Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 25, no. 1, pp. 417–436 (2021)
DOI: 10.7155/jgaa.00565

Tree Containment With Soft Polytomies

Matthias Bentert[1] *Mathias Weller*[2,3]

[1]Institut für Softwaretechnik und Theoretische Informatik, TU-Berlin, Germany
[2]LIGM, Université Gustave Eiffel, Paris, France
[3]Centre National de la Recherche Scientifique, France

Submitted: February 2020 Reviewed: July 2020 Revised: July 2020
Accepted: March 2021 Final: July 2021 Published: September 2021
Article type: Regular paper Communicated by: F. Vandin

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 NP-complete, 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 NP-completeness and polynomial-time solvability on reticulation-visible networks in which the maximum in-degree is bounded by three and two, respectively.

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 [3, 17]. 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 [14] and Huson et al. [15]). A central question when dealing with both phylogenetic trees and networks is whether or not they represent consistent information, formulated

A preliminary version of this paper appeared in [1]

E-mail addresses: matthias.bentert@tu-berlin.de (Matthias Bentert) mathias.weller@u-pem.fr (Mathias Weller)

as the question whether or not the network "displays" the tree. This problem is known as Tree Containment and it has been shown NP-hard [16, 19]. Due to its importance in the analysis of evolutionary history, attempts have been made to identify polynomial-time computable special cases [2, 6, 8, 10, 12, 16, 19, 20], as well as moderately exponential-time algorithms [11, 20]. 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 NP-complete, 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 problem can be seen to be a generalization[1] of the Cluster Containment problem [15], implying that our algorithms can be used to attack the latter as well.

Preliminaries. A *phylogenetic network* (or *network* for short) on a set X of taxa is a rooted, leaf-labeled DAG in which all vertices with in-degree (number of predecessors) at most one have out-degree (number of successors) exactly one. These vertices are called *reticulations* and the others are called *tree vertices*. A network without reticulations is called a (phylogenetic) *tree*. The *degree* of a vertex u in N is the sum of its in-degree and its out-degree. *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 (suppressing u is the inverse of subdividing pc with u). A network is called *binary* if all vertices except the root have 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 N_B contains an edge from a vertex in $\chi^{-1}(x)$ to a vertex in $\chi^{-1}(y)$). We call such a function a *contraction function* of N_B for N. We suppose that all binary resolutions are minimal, that is, they do not contain biconnected components C with exactly one incoming and one outgoing edge and $|\bigcup_{v\in C} \chi(v)| = 1$.

By default, no label occurs twice in a network, and we will make exceptions explicit by calling networks in which a label may occur up to ℓ times *ℓ-labeled* (or *multi-labeled* if ℓ is unknown

[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 (see preliminaries for the notions of (soft) display, binary trees, etc).

or infinite). In this sense, "networks" are exactly the special multi-labeled networks with $\ell = 1$ (1-labeled networks). In 1-labeled networks, we use leaves and labels (taxa) interchangeably. We abbreviate $\{x, y\}$ to xy, and $\{x, y, z\}$ to xyz. We denote the set of vertices in a network N by $V(N)$ and 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 is just a singleton $\{u\}$, then we write $u \leq_N W$ (and likewise for 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 define N_v as the subnetwork of N induced by $\{u \mid u \leq_N v\}$ and we denote the set of labels in N_v by $\mathcal{L}(N_v)$ (or $\mathcal{L}(v)$ is N is clear from the context) and abbreviate $n := |\mathcal{L}(N)|$. 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-set 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 $N|_U$ and $N||_U$ is the result of suppressing all degree-two vertices in $N|_U$. Note that $N||_U$ can be assumed to be computable in $O(|U|)$ time (see, for example [4, Section 8]). 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, where ρ_N denotes the root of N. 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*.

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 1 (see Figure 1) *Let N be a network and let T be a tree. Then,*

- *N* firmly displays *T if N contains (as a subgraph, respecting leaf-labels) a subdivision of T and*
- *N* softly displays *T 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 soft 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 1 *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 2 *A tree T displays a tree T' on the same label-set if and only if T' displays T.*

[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.

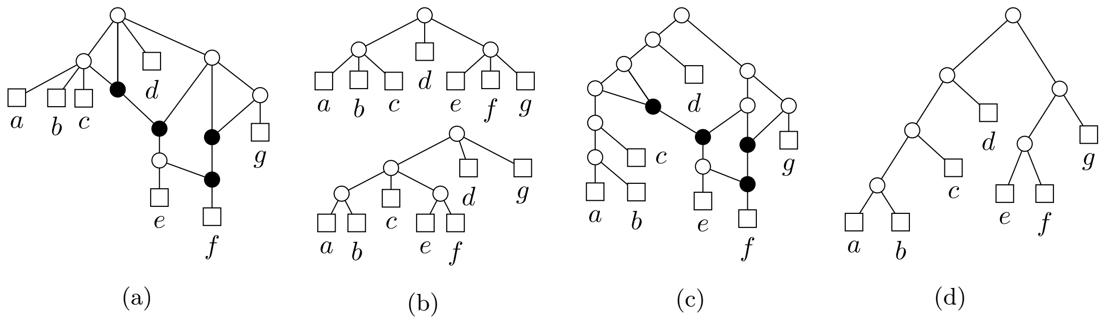


Figure 1: Illustration of firm and soft display (black vertices are reticulations, white vertices are tree-vertices, round vertices are inner vertices, boxes are leaves (with labels)). **(a)** A network N. **(b)** Trees T_1 and T_2. The former is firmly and, thus, softly displayed by N. The latter is softly, but not firmly displayed by N. **(c)** A binary resolution N_B of N displaying T_1 and T_2 softly, but not firmly. **(d)** A binary resolution T_B of T_1 that is (firmly and, thus, softly) displayed by N_B (and, thus, softly displayed by N).

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?

As a side-note, all considered problems are in NP, since a mapping of the vertices of T to the vertices of N constitutes a polynomial-time checkable certificate for the fact that N firmly/softly displays T. For (Soft) Cluster Containment, the tree T is also part of the certificate. Thus, all proofs of NP-hardness actually imply NP-completeness.

2 Display with Soft Polytomies

In this section, all trees and networks are single-labeled. For binary trees (in particular, triplets), the concept of "display" is well-researched.

Observation 3 ([5]) *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 (up to subdivision and suppression of degree-two vertices) 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 4 *Binary trees T and T' on the same label-set are isomorphic if and only if, for each $u \in V(T)$ and each $Y \subseteq \mathcal{L}(T_u)$, u has a child v with $\mathcal{L}(T_v) = Y$ if and only if* $\mathrm{LCA}_{T'}(\mathcal{L}(u))$ *has a child v' with $\mathcal{L}(T'_{v'}) = Y$.*

For a vertex u with children C and two given $v, w \in C$, we denote the operation of removing the arcs uv and uw, introducing a new vertex x, and inserting the arcs ux, xv, and xw as *splitting* v and w from u.

