose that MLS is applied to $\mathcal{P}$. We may assume that $D(\mathcal{P}')$ contains no cyclic-descendancies, otherwise MLS returns the statement $\mathcal{P}$ *contains cyclic-descendancies* and the proposition holds. Now, the only possible way that MLS does not return a rooted semi-labeled tree is if, at some iteration of FREE in the running of the algorithm, there is no minimum-weight cut that frees a label or triple node in Steps 3 and 4 of FREE. The rest of the proof consists of showing that there is always such a cut.

Let G_w denote the mixed graph inputted at an arbitrary iteration of FREE and consider FREE applied to G_w. No generality is loss in assuming that G_w is unchanged at the end of Step 1. The following is easily seen.

Lemma 7 *Let x and z be two label nodes in G_w. If there is a directed path in G_w from x to z in which each arc has infinite weight, then $x <_{\mathcal{T}} z$ for all $\mathcal{T} \in \mathcal{P}$.*

Lemma 8 *Let x and y be label nodes of G_w. Then x and y are joined by an arc (x, y) precisely if (x, y) is an arc in $D(\mathcal{P}')$.*

Proof. If the lemma does not hold, then there is an arc, (x, y) say, in $D(\mathcal{P}')$ where x and y are nodes in G_w, but (x, y) is not an arc in G_w. The only way that this could happen is that, at some previous iteration, (x, y) is deleted as part of a minimum-weight cut to free either a label or triple node. But then, as the cut has minimum-weight, x and y would be in different arc components at Step 5 of FREE in this iteration, and so the node sets of the

mixed graphs inputted to FREE at subsequent iterations contains at most one of x and y; a contradiction. Thus Lemma 8 holds. □

Lemma 9 *Let Q be an arc-path in G_w in which each node is a label node and each arc has weight ∞. Let x be the initial node of Q, and suppose that x has in-degree zero in G_w. If z is a node in Q and $z \neq x$, then $x <_{\mathcal{T}} z$ for all $\mathcal{T} \in \mathcal{P}$.*

Proof. The proof is by induction on the number k of nodes in Q. If $k = 2$, then, as x has in-degree zero, Q consists of the vertices x and z, and the arc (x, z). Since (x, z) has weight ∞, the lemma holds.

Now suppose that Lemma 9 holds for all such arc-paths beginning at x with at most $k - 1$ nodes, where $k \geq 3$. Let Q' be the arc-path obtained from Q by restricting it to the first $k-1$ nodes. Let y be the last node in Q' and let z be the last node in Q. Then, by the induction assumption, $x <_{\mathcal{T}} y$ for all $\mathcal{T} \in \mathcal{P}$. There are two cases to consider depending upon whether the last arc in Q is (i) (y, z) or (ii) (z, y).

First assume that (i) holds. Then $y <_{\mathcal{T}} z$ for all $\mathcal{T} \in \mathcal{P}$ and so, as $x <_{\mathcal{T}} y$ for all $\mathcal{T} \in \mathcal{P}$, it follows that $x <_{\mathcal{T}} z$ for all $\mathcal{T} \in \mathcal{P}$. Now assume that (ii) holds. Since $z <_{\mathcal{T}} y$ and $x <_{\mathcal{T}} y$ for all $\mathcal{T} \in \mathcal{P}$, we have, for each $\mathcal{T} \in \mathcal{P}$, either $x <_{\mathcal{T}} z$ or $z <_{\mathcal{T}} x$. If $x <_{\mathcal{T}} z$ for all $\mathcal{T} \in \mathcal{P}$, then the lemma holds. Furthermore, if $z <_{\mathcal{T}} x$ for all $\mathcal{T} \in \mathcal{P}$, then, by Lemma 8, G_w contains the arc (z, x), contradicting the assumption that x has in-degree zero in G_w. Therefore we may assume that there are trees $\mathcal{T}', \mathcal{T}'' \in \mathcal{P}$ such that $z <_{\mathcal{T}'} x$ and $x <_{\mathcal{T}''} z$. But then $D(\mathcal{P}')$ contains a cyclic-descendancy; a contradiction. Thus Lemma 9 holds. □

Lemma 10 *Let x be a label node in G_w with in-degree zero and suppose that w is edge-adjacent to x such that $\{x, w\}$ has weight ∞. Then every arc-path from x to w in G_w that contains no triple node has an arc of finite weight.*

Proof. Suppose that G_w contains an arc-path from x to w in which every arc has weight ∞ and no node is a triple node. Then, by Lemma 9, $x <_{\mathcal{T}} w$ for all $\mathcal{T} \in \mathcal{P}$, contradicting the assumption that $\{x, w\}$ has weight ∞. Thus the lemma holds. □

It follows from Lemma 10 that there is a label node in G_w that can be freed unless G_w contains a triple node. We complete the proof of Proposition 1 by considering triple nodes.

