Tree Containment With Soft Polytomies

Matthias Bentert, Josef Malík, Mathias Weller

Tree Containment Wih Soft Polytomies

Matthias Bentert[1], Josef Malík[2], and Mathias Weller[3]

[1]TU Berlin, Germany, matthias.bentert@tu-berlin.de
[3]Czech Technical University, Prague, Czech Republic, malikjo1@fit.cvut.cz
[3]LIGM, CNRS, Université Paris Est, Marne-la-Vallée, France, mathias.weller@u-pem.fr

March 14, 2018

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 $O(|T|^3)$.

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 desceibe certain processes that have been observed in nature. In the last decade, the idea of reticulate evolution, supporting events such as hybridization (occurring frequently in plants) and horizontal gene transfer (a dominating factor in bacterial evolution) [2, 14]. Reticulate evolution is described using "phylogenetic networks" (see the monographs by Gusfield [10] and Huson et al. [12]). 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 [13, 16]. Due to its importance in the analysis of evolutionaryr history, attempts have been made to identify polynomial-time computable special cases [1, 5, 6, 8, 9, 13, 16, 17], as well as moderately exponential-time algorithms [7, 17]. 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) occuring 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". 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 (with we call Soft Tree Containment). We show that it is $\mathcal{NP}$-hard, even if the network is "reticulation-visible" and all reticulation vertices have at most three parents, but if we lower this bound to two, the problem becomes solvable in cubic time.

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. The former vertices are called *tree vertices* and the latter are called *reticulations*. 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) interchangably. For brevity, we abbreviate $\{x, y\}$ to xy, and $\{x, y, z\}$ to xyz. Let N be a network with root ρ_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 u-w-path for each $u \in U$ and $w \in W$. If $u \leq_N w$, we call u an *ancestor* of v and v a *descendant* 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 U by $N|_L$ and $N||_L$ is the result of supressing all degree-two vertices in $N|_L$. 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*. *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. 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[1] 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 central to this work.

Definition 1. *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 1 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

[1] 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.

T, the two concepts coincide. Furthermore, for trees on the same label-set, the concepts of display and binary resolution coincide.

Observation 2. *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.

Observation 3. *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 4 ([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 develope alternative characterizations of the term "(softly) display". To do this, we use the following handy characterization of isomorphism for binary trees.

Observation 5. *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 1. *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. Observe that, 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.

"$\Rightarrow$": 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.

"$\Leftarrow$": 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 5, 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 of q, 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 $\mathcal{L}(q_2)$ is subset of 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 6. *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 2. *A tree T on X softly displays a tree T' on $X \Leftrightarrow$ for all $a, b, c \in X$,*
T firmly displays $ab|c \Rightarrow T'$ softly displays $ab|c$, and
T' firmly displays $ab|c \Rightarrow T$ softly displays $ab|c$

Proof. "⇒": By Observation 3, 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 6, 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 1, T does not display T.

"⇐": Towards a contradiction, assume that T does not display T'. By Lemma 1, 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 6, T firmly displays $ab|c$ and T' firmly displays $ac|b$. Applying the implications of the lemma, we get that T' softly displays $ab|c$ and T softly displays $ac|b$, thereby contradicting Observation 6. □

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

Observation 7. *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 3. *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 4, 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 $\mathrm{LCA}_{T_B}(ab) = \mathrm{LCA}_{T_B}(bc) = \mathrm{LCA}_{T_B}(bc)$ and, thus, $\mathrm{LCA}_{T_B}(ab) \leq_{T_B} \{\mathrm{LCA}_{T_B}(ac), \mathrm{LCA}_{T_B}(bc)\}$ which, by Observation 7, 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 7, $\mathrm{LCA}_{T_B}(ac) <_{T_B} \mathrm{LCA}_{T_B}(abc)$, implying that T_B displays $ac\,|\,b$. Hence, T_B displays conflicting triples, contradicting Observation 4.