Lemma 1 *Let N and T be trees on the same label-set. Then, N softly 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 softly 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 $\mathcal{L}(N_u) \nsubseteq \mathcal{L}(T_v)$, $\mathcal{L}(N_u) \nsupseteq \mathcal{L}(T_v)$ and $\mathcal{L}(N_u) \cap \mathcal{L}(T_v) \neq \varnothing$, that is, there are $x \in \mathcal{L}(N_u) \setminus \mathcal{L}(T_v)$, $y \in \mathcal{L}(N_u) \cap \mathcal{L}(T_v)$, and $z \in \mathcal{L}(T_v) \setminus \mathcal{L}(N_u)$. Since there are isomorphic binary resolutions N^B and T^B of N and T, respectively, there is a vertex u^B in N^B with $\mathcal{L}(N^B_{u^B}) = \mathcal{L}(N_u)$ and a vertex v^B in T^B with $\mathcal{L}(T^B_{v^B}) = \mathcal{L}(T_v)$. However, since $x, y \in \mathcal{L}(N_u) = \mathcal{L}(N^B_{u^B})$ and $z \notin \mathcal{L}(N_u) = \mathcal{L}(N^B_{u^B})$, we have $\mathrm{LCA}(xy) \leq_{N^B} u^B < \mathrm{LCA}(yz) = \mathrm{LCA}(xz)$, that is, N^B displays $xy|z$. Likewise, T^B displays $yz|x$. But then, Observation 3 contradicts the fact that T^B and N^B are isomorphic.

"$\Leftarrow$": Assuming that $\mathcal{L}(u) \subseteq \mathcal{L}(v)$, $\mathcal{L}(u) \supseteq \mathcal{L}(v)$ or $\mathcal{L}(u) \cap \mathcal{L}(v) = \varnothing$ holds for all $u \in V(T)$ and $v \in V(N)$, we will construct a binary tree B on the labels of N and T that is a resolution of both N and T, using Observation 3. Consider any set $\{a, b, c\}$ of three labels and observe that it cannot happen that $\mathrm{LCA}_N(ab) <_N \mathrm{LCA}_N(ac) = \mathrm{LCA}_N(bc)$ and $\mathrm{LCA}_T(ac) <_T \mathrm{LCA}_T(ab) = \mathrm{LCA}_T(bc)$ as this implies the existence of $v = \mathrm{LCA}_N(ab)$ and $u = \mathrm{LCA}_T(ac)$ with $c \in \mathcal{L}(T_u) \setminus \mathcal{L}(N_v)$, $b \in \mathcal{L}(N_v) \setminus \mathcal{L}(T_u)$, and $a \in \mathcal{L}(T_u) \cap \mathcal{L}(N_v)$, a contradiction to our assumption. To construct the binary tree B, we start with N and replace all high-degree vertices by binary trees. If $\mathrm{LCA}_N(ab) <_N \mathrm{LCA}_N(ac) = \mathrm{LCA}_N(bc)$ and $\mathrm{LCA}_T(ab) \leq_T \mathrm{LCA}_T(ac) = \mathrm{LCA}_T(bc)$, then we already display $ab|c$ in N. If $\mathrm{LCA}_N(ab) = \mathrm{LCA}_N(ac) = \mathrm{LCA}_N(bc)$ and $\mathrm{LCA}_T(ab) \leq_T \mathrm{LCA}_T(ac) = \mathrm{LCA}_T(bc)$, then we split the children of $\mathrm{LCA}_N(ab)$ that are ancestors of a and b, respectively, from $\mathrm{LCA}_N(ab)$. The resulting tree displays $ab|c$. Let B denote the result of iterating the above process for all leaf-triples in $\mathcal{L}(N) = \mathcal{L}(T)$ and observe that B is binary. Observe further that N is a contraction of B and, by Observation 1, N softly displays B. Since B displays exactly the triplets that are firmly displayed by N or T, we could also perform the same procedure but start with T (and deciding similarly whenever there are multiple options), obtaining a binary tree B' that displays exactly the same triplets as B but is displayed by T. By Observation 3, B and B' are the same tree as binary trees are uniquely defined by their displayed triplets. Hence B is displayed by N and T and, as B is binary, N softly displays T by definition. □

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

Observation 5 *Let T be a tree and let $a, b, c \in \mathcal{L}(T)$. Then,*
(a) T softly/firmly displays $ab|c$ if and only if $T|_{abc}$ does.
(b) T firmly displays $ab|c$ if and only if $\mathrm{LCA}(ab) <_T \{\mathrm{LCA}(ac), \mathrm{LCA}(bc)\}$.
(c) 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 if and only if, 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: "$\Rightarrow$": By Observation 2, 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 softly display $ab|c$. By Observation 5, 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 $c \in \mathcal{L}(v) \setminus \mathcal{L}(u)$. But since T softly displays T', this contradicts Lemma 1.

"$\Leftarrow$": Towards a contradiction, assume that T does not softly 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 5, 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 5. $\square$

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

Observation 6 *Let T be a tree, let T' be the result of contracting any arc in T, and let Y and Z be sets of leaves common to T and T'. Then,*
(a) $\mathrm{LCA}_T(Y) \leq_T \mathrm{LCA}_T(Z) \iff \mathrm{LCA}_{T'}(Y) \leq_{T'} \mathrm{LCA}_{T'}(Z)$ *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 softly displays the leaf-triplet $ab|c$ if and only if $\mathrm{LCA}(ab) \leq \{\mathrm{LCA}(bc), \mathrm{LCA}(ac)\}$.
(b) T softly displays a binary tree T_B on X if and only if T softly displays all triplets that T_B displays firmly.
(c) T softly displays a tree T' on X (and vice versa) if and only if there is a binary tree T_B on X that is softly displayed by both T and T'.
(d) A network N softly displays T if and only if N contains (as a subgraph) a tree T' on X that softly displays T.

Proof: (a) By definition, T softly displays $ab|c$ if and only if there is a binary resolution T_B of T displaying $ab|c$. By Observation 3, 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 6, is equivalent to $\mathrm{LCA}_T(ab) \leq_T \{\mathrm{LCA}_T(ac), \mathrm{LCA}_T(bc)\}$.

(b) "$\Rightarrow$": Assume towards a contradiction that a triplet $ab|c$ of T_B is not softly displayed by T. Then, $\{\mathrm{LCA}_T(ab), \mathrm{LCA}_T(ac), \mathrm{LCA}_T(bc)\}$ has a unique minimum x and, by (a), $x \neq \mathrm{LCA}_T(ab)$ (as, otherwise, T displays $ab|c$). Without loss of generality, let $x = \mathrm{LCA}_T(ac)$. As T has a binary resolution that is isomorphic to T_B, we know that T is a contraction of T_B. Hence, Observation 6 applies to T and T_B, showing that $\mathrm{LCA}_{T_B}(ac) <_{T_B} \mathrm{LCA}_{T_B}(abc)$ and, thus, T_B displays $ac|b$. But then, T_B displays conflicting triples, contradicting Observation 3.

"$\Leftarrow$": Assume towards a contradiction that T does not softly 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 softly displays $yz|x$ since all triplets displayed by T_B are softly 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 $u <_T \mathrm{LCA}_T(xyz)$.

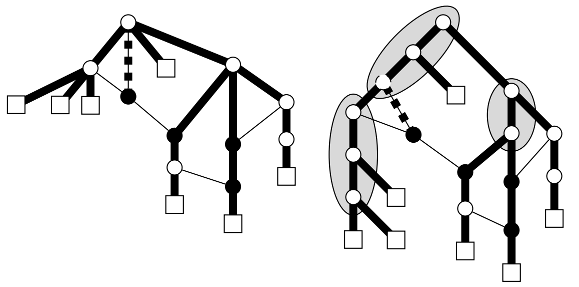


Figure 2: Illustration of the proof of Lemma 3(d). Left: N with T' (bold) and T^* (bold and dashed). Right: N_B with T_B (bold) and T_B^* (bold and dashed). The function χ maps each vertex to itself except the encircled ones that are mapped together.

