Tree Containment With Soft Polytomies

Matthias Bentert
TU Berlin, Institut für Softwaretechnik und Theoretische Informatik, Berlin, Germany
matthias.bentert@tu-berlin.de

Josef Malík
Czech Technical University, Prague, Czech Republic
josef.malik@fit.cvut.cz

Mathias Weller
CNRS, LIGM, Université Paris Est, Marne-la-Vallée, France
mathias.weller@u-pem.fr

Abstract

The Tree Containment problem has many important applications in the study of evolutionary history. Given a phylogenetic network N and a phylogenetic tree T whose leaves are labeled by a set of taxa, it asks if N and T are consistent. While the case of binary N and T has received considerable attention, the more practically relevant variant dealing with biological uncertainty has not. Such uncertainty manifests itself as high-degree vertices ("polytomies") that are "jokers" in the sense that they are compatible with any binary resolution of their children. Contrasting the binary case, we show that this problem, called Soft Tree Containment, is $\mathcal{NP}$-hard, even if N is a binary, multi-labeled tree in which each taxon occurs at most thrice. On the other hand, we reduce the case that each label occurs at most twice to solving a 2-SAT instance of size $O(|T|^3)$. This implies $\mathcal{NP}$-hardness and polynomial-time solvability on reticulation-visible networks in which the maximum in-degree is bounded by three and two, respectively.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms, Applied computing → Biological networks

Keywords and phrases Phylogenetics, Reticulation-Visible Networks, Multifurcating Trees

Digital Object Identifier 10.4230/LIPIcs.SWAT.2018.9

Acknowledgements The project leading to this work was conceived on the 2017 research retreat of the Algorithmics and Computational Complexity group of TU Berlin.

1 Introduction

With the dawn of molecular biology also came the realization that evolutionary trees, which have been widely adopted by biologists, are insufficient to describe certain processes that have been observed in nature. In the last decade, the idea of reticulate evolution, supporting gene flow from multiple parent species, arose [2, 15]. A reticulation event can be caused by, for example, hybridization (occurring frequently in plants) and horizontal gene transfer (a dominating factor in bacterial evolution). Reticulate evolution is described using "phylogenetic networks" (see the monographs by Gusfield [11] and Huson et al. [13]). A central question when dealing with both phylogenetic trees and networks is whether or not they represent consistent information, formulated as the question whether or not the network "displays" the tree. This problem is known as Tree Containment and it has been shown $\mathcal{NP}$-hard [14, 17]. Due to its importance in the analysis of evolutionary history, attempts have been made to identify polynomial-time computable special cases [6, 5, 1, 10, 14, 17, 7, 18],

16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018).
Editor: David Eppstein; Article No. 9; pp. 9:1–9:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

as well as moderately exponential-time algorithms [8, 18]. However, all of these works are limited to binary networks and trees.

In reality, we cannot hope for perfectly precise evolutionary histories. In particular, speciation events (a species splitting off another) occurring in rapid succession (only a few thousand years between speciations) can often not be reliably placed in the correct order they occurred. The fact that the correct order of bifurcations is unknown is usually modeled by multifurcating vertices and, to tell them apart from speciation events resulting in multiple species, the former are called "soft polytomies" and the latter are called "hard polytomies". Of course, the same argument holds for non-binary reticulation vertices indicating uncertainty in the order of hybridization events. Soft polytomies have a noteworthy impact on the question of whether a tree is compatible with a network: since a soft polytomy (also called "fan") on the taxa a, b, and c represents lack of knowledge regarding their history, we would consider any binary tree on the taxa a, b, and c compatible with it. In this work, we present first algorithmic results for Tree Containment with soft polytomies (which we call Soft Tree Containment). We consider the case where the network is a multi-labeled tree and show that the problem is cubic-time solvable if each label occurs at most twice (by reduction to 2-SAT) and $\mathcal{NP}$-hard, otherwise. This implies corresponding results for (single-labeled) "reticulation-visible" networks, depending on their maximum in-degree. Despite being an intermediate step in proving results for networks, multi-labeled trees are themselves important, for example when handling gene trees, in which different versions of a gene may be found in the same species.

Finally, our results have impact on the Cluster Containment problem [13] since it is a special case of our problem.[1]

Preliminaries

A *phylogenetic network* (or *network* for short) on a set X of taxa is a rooted, leaf-labeled DAG in which all vertices that do not have in-degree at most one have out-degree exactly one. These vertices are called *reticulations* and the others are called *tree vertices*. A network without reticulations is called a (phylogenetic) *tree*. By default, no label occurs twice in a network, and we will make exceptions explicit by calling networks in which a label may occur more than once *multi-labeled* (note that networks are a special case of multi-labeled networks in which each label occurs only once). This allows us to use leaves and labels (taxa) interchangeably. For brevity, we abbreviate $\{x, y\}$ to xy, and $\{x, y, z\}$ to xyz. Let N be a network with root ρ_N. We denote the set of vertices in N by $V(N)$. We define a relation "$\leq_N$" on subsets of $V(N)$ such that $U \leq_N W$ if and only if N contains a w-u-path for each $u \in U$ and $w \in W$. If $u \leq_N w$, we call u a *descendant* of w and w an *ancestor* of u. For each $v \in V(N)$, we let N_v be the subnetwork of N induced by $\{u \mid u \leq_N v\}$ and we denote the set of leaf-labels in N_v by $\mathcal{L}(v)$ and abbreviate $\mathcal{L}(N) := \mathcal{L}(\rho_N)$. Such a set is also called a *cluster* of N. Note that, if N is a tree, N_v is the subtree rooted at v. We abbreviate $n := |\mathcal{L}(\rho_N)|$. For any $X \subseteq V(N)$, we let $\mathrm{LCA}_N(X)$ be the set of least common ancestors of X, that is, the minima (wrt. $\leq_N$) among all vertices u of N with $X \leq_N u$ (in particular, if N is a tree, $\mathrm{LCA}_N(X)$ is a single vertex, not a set). If clear from context, we may drop the subscript. Note that, in trees, the LCA of any three vertices has a unique minimum. For any $U \subseteq V(N)$, we denote the result of removing all vertices v that do not have a descendant in

[1] Given a binary network N on the taxa X and some $Y \subseteq X$, Cluster Containment asks if N displays any binary tree T in which $\mathcal{L}(u) = Y$ for any u. This is equivalent to N softly displaying the tree T in which all taxa in $X \setminus Y$ are children of the root and there is another child u of the root with children Y.