"⇐": Assume towards a contradiction that T does not display T_B. By Lemma 1, 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 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,

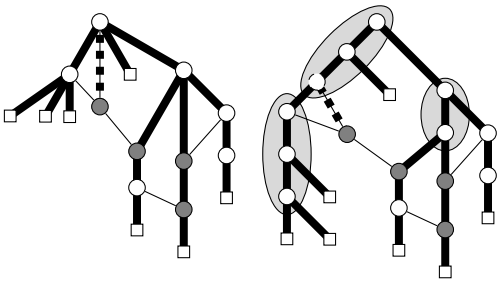


Figure 1: Illustration of the proof of Lemma 3(d). Left: N with T' (bold) and T^* (bold and dashed). Right: N_B with T_B (bold) and T_B^* (bold and dashed). χ maps each vertex to itself except the encircled ones that are mapped together.

by (b), T_B displays all triplets displayed by T'_B and, by Observation 4, $T_B = T'_B$. Conversely, by Observation 2, all binary trees on X displayed by T and T' are binary resolutions of T and T'.

(d). "⇒": By definition, there are binary resolutions N_B and T_B of N and T, respectively, such that N_B displays T_B, that is, there is a subdivision S_B of T_B that is a subgraph of N_B. Let $\chi : V(N_B) \to V(N)$ be the function mapping each vertex u in N_B to the vertex $\chi(u)$ in N that u is contracted to when forming N. Note that, for all vertices $w \in V(N)$, the vertices in $\chi^{-1}(w)$ form a connected component in N_B. To show that, the vertices in $\chi^{-1}(w)$ also form a connected component in S_B, assume that there are vertices u and v in S_B with $\chi(\mathrm{LCA}_{S_B}(uv)) \neq \chi(u) = \chi(v) =: w$. Clearly, u and v are not in the ancestor relation in S_B. Further, by Observation 1, $\chi^{-1}(w)$ contains only reticulations (and degree-two vertices) or only tree vertices of N_B. Since $\chi^{-1}(w)$ induces a connected component in N_B, the latter case implies $\chi(\mathrm{LCA}_{S_B}(uv)) = \chi(u) = \chi(v)$. In the former case, there is some $\gamma \in \chi^{-1}(w)$ with $\gamma <_{N_B} uv$ (note that $\gamma \in uv$ contradicts $\mathrm{LCA}_{S_B}(uv) \notin uv$) and, since S_B is a tree, it cannot contain all edges between γ and u and v. Thus, u or v does not have a labeled leaf below it in S_B. The vertices in $\chi^{-1}(w)$ therefore form a connected component in S_B for each $w \in V(N)$. Since (a) $\chi^{-1}(w)$ in S_B is a subset of $\chi^{-1}(w)$ in N_B and contracting it in N_B yields N and (b) $\chi^{-1}(w)$ is a connected component in S_B it follows that the result T' of contracting $\chi^{-1}(w)$ for each $w \in V(N)$ to a single vertex in S_B is a subgraph of N. Furthermore, S_B is a binary resolution of T' displaying T_B, which is a binary resolution of T and, by (c), T' displays T.