(c) By definition, T softly displays T' if and only if there are binary resolutions T_B and T'_B of T and T', respectively, such that T_B firmly displays T'_B. If such trees exist then they are equal since, by (b), T_B displays all triplets displayed by T'_B and, by Observation 3, $T_B = T'_B$. Conversely, by Observation 1, all binary trees on X that are softly 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 and note that, for each arc uv in N_B with $\chi(u) \neq \chi(v)$, the arc $\chi(u)\chi(v)$ exists in N. Note that, for all vertices w of N, the vertices in $\chi^{-1}(w)$ form a weakly connected component in N_B. To show that they also form a weakly connected component in S_B, assume that there are vertices u and v in S_B with $\chi(u) = \chi(v) =: w$ but the unique u-v-path in the undirected graph underlying S_B contains a vertex that is not in $\chi^{-1}(w)$. Then, u and v are not related by $<_{S_B}$ as, otherwise, N contains a directed cycle involving w. Now, let T' be the result of contracting all arcs xy of S_B with $\chi(x) = \chi(y)$ and note that T' is a tree (since it results from a tree by contracting weakly connected components). Further, each arc uv of S_B with $\chi(u) \neq \chi(v)$ also exists in N_B and N contains the arc $\chi(u)\chi(v)$. Thus, T' is also a subgraph of N. Concluding, N contains a subgraph T' whose binary resolution S_B firmly displays the binary resolution T_B of T, that is, T' softly 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 1, is a binary resolution of T (see Figure 2). 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 be some arbitrary edge 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

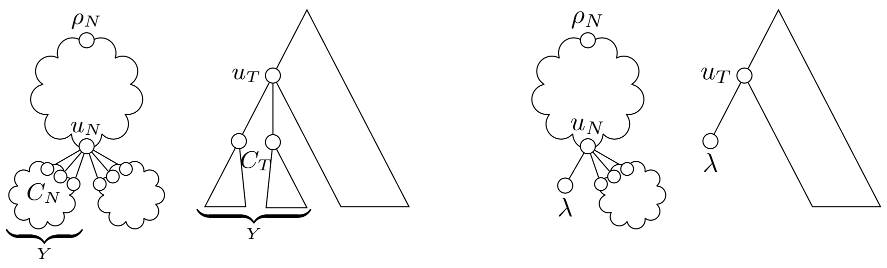


Figure 3: Illustration of Lemma 4: (N, T) left and (N_1, T_1) right.

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^*. Then, there exists 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^*. Hence, 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. Thus, xy is an edge of N^*. □

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

Observe that, if N softly/firmly displays T, then the result of removing any label from N softly/firmly displays the result of removing this label from T.

Observation 8 *Let N be a network and let T be a tree on X. Then, N softly/firmly displays T if and only if $N|_{X'}$ softly/firmly 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 the means to solving 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. Indeed, the lemma holds true even if N is a network, so we state it in this, more general form.

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 $\ell \in Y$, all ρ_N-ℓ-paths contain u_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 3).

Proof: Since "$\Rightarrow$" follows directly from Observation 8, 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, let Q be the result of deleting the leaf λ in Q_1 and identifying the root of Q_2 with u_N in Q_1 (note that Q_1 contains u_N since it is the only parent of λ in N_1) and note that Q is a subtree of N (since Q_2 contains the root of N_2). By Lemma 3, it is sufficient to prove that T and Q display all triplets displayed by T_B Towards a contradiction, assume that T_B displays a triplet $xy|z$ that T or Q does not display. Without loss of generality, let T or Q firmly display $xz|y$ (see Lemma 2) that is, $\mathrm{LCA}_T(xz) <_T \mathrm{LCA}_T(xyz)$ and $\mathrm{LCA}_Q(xz) <_Q \mathrm{LCA}_Q(xyz)$.

Case 1: none or all of x, y, z are in Y. Then, $xy|z$ is already displayed by T_i^B but not by T_i or Q_i for some $i \in \{1,2\}$. By Lemma 3(b), this contradicts T_i^B being displayed by both T_i and Q_i.

Case 2: exactly one of x, y, z is not in Y. By construction of T_B, this implies $x, y \in Y$ and $z \notin Y$ as, otherwise, T_B cannot display $xy\,|\,z$. But then, by (b), $\mathrm{LCA}_Q(xy) \leq_Q u_N \leq_Q \{\mathrm{LCA}_Q(xz), \mathrm{LCA}_Q(yz)\}$. Further, $\mathrm{LCA}_T(xy) \leq_T u_T \leq_T \{\mathrm{LCA}_T(xz), \mathrm{LCA}_T(yz)\}$ since T is a tree. Thus, by Lemma 3(a), both Q and T softly display $xy|z$, contradicting the assumption.

Case 3: exactly one of x, y, z is in Y. Suppose $x \in Y$. Then, by construction, T_B also displays $\lambda y\,|\,z$ and so does T_1^B. Thus, by Lemma 3(b), T_1 and Q_1 both softly display $\lambda y\,|\,z$, implying $\mathrm{LCA}_Q(\lambda y) \leq_Q \mathrm{LCA}_Q(\lambda yz)$ and $\mathrm{LCA}_T(\lambda y) \leq_T \mathrm{LCA}_T(\lambda yz)$. By (b), the embedding of $\lambda y|z$ in Q contains a u_N-λ-path p which can be turned into an embedding of $xy|z$ in Q by replacing p with a u_N-x-path in Q (which exists due to (b)). Thus, $xy|z$ is softly displayed by Q and, analogously, by T, contradicting our assumption. The cases the $y \in Y$ and $z \in Y$ are symmetrical. □

In the following, the operation of *splitting off* a subnetwork B with root u in a network N means to insert a new vertex u', replace each arc ux in N with $x \in V(B)$ by $u'x$, and add a new leaf labeled $\lambda \notin \mathfrak{L}(N)$ to u. This gives rise to the networks N_1 (containing the new leaf λ) and N_2. To state the reduction rule implied by Lemma 4, let a subnetwork B of a network N be called *child-including* if for all vertices v other than the root of B, all children of v are in B.

Rule 1 *Let (N, T) be an instance of* Soft Tree Containment, *let N' and T' be children-including subnetworks of N and T, respectively, such that $\mathcal{L}(N') = \mathcal{L}(T')$. Then, split off N' from N and T' from T.*

While, in general, finding N' and T' might be difficult, we do not need its full power to solve Soft Tree Containment on 1-labeled trees.

Lemma 5 *Let N and T be trees. There is an algorithm that either finds children-including subnetworks N' and T' of N and T, respectively, such that $\mathcal{L}(N') = \mathcal{L}(T')$ in $O(|\mathcal{L}(N')|)$ time, or correctly concludes that N does not display T in $O(|N| + |T|)$ time.*

Proof: Our strategy to find N' and T' is as follows. Start with N'_1 being any cherry in N with root $\rho_{N'_1}$ and T'_1 the smallest child-including subtree of T containing a and b. Note that T'_1 can be found in $O(|T'_1|)$ time with a simple DFS. If T'_1 is a cherry in T, then we output N'_1 and T'_1. Otherwise, assume that T'_1 contains a leaf $c \notin ab$. Then, without loss of generality, T firmly displays $ac|b$. If $\mathrm{LCA}_N(ac) \neq \rho_{N'_1}$, then N firmly displays $ab|c$ and, thus, cannot softly display

T. As this can be detected in $O(1)$ time, we suppose that $c \leq_N \rho_{N'_1}$. Now, let N'_2 be the smallest child-including subtree of N containing $\mathcal{L}(T'_1)$ and note that its root is $\rho_{N'_2} = \rho_{N'_1}$. Now if there is a leaf $e \in \mathcal{L}(N'_2) \setminus \mathcal{L}(T'_1)$, then this leaf has been included because of a leaf $d \in \mathcal{L}(T'_1) \setminus \mathcal{L}(N'_1)$. Thus, N firmly displays both $de|a$ and $de|b$. However, since $d \in \mathcal{L}(T'_1)$ and $e \notin \mathcal{L}(T'_1)$, we know that T firmly displays either $ad|e$ or $bd|e$. Both cases imply that N does not display T. Thus, we can either output N'_2 and T'_1 if $\mathcal{L}(N'_2) = \mathcal{L}(T'_1)$ and reject the instance, otherwise. As this can be checked in $O(|N'_2|)$ time, the claimed running time follows. $\square$

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

4 Tree Containment in Multilabeled Trees

In this section, we consider the task of deciding if a 1-labeled tree T is softly contained in a multi-labeled tree N. Note that, for multi-labeled networks and trees, leaves and labels no longer correspond one-to-one. Thus, we define the set $\mathfrak{L}(N)$ of labels in N, which may now differ from the set $\mathcal{L}(N)$ of leaves in N. In the following, the sets $\mathcal{L}(N_v)$ for each v are called the *clusters* of N. Then, we can formulate the "soft" version of the well-known Cluster Containment problem.

