MUL-Tree Pruning for Consistency and Compatibility

Christopher Hampson
Department of Informatics, King's College London, UK

Daniel J. Harvey
Graduate School of Informatics, Kyoto University, Japan

Costas S. Iliopoulos
Department of Informatics, King's College London, UK

Jesper Jansson
Graduate School of Informatics, Kyoto University, Japan

Zara Lim
Department of Informatics, King's College London, UK

Wing-Kin Sung
Department of Chemical Pathology, The Chinese University of Hong Kong, China
Hong Kong Genome Institute, Hong Kong Science Park, Shatin, China
Laboratory of Computational Genomics, Li Ka Shing Institute of Health Sciences,
The Chinese University of Hong Kong, China

Abstract

A multi-labelled tree (or MUL-tree) is a rooted tree leaf-labelled by a set of labels, where each label may appear more than once in the tree. We consider the MUL-tree Set Pruning for Consistency problem (MULSETPC), which takes as input a set of MUL-trees and asks whether there exists a perfect pruning of each MUL-tree that results in a consistent set of single-labelled trees. MULSETPC was proven to be NP-complete by Gascon et al. when the MUL-trees are binary, each leaf label is used at most three times, and the number of MUL-trees is unbounded. To determine the computational complexity of the problem when the number of MUL-trees is constant was left as an open problem.

Here, we resolve this question by proving a much stronger result, namely that MULSETPC is NP-complete even when there are only two MUL-trees, every leaf label is used at most twice, and every MUL-tree is either binary or has constant height. Furthermore, we introduce an extension of MULSETPC that we call MULSETPComp, which replaces the notion of consistency with compatibility, and prove that MULSETPComp is NP-complete even when there are only two MUL-trees, every leaf label is used at most thrice, and every MUL-tree has constant height. Finally, we present a polynomial-time algorithm for instances of MULSETPC with a constant number of binary MUL-trees, in the special case where every leaf label occurs exactly once in at least one MUL-tree.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases multi-labelled tree, phylogenetic tree, consistent, compatible, pruning, algorithm, NP-complete

Digital Object Identifier 10.4230/LIPIcs.CPM.2023.14

Funding *Daniel J. Harvey*: Partially funded by JSPS KAKENHI grant 22H03550.
Jesper Jansson: Partially funded by JSPS KAKENHI grant 22H03550.

1 Introduction

In evolutionary biology, leaf-labelled (phylogenetic) trees are commonly employed to describe the evolution of species using leaf labels to represent different species [11]. Comparisons of these structures are used particularly in phylogenetic inferences – similarities may indicate evolutionary patterns, whereas differences may highlight genetic mutations. The measure of similarity between phylogenetic trees has been defined by multiple alternate metrics, such as the Robinson-Foulds distance [29], subtree pruning and regraft (SPR) distances [5, 34], and maximum agreement subtrees [2, 7, 12]. Other problems related to phylogenetic trees include constructing supertrees [1, 3, 4, 33] or consensus trees [6, 11, 21] which can determine relations or interactions between smaller phylogenetic trees.

Phylogenetic trees are classically described as single-labelled trees, where no label appears on the leaves of the tree more than once. Typically, construction or comparison algorithms of such phylogenetic trees make use of this property to reduce computational costs. Multi-labelled trees (or MUL-trees) are a generalisation of single-labelled trees in which multiple leaves may be labelled by the same label. MUL-trees can be useful to depict genome duplication, lineage sorting, or lateral gene transfer [23]. Other applications include the construction of phylogenetic networks by folding operations [17, 18, 19], biogeography [13, 24, 25], the study of host-parasite cospeciation [26], and gene evolution studies [23, 27, 30].

MUL-trees have been far less investigated than their single-labelled counterparts and many computational problems become NP-hard when extended to MUL-trees. For example, the majority rule consensus tree for a set of k single-labelled trees with n leaf labels each can be computed in $O(nk)$ time [21], but is NP-hard to compute for MUL-trees [8]. Other approaches convert MUL-trees into single-labelled trees which can be input to existing algorithms [20, 30]. A few polynomial-time algorithms do exist for MUL-trees – Cui et al.[8] presented a $O(n^2k + nk^2)$-time algorithm for building a majority rule consensus MUL-tree in which each leaf label occurs at most twice, based on a reduction to the Perfect Phylogeny Haplotyping problem [10]. Furthermore, the maximum agreement subtree (MAST) distance between two MUL-trees can be computed in quadratic time, though it also becomes NP-complete when generalised to more than two MUL-trees [13, 22].

This paper investigates MUL-trees by considering the *MUL-tree Set Pruning for Consistency* problem (MULSETPC), which takes as input a set of MUL-trees, and outputs whether or not there exists a pruning of the MUL-trees which gives a consistent set of single-labelled trees (see Section 2 for formal definitions). Gascon et al. showed that, in general, MULSETPC is NP-complete via a polynomial reduction from 3-SAT [15, 16]. However, their reduction from an instance of 3-SAT with m variables and z clauses gives an instance of MULSETPC containing $m + z + 1$ MUL-trees; moreover these MUL-trees may have labels occurring three times. Here we prove that MULSETPC is still NP-complete, even when restricted to instances involving only two MUL-trees, or in which every leaf label appears at most twice within a MUL-tree. This totally resolves the open question of Gascon et al. [15, 16] regarding the parameterised complexity of MULSETPC when the parameter is the number of input trees. Also, we identify tractable fragments of MULSETPC which can be solved in polynomial time, in short, instances in which each label appears exactly once in at least one MUL-tree.