"⇐": By (c), there is a binary tree T_B on X displayed by both T' and T. We construct a binary resolution N_B of N such that N_B displays T_B which, by Observation 2, is a binary resolution of T (see Figure 1). To this end, let T^* be any spanning subgraph of N that is a tree and contains T' as a subgraph, and let $Y := E(N) \setminus E(T^*)$ be the set of edges in N that are missing in T^*. Since T_B is a binary resolution of T', there is a binary resolution T_B^* of T^* that contains a subdivision of T_B. Finally, we construct N_B from T_B^* by adding representations of all edges $uv \in Y$. To this end, let $\chi : V(T_B^*) \to V(T^*)$ be a function mapping each vertex x of T_B^* to a vertex of T^* such that T^* can be obtained from T_B^* by contracting all edges uv in T_B^* with $\chi(u) = \chi(v)$. Note that χ is surjective. Let $uv \in Y$. In order to keep T_B^* binary while adding a representation of uv, we will first create two new vertices u_B and v_B in T_B^* and then add the edge $u_B v_B$. If $|\chi^{-1}(u)| = 1$, then let y be the unique vertex in $\chi^{-1}(u)$ and, if y is the root of T_B^*, then add a new root u_B to T_B^*, and make y its only child, otherwise subdivide the edge between y and its parent with a new vertex u_B. In both cases, add u_B to $\chi^{-1}(u)$. If $|\chi^{-1}(u)| > 1$, then subdivide any edge between vertices in $\chi^{-1}(u)$, call the new vertex u_B and add it to $\chi^{-1}(u)$. Then, construct a new vertex v_B corresponding to v in an analogous way and add the edge $u_B v_B$. Let N_B denote the result of repeating this operation for all edges $uv \in Y$. Since N_B results from T_B^* by a series of subdivisions and edge additions, we know that N_B contains a subdivision of T_B^* and, thus, displays T_B. It remains to show that N_B is a binary resolution of N, that is, N is a contraction of N_B. Indeed, we show that N is equal to the result N^* of contracting all $u, v \in V(N_B)$ with $\chi(u) = \chi(v)$. First, since T^* spans N, the image of χ equals $V(N)$ and, thus, $V(N^*) = V(N)$. Second, assume that N^* contains an edge xy that is not in N and, thus, not in T^*. But then, there is an edge $x_B y_B \in E(N_B^*)$ such that $\chi(x_B) = x$ and $\chi(y_B) = y$. Since xy is not in T^*, we know that $x_B y_B$ is not in T_B^*, so $x_B y_B$ has been added by the procedure above. By construction, there is an edge uv in N such that $x_B \in \chi^{-1}(u)$ and $y_B \in \chi^{-1}(v)$, implying that $u = x$ and $v = y$, contradicting xy not being an edge of N. Third, assume that N contains an edge xy that is not in N^*. Then, xy is not in T^*. Thus, by the construction above, N_B contains vertices $x_B \in \chi^{-1}(x)$ and $y_B \in \chi^{-1}(y)$ such that $x_B y_B$ is an edge of N_B. But then, xy is an edge of N^*. □

Figure 2: Illustration of Lemma 4: (N, T) left and (N_1, T_1) right.

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 8. *Let N be a network that displays a tree T and let N' be the result of contracting an edge between two tree-vertices or two reticulations of N. Then, 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 9. *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 1 already provides a 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 4. *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.

Proof. Since "$\Rightarrow$" follows directly from Observation 9, we only show "$\Leftarrow$". By Lemma 3, 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 3(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 3(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,

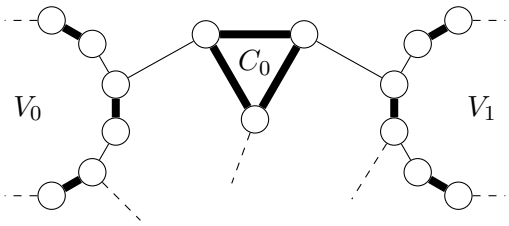


Figure 3: Illustration of Construction 1. Bold edges indicate E_1.

$z \in Y$. But then, $\mathrm{LCA}_{T_B}(yz) \leq_{T_B} \mathrm{LCA}_{T_B}(Y) < \mathrm{LCA}_{T_B}(xy)$ which, by Lemma 3(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 8). Thus, T^* is a subtree of N that displays T and, by Lemma 3(d) N displays T. □

In the following, the operation of *splitting off* a vertex u in a network N means to duplicate u, remove all incoming edges from one copy of u and remove all outgoing edges from the other copy of u, thus giving rise to N_1 (where a copy of u is a leaf) and N_2 (where a copy of u is the root). Lemma 4 gives rise to the following reduction rule.

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

Note that Reduction Rule 1 Further, note that the vertex u is a cut-vertex in N. Thus, the biconnected components of N are not modified by application of Reduction Rule 1 and, since biconnected components can be found in linear time [11], we conclude that Reduction Rule 1 can be exhaustively applied in linear time.

Theorem 1. Weak Tree Containment *can be solved in linear time if N and T are trees.*

4 Tree Containment in Multilabeled Trees

To show that 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 [15]. For our reduction, we allow (V, E_2) 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. [15].