U by $N|_L$ and $N||_L$ is the result of *suppressing* all degree-two vertices in $N|_L$. Suppressing a vertex u in N with unique parent p and unique child c refers to the act of removing u and adding the edge pc, unless this edge already exists. Note that, if N is a tree, then $N|_L$ is the smallest subtree of N containing the vertices in L and the root of N and $N||_L$ is the smallest topological minor of N containing the vertices in L and the root of N. A vertex u in N is called *stable* on v if all ρ_N-v-paths contain u. If, for each reticulation u in N there is some leaf ℓ such that u is stable on ℓ, then N is called *reticulation visible.* A network is *binary* if all vertices except the root have degree (=in-degree + out-degree) at most three and the root has degree two. A binary network N_B on three leaves a, b, and c is called a *triplet* and we denote it by $ab|c$ if c is a child of the root of N_B. N_B is called *binary resolution* of a network N if N is a contraction of N_B. In this case, there is a surjective function $\chi : V(N_B) \to V(N)$ such that, contracting all edges uv of N_B with $\chi(u) = \chi(v)$ results in N (more formally, for each $x, y \in V(N)$, the edge xy exists in N if and only if there is an edge between $\chi^{-1}(x)$ and $\chi^{-1}(y)$ in N_B). We call such a function *contraction function* of N_B for N. We suppose that all binary resolutions are minimal, that is, they do not contain biconnected components with exactly one incoming and one outgoing edge. Observe that, when contracting edges of N_B to form N, we never create vertices with in-degree *and* out-degree more than one.

▶ **Observation 1.** *Let N_B be a binary resolution of a network N, let χ be a contraction function of N_B for N, and let $u \in V(N)$. Then, $\chi^{-1}(u)$ does not contain a reticulation and a tree vertex with out-degree more than one.*

If N contains a subgraph S that is isomorphic[2] to a tree T, then we simply say that N contains a subdivision of T. Slightly abusing notation, we consider each vertex $v \in V(T)$ equal to the vertex of S (and, thus, of N) that v is mapped to by an isomorphism. Thus, S consists of $V(T)$ and some vertices of in- and out-degree one. The following definition is paramount.

▶ **Definition 2.** Let N be a network and let T be a tree. Then,
- N *firmly displays* T if and only if N contains a subdivision of T and
- N *softly displays* T if and only if there are binary resolutions N_B of N and T_B of T such that N_B firmly displays T_B.

Definition 2 is motivated by the concept of "hard" and "soft" polytomies (that is, high degree vertices): In phylogenetics, a polytomy is called *firm* or *hard* if it corresponds to a split of multiple species at the same time and *soft* if it represents a set of binary speciations whose order cannot be determined from the available data. In this sense, a polytomy is compatible with another if and only if there is a biological "truth", that is, a binary resolution, that is common to both. Note that, for binary N and T, the two concepts coincide. Furthermore, for trees on the same label-set, the concepts of display and binary resolution coincide.

▶ **Observation 3.** *Let T and T_B be trees on the same leaf-label set and let T_B be binary. Then, T softly displays T_B if and only if T_B is a binary resolution of T.*

Throughout this work we will mostly use the soft variant and we will refer to it simply as "display" for the sake of readability. Note that a binary tree displays another binary tree if and only if they are isomorphic. Thus, in the special case that N is a tree, the "display" relation is symmetrical, leading to the following observation.

[2] In this work, "isomorphic" always refers to isomorphism respecting leaf-labels, that is, all isomorphisms must map a leaf of label λ to a leaf of label λ.

▶ **Observation 4.** *A tree T displays a tree T' if and only if T' displays T.*

Finally, the central problem considered in this work is the following.

Soft Tree Containment
Input: A network N and a tree T
Question: Does N softly display T?

2 Display with Soft Polytomies

The concept of "display" is well-researched for binary trees, in particular, triplets.

▶ **Observation 5** ([4])**.** *Let T_B be a binary tree and let $a, b, c \in \mathcal{L}(T_B)$. Then, T_B displays $ab|c$ if and only if $\mathrm{LCA}(ab) < \mathrm{LCA}(bc) = \mathrm{LCA}(ac)$. Indeed, T_B is uniquely identified by the set D of displayed triplets, that is, T_B is the only binary tree displaying the triplets in D.*

However, the "display"-relation with soft polytomies lacks a solid mathematical base in the literature. In this section, we develop alternative characterizations of the term "(softly) display". To do this, we use the following characterization of isomorphism for binary trees.

▶ **Observation 6.** *Binary trees T_B and T'_B on the same label-set are isomorphic if and only if, for each $u \in V(T_B)$ and each $Y \subseteq \mathcal{L}(u)$, u has a child v with $\mathcal{L}(v) = Y$ if and only if $\mathrm{LCA}_{T'_B}(\mathcal{L}(u))$ has a child v' with $\mathcal{L}(v') = Y$.*

▶ **Lemma 7.** *Let N and T be trees. Then, N displays T if and only if, for all $u \in V(T)$ and $v \in V(N)$, it holds that $\mathcal{L}(u) \subseteq \mathcal{L}(v)$, $\mathcal{L}(u) \supseteq \mathcal{L}(v)$ or $\mathcal{L}(u) \cap \mathcal{L}(v) = \varnothing$.*

Proof. Since each label appears only once in N and T, it holds that N displays T if and only if there are binary resolutions N^B of N and T^B of T such that N^B and T^B are isomorphic.

"⇒": Let N softly display T. Towards a contradiction, assume that there are $u \in V(N)$ and $w \in V(T)$ such that $\mathcal{L}(u) \not\subseteq \mathcal{L}(v)$, $\mathcal{L}(u) \not\supseteq \mathcal{L}(v)$ and $\mathcal{L}(u) \cap \mathcal{L}(v) \neq \varnothing$, that is, there are $x \in \mathcal{L}(u) \setminus \mathcal{L}(w)$, $y \in \mathcal{L}(u) \cap \mathcal{L}(w)$, and $z \in \mathcal{L}(w) \setminus \mathcal{L}(u)$. Since there are binary resolutions N^B and T^B of N and T, respectively, such that N^B and T^B are isomorphic, there is a vertex u' in N^B with $\mathcal{L}(u') = \mathcal{L}(u)$ and a vertex v' in T with $\mathcal{L}(v') = \mathcal{L}(v)$. Since N^B and T^B are trees and each leaf-label only appears once in each of them, $N^B_{u'}$ contains the leaves x and y but not the leaf z. Analogously, $T^B_{v'}$ contains the leaves y and z but not the leaf x, contradicting N^B being isomorphic to T^B.

"⇐": In order to show the contraposition, suppose that N does not softly display T. Since N does not softly display T, for any binary resolutions N^B of N and T^B of T, it holds that N^B and T^B are not isomorphic. By Observation 6, there are vertices $p \in V(N^B)$ and $q := \mathrm{LCA}_{T^B}(\mathcal{L}(p))$ with children p_1, p_2 and q_1, q_2, respectively, such that $\mathcal{L}(p_1) \neq \mathcal{L}(q_1)$ and $\mathcal{L}(p_1) \neq \mathcal{L}(q_2)$. We will use the fact that $\mathcal{L}(p_1) \uplus \mathcal{L}(p_2) = \mathcal{L}(p) = \mathcal{L}(q) = \mathcal{L}(q_1) \uplus \mathcal{L}(q_2)$.