We also present a generalisation of MULSETPC called MULSETPComp, which asks for a *compatible* set of trees instead of a *consistent* set of trees. Tree compatibility is a generalisation of tree consistency where we allow the supertree displaying the set to instead display a refinement of each tree rather than the tree itself–again, a more rigorous definition is given later. Tree compatibility is relevant when determining the existence of a supertree for a given set of phylogenetic trees [1, 31]. This is because experimental data often contains uncertainty, which can be expressed by non-binary nodes in the tree; this necessitates the use of compatibility. Compatibility is also relevant to other questions such as the incomplete directed perfect phylogeny problem [28]. Our interest in tree compatibility was partially motivated by recent improvements in compatibility testing [9]. Here, we prove that MULSETPComp is NP-complete, even when restricted to instances involving only two MUL-trees, or in which every leaf label appears at most thrice within a MUL-tree.

The rest of the paper is organized as follows. In Section 2, we introduce the preliminary notation and definitions. In Section 3 we present the improved NP-completeness proof for MULSETPC by reduction from the Boolean 3-SAT problem. In Section 4 we give a NP-completeness proof for MULSETPComp by reduction from the Exact 3-cover with Multiplicity 3 problem. Section 5 contains our polynomial time results for tractable instances of MULSETPC. Finally, in Section 6 we present our conclusions and a few open problems.

2 Preliminaries

We shall use the following standard definitions on trees.

▶ **Definition 1** (Basic tree definitions). *All trees we consider are rooted and unordered. If x, y are nodes in a tree T, then y is an* ancestor *of x (and x a* descendant *of y) if y lies on the unique path from x to the root of T. We denote this by $x \leq y$. Additionally, if $y \neq x$ then y is a* proper ancestor *of x, which we denote by $x < y$. If $x < y$ and y is adjacent to x then y is the* parent *of x and x a* child *of y. If x and x' are both children of y then x and x' are* siblings. *The* lowest common ancestor *of nodes x and y, denoted $\mathrm{lca}_T(x, y)$, is the node z such that $x \leq z$, $y \leq z$, and no proper descendant of z also satisfies these properties. The empty tree, denoted by $T_\emptyset$, is the unique tree which contains no nodes.*

We use the following definition of leaf-labelled trees, which takes the definitions of Gascon et. al. [15] and generalises them to the case where the tree may not be binary.

▶ **Definition 2** (Leaf-labelled trees). *A* leaf-labelled tree $(T, \mathcal{X})$ *is a (rooted, unordered) tree T where no node has exactly one child and where each leaf has been assigned a label from a set of labels $\mathcal{X}$. (We will sometimes refer to T as a* leaf-labelled tree on *$\mathcal{X}$, and omit $\mathcal{X}$ if it is clear from context.) A leaf-labelled tree is a* single-labelled tree *if every label in $\mathcal{X}$ is used at most once. Alternatively, a* multi-labelled tree *or* MUL-tree *is a leaf-labelled tree where we allow each label in $\mathcal{X}$ to label multiple leaves. We say a MUL-tree has* multiplicity k *if each leaf label appears at most k times. Let $L(T) \subseteq \mathcal{X}$ denote the set of leaf labels appearing in T and let $D(T) \subseteq L(T)$ denote the set of leaf labels appearing only once in T.*

Note that in a single-labelled tree we sometimes abuse notation and identify the leaf and the leaf label. We do the same in a MUL-tree only if the context is clear and there is no possibility of confusion.

If u is a node of T then T^u denotes the subtree rooted at u containing u and all its descendants, maintaining the same leaf-labelling as T on the remaining leaves. Let $D^u := D(T^u)$.

▶ **Definition 3** (Pruning and Perfect Pruning). *Given a leaf-labelled tree T, let y denote a leaf node of T and x the parent of y. We* prune *the leaf y in the following manner:*

- *Delete the leaf y.*
- *If x still has at least two children, do nothing else.*
- *Alternatively, if x now has only one child and is not the root, suppress the vertex x.*
- *Finally, if x has only one child z and is the root, delete x and make z the new root.*

A perfect pruning *of T is a single-labelled tree T' such that $L(T') = L(T)$, created by (possibly repeated) prunings of T. That is, for every label that appears more than once in T, we prune away all but exactly one copy of the label to obtain a single-labelled tree. If T is a single-labelled tree, then its only perfect pruning is itself.*

Given a leaf-labelled tree T and a set of leaf labels $L' \subseteq L(T)$, let $T\restriction_{L'}$ denote the leaf-labelled tree constructed by pruning from T every leaf labelled by a label from $L(T) - L'$. (That is, only leaves labelled by L' remain.) Two leaf-labelled trees T_1 and T_2 are *leaf-label isomorphic* if there is an isomorphism between T_1 and T_2 which preserves the labelling of the leaves. We say that a leaf-labelled tree T on L *displays* a single-labelled tree T' on $L' \subseteq L$ if there exists a perfect pruning T^* of T such that $T^*\restriction_{L'}$ is leaf-label isomorphic to T'.

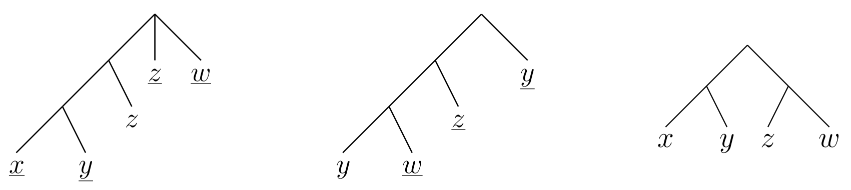
▶ **Definition 4** (Refinement). *Given single-labelled trees T and T^*, we say T^* is a refinement of T if T can be obtained from T^* by (possibly repeated) contractions of non-leaf edges, where we treat a contraction as merging the child node into the parent node. We write $T \leq T^*$.*

What follows is the definition of a *consistent set*, and the very similar definition of a *compatible set.*

▶ **Definition 5** (Consistent Set). *Consider a set of single-labelled trees $T_1, \ldots, T_k$ with corresponding label sets $L_1, \ldots, L_k$. We say this set is* consistent *if there exists a single-labelled tree T on label set $L = \bigcup_{i=1}^{k} L_i$ such that for every $i = 1, \ldots, k$, T displays T_i.*

Note in the above definition that if $L_1 = \cdots = L_k$ then $L = L_1$ and so the set $T_1, \ldots, T_k$ is consistent if and only if the trees are pairwise leaf-label isomorphic.