> Soft Cluster Containment (SCC)
>
> **Input:** a network N, some $\mathcal{C} \subseteq \mathfrak{L}(N)$
> **Question:** does N softly display a 1-labeled tree on $\mathfrak{L}(N)$ of which $\mathcal{C}$ is a cluster?

We show that the Soft Cluster Containment problem, a special case of Soft Tree Containment, is NP-hard even on 3-labeled trees. We complement this result with an $O(n^3)$-time algorithm for 2-labeled trees.

To get started, observe that contracting arcs cannot introduce new clusters to trees.

Observation 9 *Let T be a tree and let T' result from contracting an arc uv in T. Let $\mathcal{C}$ and $\mathcal{C}'$ be the clusters of T and T', respectively. Then, $\mathcal{C}' \subsetneq \mathcal{C}$.*

With Observation 9, we can show that SCC is indeed a special case of Soft Tree Containment.

Proposition 1 **([7])** *Let N be a 1-labeled network and let $\mathcal{C} \subseteq \mathcal{L}(N)$. Let T be the tree with root ρ_T that is parent to all leaves in $\mathcal{L}(N) \setminus \mathcal{C}$, as well as a vertex v that is parent to all leaves in $\mathcal{C}$. Then, N softly displays T if and only if N softly displays a tree with cluster $\mathcal{C}$.*

Proof: Since $\mathcal{C}$ is a cluster of T, it suffices to show "$\Leftarrow$". Let T' be a tree with cluster $\mathcal{C}$ that is softly displayed by N, that is, there is a binary resolution T^B of T' that is softly displayed by N. Now, T' can be contracted to T and, thus, T^B can be contracted to T. But then, T^B is a binary resolution of T displayed by N. $\square$

In the following, we reduce Monotone k-SAT to SCC on k-labeled trees. Since Monotone k-SAT is well known to be NP-complete even for $k = 3$ [9], the desired hardness of SCC on 3-labeled trees follows. Note that each clause in a monotone boolean formula is either "positive" (containing only non-negated variables) or "negative" (containing only negated variables).

Construction 1 (See Figure 4) *Given an instance φ of Monotone k-SAT on n variables $x_1, x_2, \ldots, x_n$ and m clauses $c_1, c_2, \ldots, c_m$. Let Z be the set of pairs (x_i, c_j) such that c_j contains x_i or $\neg x_i$. We construct a multi-labeled tree N on the vertex set $\{\rho_N\} \cup \bigcup_i \{x_i\} \cup Z$ and*

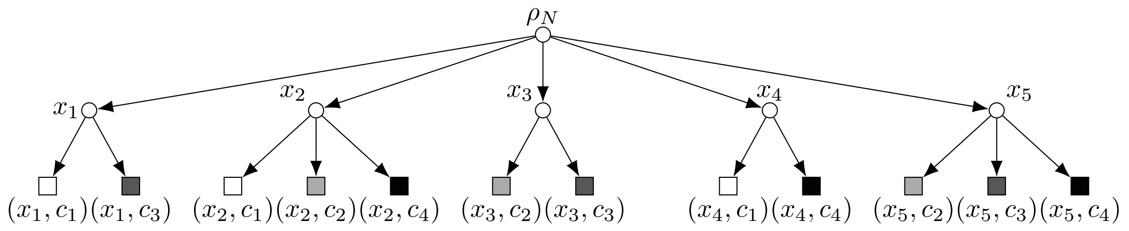


Figure 4: Illustration of Construction 1 for $\varphi = (x_1x_2x_4) \wedge (x_2x_3x_5) \wedge (\overline{x_1}\,\overline{x_3}\,\overline{x_5}) \wedge (\overline{x_2}\,\overline{x_4}\,\overline{x_5})$; $\mathcal{C} = \{$□, ▨$\}$.

arcs $\bigcup_i\{(\rho_N, x_i)\}$ and $\{(x_i,(x_i,c_j)) \mid (x_i,c_j) \in Z\}$. Further, each leaf $(x_i,c_j) \in Z$ is labeled by c_j. Finally, the cluster $\mathcal{C}$ consists of all clauses containing only positive literals.

Construction 1 constructs k-labeled trees since each clause contains at most k variables.

Lemma 6 *Construction 1 is correct, that is, the constructed tree N softly displays a tree T on $\mathfrak{L}(N)$ containing $\mathcal{C}$ as a cluster if and only if φ is satisfyable.*

Proof: "$\Leftarrow$": Let β be an assignment satisfying φ and let ψ map each clause c_j to a variable x_i satisfying c_j (that is, ψ maps c_j to some x_i with $\beta(x_i) = 0 \iff x_i$ occurs negated in c_j). Note that positive variables can only satisfy positive clauses, so $\beta(\psi(c_j)) = 1 \iff c_j \in \mathcal{C}$. Let $T := N|_{\psi^{-1}}$ be the result of deleting all leaves (x_i,c_j) with $\psi(c_j) \neq x_i$ and let ρ_T be the root of T. Finally, let B result from T by 1. adding a new vertex v with arc $\rho_T v$ and 2. replacing each arc $\rho_T x_i$ for which $\beta(x_i) = 1$ with vx_i. Observe that all labels occur in T and B and T is a contraction of B. To prove the "$\Leftarrow$"-direction of the lemma, we show that $\mathfrak{L}(B_v) = \mathcal{C}$, that is, $\mathcal{C}$ is already a cluster of B and, thus, of any binary resolution of B. To this end, note that

$$\begin{aligned} B_v \text{ contains a label } c_j &\iff (\psi(c_j), c_j) <_B v \\ &\iff \beta(\psi(c_j)) = 1 \\ &\iff c_j \in \mathcal{C}. \end{aligned}$$

"$\Rightarrow$": Without loss of generality, we suppose that each variable occurs non-negated in φ (that is, all variables of φ occur in some clause of $\mathcal{C}$) since variables that only occur negated can just be set to false without affecting satisfyability. Let B be a binary resolution of some 1-labeled subtree T of N on the same label set such that $\mathcal{C}$ is a cluster of B. Towards a contradiction, assume that φ is not satisfyable. We construct an assignment β setting a variable x_i to true if and only if there is a label $c_j \in \mathcal{C}$ such that (x_i,c_j) is a leaf of B. Assume towards a contradiction that some clause c_j of φ is not satisfied by β. Since B is on the same label set as N, we know that B contains a leaf (x_i,c_j) labeled c_j. If $c_j \in \mathcal{C}$, then c_j is a positive clause and, by definition, $\beta(x_i) = 1$, thereby satisfying c_j. Otherwise, $c_j \notin \mathcal{C}$, implying that c_j is a negative clause containing the literal $\neg x_i$. Since c_j is not satisfied by β, we have $\beta(x_i) = 1$, implying that there is a leaf (x_i, c_ℓ) in B with $c_\ell \in \mathcal{C}$. Since all variables occur in a clause in $\mathcal{C}$, we know that T also contains a leaf $(x_{i'}, c_{\ell'})$ with $i' \neq i$ and $c_{\ell'} \in \mathcal{C}$. By construction of N, we know that T firmly displays $c_j c_\ell | c_{\ell'}$ and, by Lemma 2, so does B, contradicting that $\mathcal{C}$ is a cluster of B. □