Case 1: $\mathcal{L}(p_i) \subsetneq \mathcal{L}(q_j)$ for any i, j. Then, there are taxa

$$\begin{aligned} &x \in \mathcal{L}(p_i) \cap \mathcal{L}(q_j) = \mathcal{L}(q_j) \setminus \mathcal{L}(p_{3-i}) \\ &y \in \mathcal{L}(q_j) \setminus \mathcal{L}(p_i) = \mathcal{L}(q_j) \cap \mathcal{L}(p_{3-i}), \text{ and} \\ &z \in \mathcal{L}(q_{3-j}) = \mathcal{L}(q_{3-j}) \setminus \mathcal{L}(p_i) = \mathcal{L}(p_{3-i}) \setminus \mathcal{L}(q_j). \end{aligned}$$

The case where $\mathcal{L}(q_j) \subsetneq \mathcal{L}(p_i)$ holds is analogous.

Case 2: None of $\mathcal{L}(p_1)$, $\mathcal{L}(p_2)$, $\mathcal{L}(q_1)$, and $\mathcal{L}(q_2)$ are subsets of one another. Then, there are taxa x, y, z such that $x \in \mathcal{L}(p_1) \cap \mathcal{L}(q_1)$ $y \in \mathcal{L}(q_1) \setminus \mathcal{L}(p_1)$, and $z \in \mathcal{L}(q_1) \setminus \mathcal{L}(p_1)$. ◀

We can relate the two forms of "display" for triplets in non-binary trees.

▶ **Observation 8.** *Let T be a tree and let $a, b, c \in \mathcal{L}(T)$. Then,*
(a) *T firmly displays $ab|c$ if and only if* $\mathrm{LCA}(ab) <_T \{\mathrm{LCA}(ac), \mathrm{LCA}(bc)\}$.
(b) *T firmly displays $ac|b$ or $bc|a$ if and only if T does not softly display $ab|c$.*

▶ **Lemma 9.** *A tree T on X softly displays a tree T' on X $\Leftrightarrow$ for all $a, b, c \in X$,*

$$T \text{ firmly displays } ab|c \Rightarrow T' \text{ softly displays } ab|c, \text{ and}$$
$$T' \text{ firmly displays } ab|c \Rightarrow T \text{ softly displays } ab|c$$

Proof. "⇒": By Observation 4, it suffices to show the first of the claimed implications, so let $\mathrm{LCA}_T(ab) <_T \mathrm{LCA}_T(abc)$ and assume towards a contradiction that T' does not display $ab|c$. By Observation 8, we can suppose without loss of generality that T' firmly displays $ac|b$. But then, for $u := \mathrm{LCA}_T(ab)$ and $v := \mathrm{LCA}_{T'}(ac)$, we have $a \in \mathcal{L}(u) \cap \mathcal{L}(v)$, $b \in \mathcal{L}(u) \setminus \mathcal{L}(v)$, and $c \in \mathcal{L}(v) \setminus \mathcal{L}(u)$. Thus, by Lemma 7, T does not display T.

"⇐": Towards a contradiction, assume that T does not display T'. By Lemma 7, there are $u \in V(T)$ and $v \in V(T')$ and $a, b, c \in X$ such that $a \in \mathcal{L}(u) \cap \mathcal{L}(v)$, $b \in \mathcal{L}(u) \setminus \mathcal{L}(v)$, and $c \in \mathcal{L}(v) \setminus \mathcal{L}(u)$. Thus, $\mathrm{LCA}_T(ab) <_T \mathrm{LCA}_T(abc)$ and $\mathrm{LCA}_{T'}(ac) <_{T'} \mathrm{LCA}_{T'}(abc)$. By Observation 8, T firmly displays $ab|c$ and T' firmly displays $ac|b$. With the implications of the lemma, we get that T' softly displays $ab|c$ and T softly displays $ac|b$, contradicting Observation 8. ◀

The final ingredient to our alternative characterization is the observation that, in (multi-labeled) trees, edge contraction does not change the ancestor relation.

▶ **Observation 10.** *Let T be a tree, let T' be the result of contracting a vertex u onto its parent v, and let Y and Z be sets of leaves common to T and T'. Then,*
(a) $\mathrm{LCA}_T(Y) \leq_T \mathrm{LCA}_T(Z) \Leftrightarrow \mathrm{LCA}_{T'}(Y) \leq_{T'} \mathrm{LCA}_{T'}(Z)$ *and*
(b) $\mathrm{LCA}_T(Y) <_T \mathrm{LCA}_T(Z) \Leftarrow \mathrm{LCA}_{T'}(Y) <_{T'} \mathrm{LCA}_{T'}(Z)$.
We can now prove the following alternative definition of "display".

▶ **Lemma 11.** *Let T be a tree on the label-set X.*
(a) *T displays the leaf-triplet $ab|c$ if and only if* $\mathrm{LCA}(ab) \leq \{\mathrm{LCA}(bc), \mathrm{LCA}(ac)\}$.
(b) *T displays a binary tree T_B on X if and only if T displays all triplets displayed by T_B.*
(c) *T displays a tree T' on X (and vice versa) if and only if there is a binary tree T_B on X displayed by both T and T'.*
(d) *A network N displays T if and only if N contains (as subgraph) a tree T' on X that displays T.*

Proof. (a) By definition, T displays $ab|c$ if and only if there is a binary resolution T_B of T displaying $ab|c$. By Observation 5, T_B displays $ab|c$ if and only if $\mathrm{LCA}_{T_B}(ab) <_{T_B} \mathrm{LCA}_{T_B}(abc) = \mathrm{LCA}_{T_B}(ac) = \mathrm{LCA}_{T_B}(bc)$. Now, since T_B is binary, we cannot have that $\mathrm{LCA}_{T_B}(ab) = \mathrm{LCA}_{T_B}(bc) = \mathrm{LCA}_{T_B}(bc)$ and, thus, $\mathrm{LCA}_{T_B}(ab) \leq_{T_B} \{\mathrm{LCA}_{T_B}(ac), \mathrm{LCA}_{T_B}(bc)\}$ which, by Observation 10, is equivalent to $\mathrm{LCA}_T(ab) \leq_T \{\mathrm{LCA}_T(ac), \mathrm{LCA}_T(bc)\}$.

(b) "⇒": Assume towards a contradiction that a triplet $ab\,|\,c$ of T_B is not displayed by T and recall that $\{\mathrm{LCA}_T(ab), \mathrm{LCA}_T(ac), \mathrm{LCA}_T(bc)\}$ has a unique minimum x. Since, by (a), $\mathrm{LCA}_T(ab) \not\leq_T \mathrm{LCA}_T(abc)$, we have $x <_T \mathrm{LCA}_T(ab) \leq_T \mathrm{LCA}_T(abc)$. Without loss of generality, let $x = \mathrm{LCA}_T(ac)$. Then, by Observation 10, $\mathrm{LCA}_{T_B}(ac) <_{T_B} \mathrm{LCA}_{T_B}(abc)$, implying that T_B displays $ac\,|\,b$. Hence, T_B displays conflicting triples, contradicting Observation 5.