▶ **Definition 6** (Compatible Set). *Consider a set of single-labelled trees $T_1, \ldots, T_k$ with corresponding label sets $L_1, \ldots, L_k$. We say this set is* compatible *if there exists a single-labelled tree T on label set $L = \bigcup_{i=1}^{k} L_i$ such that for every $i = 1, \ldots, k$, $T\restriction_{L_i}$ is a refinement of T_i.*



(a) T_1. **(b)** T_2. **(c)** T^*.

Figure 1 Let T_1, T_2, and T^* be the three trees with $\mathcal{X} = \{x, y, z, w\}$ shown above. If we prune away the non-underlined labels in T_1 and T_2 then $T^*\restriction_{L(T_i)}$ is a refinement of the pruned T_i for $i \in \{1, 2\}$, which shows that there exists a perfect pruning of $\{T_1, T_2\}$ giving a compatible set of trees. In contrast, there is no perfect pruning of $\{T_1, T_2\}$ giving a consistent set of trees because neither of the two single-labelled subtrees with leaf labels y, z, and w displayed by T_1 is also displayed by T_2. Note, however, that if the label w in T_1 is changed to x then $\{T_1, T_2\}$ becomes consistent since this label can be pruned along with one leaf labelled by z to obtain a perfect pruning of T_1 which is displayed by T^*.

Note that every consistent set of trees is also a compatible set, but the converse does not hold in general.

The MUL-set pruning for consistency problem can then be defined as follows:

MUL-tree Set Pruning for Consistency (MULSETPC) Problem:
Input: $(\mathcal{M}, \mathcal{X})$ where $\mathcal{M}$ is a set of MUL-trees on $\mathcal{X}$.
Output: $\exists$? a perfect pruning of each tree of $\mathcal{M}$ resulting in a consistent set of trees.

We introduce the following problem, which substitutes *compatible* for *consistent* sets:

MUL-tree Set Pruning for Compatibility (MULSETPComp) Problem:
Input: $(\mathcal{M}, \mathcal{X})$ where $\mathcal{M}$ is a set of MUL-trees on $\mathcal{X}$.
Output: $\exists$? a perfect pruning of each tree of $\mathcal{M}$ resulting in a compatible set of trees.

See Figure 1 for an example that illustrates the difference between MULSETPC and MULSETPComp.

3 NP-completeness for MULSETPC instances with two MUL-trees and multiplicity 2

In this section we consider the MULSETPC problem, and show that it is NP-complete even when considering a heavily restricted set of instances. Specifically, we consider instances with at most two MUL-trees, where the multiplicity is 2, and where the MUL-trees are binary. The core of our proof will be a reduction from 3-SAT [14].

3-satisfiability (3-SAT) Problem:
Input: A Boolean set of clauses $C = (C_1 \wedge C_2 \wedge \ldots \wedge C_z)$ on a finite set of literals $\{l_1, l_2, \ldots, l_m\}$ where each clause is in conjunctive normal and contains 3 literals.
Output: $\exists$? a satisfying valuation $\mathcal{V}$ of C.

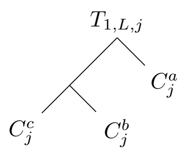
Our first goal is to construct, given an instance of 3-SAT, two MUL-trees T_1 and T_2 which we will use in our corresponding instance of MULSETPC. Our set of leaf labels $\mathcal{X}$ consists of the following:

- $\{l_i, \bar{l}_i \mid i = 1, \ldots, m\}$, the set of literals,
- $\{F_i, V_i \mid i = 1, \ldots, m\}$, a pair of "dummy" labels for each variable and
- $\mathcal{P} := \{C_j^1, C_j^2, C_j^3 \mid j = 1, \ldots, z\}$, a triple of "position" labels representing the three places of each clause of C.

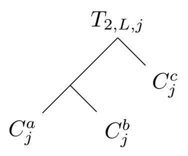
Define $h := \mathcal{P} \to \{l_i, \bar{l}_i \mid i = 1, \ldots, m\}$ as the function which maps a position label C_j^x to the literal found in that position. For example, if $C_1 = (l_1 \vee l_4 \vee \bar{l}_6)$ then $h(C_1^1) = l_1, h(C_1^2) = l_4$ and $h(C_1^3) = \bar{l}_6$. We treat $\mathcal{P}$ as being ordered first by the index of the clause and then by the position. Let $H(l_i) := \{C_j^x \mid h(C_j^x) = l_i\}$, and define $H(\bar{l}_i)$ similarly.

Our trees T_1, T_2 will be constructed from four types of subtrees $T_{1,L,j}, T_{2,L,j}, T^*_{1,R,i}$ and $T^*_{2,R,i}$, where $i = 1, \ldots, m$ and $j = 1, \ldots, z$. See Figure 2 for the subtrees $T_{1,L,j}, T_{2,L,j}, T_{1,R,i}$ and $T_{2,R,i}$; we now explain how to construct $T^*_{1,R,i}$ and $T^*_{2,R,i}$ from $T_{1,R,i}$ and $T_{2,R,i}$.

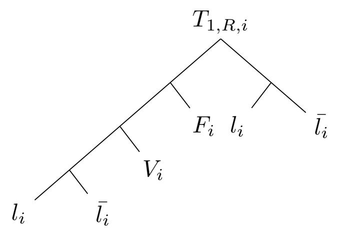
Let v_i denote the leaf labelled V_i in $T_{1,R,i}$ and let u_i denote the parent of v_i. Let u_i^+ and u_i^- denote the parents of leaves labelled l_i and $\bar{l}_i$ respectively in $T_{2,R,i}$. Let v_i^+ denote the child of u_i^+ labelled by V_i, and v_i^- denote the child of u_i^- labelled by V_i.