Theorem 2 *Soft Cluster Containment is* NP*-hard even if the input network is a 3-labeled tree.*

Corollary 1 *Soft Tree Containment is* NP*-hard, even if the input network N is a 3-labeled tree.*

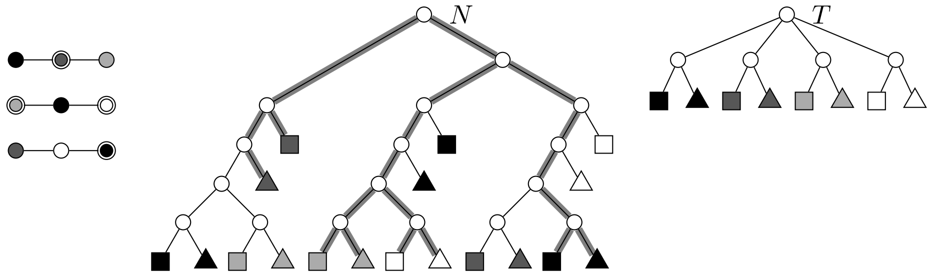
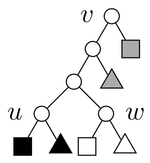


Figure 5: Illustration of Construction 2. **Left:** the initial instance of 2-UNION INDEPENDENT SET with 4 colors (●,●,●,○) and a size-4 solution encircled. **Right:** the non-binary tree T (boxes and triangles indicating label i_1 and i_2 for a color i). **Middle:** the binary multi-labeled tree N with a subdivision of T (bold, gray) corresponding to the solution to the left instance.

Binary Networks. The instance (N,T) of SOFT TREE CONTAINMENT constructed by Construction 1 and Proposition 1 uses polytomies in both N and T. We can, however, strengthen the hardness to also work if N is a binary 3-labeled trees. To this end, we reduce 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 k disjoint K_2s and K_3s and (V, E_2) is a collection of disjoint P_3s [18, Lemma 2]. We reduce this version of 2-UNION INDEPENDENT SET to SOFT TREE CONTAINMENT for multi-labeled 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). Further, in the construction of van Bevern et al. [18], no P_3 contains the same color twice.

Construction 2 (See Figure 5) *Given a vertex-colored collection G of P_3s, 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 labeled 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. Note that $u, v, w \in V(G) \cap V(N)$.*



Lemma 7 *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. We show that the set Q of vertices of $V(G)$ that are parents of leaves in S contains all colors occurring in G and is independent in G (thus, it contains a size-k colorful independent set of G). First, assume that Q avoids some color i. However, since S displays T, we know that S contains leaves ℓ_1 and ℓ_2 with colors i_1 and i_2, respectively. But, by construction, one of them has a parent in $V(G)$, contradicting Q avoiding the color i. Second, assume that Q is not independent in G, that is, there are vertices u and v in Q such that u is the center of the P_3 containing v. Let i and j be the respective labels of u and v. By construction, S_u contains a leaf labeled i_1 and a leaf labeled j_1. But then, $j_1 i_1 | i_2$ is firmly displayed by S but not softly displayed by T, thereby contradicting Lemma 3(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. We show that S softly displays T. To this end, assume that S firmly displays a triplet $xy|z$ that T does not display softly. By Observation 5(c), T firmly displays $xz|y$ or $yz|x$. Without loss of generality, let T firmly display $xz|y$. By Observation 5(b), $\mathrm{LCA}_T(xz) <_T \{\mathrm{LCA}_T(xy), \mathrm{LCA}_T(yz)\}$. By construction, $x = i_1$, $z = i_2$, and $y = j_1$ for some colors $i \neq j$. By Lemma 3(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 3 *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-Satisfyability). This allows us to state the following generalization of Theorem 3.

Corollary 2 *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 multi-labeled trees.*

Corollary 2 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 multi-labeled 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 3 *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 3 (Corollary 1) and Corollary 3 stand in contrast with results for (Strong) Tree Containment, which is linear-time solvable in both cases [12, 20].

4.1 Solving k-Labeled Trees

To solve Soft Tree Containment for instances where N is a k-labeled tree, we compute a mapping $M : V(T) \to 2^{V(N)}$ such that $M(u)$ contains the at most k minima (with respect to $\leq_N$) among all vertices v of N such that N_v displays T_u. A 1-labeled subtree S of N that softly/firmly displays T is called *canonical* for some $u \in V(T)$ if $\mathrm{LCA}_S(\mathcal{L}(T_u)) \in M(u)$ (that is, for all proper descendants w of $\mathrm{LCA}_S(\mathcal{L}(T_u))$ in N, we have that N_w does not softly/firmly display T_u) and *canonical* for T if it is canonical for all $u \in V(T)$. We show that, softly displaying is equivalent to having such a canonical subtree.

Lemma 8 *A k-labeled tree N softly/firmly displays a 1-labeled tree T if and only if N has a canonical subtree for T.*

Proof: As "$\Leftarrow$" is evident, we just prove "$\Rightarrow$". Let S be a 1-labeled subtree of N that softly/firmly displays T and suppose that S is not canonical for T. Let $u \in V(T)$ have minimum distance to the root of T among all vertices for which S is not canonical. We will construct a 1-labeled subtree S' of N that softly/firmly displays T where all vertices, for which S is canonical but S' is not, are strict descendants of u in T. Then, iterating this construction eventually yields a canonical subtree for T.

Let x be defined as $\mathrm{LCA}_S(\mathcal{L}(T_u))$. Since N_x softly/firmly displays T_u, there is some $y \in M(u)$ with $y <_N x$ and N_y softly/firmly displays T_u, that is, N_y contains a subtree S'_y that softly/firmly displays T_u. Let S' result from S by 1. removing all vertices $q <_S x$ with $\mathcal{L}(S'_q) \subseteq \mathcal{L}(T_u)$, 2. attaching the unique x-y-path in N to x, and 3. attaching S'_y to y. To show that S' softly/firmly displays T, consider leaves a, b, and c of T and recall that $\mathcal{L}(T_u) = \mathcal{L}(S'_y)$.

Case 1: T_u contains at most one of abc. Then $S'|_{abc} = S|_{abc}$ and, thus, S' softly/firmly displays $ab|c$ if and only if S does.

Case 2: T_u contains all of abc. Then, S'_y also contains all of abc and S' softly/firmly displays $ab|c$ if and only if S'_y does.

Case 3: T_u contains exactly two of abc. Then, T softly/firmly displays $ab|c$ if and only if $abc \cap \mathcal{L}(T_u) = ab$ if and only if T firmly displays $ab|c$ if and only if $abc \cap \mathcal{L}(S'_y) = ab$ if and only if S' firmly displays $ab|c$.

For firm display, this directly implies that S' firmly displays T. For soft display, Lemma 2 implies that S' softly displays 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 softly displays T_u. For each $v \in V(T_u) \setminus \{u\}$ 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 set $y := \rho$. In both cases, we call the unique x-y-path in N_ρ the *ascending path* of x wrt. v and we omit mentioning v if it is clear from the context. A crucial lemma about ascending paths is the following.