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 $w_{J_i[\ell]}^i$ 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) consist of all triangles and all edges $\{u_i^j, \overline{u}_i^j\}$ while E_2 contains all other edges. See Figure 3 for an illustration.*

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 5. *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).*

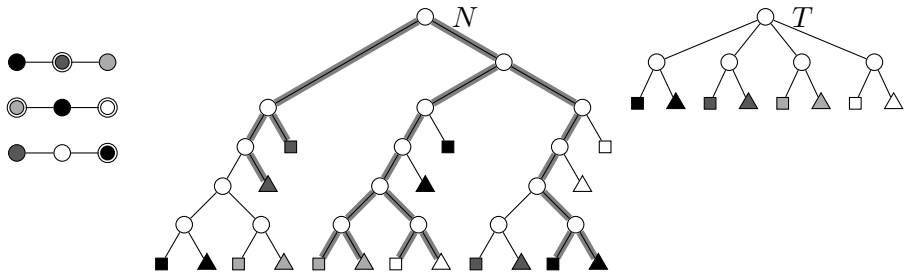


Figure 4: Illustration of Construction 2. **Left:** the initial instance of 2-Union Independent Set with 4 colors (● , ● , ● , ○) and a size-4 solution encircled. **Right:** the non-binary tree T (boxes and triangles indicating label i_1 and i_2 for a color i). **Middle:** the binary multi-labeled tree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

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, E) are incident with some edge in E_1. Therefore, (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.

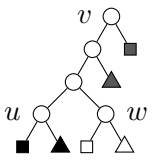
"$\Leftarrow$": Let I be an independent set of size k in (V, E). Then, for each i, I contains either u_i^1 or $\overline{u}_i^1$. By construction of V_i, it holds that if $u_i^h \in I$ for some h, then $u_i^\ell, v_i^\ell \in I$ for all $\ell \leq |J_i|$. Analogously, if $\overline{u}_i^h \in I$ for 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 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 occurances of each color equals the size of its corresponding clique in (V, E_1).

Construction 2 (See Figure 4)**.** *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 occuring 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 on the right, 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 6. *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 3 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[2], then we modify S to include the sibling/uncle of u_1 in N into S instead of u_2. 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 1(b).

"⇐": 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 1(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 1(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 1(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 2. Soft Tree Containment *is NP-hard, even if N is a binary 3-labeled tree.*

Note that the number of occurances of each label in N equals the number of occurances 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 2.

Corollary 1. *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 1 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 2. Soft Tree Containment *is NP-hard on reticulation-visible networks, even if the maximum in-degree is three.*

Theorem 2 and Corollary 2 stand in contrast with results for (Strong) Tree Containment, which is linear-time solvable in both cases [9, 17].

4.1 2-Labeled Trees

In the following, N is a 2-labeled tree and T is a (single-labeled) tree. To solve 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

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

(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 7. *N displays T if and only if N has a canonical subtree for T.*

Proof. As "$\Leftarrow$" is evident, we just prove "$\Rightarrow$". 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, checking 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 cruicial lemma about ascending paths is the following.

Lemma 8. *Let S be a canonical subtree of N' for 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 point of r is below the lowest point of q and the lemma holds. This allows us to 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 1. □

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[3].

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 $x_{v\to y} \Rightarrow \neg x_{w\to z}$.

By definition of $M(u)$ it holds that no two vertices in $M(u)$ can be in an ancestor-descendant relation. For this reason we can ignore all ancestors of a vertex ρ that satisfies $\varphi_{u\to\rho}$.

Lemma 9. *$\varphi_{u\to\rho}$ is satisfiable if and only if N_ρ displays T_u.*

[3]Note that 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.

Proof. "⇐": 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 8.

"⇒": 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

$$w \leq_T v \Rightarrow \psi(w) \leq_N \psi(v) \Rightarrow \psi(w) \leq_S \psi(v) \tag{1}$$

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 1, 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 $\mathrm{LCA}_S(ac) \leq_S z <_S \{\mathrm{LCA}_S(ab), \mathrm{LCA}_S(bc)\}$ and $\mathrm{LCA}_S(ab) \leq_S \psi(p_\alpha)$ 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 3. *On 2-labeled trees,* Soft Tree Containment *can be solved in $O(n^3)$ time.*

Proof. As correctness follows from Lemma 9, 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 linear time, that is, $O(\deg_T(u))$. 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 3 implies [4] that we can solve bifurcating reticulation-visible networks in polynomial time, complementing Corollary 2

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

[4] See [17] for a the corresponding reduction.