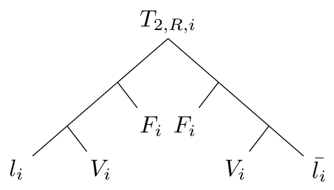
(a) $T_{1,L,j}$.



(b) $T_{2,L,j}$.



(c) $T_{1,R,i}$.



(d) $T_{2,R,i}$.

Figure 2 Subtrees $T_{1,L,j}, T_{2,L,j}, T_{1,R,i}, T_{2,R,i}$ for MULSETPC.

Initialise $T^*_{1,R,i}$ as a copy of $T_{1,R,i}$, then do the following:

- If $H(l_i) \cup H(\bar{l}_i) = \emptyset$, make no further changes.
- Otherwise, subdivide the edge $u_i v_i$ $|H(l_i) \cup H(\bar{l}_i)|$ times, and add to each new node an adjacent leaf. Label these leaves with $H(l_i) \cup H(\bar{l}_i)$, respecting the ordering such that the first label is closest to u_i.

Initialise $T^*_{2,R,i}$ as a copy of $T_{2,R,i}$, and then:

- If $H(l_i) \neq \emptyset$, then subdivide $u_i^+ v_i^+$ $|H(l_i)|$ times and add an leaf adjacent to each new node. Label these leaves with $H(l_i)$, respecting the ordering so that the first label is closest to u_i^+.
- If $H(\bar{l}_i) \neq \emptyset$, repeat the previous step, substituting $H(\bar{l}_i)$ for $H(l_i)$ and $u_i^- v_i^-$ for $u_i^+ v_i^+$.

Construct $T_{1,L}$ by taking a complete binary tree on z leaves (recall z is the number of clauses), suppressing any nodes with exactly one child, and then identifying the root of each $T_{1,L,j}$ (ordered by j) with exactly one of the leaves (ordered left-to-right). Construct $T_{2,L}$ in the same fashion, substituting $T_{2,L,j}$ for $T_{1,L,j}$. Construct $T_{1,R}$ by taking a complete binary tree on m leaves, suppressing any nodes with exactly one child, and then identifying the root of each $T^*_{1,R,i}$ (ordered by i) with exactly one of the leaves (ordered left-to-right). Again, construct $T_{2,R}$ in the same fashion, substituting $T^*_{2,R,i}$ for $T^*_{1,R,i}$. Finally, construct T_1 by taking a root node $r(T_1)$ and adding an edge to the roots of $T_{1,L}$ and $T_{1,R}$; construct T_2 in the obvious equivalent fashion.

Given an instance C of 3-SAT, we create our instance of MULSETPC, $(\{T_1, T_2\}, \mathcal{X})$. Note that T_1 and T_2 have multiplicity 2; each leaf label $C_j^x \in \mathcal{P}$ appears once in $T_{1,L,j}$ and $T_{2,L,j}$ and once in $T^*_{1,R,i}$ and $T^*_{2,R,i}$ for the single value of i such that $C_j^x \in H(l_i) \cup H(\bar{l}_i)$. By inspection, the labels of $\mathcal{X} - \mathcal{P}$ also appear at most twice. Hence the instance $(\{T_1, T_2\}, \mathcal{X})$ contains two binary MUL-trees with multiplicity 2. It suffices to now show the reduction, in two parts.

▶ **Lemma 7.** *If C is a satisfied instance of 3-SAT then the corresponding instance $(\{T_1, T_2\}, \mathcal{X})$ of* MULSETPC *admits a perfect pruning giving a consistent set of trees.*

Proof. Suppose that C is satisfiable. Then there exists a valuation of every variable which satisfies every clause of C; fix one such valuation and label it $\mathcal{V}$. For each clause C_j, *mark* one of the position labels C_j^x for $x \in \{1, 2, 3\}$ such that $h(C_j^x)$ is valued true by $\mathcal{V}$. Since $\mathcal{V}$ satisfies every clause, we will always be able to choose a label to mark; if there are multiple legitimate choices choose arbitrarily. Refer to any label C_j^x we have not marked as *unmarked*.

By our construction of T_1 and T_2, if we show that after pruning each $T_{1,L,j}$ and $T^*_{1,R,i}$ is leaf-label isomorphic to $T_{2,L,j}$ and $T^*_{2,R,i}$ respectively, then T_1 is leaf-label isomorphic to T_2.

Consider first the labels of $\mathcal{P}$. Prune from each $T_{1,L,j}$ and $T_{2,L,j}$ the one marked label C_j^x, and leave the two unmarked labels C_j^x. We must keep the other copy of the marked C_j^x and prune away the other copies of the unmarked C_j^x in whichever $T^*_{1,R,i}$ and $T^*_{2,R,i}$ they appear. Consider the sets $H(l_i)$ and $H(\bar{l}_i)$. If C_j^x is marked, then $h(C_j^x)$ is true, and so at most one of $H(l_i), H(\bar{l}_i)$ contains a marked label. Keeping this information in mind, we can now simply look at the subtrees themselves.

- After this pruning each $T_{1,L,j}$ will be leaf-label isomorphic to the corresponding $T_{2,L,j}$, by inspection.
- Consider $T^*_{1,R,i}$ and $T^*_{2,R,i}$. We have already pruned away unmarked labels of $\mathcal{P}$. If l_i is true, prune the copy of l_i in $T^*_{1,R,i}$ closest to the root of $T^*_{1,R,i}$ and the copy of $\bar{l}_i$ furthest from the root; if $\bar{l}_i$ is true do the opposite. In $T^*_{2,R,i}$ prune the copies of F_i, V_i closer to the literal l_i or $\bar{l}_i$ which evaluates as false. Hence we have pruned away the extra copies of each leaf label. It is clear, mostly by inspection, that $T^*_{1,R,i}$ is leaf-label isomorphic to $T^*_{2,R,i}$; the most important point is that since at most one of $H(l_i)$ and $H(\bar{l}_i)$ contains a marked label, at least one of these sets has been pruned away entirely in $T^*_{2,R,i}$ (specifically the set closer to the false literal).