Lemma 9 *Let N' be a multi-labeled tree displaying a 1-labeled tree T' and let S be a canonical subtree of N' for T'. Let $u, v \in V(T')$ not be siblings in T'. Let $\mathrm{LCA}_S(\mathcal{L}(T'_u))$ and $\mathrm{LCA}_S(\mathcal{L}(T'_v))$ have ascending paths r and q wrt. u and v, respectively. Then, r and q are arc-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 $\leq_{N'}$) 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 r and q share an internal vertex z and thus, $\mathcal{L}(u) \uplus \mathcal{L}(v) \subseteq \mathcal{L}(z)$. 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 $y := \mathrm{LCA}_S(\mathcal{L}(p)) \in M(p)$ and, thus, r ends in y. As z is an internal vertex of r, it holds that $z <_S y$, 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, we compute $N|_L$ where L is the set of leaves of N whose label occurs in T_u. Then, it holds for all $v \in V(T_u)$ that $M(v) \subseteq V(N|_L)$. 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 k-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 $\bigvee_{z\in M(v)} x_{v\to z}$
(iii) for each $y, z \in M(v)$ add the clause $x_{v\to y} \Rightarrow \neg x_{v\to z}$.
(iv) 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 $x_{v\to y} \Rightarrow \neg x_{p\to z}$.
(v) 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 in N_ρ share an arc, add the clause $x_{v\to y} \Rightarrow \neg x_{w\to z}$.

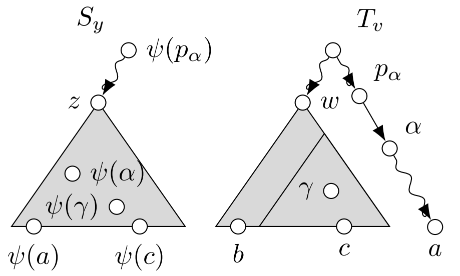
By definition of $M(u)$, no two vertices in $M(u)$ can be in an ancestor-descendant relation. Thus, we can assume that no strict descendant z of our current ρ satisfies $\varphi_{u\to z}$.

Lemma 10 *$\varphi_{u\to\rho}$ is satisfyable 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 $y = \mathrm{LCA}_S(\mathcal{L}(T_v))$. Since the LCA of $\mathcal{L}(T_v)$ in S is unique, all clauses of types (ii) and (iii) are satisfied by β. If a clause of type (iv) is not satisfied, then there is some v with parent p in T_u such that $y \not\leq_N z$ for some $y \in M(v)$ and $z \in M(p)$ and $\beta(x_{v\to y}) = \beta(x_{p\to z}) = 1$. Since $\mathcal{L}(T_p) \supseteq \mathcal{L}(T_v)$, we know that $y \leq_S z$ and, as S is a subtree of N, we have $y \leq_N z$, contradicting $y \not\leq_N z$. If a clause of type (v) 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 $y = \mathrm{LCA}_S(\mathcal{L}(T_v))$ and $z = \mathrm{LCA}_S(\mathcal{L}(T_w))$ in N_ρ share an arc. This contradicts Lemma 9.

"$\Rightarrow$": Let β be a satisfying assignment for $\varphi_{u\to\rho}$ and let $\psi := \{(v, y) \mid \beta(x_{v\to y}) = 1\}$. Since β satisfies the clauses of type (iii), ψ describes a function $V(T) \to V(N)$. Let $Y := \psi(V(T))$ 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 (iv). Thus, for all $v, w \in V(T_u) - u$, we have $w \leq_T v$, which implies $\psi(w) \leq_N \psi(v)$, which then implies that $\psi(w) \leq_S \psi(v)$. For all $(v, y) \in \psi \cup \{(u, \rho)\}$, we show that S_y is a canonical subtree of N_y for T_v. The proof is by induction on the height of v in T. If v is a leaf in T, then $M(v)$ contains all leaves in N (and, thus, the leaf in S) with the same label and the claim follows from $y \in M(v)$. Otherwise, suppose that the claim holds for all $w <_T v$. Note that, if S_y displays T_v, then either $v \neq u$ and we have $y \in M(v)$ or $v = u$ and we can assume that for no proper descendant z of u in N it holds that N_z displays T_u (implying $y \in M(u)$ by definition). In both cases, it suffices to show that S_y displays T_v to show that S_y is canonical for 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 $z \in V(S_y)$ and leaves $a \in \mathcal{L}(S_z) \setminus \mathcal{L}(T_w)$, $b \in \mathcal{L}(T_w) \setminus \mathcal{L}(S_z)$, and $c \in \mathcal{L}(T_w) \cap \mathcal{L}(S_z)$.

[3]The construction uses implications ($(x \Rightarrow y) := (\neg x \vee y)$), which can be formulated as clauses with two variables as shown.



Since $\psi(a), \psi(c) <_S z$, there is a highest ancestor α of a in T with $\psi(\alpha) \leq_S z$ and a highest ancestor γ of c in T with $\psi(\gamma) \leq_S z$. Since $b \not<_S z$, it holds that $b \not\leq_S \psi(\alpha)$ and $b \not\leq_S \psi(\gamma)$, that is, $b \notin \text{LCA}_S(T_\alpha) \cup \text{LCA}_S(T_\gamma)$, so $\alpha <_T w$ and $\gamma \not\leq_T w$, implying that α and γ are not siblings in T. Further, there are parents p_α and p_γ of α and γ, respectively, with $\psi(p_\alpha) \not\leq_S z$ and $\psi(p_\gamma) \not\leq_S z$ by the maximality of α and γ (see figure on the right). But this means that the ascending paths of $\psi(\alpha)$ and $\psi(\gamma)$ in N share the arc that is incoming to z in S, contradicting (v). $\square$

Theorem 4 *In $O((kn)^3)$ time, we can decide if a k-labeled tree N softly displays a 1-labeled tree T using $O(kn)$ queries of size $O((kn)^2)$ to k-SAT.*

Proof: As correctness follows from Lemma 10, we next analyze the running time and the number and size of all of the k-SAT formulas. First, we precompute for each vertex w of N the sets of vertices and leaves below w in N, as well as all pairwise lowest common ancestors in a simple bottom-up manner in $O(n^2)$ time. Then, we compute $M(u)$ for each leaf u of T in $O(n)$ time using a map of labels to leaves in T. During all computations, we maintain a graph G on the vertex set $V(G) \subseteq V(T) \times V(N)$ containing an edge $\{(u, v), (x, y)\}$ if and only if u and x are not siblings and the so-far constructed ascending paths of v and y wrt. u and x, respectively, overlap. Since $|M(u)| \leq k$ for all $u \in V(T)$, this graph contains $O(kn)$ vertices and $O((kn)^2)$ edges at all times. To maintain G, each vertex of N keeps track of the ascending paths it is contained in and from which child each path ascended. To this end, whenever a k-SAT instance $\varphi_{u\to\rho}$ is deemed unsatisfiable, we first check each pair of ascending paths from different children of ρ and add an edge to G if they correspond to non-siblings in T. Second, we add all ascending paths that ρ is contained in to the bucket labeled ρ of its parent's path list. Whenever a k-SAT instance $\varphi_{u\to\rho}$ is deemed satisfyable, a new ascending path of ρ wrt. u is noted in the list of ρ and all ascending paths wrt. children of u are deleted from the list of ρ.

In the follwing, we first argue why it suffices to check $O(kn)$ such pairs to decide whether the root of T is displayed in N and then we show how to construct $\varphi_{u\to\rho}$ for any pair $(u, \rho) \in V(T) \times V(N)$ in $O((kn)^2)$ time. This also shows that G can be maintained in $O((kn)^3)$ time since for each of the $O(kn)$ k-SAT instances, we use $O((kn)^2)$ time to maintain G.