"⇐": Assume towards a contradiction that T does not display T_B. By Lemma 7, there are vertices $u \in V(T)$ and $v_B \in V(T_B)$ such that $\mathcal{L}(u)$ and $\mathcal{L}(v_B)$ intersect, but are not in the

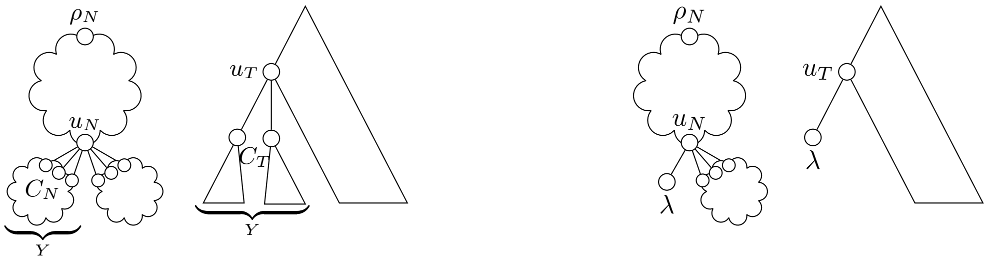


Figure 1 Illustration of Lemma 14: (N, T) left and (N_1, T_1) right.

subset relation, that is, there are $x \in \mathcal{L}(u) \setminus \mathcal{L}(v_B)$, $y \in \mathcal{L}(v_B) \setminus \mathcal{L}(u)$ and $z \in \mathcal{L}(u) \cap \mathcal{L}(v_B)$. Thus, $x, z <_T \mathrm{LCA}_T(xz) \leq_T u <_T \mathrm{LCA}_T(xyz)$ and $y, z <_{T_B} \mathrm{LCA}_{T_B}(yz) \leq_{T_B} v_B <_{T_B} \mathrm{LCA}_{T_B}(xyz)$. Then, by (a), T_B displays $yz \,|\, x$ implying that T displays $yz \,|\, x$ since all triplets displayed by T_B are displayed by T. By (a), we have $\mathrm{LCA}_T(yz) \leq_T \mathrm{LCA}_T(xz)$, implying $x, y, z <_T \mathrm{LCA}_T(xz) \leq_T u$, which contradicts $u <_T \mathrm{LCA}_T(xyz)$.

(c) By definition, T displays T' if and only if there are binary resolutions T_B and T'_B of T and T_B, respectively, such that T_B displays T'_B. Note that, if such trees exist then they are equal since, by (b), T_B displays all triplets displayed by T'_B and, by Observation 5, $T_B = T'_B$. Conversely, by Observation 3, all binary trees on X displayed by T and T' are binary resolutions of T and T'.

(d) We defer this proof to the full version of this paper. ◀

Note that, if N contains a subdivision S of T, then any reticulation in N that is in S has in- and out-degree one in S. Further, contracting an edge between two tree vertices of N cannot break softly displaying T.

▶ **Observation 12.** *Let N be a network that displays a tree T. Then, the result of contracting an edge between two tree-vertices or two reticulations of N displays T.*

Also note that, if N displays T, then the result of removing any label from N displays the result of removing this label from T.

▶ **Observation 13.** *Let N be a network and let T be a tree on X. Then, N displays T if and only if $N|_{X'}$ displays $T|_{X'}$ for each $X' \subseteq X$.*

3 Single-Labeled Trees

In a first step, we suppose that N is a tree. While Lemma 7 already provides the means to solve this case in polynomial time, we aim to be more efficient. If N and T are both binary, this special case is solved using the folklore "cherry reduction": remove a pair of leaves that are siblings in both N and T and label their parents in N and T with the same new label λ. Here, we prove an analog for non-binary trees that allows solving the case that N is a tree in linear time.

▶ **Lemma 14.** *Let N be a network on X with root ρ_N, let T be tree on X, let $u_N \in V(N)$ and $u_T \in V(T)$ and let C_N and C_T be sets of children of u_N and u_T, respectively, such that*
(a) $\bigcup_{c \in C_N} \mathcal{L}(c) = \bigcup_{c \in C_T} \mathcal{L}(c) =: Y$, *and*
(b) *for all $\lambda \in Y$, all ρ_N-λ-paths contain some $c \in C_N$.*
Let $\lambda \in Y$, let $N_1 := N||_{X \setminus (Y - \lambda)}$, let $T_1 := T||_{X \setminus (Y - \lambda)}$, let $N_2 := N||_Y$, and let $T_2 := T||_Y$. Then, N displays T if and only if N_1 displays T_1 and N_2 displays T_2 (see Figure 1).

Proof. Since "⇒" follows directly from Observation 13, we only show "⇐". By Lemma 11, for each $i \in \{1,2\}$, there is a tree Q_i in N_i (containing the root of N_i) that displays T_i and there is a binary tree T_i^B that is displayed by both Q_i and T_i. We show that the binary tree T_B resulting from replacing the leaf λ in T_1^B by T_2^B is displayed by both T and a subtree Q of N. To this end, note that T is the result of replacing the leaf λ in T_1 by T_2 and let Q be the result of replacing the leaf λ in Q_1 by Q_2. Since T_i^B is displayed by both T_i and Q_i for all $i \in \{1,2\}$, the following argument holds for both T and Q, but we only state it for T. To show that T displays T_B, it suffices to prove that T displays all triplets displayed by T_B (by Lemma 11(b)). Towards a contradiction, assume that T_B displays a triplet $xy|z$ that T does not display.

Case 1: $x, y \in Y$. If z is also in Y, then $xy|z$ is displayed by T_2^B and, thus, by T_2 and by T. If $z \notin Y$, then $\mathrm{LCA}_T(xy) \leq_T \mathrm{LCA}_T(Y) \leq_T u_T \leq_T \{\mathrm{LCA}_T(xz), \mathrm{LCA}_T(yz)\}$ by (a) (and (b) when arguing for Q instead of T) and, by Lemma 11(a), T displays $xy|z$.

Case 2: x or y is not in Y. Without loss of generality, let $x \notin Y$. If also $y \notin Y$, then λ can take the role of z in the assumption, that is, T_B displays $xy|\lambda$ but T does not. But then, T_B^1 displays $xy|\lambda$ but T_1 does not, contradicting the fact that T_1 displays T_B^1. Thus, $y \in Y$ and, completely analogously, $z \in Y$. But then, $\mathrm{LCA}_{T_B}(yz) \leq_{T_B} \mathrm{LCA}_{T_B}(Y) < \mathrm{LCA}_{T_B}(xy)$ which, by Lemma 11(a), contradicts T_B displaying $xy|z$.

Finally, let T^* be the result of contracting $\mathrm{LCA}_Q(Y)$ (that is, the former root of T_2^*) onto its parent in Q. Then, T^* is a subtree of N since N is (isomorphic to) the result of replacing ℓ by N_2 in N_1 and contracting the the root of N_2 onto its parent in the result. Since Q displays T_B, so does T^* (by Observation 12). Thus, T^* is a subtree of N that displays T and, by Lemma 11(d) N displays T. ◀

In the following, the operation of *splitting off* a subnetwork B with root u in a network N means to

- remove B and
- add a new leaf labeled $\lambda \notin X$ to u.

This gives rise to the networks N_1 (containing the new leaf λ) and $N_2 := B$. Lemma 14 implies correctness of the following reduction rule.