Hence, after pruning, T_1 and T_2 are leaf-label isomorphic, and thus $\{T_1, T_2\}$ is consistent. ◀

See Figure 3 for an illustration of the reduction in Lemma 7.

▶ **Lemma 8.** *If C is not a satisfied instance of 3-SAT then the corresponding instance $(\{T_1, T_2\}, \mathcal{X})$ of* MULSETPC *do not admit perfect prunings giving a consistent set of trees.*

We omit the proof of Lemma 8 for reasons of space. We now prove our main result.

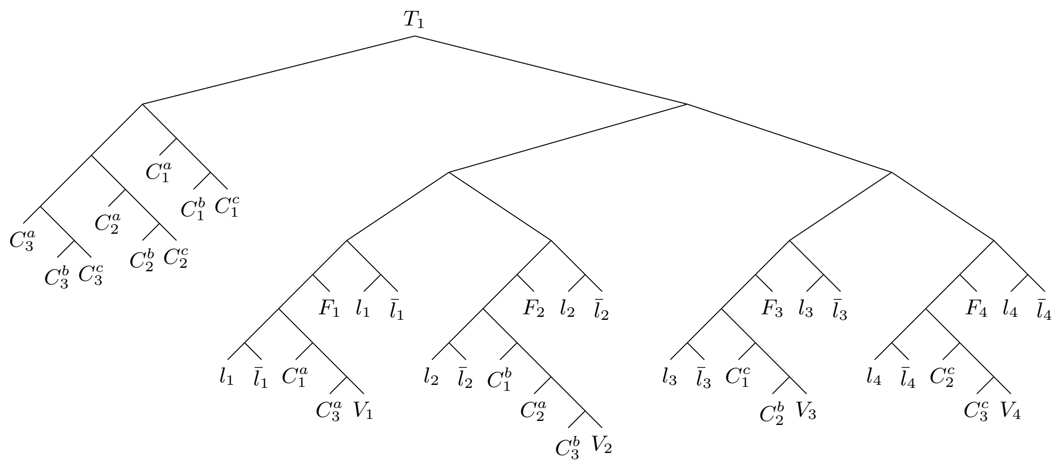
▶ **Theorem 9.** *The* MULSETPC *problem is NP-complete, even restricted to instances containing at most two binary MUL-trees with multiplicity 2.*

Proof. Note first that MULSETPC is in NP, since given a set of pruned leaves, a perfect pruning can be constructed in polynomial time, and the consistency of the set of trees determined in polynomial time using the BUILD algorithm [1, 15].

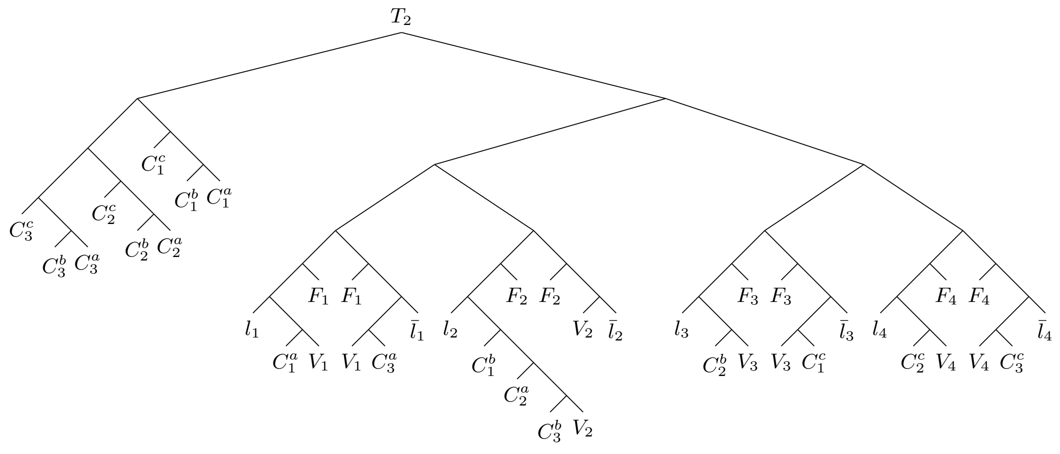
The result then follows directly from Lemma 7 and Lemma 8. ◀

It is also possible to swap our requirement that the MUL-trees are binary for an alternative requirement that the MUL-trees have height at most 5. Proving this result is very similar to the binary case, but we omit it here on grounds of space; the proof will appear in the journal version of this article. Thus we get the following result.

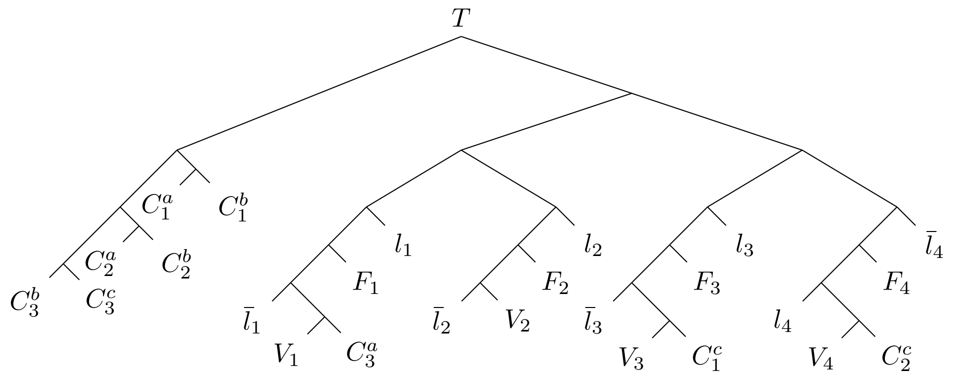
▶ **Theorem 10.** *The* MULSETPC *problem is NP-complete, even restricted to instances containing at most two MUL-trees with multiplicity 2 and height at most 5.*



(a) The MUL-tree T_1.