First, we argue that it suffices to construct $O(kn)$ instances of k-SAT to decide whether N softly displays T (see Algorithm 1). Since we need to know $M(v)$ for all $v <_T u$ to compute $\varphi_{u\to\rho}$ for any $u \in V(T)$ and $\rho \in V(N)$, we will go bottom-up "simultaneously" in N and T. In the following, let $u \in V(T)$ be such that $M(u_i)$ is known for all children u_i of u in T. Using our precomputed LCA-lookup table, we can find the at most k lowest vertices ρ_i in N that have a descendant in $M(u_i)$ for each u_i in $O(k \cdot \deg_T(u))$ time[4]. Let C denote this set. If all ρ_i are distinct, that is, $|C| = k$, then $M(u) \subseteq V(N||_C)$ since no vertex $\rho \notin C$ has a path to a leaf in T_u that avoids C. Thus, k k-SAT instances suffice in this case, summing to a total of $O(kn)$ such instances. In the following, suppose $|C| < k$ and note that only subtrees of N intersecting C can softly display T_u. Thus, for each $\rho \in C$, we can "climb" N looking at all ancestors p of ρ in ascending order. For each such p, we test in $O(n)$ time whether any child p_i of p with $\rho \not\leq_N p_i$ is ancestor of a leaf ℓ whose label occurs in T_u and, if so, we construct $\varphi_{u\to p}$ and test its satisfyability. In order to show that the number of such constructions is $O(kn)$, we show that such a construction can only occur

[4]This can be done by iteratively computing a set of candidates C_i by keeping the lowest nodes of $N||_{C_{i-1}\cup M(u_i)}$ that have a descendant in both C_{i-1} and $M(u_i)$ (choosing $C_0 := M(u_0)$).

Algorithm 1: Construction of $O(k)$ k-SAT instances to compute $M(u)$.

```
Input: A multi-labeled tree $N$, a tree $T$, and a vertex $u \in V(T)$
Output: $M(u)$
$C := M(u_0)$;                                   // candidate set; $|C| \leq k$ at all times
foreach child $u_i$ of $u$ do $C := \min_N\{\mathrm{LCA}_N(xy) \mid x \in C \wedge y \in M(u_i)\}$;
compute $N||_C$;
if $|C| < k$ then
    foreach $\rho \in C$ do
        while ($\varphi_{u\to\rho}$ is not satisfyable) and ($\rho$ is not marked) do  // $O(kn)$ loops total
            $L' := \mathcal{L}(\rho)$;
            while $\mathcal{L}(\rho) \setminus L'$ contains no label in $\mathfrak{L}(u)$ do  // $O(kn^2)$ checks à $O(n)$ time
                mark $\rho$;
                $\rho \leftarrow \mathrm{parent}(\rho)$;
        if $\rho$ is not marked then add $\rho$ to $M(u)$;
    return $M(u)$
else return $\{x \in V(N||_C) \mid \varphi_{u\to x}$ satisfyable$\}$;
```

once for each leaf ℓ. Towards a contradiction, assume that a leaf ℓ incurs two such constructions and let u' and u'' be the vertices in T for which they occur, that is, there are $p', p'' \in V(N)$ such that $\varphi_{u'\to p'}$ and $\varphi_{u''\to p''}$ are constructed due to ℓ. But then, ℓ is a leaf of both $T_{u'}$ and $T_{u''}$ so, without loss of generality, $u' <_T u''$ (it is not hard to see that $u' \neq u''$ by the algorithm described above), implying $M(u') \leq_N M(u'')$ by construction. Then, however, $p \leq_N M(u')$ and $\mathrm{LCA}_N(\bigcup_i M(u_i'')) <_N p$, contradicting $M(u') \leq_N M(u'')$. Thus, the overall number of k-SAT instances constructed by the described algorithm is $O(kn)$.

Finally, the size of $\varphi_{u\to\rho}$ is dominated by the $O((kn)^2)$ possible clauses of type (v), which can be generated in constant-time per clause by considering the $O((kn)^2)$ edges of G and picking up a clause $x_{v\to y} \Rightarrow \neg x_{w\to z}$ for each edge $\{(v, y), (w, z)\}$ with $v <_T u$ and $w <_T u$. □

Corollary 4 *A 2-labeled tree N can be verified to softly display a 1-labeled tree T in $O(n^3)$ time.*

5 Extended NP-hardness

Complementing the main result of Section 4.1, we now show that Soft Tree Containment remains NP-hard on even very restricted classes of 2-labeled networks. To this end, we will modify Construction 2 to construct 2-labeled networks instead of 3-labeled trees. We will make heavy use of the *merge* operation on two nodes x and y of N defined as removing N_x and making all former parents of x new parents of y. We denote the result of this operation by $N|_{x\to y}$ and note that it still contains the vertex y. If x and y have the same label λ, then y retains this label. If at most one of x and y has a label, then y gets no label. If x and y have different labels, the merge operation is undefined.

Lemma 11 *Let N be a multi-labeled network and let T be a 1-labeled tree. Let $u, v \in V(N)$ and $w \in V(T)$ be distinct such that N_u, N_v, and T_w are isomorphic. Let $x \leq_N u$ and $y \leq_N v$ such that N_x is isomorphic to a subtree of N_y. Then, N softly/firmly displays T if and only if $N' := N|_{x\to y}$ does.*

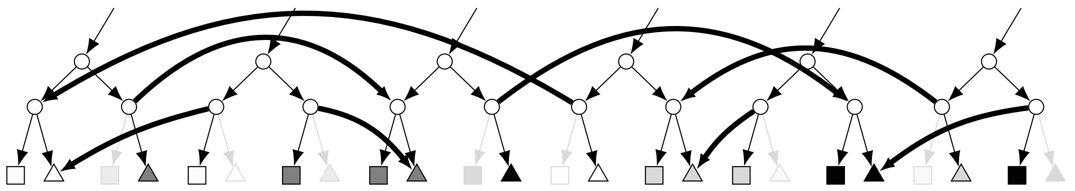


Figure 6: Illustration of the multiplicity reduction for Soft Tree Containment. Removed leaves are faded out and newly introduced arcs are bold. Note the tree-child property after expansion of nodes with in- and out-degree two.

Proof: "⇒": Let S be a 1-labeled subtree of N that softly/firmly displays T. Suppose that S contains both u and v and let S' be the result of replacing S_u and S_v by N_u and N_v, respectively, in S. Since S is a subtree of S', we know that S' is a multi-labeled tree that softly/firmly displays T. By Lemma 8, S' contains a canonical subtree S^* displaying T and, since $N_u = N_v = T_w$, we know that S^* contains at most one of u and v. Thus, we can suppose without loss of generality that S contains at most one of u and v. If S does not contain x, then S is a subtree of $N - V(N_x)$, which is a subtree of N'. Otherwise, $x \in V(S)$ implying $u \in V(S)$ and, thus, $v \notin V(S)$ and $y \notin V(S)$. Note that, by definition, N_y softly/firmly displays S_x. Now, since $N'_y = N_y$, there is a subtree S'_y of N'_y that softly/firmly displays S_x. Then, replacing S_x by S'_y in S yields a subtree of N' that softly/firmly displays T.