Remarks: in practice, it will be much faster to not consider the whole subtree $N|_L$ but only $N||_Q$ where $Q := \bigcup_i M(u_i)$. In general though, vertices of Q might be mutually incompatible, that is, two children u_1 and u_2 of u might map to $y_1 \in M(u_1)$ and $y_2 \in M(u_2)$ with $y_1 \leq_N y_2$. If this occurs, it is unclear whether or not both T_{u_1} and T_{u_2} can be displayed by N_{y_2} at the same time or not, as this depends on the ascending paths of the vertices in N that the children of u_1 and u_2 map to. Thus, we would have to "develope" y_1 and y_2, where developing y_1 means to replace y_1 with the subtree rooted at y_1 restricted to $\bigcup_i M(u_{1,i})$ where $u_{1,i}$ are the children of u_1 in T. In theory, we might face another incompatibility at the level of the children of u_1 and this may, in theory, continue down to the leaves of T. Thus, in a worst-case consideration, we can just start with the full subtree $N|_L$ while in practice, the above strategy of developing incompatabilities will be much faster.

Acknowledgements. The project leading to this work was conceived on the 2017 research retreat of the algorithms and complexity group of TU-Berlin.

References

[1] M. Bordewich and C. Semple. Reticulation-visible networks. *Advances in Applied Mathematics*, (78):114–141, 2016.
[2] J. M. Chan, G. Carlsson, and R. Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences*, 110(46):18566–18571, 2013.
[3] R. Cole, M. Farach-Colton, R. Hariharan, T. Przytycka, and M. 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, K. Huber, J. Koolen, V. Moulton, and A. Spillner. *Basic Phylogenetic Combinatorics.* Cambridge University Press, 2004.
[5] J. Fakcharoenphol, T. Kumpijit, and A. 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] P. Gambette, A. D. M. Gunawan, A. Labarre, S. Vialette, and L. Zhang. Locating a tree in a phylogenetic network in quadratic time. volume 9029 of *LNCS*, pages 96–107. Springer, 2015.
[7] A. D. Gunawan, B. Lu, and L. Zhang. A program for verification of phylogenetic network models. *Bioinformatics*, 32(17):i503–i510, 2016.
[8] A. D. Gunawan, B. DasGupta, and L. Zhang. A decomposition theorem and two algorithms for reticulation-visible networks. *Information and Computation*, (252):161–175, 2017.
[9] A. D. M. Gunawan. Solving tree containment problem for reticulation-visible networks with optimal running time. *CoRR*, abs/1702.04088, 2017.
[10] D. Gusfield. *ReCombinatorics: the algorithmics of ancestral recombination graphs and explicit phylogenetic networks.* MIT Press, 2014.
[11] J. Hopcroft and R. Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6): 372–378, June 1973. ISSN 0001-0782.
[12] D. H. Huson, R. Rupp, and C. Scornavacca. *Phylogenetic networks: concepts, algorithms and applications.* Cambridge University Press, 2010.
[13] I. A. Kanj, L. Nakhleh, C. Than, and G. Xia. Seeing the trees and their branches in the network is hard. *Theoretical Computer Science*, 401(1-3):153–164, 2008.
[14] T. J. Treangen and E. P. Rocha. Horizontal transfer, not duplication, drives the expansion of protein families in prokaryotes. *PLoS Genet*, 7(1):e1001284, 2011.
[15] R. van Bevern, M. Mnich, R. Niedermeier, and M. Weller. Interval scheduling and colorful independent sets. *J. Scheduling*, 18(5):449–469, 2015. doi: 10.1007/s10951-014-0398-5.
[16] L. Van Iersel, C. Semple, and M. Steel. Locating a tree in a phylogenetic network. *Information Processing Letters*, 110(23):1037–1043, 2010.
[17] M. Weller. Linear-time tree containment in phylogenetic networks. *CoRR*, abs/1702.06364, 2017.