T_2
C_1^c C_1^b C_1^a
C_2^c C_2^b C_2^a
C_3^c C_3^b C_3^a
F_1 F_1 l_1 $\bar{l}_1$
C_1^a V_1 V_1 C_3^a
F_2 F_2 l_2 V_2 $\bar{l}_2$
C_1^b C_2^a C_3^b V_2
F_3 F_3 l_3 $\bar{l}_3$
C_2^b V_3 V_3 C_1^c
F_4 F_4 l_4 $\bar{l}_4$
C_2^c V_4 V_4 C_3^c

(b) The MUL-tree T_2.


T
C_1^a C_1^b
C_2^a C_2^b
C_3^b C_3^c
l_1 F_1 $\bar{l}_1$ V_1 C_3^a
l_2 F_2 $\bar{l}_2$ V_2
l_3 F_3 $\bar{l}_3$ V_3 C_1^c
$\bar{l}_4$ F_4 l_4 V_4 C_2^c

(c) Tree T displays the two MUL-trees T_1 and T_2 after they have been perfectly pruned.

Figure 3 Illustrating the reduction from 3-SAT in Lemma 7. The two MUL-trees T_1 and T_2 are constructed from $C = (C_1 \wedge C_2 \wedge C_3)$ where $C_1 = (l_1 \vee l_2 \vee \bar{l}_3)$, $C_2 = (l_2 \vee l_3 \vee l_4)$, and $C_3 = (\bar{l}_1 \vee l_2 \vee \bar{l}_4)$. Figure 3c shows the corresponding tree T which displays the pruned T_1 and T_2 corresponding to a satisfiable assignment $\bar{l}_1 = \bar{l}_2 = \bar{l}_3 = l_4 = true$ with marked labels C_1^c, C_2^c and C_3^a.

4 NP-completeness for MULSETPComp

In this section, we extend our previous results regarding the problem MULSETPC to the similar problem MULSETPComp. We present the following two theorems.

▶ **Theorem 11.** MULSETPComp *is NP-complete, even when restricted to instances containing at most two MUL-trees with multiplicity at most 3 and where the MUL-trees have height at most 4.*

▶ **Theorem 12.** MULSETPComp *is NP-complete, even when restricted to instances containing at most two MUL-trees with height at most 3 and where one MUL-tree is a single-labelled tree containing all leaf labels.*

As was the case in Section 3, these two results have very similar proofs. We shall prove Theorem 11, but omit the proof of Theorem 12 for space reasons. Here, we reduce from X3C3, also known to be NP-complete [14].

Exact 3-cover with multiplicity 3 (X3C3) Problem:
Input: A set $X = \{x_1, \dots, x_{3q}\}$ and a collection $C = \{S_1, \dots, S_k\}$ of 3-element subsets of X, such that any element of X appears in at most three sets in C.
Output: $\exists$? an exact cover for X.

As before, our first goal is to construct, given an instance (X, C) of X3C3, a set of two MUL-trees we shall use to construct our corresponding instance of MULSETPComp.

Recall that $|X| = 3q$ and that $k := |C|$. Let $m := k - q$, the number of sets of C *not* chosen to be part of our exact 3 cover. Define $A := \{a^i_j | i = 1, \dots, m,\ j = 1, \dots, k\}$ and $B = \{b^i_j | i = 1, \dots, m,\ j = 1, \dots, k\}$. Let $A^i = \{a^i_j | j = 1 \dots, k\}$ and $A_j = \{a^i_j | i = 1, \dots, m\}$, and define B^i, B_j similarly. The labels in $A \cup B$ are another set of "dummy" labels we use for technical reasons. Let $Y = X \cup A \cup B$. We may assume that $q \geq 3$. We may also assume that k is even; if not, add to X three additional elements $\{x_{3q+1}, x_{3q+2}, x_{3q+3}\}$ (which increases q by 1) and add to C a additional set $S' = \{x_{3q+1}, x_{3q+2}, x_{3q+3}\}$ (which increases $k = |C|$ by 1). It is clear this modified instance contains an exact 3-cover if and only if the original instance did.

We denote our two trees T_1 and T_2. Our corresponding instance of MULSETPComp for Theorem 11 will be $(\{T_1, T_2\}, Y)$.

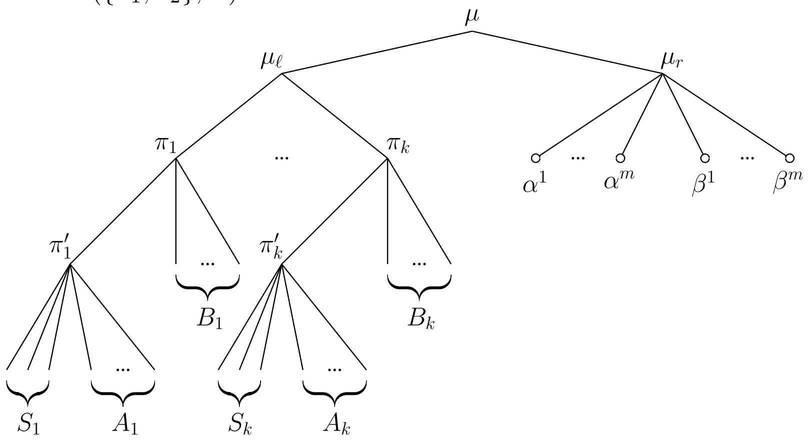


Figure 4 The tree T_1 for MULSETPComp.