"⇐": Let S' be a 1-labeled subtree of N' that softly/firmly displays T. By Lemma 8, we suppose that S' is canonical for T. Further, we suppose that S' contains an arc py where p is a parent of x but not y in N as, otherwise, S' is also a subtree of N. Let $z := \mathrm{LCA}_{S'}(\mathfrak{L}(T_w))$ and note that $z \geq_{S'} y$ since $\mathfrak{L}(S'_r) \subseteq \mathfrak{L}(N'_r) \subseteq \mathfrak{L}(N_x) \cup \mathfrak{L}(N_y) \subseteq \mathfrak{L}(N_u) \cup \mathfrak{L}(N_v) = \mathfrak{L}(T_w)$. First, if $x = u$ (implying $y = v$), then $N'_y = N_y = N_v = T_w$. Further, $z \leq_{S'} y$ since S' is canonical and $N'_y = N_y = T_w$. But then, the result S of replacing S'_y with N_u in S' yields a subtree S of N that softly/firmly displays T_w. Second, if $x <_N u$, then N_u contains the arc px. Then, $p \in V(S') \cap V(N'_u) \setminus V(N'_y)$, implying $u \in V(S')$ since N_u (and, thus, $N'_u - V(N'_y)$) is free of reticulations. Further, $\mathfrak{L}(S'_u) \subseteq \mathfrak{L}(N_u) \cup \mathfrak{L}(N_y) = \mathfrak{L}(T_w)$. Thus, S'_u cannot softly/firmly display T_q for any $q \not\leq_T w$. But then, replacing S'_u with N_u in S' yields a subtree S of N that softly/firmly displays T. □

Lemma 11 can be used to turn multi-labeled trees into 2-labeled networks. In the following, we show that, for the 3-labeled trees constructed by Construction 2, this can be done such that the resulting network is tree-child. To this end, we construct a mapping γ that maps each leaf whose label λ occurs thrice in N to a vertex u of N such that either u is a leaf labeled λ or u is the root of a cherry containing a leaf labeled λ. Observe that the result of the merge is tree-child if no two siblings in N are involved (in the domain or image) in a mapping in γ. The mapping γ is defined as follows. Let $i_\square$ and $i_\triangle$ be two labels of the same color i in N (see Construction 2). First, suppose $i_\square$ and $i_\triangle$ occur in leaves $\ell_\square$ and $\ell_\triangle$ that are at distance three in N. Then, either $i_\square$ and $i_\triangle$ occur at most twice, or we can find occurrences $\ell'_\square \neq \ell_\square$ and $\ell'_\triangle \neq \ell_\triangle$ of $i_\square$ and $i_\triangle$ that are not siblings. Then, we let γ map $\ell'_\square$ to $\ell_\square$ and $\ell'_\triangle$ to $\ell_\triangle$. Second, suppose $i_\square$ and $i_\triangle$ occur only in cherries together, then let $\{\ell^j_\square, \ell^j_\triangle\}$ denote these cherries and let ρ^j be their roots, for $1 \leq j \leq 3$. Then, we let γ map $\ell^2_\triangle$ to $\ell^1_\triangle$ and $\ell^3_\square$ to ρ^1 (see Figure 6). Note that, to merge $\ell^3_\square$ with ρ^1, we apply Lemma 11 with $u = \rho^3$, $x = \ell^3_\square$ and $v = y = \rho^1$. Further, note that we have to choose ρ^1 among the three cherries of color i such that its sibling is not chosen for some other color i'. Observe that we may assign one of two colors to such "double cherries", but each color can be assigned to one of three cherries. By Hall's theorem, an assignment saturating the colors always exists and can be found in polynomial time. Observe that, in this way, no two siblings in N are involved in γ.

In a similar way, Hall's theorem also allows us to construct a mapping γ between the leaves of any 3-labeled tree constructed by Construction 1 such that each color is mapped to two leaves and all nodes have at most one child involved in γ.

Corollary 5 *Soft Tree Containment is NP-hard, even on 2-labeled binary tree-child networks N. Soft Cluster Containment is NP-hard, even on 2-labeled tree-child networks N.*

6 Conclusion

We initiated research into a practically relevant variant of the Tree- and Cluster Containment problems handling soft polytomies. We lay the mathematical foundation to dealing with soft polytomies and showed that the (classical) complexity of both problems are hard, even on classes for which their "firm"-versions are polynomial-time solvable. Hope lies in the special case of 2-labeled trees, which has been used as building block for algorithms for the "firm"-versions and which we show to be solvable in cubic time. 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 [11, 20] 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 (Corollary 3) is transferable to multi-labeled networks, leaving many open questions in this direction. Finally, given the close relationship between Soft Tree Containment and (FIRM) Cluster Containment, we hope to apply ideas and methods used for (FIRM) Cluster Containment to also attack Soft Tree Containment. In particular, we hope that the ideas in Theorem 4 can be adapted since Cluster Containment seems to exhibit a close relationship to SAT [13]—similar to what we exploited to prove Theorem 4.

References

[1] M. Bentert, J. Malík, and M. Weller. Tree containment with soft polytomies. In *16th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2018, June 18-20, 2018, Malmö, Sweden*, volume 101 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SWAT.2018.9.

[2] M. Bordewich and C. Semple. Reticulation-visible networks. *Advances in Applied Mathematics*, 78:114–141, 2016. doi:10.1016/j.aam.2016.04.004.

[3] J. M. Chan, G. Carlsson, and R. Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences*, 110(46):18566–18571, 2013. doi:10.1073/pnas.1313480110.

[4] 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. doi:10.1137/S0097539796313477.

[5] A. Dress, K. Huber, J. Koolen, V. Moulton, and A. Spillner. *Basic Phylogenetic Combinatorics*. Cambridge University Press, 2004. doi:10.1017/CBO9781139019767.

[6] 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. doi:10.1109/JCSSE.2015.7219820.

[7] P. Gambette. personal communication, 2018.

[8] 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. doi:10.1007/978-3-319-16706-0_12.

[9] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978. doi:10.1016/S0019-9958(78)90562-4.

[10] 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. doi:10.1016/j.ic.2016.11.001.

[11] A. D. Gunawan, B. Lu, and L. Zhang. A program for verification of phylogenetic network models. *Bioinformatics*, 32(17):i503–i510, 2016. doi:10.1093/bioinformatics/btw467.

[12] A. D. M. Gunawan. Solving the tree containment problem for reticulation-visible networks in linear time. In *Proceedings of the 5th International Conference on Algorithms for Computational Biology AlCoB*, volume 10849 of *Lecture Notes in Computer Science*, pages 24–36. Springer, 2018. doi:10.1007/978-3-319-91938-6_3.

[13] A. D. M. Gunawan, B. Lu, and L. Zhang. Fast methods for solving the cluster containment problem for phylogenetic networks. *CoRR*, abs/1801.04498, 2018.

[14] D. Gusfield. *ReCombinatorics: the algorithmics of ancestral recombination graphs and explicit phylogenetic networks.* MIT Press, 2014. doi:10.7551/mitpress/9432.001.0001.

[15] D. H. Huson, R. Rupp, and C. Scornavacca. *Phylogenetic networks: concepts, algorithms and applications.* Cambridge University Press, 2010. doi:10.1093/sysbio/syr054.

[16] 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. doi:10.1016/j.tcs.2008.04.019.

[17] 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. doi:10.1371/journal.pgen.1001284.

[18] 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.

[19] L. Van Iersel, C. Semple, and M. Steel. Locating a tree in a phylogenetic network. *Information Processing Letters*, 110(23):1037–1043, 2010. doi:10.1016/j.ipl.2010.07.027.

[20] M. Weller. Linear-time tree containment in phylogenetic networks. In *Proceedings of the 16th International Conference on Comparative Genomics RECOMB-CG*, volume 11183 of *Lecture Notes in Computer Science*, pages 309–323. Springer, 2018. doi:10.1007/978-3-030-00834-5_18.