Lemma 11 *Let Q be an arc-path in G_w starting at label node x, ending at label node y, and having the property that each arc has weight ∞. Let $\mathcal{T} \in \mathcal{P}$ and suppose that $x||_{\mathcal{T}}y$. Then either there is a label node in Q that is ancestor of both x and y in $\mathcal{T}$ or there is a triple node $\overline{ab|c}$ in Q such that $\mathrm{mrca}_{\mathcal{T}}(a,b)$ is an ancestor of $\mathrm{mrca}_{\mathcal{T}}(x,y)$.*

Proof. Since each arc in Q has weight ∞, each label node in Q is a label of $\mathcal{T}$. Now, by considering Q and, in particular, the position of the label nodes in this path in $\mathcal{T}$ as one follows it from x to y, it is easily seen that one of the two outcomes in the lemma must hold. □

Now let $\mathcal{T}$ be a tree in $\mathcal{P}$ and let $\overline{ab|c}$ be a triple node in G_w. Relative to G_w, we say that $ab|c$ is *maximal in $\mathcal{T}$* if there is no triple node $\overline{a'b'|c'}$ in G_w such that $\mathrm{mrca}_{\mathcal{T}}(a',b')$ is a strict ancestor of $\mathrm{mrca}_{\mathcal{T}}(a,b)$.

Suppose that no label node of G_w can be freed and suppose, to the contrary, that no triple node of G_w can be freed. We next establish three properties of a maximal triple in G_w. These properties will be repeatedly use to complete the proof of the proposition.

Lemma 12 *Let $\mathcal{T} \in \mathcal{P}$, and let $\overline{ab|c}$ be a triple node in G_w that is maximal in $\mathcal{T}$.*

(I) *Let Q be an arc-path in G_w either from a to c or from b to c in which each arc has weight ∞. Then there is a label node, x say, in Q that is ancestor of both $\mathrm{mrca}_{\mathcal{T}}(a,b)$ and c in $\mathcal{T}$.*

(II) *Let x in (I) be chosen to be the closest such label to the root of $\mathcal{T}$. Then x does not have in-degree zero in G_w.*

(III) *Let z be a label node in G_w such that either z has in-degree zero or z is arc-adjacent to a triple node, and there is a directed path in G_w from z to x. Then $z \in \mathcal{L}(\mathcal{T})$, and the label node in G_w, say x', that is an ancestor of z in $\mathcal{T}$ and is the closest such label to the root of $\mathcal{T}$ satisfies the following:*

(i) *x is non-comparable to x' in $\mathcal{T}$, and*

(ii) *there is a triple node in G_w such that x' is an ancestor of each of the labels that make up this triple in $\mathcal{T}$.*

Proof. By Lemma 11, either (I) holds or there is a triple node $\overline{a'b'|c'}$ in Q such that $\mathrm{mrca}_{\mathcal{T}}(a', b')$ is an ancestor of $\mathrm{mrca}_{\mathcal{T}}(a, c)$ (and therefore also of $\mathrm{mrca}_{\mathcal{T}}(b, c)$). But then $\mathrm{mrca}_{\mathcal{T}}(a', b')$ is a strict ancestor of $\mathrm{mrca}_{\mathcal{T}}(a, b)$, contradicting the maximality of $ab|c$ in $\mathcal{T}$. Thus (I) holds.

To see (II), suppose that x has in-degree zero in G_w. Then, as no label nodes in G_w can be freed, there is a node, w say, in G_w that is edge-adjacent to x with $\{x, w\}$ having weight ∞, and there is an arc-path Q_x in G_w from x to w in which each arc has weight ∞. By Lemma 11, either there is a label node in Q_x that is an ancestor of both x and w in $\mathcal{T}$ or there is a triple node $\overline{a'b'|c'}$ in Q_x such that $\mathrm{mrca}_{\mathcal{T}}(a', b')$ is an ancestor $\mathrm{mrca}_{\mathcal{T}}(x, w)$. The first possibility contradicts the choice of x, while the second possibility contradicts the maximality of $ab|c$. Hence (II) holds.