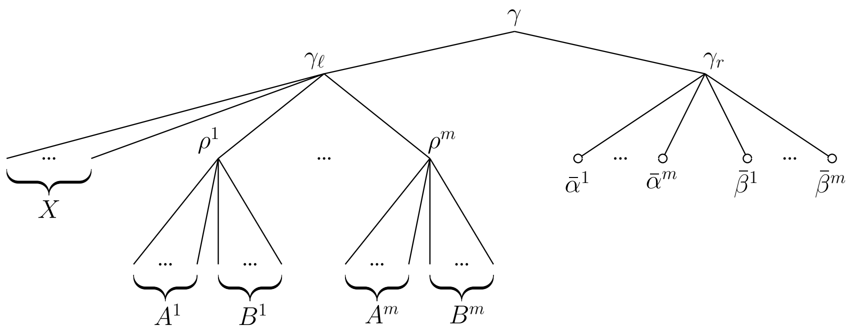


Figure 5 The tree T_2 for MULSETPComp.

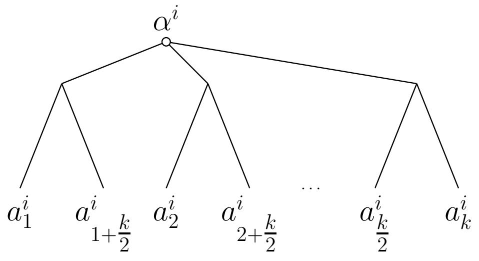


Figure 6 The subtree of T_1 rooted by α^i.

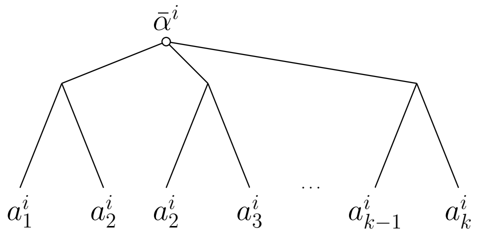


Figure 7 The subtree of T_2 rooted by $\bar{\alpha}^i$.

See Figures 4 and 6 for the construction of T_1, and Figures 5 and 7 for the construction of T_2. Note that a node labelled α^i in Figure 4 is the root of the appropriate subtree from Figure 6, not a leaf labelled by α^i. An equivalent statement holds for $\bar{\alpha}^i, \beta^i$ and $\bar{\beta}^i$, where the subtree rooted at β^i is found by taking the subtree of Figure 6 and replacing each a^i_j with b^i_j. (An equivalent statement holds for $\bar{\beta}^i$ and Figure 7). Note the following other useful facts about our construction:

- Trees T_1 and T_2 have the same set of leaf labels. Hence a perfect pruning of T_1 and T_2 is a compatible set of trees if and only if there exists a tree T^* on the same leaf label set which is a refinement of both perfect prunings.
- In T_1 any label of X may appear at most three times, since any x_i may appear in at most three sets of C. Every label of A and B appears once in $T_1^{\mu_\ell}$ and once in $T_1^{\mu_r}$. Hence T_1 has multiplicity 3. In T_2 any label of X appears only once, and the labels of A and B appear at most thrice, once in $T_2^{\gamma_\ell}$ and once or twice in $T_2^{\gamma_r}$. Hence T_2 has multiplicity 3.
- The heights of both MUL-trees can be determined by inspection.

We will need the following two technical lemmas – as the proofs are straightforward we omit them.

▶ **Lemma 13.** *Consider a set of elements $X = \{x_1, \dots x_k\}$ together with a subset $X' \subset X$ such that $|X'| = k - 1$ and a collection of sets $\mathcal{X} = \{x_i, x_{i+i}\}_{i \in [k-1]}$. Then it is possible to construct a set equal to X' by choosing one element from each set in $\mathcal{X}$.*

▶ **Lemma 14.** *Let T, T^* be single-labelled trees such that $T \leq T^*$, and let u, x, y be leaf nodes in T (and hence also in T^*). If $\mathrm{lca}_T(u,x) < \mathrm{lca}_T(u,y)$ then $\mathrm{lca}_{T^*}(u,x) < \mathrm{lca}_{T^*}(u,y)$.*

The following two lemmas form the core of our main result.

▶ **Lemma 15.** *If (X, C) is an instance of X3C3 that allows an exact 3-cover C' then the corresponding instance $(\{T_1, T_2\}, Y)$ of* MULSETPComp *admits a perfect pruning giving a compatible set of trees.*

Proof. Let $\mathcal{I} \subset [k]$ denote the set of m indices i of those S_i we did not choose as part of our exact 3-cover, that is the sets $S_i \in C - C'$. Let $\psi : \mathcal{I} \to [m]$ be an arbitrary isomorphism. Prune the tree T_1 as follows:

- For each S_i (or equivalently, each subtree rooted at π_i):
  - If $S_i \in C'$ then keep the labels of S_i as the children of π'_i but prune away all other leaf labels in the subtree $T_1^{\pi_i}$. After pruning there are three leaves (with labels corresponding to the elements of S_i) as children of π'_i, which is itself a child of μ_ℓ.
  - If $S_i \notin C'$ then prune away all leaf labels in $T_1^{\pi_i}$ except $a_i^{\psi(i)}$ and $b_i^{\psi(i)}$. After pruning, the remaining leaves will be children of π_i.
- In each $T_1^{\alpha^j}$, prune away a_i^j if the leaf label appears in $T_1^{\mu_\ell}$. Since m sets of C are not in C', there are m leaf labels of the form a_i^j appearing in $T_1^{\mu_\ell}$, specifically $a_i^{\psi(i)}$ for the m values of i for which $\psi(i)$ is defined, or equivalently for all $\psi(i) = 1, \ldots, m$. Hence we must prune away one leaf label in each $T_1^{\alpha^j}$.
- Repeat the previous step for each $T_1^{\beta^j}$ – the argument is identical.