▶ **Reduction Rule 1.** *Let (N, T) be an instance of* Soft Tree Containment*, let B be a lowest biconnected component (such that B does not consist of a leaf and a non-leaf) or a cherry of N with root u. Then, split off B from N and split-off $T_{\mathrm{LCA}_T(\mathcal{L}(u))}$ from T (giving the new leaf in N and T the same new label λ).*

Note that Reduction Rule 1 can be applied exhaustively in linear time. This is because

(a) biconnected components can be found in linear time [12], and

(b) no biconnected component of N (except B) is modified by application of Reduction Rule 1.

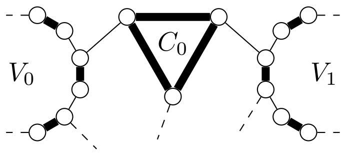
Now, if N is a (single-labeled) tree, then Reduction Rule 1 splits-off only cherries from N and each such cherry can be checked against the subtree split-off from T in linear time.

▶ **Theorem 15.** Soft Tree Containment *can be solved in linear time if N and T are trees.*

4 Tree Containment in Multilabeled Trees

To show that Soft Tree Containment is NP-hard even when restricting N to be a multilabeled tree, we reduce from 2-Union Independent Set, which asks if a graph $(V, E_1 \cup E_2)$ has a size-k independent set, and which is NP-hard even if (V, E_1) is a collection of disjoint K_2s (that is, a matching) and (V, E_2) is a collection of disjoint P_2s and P_3s [16]. For our reduction, we allow (V, E_1) to also contain K_3s and demand that k equals the number of cliques in (V, E_1). To prove that this variant remains NP-hard, we slightly modify the reduction from 3-SAT given by van Bevern et al. [16].

▶ **Construction 1.** *Consider an instance φ with n variables x_i and m clauses c_j of* 3-SAT *such that each variable occurs at least twice in φ and at most once in each clause. For each variable x_i, let J_i be the list of indices of clauses that contain x_i or $\neg x_i$ and let $J_i[\ell]$ denote the ℓ^{th} element of this list. Construct a graph (V, E) as follows. For each variable x_i, construct a cycle V_i of $2|J_i|$ vertices: $(u_i^1, \overline{u}_i^1, u_i^2, \overline{u}_i^2, \ldots)$. For each clause c_j on the variables x_i, x_k, x_ℓ, construct a triangle $C_j = (w_j^i, w_j^k, w_j^\ell)$. For each variable x_i and each $\ell \leq |J_i|$, connect $w^i_{J_i[\ell]}$ to $\overline{u}_i^\ell$ if $c_{J_i[\ell]}$ contains x_i, and to u_i^ℓ if $c_{J_i[\ell]}$ contains $\neg x_i$. Now, (V, E_1) (bold in the figure) consist of all triangles and all edges $\{u_i^j, \overline{u}_i^{j+1 \bmod |J_i|}\}$ while E_2 contains all other edges.*



Note that (V, E_1) consists of disjoint K_2s and K_3s and (V, E_2) consist exclusively of P_3s. Also note that this generalizes to k-SAT but (V, E_1) becomes a collection of disjoint K_2s and K_ks.

▶ **Lemma 16.** *φ is satisfiable if and only if (V, E) has a size-k independent set, where k is the number of cliques in (V, E_1).*

Proof. Note that
- k equals the number of cliques in (V, E_1),
- each clique contains at most one independent vertex, and
- all vertices in V are incident with some edge in E_1.

Hence, (V, E) contains a size-k independent set, if and only if a largest independent set in (V, E) contains exactly one vertex of each clique in (V, E_1). We will first show that if (V, E) contains an independent set of size k, then φ is satisfiable and afterwards the other direction.

"$\Leftarrow$": Let I be an independent set of size k in (V, E). Then, for each i, I contains either u_i^1 or $\overline{u}_i^1$. By construction of V_i, it holds that if $u_i^h \in I$ for some h, then $u_i^\ell, v_i^\ell \in I$ for all $\ell \leq |J_i|$. Analogously, if $\overline{u}_i^h \in I$ for some h, then $\overline{u}_i^\ell, \overline{v}_i^\ell \in I$ for all $\ell \leq |J_i|$. Consider any vertex w_j^i in the clause gadgets that is in I. Then, w_j^i has a unique neighbor in the variable gadget of x_j which is either u_j^h for some h if $\neg x_j$ occurs in clause i or $\overline{u}_j^\ell$ otherwise. If the neighbor is u_j^h, then all vertices $\overline{u}_j^\ell$ with $1 \leq \ell \leq |J_j|$ are in I and otherwise all vertices u_j^ℓ.

We set x_i to true if u_i^1 is in I and to false if $\overline{u}_i^1$ is in I. Consider any clause c_j in φ. The literal whose corresponding vertex is in I is then set to true as its neighboring vertex u is not in I and u has a neighbor u_i^h for some h if x_i occurs in c_j and a neighbor $\overline{u}_i^h$ for some h if $\neg x_i$ appears in c_j. Since each clause has at least one variable set to true, φ is satisfiable.

"$\Rightarrow$": We will now show that if φ is satisfiable, then (V, E) contains an independent set of size k. Let β be a satisfying assignment for φ. We construct an independent set I for (V, E) as follows. For each x_i and each $\ell \leq |J_i|$, the set I contains the vertices u_i^ℓ and v_i^ℓ if $\beta(x_i) = 1$, and the vertices $\overline{u}_i^\ell$ and $\overline{v}_i^\ell$, otherwise. For each clause c_j we pick one literal that is satisfied by our assignment of the variables and put the corresponding vertex into I. Observe that I is of size k as exactly one vertex of each clique in (V, E_1) is in I. Further, I is independent since, in each variable gadget, we pick every second vertex and, if a vertex

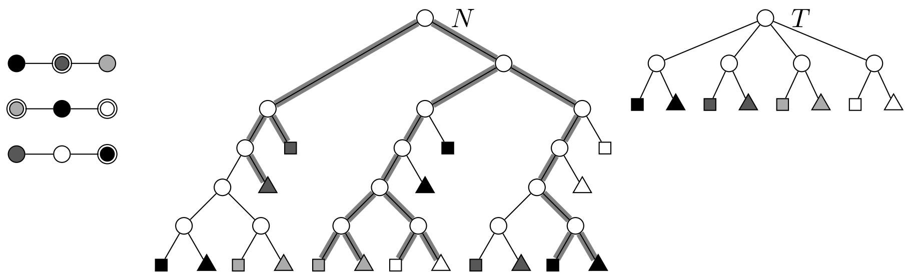
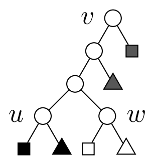


Figure 2 Illustration of Construction 2. **Left:** the initial instance of 2-Union Independent Set with 4 colors (●,●,●,○) and a size-4 solution encircled. **Right:** the non-binary tree T (boxes and triangles indicating label i_1 and i_2 for a color i). **Middle:** the binary multi-labeled tree N with a subdivision of T (bold, gray) corresponding to the solution to the left instance.

in a clause gadget is picked, then its neighbor in the corresponding variable gadget is not picked. ◀

We reduce this version of 2-Union Independent Set to Soft Tree Containment for multilabeled trees. To this end, we use an equivalent formulation where each clique in (V, E_1) is represented by a color. The problem then becomes the following: Given a vertex-colored collection of P_3s, select exactly one vertex per color such that all selected vertices are independent. Note that the number of occurrences of each color equals the size of its corresponding clique in (V, E_1).