Consider (III). If z is arc-adjacent to a triple node, then $z \in \mathcal{L}(\mathcal{T})$. Furthermore, if z is a label node, then z cannot be freed and so z is edge-adjacent to a label node, say y, in G_w and the edge $\{z, y\}$ has weight ∞, so $z \in \mathcal{L}(\mathcal{T})$. Since $D(\mathcal{P}')$ has no directed cycles and there is a directed path in G_w from z to x, it follows that x is not an ancestor of z in $\mathcal{T}$ and so x is non-comparable to z in $\mathcal{T}$. Thus, because of the choice of x, any ancestor of z in $\mathcal{T}$ that is a label node in G_w is also non-comparable to x. Hence, to complete the proof of (III), it suffices to show that x' satisfies (ii).

First assume that z has in-degree zero. Then, by Lemma 10, there is a triple node $\overline{rs|t}$ on an arc-path from z to y in which each arc has weight ∞. Without loss of generality, we may assume that this is the first such triple node and that r appears before s on this path. By Lemma 9, $z <_{\mathcal{T}} r$. In particular, x is non-comparable to r in $\mathcal{T}$. Let $\overline{a'b'|c'}$ be a triple node in G_w that is maximal in $\mathcal{T}$ and has the property that $\mathrm{mrca}_{\mathcal{T}}(a', b')$ is an ancestor of $\mathrm{mrca}_{\mathcal{T}}(r, s)$ in $\mathcal{T}$. Since no triple node can be freed and $x' <_{\mathcal{T}} r$, it follows by (I) that x' is an ancestor of both $\mathrm{mrca}(a', b')$ and c' in $\mathcal{T}$.

Now assume that z is arc-adjacent to a triple node in G_w. Making use of this triple node instead of $\overline{rs|t}$ as in the previous paragraph and again using (II), we deduce that there is a triple node in G_w that enables x' to satisfy (ii). Thus (III), and therefore Lemma 12, holds. □

Noting that (I)–(III) holds for all triple nodes in G_w that are maximal in $\mathcal{T}$, we complete the proof of the proposition by repeatedly using (I)–(III) to obtain a contradiction to our assumption that G_w has no triple nodes that can be freed. Before making this completion, observe that, for (III), one can always find such an element z by starting at x and continually traversing arcs in the opposite direction until there are no such arcs to traverse. Since G_w has no directed cycles, this process must eventually stop.

Let $\overline{a_ib_i|c_i}$ be a triple node in G_w such that $a_ib_i|c_i$ is maximal in $\mathcal{T}$. Using (I), let x_i be the label node in G_w that is an ancestor of $\mathrm{mrca}_{\mathcal{T}}(a_i, b_i)$ and c_i in $\mathcal{T}$ and, amongst all such nodes, it is the closest label to the root of $\mathcal{T}$. By (II), x_i does not have in-degree zero in G_w. Let z_i be a label node of G_w such that either z_i has in-degree zero or z_i is arc-adjacent to a triple node, and there is a directed path in G_w from z_i to x_i. Let x_{i+1} be the label node in G_w that is an ancestor of z_i and is the closest such label to the root of $\mathcal{T}$. By (III), x_{i+1} is non-comparable to x_i in $\mathcal{T}$ and there is a triple node $\overline{a_{i+1}b_{i+1}|c_{i+1}}$ in G_w that is maximal in $\mathcal{T}$ and has the property that x_{i+1} is an ancestor of both $\mathrm{mrca}_{\mathcal{T}}(a_{i+1}, b_{i+1})$ and c_{i+1} in $\mathcal{T}$.

Beginning with $i = 1$ and repeatedly applying the process in the previous paragraph, we obtain a sequence of labels $x_1, x_2, \ldots$ in $\mathcal{T}$ such that, for all i, the labels x_i and x_{i+1} are non-comparable. Since $\mathcal{T}$ is finite, it follows that, for some distinct i and j, the label nodes x_i and x_j are equal, where $i < j$. Without loss of generality, we may assume that $x_i, x_{i+1}, \ldots, x_{j-1}$ are pairwise distinct. Consider the sequence

$$x_i, z_i, x_{i+1}, z_{i+1}, \ldots, x_{j-1}, z_{j-1}, x_j = x_i.$$

For all $l \in \{i+1, i+2, \ldots, j\}$, x_l is an ancestor of z_{l-1} in $\mathcal{T}$. This implies that, for all l, there is a directed path in $D(\mathcal{P}')$ from x_l to z_{l-1}. Moreover, by construction, there is also a directed path in $D(\mathcal{P}')$ from z_{l-1} to x_{l-1} for all l. Thus, as $x_i = x_j$, the mixed graph $D(\mathcal{P}')$ contains a directed cycle; a contradiction. It now follows that there is a triple node of G_w that can be freed. This completes the proof of Proposition 1.

Proof of Theorem 5

First recall the description of how AncestralBuild can be obtained from MLS in the remarks following the description of MLS in Section . It is

easily checked that to establish the theorem, it suffices to show that, for all i, in iteration i of FREE$'$ and FREE,

(i) the set of free nodes in Step 1 of FREE$'$ is equal to the set of free nodes in Step 2a of FREE, and

(ii) up to triple nodes, the node sets of the arc components in Step 3 of FREE$'$ is the same as that of the node sets of the arc components in Step 2c of FREE.

Observe that if (i) and (ii) hold at iteration i, then, up to rooted triple nodes and weightings, the input at iteration $i+1$ of FREE$'$ and FREE coincide. Note that, in the proof, as in the description of MLS, we will assume that the weighted-descendancy graph of $\mathcal{P}'$ is connected. By applying the proof to each of the connected components of this graph, it is easily seen that no generality is loss in making this assumption.

Suppose that either (i) or (ii) does not hold at some iteration. Let j be the first such iteration. Let G (resp. G_w) be the subgraph inputed into FREE$'$ (resp. FREE) at iteration j. If $j = 1$, then these subgraphs are $D(\mathcal{P}')$ and $D_w(\mathcal{P}')$, respectively. Let $\mathcal{T}$ be the rooted semi-labeled tree returned by ANCESTRALBUILD. We begin with an observation. Since the node sets of the arc components found in Step 3 of FREE$'$ are not constrained by triple nodes and arcs with weight ∞, there is exactly one arc component at the end of Step 1 of FREE in the first j iterations; otherwise, we contradict the fact that (ii) holds in the previous iteration. Note that, as the weighted-descendancy graph of $\mathcal{P}'$ is connected, FREE is not called in Step 1 in the first iteration of FREE.

We complete the proof by first establishing by contradiction that $j \neq 1$, and so $j \geq 2$. For $j \geq 2$, the proof by contradiction is similar and makes use of the fact that (i) and (ii) hold for $j-1$. Because of this similarity, we omit the proof of this case.

Let $j = 1$, and first suppose that (i) does not hold. Let $\mathcal{S}'_0$ and $\mathcal{S}_0$ denote the set of free nodes in Steps 1 and 2a of FREE$'$ and FREE, respectively. Because of the additional constraints imposed by the triple nodes, and arcs and edges with weight ∞ in FREE, it follows that $\mathcal{S}_0$ is a proper subset of $\mathcal{S}'_0$. Let $a \in \mathcal{S}'_0 - \mathcal{S}_0$. Then either there is a triple node $\overline{ab|c}$, an arc (c, a) with weight ∞, or an edge $\{a, b\}$ with weight ∞ in $D_w(\mathcal{P}')$. If there is such

a triple node, then, as $\mathcal{T}$ ancestrally displays $ab|c$, the nodes a and b are in one arc component of $D(\mathcal{P}')$ and c is in a separate arc component. But then, by construction, a is an ancestor of b in $\mathcal{T}$; a contradiction. If there is such an arc (c, a), then c is an ancestor of a in $\mathcal{T}$; a contradiction. Lastly, if there is such an edge $\{a, b\}$, then, as $a||_{\mathcal{T}}b$, it follows that a and b are in separate arc components of $D(\mathcal{P}')$. Now a and b are in the same arc component of $D_w(\mathcal{P}')$, so, by construction, there is an arc in $D_w(\mathcal{P}')$ with weight ∞ on every arc-path from a to b in $D_w(\mathcal{P}')$. It now follows that for one of these arcs with weight ∞, (p, q) say, nodes p and q are in separate arc components in $D(\mathcal{P}')$. But then, by construction, either p is not an ancestor of q in $\mathcal{T}$ if p is a label node or $\mathcal{T}$ does not ancestrally display the rooted triple corresponding to p if p is a triple node; a contradiction. Thus (i) holds for $j = 1$.