▶ **Construction 2** (See Figure 2). *Given a vertex-colored collection G of P_3s constructed by Construction 1, we construct a multi-labeled tree N and a tree T as follows. Construct T by first creating a star that has exactly one leaf of each color occurring in G and then, for each leaf x with color i, adding two new leaves colored i_1 and i_2, respectively, and removing the color from x. Construct N from G as follows: For each P_3 (u, v, w) where black, gray, and white denote the colors of u, v, and w, respectively, construct the binary tree depicted below, where a box or a triangle colored i represents color i_1 or i_2, respectively. Then, add any binary tree on $|V(G)|$ leaves and identify its leaves with the roots of the constructed subtrees. Notice $u, v, w \in V(G) \cap V(N)$.*



▶ **Lemma 17.** *Construction 2 is correct, that is, N displays T if and only if the given collection G of P_3s has a colorful independent set using each color exactly once.*

Proof. Note that N is binary and let k be the number of colors in G.

"⇒": Let N display T, that is, N contains a binary tree S displaying T which, by Lemma 11 is equivalent to T displaying S. Consider any color i occurring in G. Then, S contains leaves u_1 and u_2 in S labeled i_1 and i_2, respectively, and we denote their least common ancestor in S by u^i. If u_1 and u_2 are neither siblings, nor in an uncle-nephew-relation[3], then we modify S to include the sibling/uncle of u_1 in N into S instead of u_2.

[3] Two vertices are in an *uncle-nephew relation* if the sibling of one is the parent of the other

Thus, we do not lose generality by assuming that u_1 and u_2 are either siblings or in an uncle-nephew-relation. We show that the set $Q = \bigcup_i u^i$ is a size-k colorful independent set in G. First, for each color i, we know that S contains exactly one leaf labeled i_1 and one leaf labeled i_2, so u^i is unique and, by construction of N, no two u^i coincide, implying that Q contains exactly one vertex of each color. Towards a contradiction, suppose that Q is not independent in G, that is, there are colors i and j such that u^i and u^j are adjacent in G. Without loss of generality, u^i is the center of a P_3 in G, implying that S contains the subtree $((((j_1, j_2), i_1), i_2)$ (that is, a caterpillar with leaves labeled j_1, j_2, i_1, i_2 in preorder). But then, $j_1 i_1 | i_2$ is displayed by S but not by T, thereby contradicting Definition 2(b).

"$\Leftarrow$": Let Q be a size-k colorful independent set of G, let L be the set of leaves that, for each $u \in Q$ of color i, contains the leaves labeled i_1 and i_2 in N_u, and let $S := N|_L$. Note that S is a subgraph of N and, as N is binary, S is a subdivision of a binary tree. Since Q contains exactly one vertex of each color in G, we know that S contains all labels that occur in T. By Definition 2(d), to show that N displays T, it suffices to show that S displays T. To this end, assume that S displays a triplet $xy|z$ that T does not display. Then Definition 2(a) lets us assume $\mathrm{LCA}_T(xz) <_T \{\mathrm{LCA}_T(xy), \mathrm{LCA}_T(yz)\}$ without loss of generality. Thus, $x = i_1$, $z = i_2$, and $y = j_1$ for colors $i \neq j$. By Definition 2(a), we have $\mathrm{LCA}_S(i_1 j_1) \leq_S \mathrm{LCA}_S(i_1 i_2)$. Then, i_1 and i_2 cannot form a cherry in S and, thus, $S|_{\{i_1,i_2,j_1,j_2\}}$ is the subtree $(((j_1, j_2), i_1), i_2)$. By construction of S, this implies that Q contains two vertices of a P_3 in G, one of color i and one of color j, and the latter is in the middle, contradicting independence of Q in G. ◀

▶ **Theorem 18.** Soft Tree Containment *is NP-hard, even if N is a binary 3-labeled tree.*

Note that the number of occurrences of each label in N equals the number of occurrences of each color in G which, in turn, equals the size of a largest clique in (V, E_1) (instance of 2-Union Independent Set), which equals the size of a largest clause (instance of 3-SAT), we can state the following generalization of Theorem 18.

▶ **Corollary 19.** *For each k, k-SAT reduces to* Soft Tree Containment *on binary k-labeled trees. Further, CNF-SAT reduces to* Soft Tree Containment *on binary multilabeled trees.*

Corollary 19 immediately raises the question of what happens in the case that N is a 2-labeled tree and we address this question in Section 4.1. Note that, for Soft Tree Containment, the case that N is a multilabeled tree reduces straightforwardly to the case that N is a reticulation-visible network, simply by merging all leaves with the same label i into one reticulation and adding a new child labeled i to it.

▶ **Corollary 20.** Soft Tree Containment *is NP-hard on reticulation-visible networks, even if the maximum in-degree is three and the maximum out-degree is two.*

Theorem 18 and Corollary 20 stand in contrast with results for (Strong) Tree Containment, which is linear-time solvable in both cases [18, 7].

4.1 2-Labeled Trees

In the following, N is a 2-labeled tree and T is a (single-labeled) tree. To solve Soft Tree Containment in this case, we compute a mapping $M : V(T) \to 2^{V(N)}$ such that $M(u)$ contains the at most two minima (with respect to $\leq_N$) among all vertices v of N such that N_v displays T_u. If N displays T, there is a single-labeled subtree S of N that displays T. If, for each $u \in V(T)$, we have $\mathrm{LCA}_S(\mathcal{L}(u)) \in M(u)$, then we call S *canonical* for T. We show that such a canonical subtree always exists.

▶ **Lemma 21.** *N displays T if and only if N has a canonical subtree for T.*

Proof. As "⇐" is evident, we just prove "⇒". To this end, let S be a single-labeled subtree of N that is a subdivision of T. If S is not canonical, then there is some $u \in V(T)$ with $x := \mathrm{LCA}_S(\mathcal{L}(u)) \notin M(u)$. Since S_x displays T_u, so does N_x. Thus, by definition of M, there is some $y \in M(u)$ with $y <_N x$ (recall that $x \notin M(u)$). But then, we can replace the subtree of S rooted at x with the unique x-y-path in N and the subtree of N_y displaying T_u. Iterating this construction yields a canonical subtree of N for T. ◀

To compute M, we consider vertices $u \in V(T)$ and $\rho \in V(N)$ in a bottom-up manner and check if N_ρ displays T_u. For each $v \in V(T_u)$ with parent p in T_u, each $x \in M(v)$ has at most one ancestor y in $M(p)$ since M contains only minima. For $v = u$, we let $y := \rho$. In both cases, we call the unique x-y-path in N_ρ the *ascending path* of x. A crucial lemma about ascending paths is the following.