Now assume that (ii) does not hold for $j = 1$. Let $\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_{k'}$ and $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ denote the node sets of the arc components of $D(\mathcal{P}')\backslash\mathcal{S}'_0$ in Step 3 of FREE$'$ and $D_w(\mathcal{P}')\backslash\mathcal{S}_0$ in Step 2c of FREE, respectively. Because of the additional constraints imposed by the triple nodes and arcs with weight ∞, it follows that there is a set, $\mathcal{S}_i$ say, such that, up to triple nodes, it is the union of at least two sets amongst $\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_k$. In particular, either there is a triple node $\overline{ab|c}$ with a in one of these sets and b in another, or an arc (c, a) with weight ∞ with c in one these sets and a in another. But then, again by construction, either $\mathcal{T}$ does not ancestrally display $ab|c$ or c is not ancestor of a in $\mathcal{T}$; a contradiction. Therefore (ii) holds for $j = 1$.

Freeing nodes using a list-based weighting scheme is NP-hard

Let $\mathcal{P}$ be a collection of rooted semi-labeled trees and let $\mathcal{P}'$ be a collection of rooted fully-labeled trees that is obtained from $\mathcal{P}$ by adding distinct new labels. When MLS meets topological conflicts amongst source trees, it frees either label nodes or a triple node by removing links from a subgraph of $D_w(\mathcal{P}')$. This leads to the rooted semi-labeled tree eventually returned by MLS to contradict the topological signal given by the trees in $\mathcal{P}'$ that induce these particular links. The intuition here is that one seeks to remove a set of links that contradict a minimum number of trees in $\mathcal{P}'$. MLS proceeds by weighting each link with the number of trees that support it and, when "stuck", removing only links belonging to a minimum-weight cut to ensure a small number of source trees will be contradicted. But, it can be that

the same tree contributes to several links in a minimum-weight cut, and so to contradict a minimum number of trees in $\mathcal{P}'$, a finer weighting scheme is required. The alternative and very natural way to "weight" the links of $D(\mathcal{P}')$ is to assign each link the list of trees in $\mathcal{P}'$ that induced this link. Then, when confronted by topological conflicts, the algorithm would be able to identify a cut that precisely involves a minimum number of trees in $\mathcal{P}'$. This number is the size of the union of the lists of the links that will be deleted if this cut is chosen. However, as we show next, finding a minimum-weight cut to free a label node under this weighting scheme is an NP-hard problem.

More formally, let $D_l(\mathcal{P}')$ be the graph obtained from the weighted-descendancy graph $D_w(\mathcal{P}')$ of $\mathcal{P}'$ by replacing the weight of each arc (c, a) with the set

$$\{\mathcal{T} \in \mathcal{P}' : c <_{\mathcal{T}} a\}$$

and that of each edge $\{a, b\}$ with the set

$$\{\mathcal{T} \in \mathcal{P}' : b||_{\mathcal{T}} a\}.$$

Note that the ∞ weight is replaced with the set $\mathcal{P}'$. The graph $D_l(\mathcal{P}')$ is the *list-descendancy graph* of $\mathcal{P}'$ and we refer to the above sets as *lists*. We will now show that applying MLS to $\mathcal{P}$, replacing $D_w(\mathcal{P}')$ with $D_l(\mathcal{P}')$ and using the above list-based weighting scheme, leads to an NP-hard problem. To establish this hardness result, we use the classical NP-hard problem Vertex Cover:

Vertex Cover
Input: An undirected graph $G = (V, E)$.
Solution: A minimum-sized subset $V_m \subseteq V$ such that, for each edge $\{u, v\} \in E$, at least one of u and v belongs to V_m.

Let $G = (V, E)$ be an instance of Vertex Cover and arbitrarily assign a direction to each edge of G, thus viewing G as a directed graph. Let

$$F = \bigcup_{e \in E} \{e_1, e_2, e_3, e_4\}.$$

We now construct a collection of rooted fully-labeled trees $\mathcal{P}$ whose label set is $V \cup F \cup \{x\}$, where x is a distinguished label. In particular, $\mathcal{P}$ consists of the following trees:

(i) For each edge $e \in E$, the trees $(x, e_3)e_1$, $(x, e_3)e_2$, are in $\mathcal{P}$, where $(a, b)c$ is the rooted fully-labeled tree consisting of two leaves labeled a and b, and a root labeled c.