▶ **Lemma 22.** *Let S be a canonical subtree of some N' for some T' and let $u, v \in V(T')$ not be siblings. Let $\mathrm{LCA}_S(\mathcal{L}(u))$ and $\mathrm{LCA}_S(\mathcal{L}(v))$ have ascending paths r and q, respectively. Then, r and q are edge-disjoint.*

Proof. Note that, if $u <_{T'} v$ then $\mathrm{LCA}_S(\mathcal{L}(p)) \leq_S \mathrm{LCA}_S(\mathcal{L}(v))$ where p is the parent of u in T'. Thus, the highest vertex of r (with respect to $\leq_{N_\rho}$) is a descendant of the lowest vertex of q and, hence, the lemma holds. Thus, we suppose in the following that u and v are incomparable in T'.

Towards a contradiction, assume that there is a vertex $z \in V(S)$ that is internal vertex of both r and q and, hence, is an ancestor of both u and v in T'. Then, $\mathcal{L}(u) \uplus \mathcal{L}(v) \subseteq \mathcal{L}(z)$. Further, since u and v are not siblings, one of u and v has a parent $p <_{T'} \mathrm{LCA}_{T'}(uv)$. Without loss of generality, let p be the parent of u, implying $\mathcal{L}(p) \cap \mathcal{L}(z) \supseteq \mathcal{L}(u) \neq \varnothing$ and $\mathcal{L}(z) \setminus \mathcal{L}(p) \supseteq \mathcal{L}(v) \neq \varnothing$. Since S is canonical, we have $\mathrm{LCA}_S(\mathcal{L}(p)) \in M(p)$ and, thus, the ascending path r of u ends in $\mathrm{LCA}_S(\mathcal{L}(p))$. Hence, as z is an internal vertex of r, it holds that $z <_S \mathrm{LCA}_S(\mathcal{L}(p))$, implying $\mathcal{L}(p) \setminus \mathcal{L}(z) \neq \varnothing$. Since S displays T', the three established relations between $\mathcal{L}(p)$ and $\mathcal{L}(z)$ contradict Lemma 7. ◀

Clearly, N displays T if and only if $M(\rho_T) \neq \varnothing$, where ρ_T is the root of T. Further, computation of $M(u)$ is trivial if u is a leaf. Thus, in the following, we show how to compute $M(u)$ given $M(v)$ for all $v \in V(T_u) - u$.

In a first step, compute $N|_L$ where L is the set of leaves of N whose label occurs in T_u. Then, we know that $M(v) \subseteq V(N|_L)$ for all $v \in V(T_u)$. Second, we mark all vertices ρ in $N|_L$ such that, for each child u_i of u in T, there is some $x_i \in M(u_i)$ with $x_i \leq_{N|_L} \rho$. For each marked vertex ρ in a bottom-up manner, we test whether N_ρ displays T_u using the following formulation as a 2-SAT problem[4].

▶ **Construction 3.** *Construct $\varphi_{u\to\rho}$ as follows. For each $v \in V(T_u) - u$,*

(i) *for each $y \in M(v)$, introduce a variable $x_{v\to y}$.*
(ii) *add the clause $\bigoplus_{y\in M(v)} x_{v\to y}$ (recall that $|M(v)| \leq 2$).*
(iii) *if the parent p of v in T_u is not u then, for all $y \in M(v)$ and all $z \in M(p)$ with $y \not\leq_N z$, add the clause $x_{v\to y} \Rightarrow \neg x_{w\to z}$.*
(iv) *for each $w \in V(T_u) - u$ that is not a sibling of v and each $y \in M(v)$ and each $z \in M(w)$ such that the ascending paths of y and z share an edge, add the clause $x_{v\to y} \Rightarrow \neg x_{w\to z}$.*

[4] We are using the XOR operation ($(x \oplus y) := (x \vee y) \wedge (\neg x \vee \neg y)$) as well as implications ($(x \Rightarrow y) := (\neg x \vee y)$) in the construction, which can be formulated as clauses with two variables as shown.

By definition of $M(u)$, no two vertices in $M(u)$ can be in an ancestor-descendant relation. Thus, we can ignore all ancestors of a vertex ρ that satisfies $\varphi_{u\to\rho}$ and we can assume that no strict ancestor of our current ρ satisfies $\varphi_{u\to z}$.

▶ **Lemma 23.** *$\varphi_{u\to\rho}$ is satisfiable if and only if N_ρ displays T_u.*

Proof. "$\Leftarrow$": Let S be a canonical subtree of N_ρ for T_u and let β be an assignment for $\varphi_{u\to\rho}$ that sets each $x_{v\to y}$ to 1 if and only if $y = \mathrm{LCA}_S(\mathcal{L}(v))$. Since the LCA of $\mathcal{L}(v)$ in S is unique, all clauses of type (ii) are satisfied by β. If a clause of type (iii) is not satisfied, then there is some v with parent p in T_u such that $y \leq_N z$ for some $y \in M(v)$ and $z \in M(p)$ and $\beta(x_{v\to y}) = 1$ and $\beta(x_{p\to z}) = 0$. Let $z' \in M(p) - z$ with $\beta(x_{p\to z'}) = 1$, which exists since all clauses of type (ii) are satisfied. Since $\mathcal{L}(p) \supseteq \mathcal{L}(v)$, we know that $y \leq_S z'$ and, as S is a subtree of N, we have $y \leq_N z'$, implying $z \leq_N z'$ or $z' \leq_N z$, which contradicts the construction of M. If a clause of type (iv) is not satisfied, then there are $x_{v\to y}$ and $x_{w\to z}$ such that v and w are not siblings in T, $\beta(x_{v\to y}) = \beta(x_{w\to z}) = 1$, and the ascending paths of $y = \mathrm{LCA}_S(\mathcal{L}(v))$ and $z = \mathrm{LCA}_S(\mathcal{L}(w))$ share an edge. But this contradicts Lemma 22.

"$\Rightarrow$": Let β be a satisfying assignment for $\varphi_{u\to\rho}$. Let $\psi \subseteq V(T) \times V(N)$ be a relation such that $(v, y) \in \psi$ if and only if $\beta(x_{v\to y}) = 1$. Since β satisfies the clauses of type (ii), ψ describes a function and, slightly abusing notation, we call this function ψ. Let Y be the image of ψ and let $S := N|_{Y\cup\{\rho\}}$. Note that, for all $v <_T u$ with parent $p \neq u$, we know that $\psi(v) \leq_N \psi(p)$, since β satisfies the clauses of type (iii). Thus, for all $v, w \in V(T_u) - u$, we have $w \leq_T v \Rightarrow \psi(w) \leq_N \psi(v) \Rightarrow \psi(w) \leq_S \psi(v)$ We show for all $(v, y) \in \psi \cup \{(u, \rho)\}$ that $y = \mathrm{LCA}_S(\mathcal{L}(v))$ and S_y is a canonical subtree of N_y for T_v. The proof is by induction on the height of v in T. Clearly, if v is a leaf, y is a leaf with the same label and the claim follows. Otherwise, suppose that the claim holds for all $w <_T v$. Towards a contradiction, assume that S_y does not display T_v. By Lemma 7, there are $w \in V(T_v)$ and $z \in V(S_y)$ such that there are leaves $a \in \mathcal{L}(z) \setminus \mathcal{L}(w)$, $b \in \mathcal{L}(w) \setminus \mathcal{L}(z)$, and $c \in \mathcal{L}(w) \cap \mathcal{L}(z)$. Note that $\mathrm{LCA}_T(bc) \leq_T w <_T \{\mathrm{LCA}_T(ab), \mathrm{LCA}_T(ac)\}$. Let α be the highest ancestor of a in T with $b \not\leq_T \alpha$ and let p_α be its parent in T. Let γ be the highest ancestor of c in T with $b \not\leq_T \gamma$ and let p_γ be its parent in T. Since $b, c <_T w$ and $a \not\leq_T w$, we know that $p_\gamma <_T p_\alpha$, implying that α and γ are not siblings in T. Then, as $\mathrm{LCA}_S(ac) \leq_S z <_S \{\mathrm{LCA}_S(ab), \mathrm{LCA}_S(bc)\}$, $\mathrm{LCA}_S(ab) \leq_S \psi(p_\alpha)$, and $\mathrm{LCA}_S(bc) \leq_S \psi(p_\gamma)$, we know that the ascending paths of $\psi(\alpha)$ and $\psi(\gamma)$ share an edge, contradicting (iv). ◀

▶ **Theorem 24.** Soft Tree Containment *can be solved in $O(n^3)$ time on instances (N, T) for which N is a 2-labeled tree.*

Proof. As correctness follows from Lemma 23, we only show the running time. To this end, note the $N|_L$ can be computed in $O(|L|) = O(|\mathcal{L}(u)|)$ time (see, for example [3, Section 8]). To mark all vertices of $N|_L$ that, for each child u_i of u in T, have an ancestor in $M(u_i)$, we compute the restriction of $N|_L$ to $\bigcup_i M(u_i)$. Again, this can be done in $O(\deg_T(u))$ time. For each vertex in this restriction, we can store the set of leaves that descend from it. In a bottom-up manner, we can thus mark the correct vertices in $O(\deg_T(u)^2)$ time.

We construct $\varphi_{u\to\rho}$ for each pair (u, ρ) as follows. To check $y \not\leq_N z$ efficiently in Construction 3(iii), we can prepare a 0/1-matrix with an entry for each pair of vertices in N. This table has size $O(n^2)$ and can be computed in the same time by a simple bottom-up scan of N. To construct the clauses of type (iv), we first order the vertices in N_ρ. For each v in this order, we construct its ascending path in $O(|N_\rho|)$ time and store v in all edges on this path. Thus, when constructing the clauses of type (iv) for a vertex v, we can merge the lists of vertices whose ascending path shares an edges with that of v. Thus, $\varphi_{u\to\rho}$ can be constructed and solved in $O(|N_\rho|^2) = O(|\mathcal{L}(u)|^2)$ time and the total time to decide whether N displays T is $O(\sum_{u\in V(T)} |\mathcal{L}(u)|^2) = O(n^3)$. ◀

Theorem 24 implies [5] that we can solve bifurcating reticulation-visible networks in polynomial time, complementing Corollary 20.

▶ **Corollary 25.** Soft Tree Containment *can be solved in* $O(n^3)$ *time on reticulation-visible networks of in-degree at most two.*

5 Conclusion

We introduced a practically relevant variant of the Tree Containment problem handling soft polytomies and showed that its (classical) complexity depends heavily on the maximum in-degree in the network. Multiple avenues are opened for future work. Motivated by our hardness result, the search for parameterized or approximative algorithms is a logical next step. Previous work for Tree Containment [8, 18] might lend promising ideas and parameterizations to this effort. While multi-labeled trees were our starting point to analyze Soft Tree Containment, only the hardness result (Theorem 20) is transferable to multi-labeled networks, leaving many open questions in this direction. Finally, given the close relationship to Cluster Containment, (see Section 1), we hope to apply ideas and methods used there to also attack Soft Tree Containment. In particular, we hope that the ideas in Theorem 24 can be adapted since Cluster Containment seems to exhibit a close relationship to SAT [9]—similar to what we exploited to prove Theorem 24.

References

1 Magnus Bordewich and Charles Semple. Reticulation-visible networks. *Advances in Applied Mathematics*, 78:114–141, 2016.

2 Joseph Minhow Chan, Gunnar Carlsson, and Raul Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences*, 110(46):18566–18571, 2013.

3 Richard Cole, Martin Farach-Colton, Ramesh Hariharan, Teresa Przytycka, and Mikkel Thorup. An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. *SIAM Journal on Computing*, 30(5):1385–1404, 2000.

4 A Dress, Katharina Huber, J Koolen, Vincent Moulton, and A Spillner. *Basic Phylogenetic Combinatorics*. Cambridge University Press, 2004.

5 Jittat Fakcharoenphol, Tanee Kumpijit, and Attakorn Putwattana. A faster algorithm for the tree containment problem for binary nearly stable phylogenetic networks. In *12th International Joint Conference on Computer Science and Software Engineering (JCSSE'15)*, pages 337–342. IEEE, 2015.

6 Philippe Gambette, Andreas D. M. Gunawan, Anthony Labarre, Stéphane Vialette, and Louxin Zhang. Locating a tree in a phylogenetic network in quadratic time. In *Proceedings of the 19th Annual International Conference on Research in Computational Molecular Biology (RECOMB'15)*, volume 9029 of *LNCS*, pages 96–107. Springer, 2015.

7 Andreas D. M. Gunawan. Solving tree containment problem for reticulation-visible networks with optimal running time. *CoRR*, abs/1702.04088, 2017.

8 Andreas D. M. Gunawan, Bingxin Lu, and Louxin Zhang. A program for verification of phylogenetic network models. *Bioinformatics*, 32(17):i503–i510, 2016.

9 Andreas D. M. Gunawan, Bingxin Lu, and Louxin Zhang. Fast methods for solving the cluster containment problem for phylogenetic networks. *CoRR*, 1801.04498, 2018.

[5] See [18] for the corresponding reduction.

10 Andreas D.M. Gunawan, Bhaskar DasGupta, and Louxin Zhang. A decomposition theorem and two algorithms for reticulation-visible networks. *Information and Computation*, 252:161–175, 2017.

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

12 John Hopcroft and Robert Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, 1973.

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

14 Iyad A Kanj, Luay Nakhleh, Cuong Than, and Ge Xia. Seeing the trees and their branches in the network is hard. *Theoretical Computer Science*, 401(1-3):153–164, 2008.

15 Todd J Treangen and Eduardo PC Rocha. Horizontal transfer, not duplication, drives the expansion of protein families in prokaryotes. *PLoS Genet*, 7(1):e1001284, 2011.

16 René van Bevern, Matthias Mnich, Rolf Niedermeier, and Mathias Weller. Interval scheduling and colorful independent sets. *J. Scheduling*, 18(5):449–469, 2015. `doi:10.1007/s10951-014-0398-5`.

17 Leo Van Iersel, Charles Semple, and Mike Steel. Locating a tree in a phylogenetic network. *Information Processing Letters*, 110(23):1037–1043, 2010.

18 Mathias Weller. Linear-time tree containment in phylogenetic networks. *CoRR*, 1702.06364, 2017.