(ii) For each $u \in V$, the following tree $\mathcal{T}_u$ is in $\mathcal{P}$. The root of $\mathcal{T}_u$ is labeled u, and has a first child labeled x and then a child labeled e_3 for each arc e leaving u in G with each such e_3 node itself having e_4 as single child. Furthermore, for each arc e' coming into u in G, the node labeled x has a child labeled e'_4.

Clearly, in the size of G, the set $\mathcal{P}$ can be constructed in polynomial time and its size is polynomial. Now consider MLS applied to $\mathcal{P}$ under the above list-based weighting scheme. Since $\mathcal{P}$ is fully-labeled, $D_l(\mathcal{P}') = D_l(\mathcal{P})$. Note that $D_l(\mathcal{P}')$ has no triple nodes, and it is arc connected because of x. At the completion of the first iteration of FREE, every node of the form e_1, e_2 is deleted as well as all nodes in V. Again because of x, the resulting graph, $D'_l(\mathcal{P}')$ say, has exactly one arc component. During the second iteration of FREE, x is considered as a possible node that can be freed, as are each of the nodes of the form e_3 which are linked to x by edges. We next show that G has a minimum-sized vertex cover of size m if and only if the size of a minimum-weight cut to free x in $D'_l(\mathcal{P}')$ is m, thus showing in general that finding a minimum-weight cut to free a label node under this alternative weighting scheme is an NP-hard problem.

Now, in $D'_l(\mathcal{P}')$, the node x is edge-adjacent precisely to each of the nodes of the form e_3. Furthermore, for each $e \in E$, there is an arc-path involving x, e_3 and e_4 consisting of two arcs (an arc from e_3 to e_4 and one from x to e_4), and these arc-paths share no arcs. Thus any minimum-weight cut to free x corresponds to either deleting the edge $\{x, e_3\}$ or deleting one of the arcs (x, e_4) and (e_3, e_4) for all e.

For an edge of the form $\{x, e_3\}$, its corresponding list includes two trees arising via (i) and one tree arising via (ii). For each such tree from (i), this is the only link whose list contains it, while the tree from (ii) is $\mathcal{T}_u$ where u is the node in G from which e starts. On the other hand, for all e, the lists of each of (x, e_4) and (e_3, e_4) consist of exactly one tree (these two trees differing from one another). Thus, to free x, it is always more parsimonious to remove the support of either (x, e_4) or (e_3, e_4), than removing the support of the edge $\{x, e_3\}$. It now follows that if C_m is the union of the lists of the links that will be deleted in a minimum-weight cut to free x, then C_m includes

the single tree in the list associated with one of the arcs (x, e_4) and (e_3, e_4) for each e in G, and no tree arising via (i). Observing that the subset

$$\{u : \mathcal{T}_u \in C_m\}$$

of V is a vertex cover of G, it follows that if V_m is a solution to VERTEX COVER for G, then $|V_m| \leq |C_m|$.

Now suppose that V_m is a minimum-sized vertex cover of G. Let

$$C = \{\mathcal{T}_u : u \in V_m\}.$$

There is a one to one correspondence between the edges of G and the above arc-paths joining x and nodes of the form e_3. In particular, this correspondence assigns the edge $e = \{v, w\}$ of G with the arc-path in which the list of one arc consists of $\mathcal{T}_v$ and the list of the other arc consists of $\mathcal{T}_w$. It now follows that C frees x, and so if C_m is a minimum-weight cut that frees x, then $|V_m| \geq |C_m|$. We conclude that $|V_m| = |C_m|$ and so finding a minimum-weight cut that frees a label node under the above weighting is an NP-hard problem.

References for the source trees of the Phocidae data set

Berta and Wyss (1994), Bininda-Emonds and Russell (1996), Bogdanov and Pastukhov (1982), Burns and Fay (1970), Chapskii (1955), Hendey (1972), Lento et al. (1995), Ling (1978), Mouchaty et al. (1995), Nojima (1990), Perry et al. (1995), Sarich (1976), Slade et al. (1994), Wozencraft (1993).