Denote the pruned version of T_1 by T'_1. Every leaf label of X appears once; this follows directly from C' being an exact 3 cover. All other leaf labels appear only once by inspection; hence this is a perfect pruning.

We now prune T_2 as follows:

- For each $T_2^{\rho^j}$, prune away all leaf labels except $a^j_{\psi^{-1}(j)}$ and $b^j_{\psi^{-1}(j)}$; after pruning ρ^j has two children, both leaves.
- For each $T_2^{\bar{\alpha}^j}$, we wish to prune this subtree to create a $(k-1)$ leaf star with all labels a_i^j for our fixed j except $a^j_{\psi^{-1}(j)}$. This is possible due to Lemma 13; from each pair of leaf labels rooted by a child of $\bar{\alpha}^j$ we pick one leaf to keep and one to prune away such that we keep one copy of everything except $a^j_{\psi^{-1}(j)}$.
- Repeat the previous step for each $T_2^{\bar{\beta}^j}$ – as before the argument is identical.

There is one copy of each leaf label of X in T_2, which we do not prune. We prune the leaf labels of $A \cup B$ in T_2 so that the leaf labels of $A \cup B$ in $T_2^{\gamma_r}$ are exactly those that do not appear in $T_2^{\gamma_\ell}$. Hence this is a perfect pruning, which we denote T'_2.

We now show that T'_2 can be constructed from T'_1 by repeated non-leaf edge contractions, which will show $\{T'_1, T'_2\}$ form a compatible set (with $T^* := T'_1$).

In T'_1 contract every remaining π'_i (one for each of the q sets $S_i \in C'$) into its parent μ_ℓ. Furthermore in each $T_1^{\alpha^i}$ and $T_1^{\beta^i}$, contract every non-leaf edge to create a star rooted at α_i or β_i. By inspection, we can see $T'_2 \leq T'_1$, giving our required compatible set of trees. ◀

See Figures 8, 9, 10, and 11 for an example of a pruning as in Lemma 15.

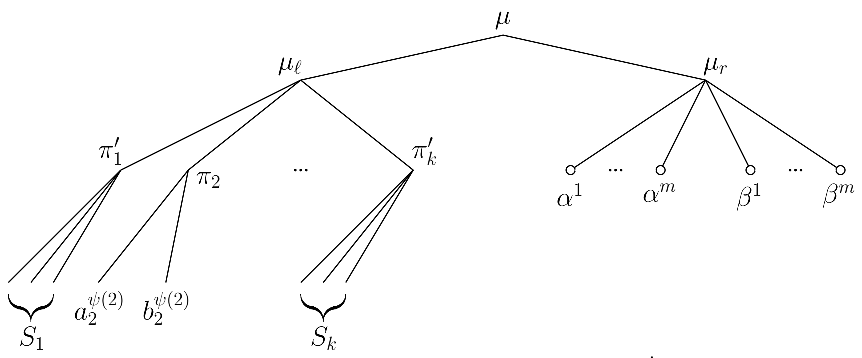


Figure 8 As an illustrated example, a possible perfect pruning T'_1 as in Lemma 15. In this example, $S_1, S_k \in C'$, but $S_2 \notin C'$.

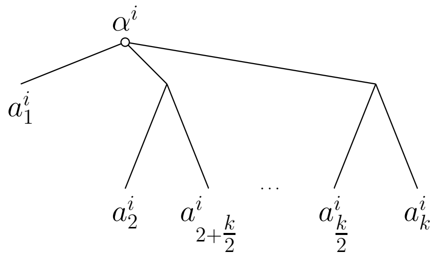


Figure 9 A subtree of T'_1 in the pruning from Figure 8. In this example, $\psi(1 + \frac{k}{2}) = i$.

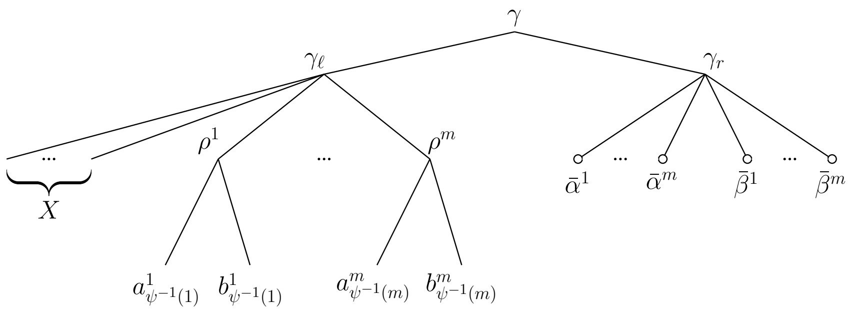


Figure 10 A possible perfect pruning T'_2, corresponding to the perfect pruning of Figure 8.

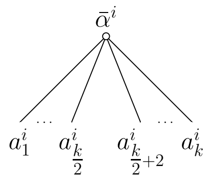


Figure 11 A subtree of T'_2 in the pruning from Figure 10. Again, here $\psi^{-1}(i) = \frac{k}{2} + 1$.

▶ **Lemma 16.** *If (X, C) is an instance of X3C3 that does not allow an exact 3-cover C' then the corresponding instance $(\{T_1, T_2\}, Y)$ of* MULSETPComp *does not admit a perfect pruning giving a compatible set of trees.*

We omit the proof of Lemma 16 due to space concerns. We now prove Theorem 11.

Proof of Theorem 11. Note first that MULSETPComp is in NP, since given a set of pruned leaves, a perfect pruning can be constructed in polynomial time. The compatibility of this set of trees can be determined in polynomial time using the BUILDST algorithm [9]. The result then follows directly from Lemma 15 and Lemma 16. ◀

We close this section with the following related result.

▶ Remark 17. MULSETPComp is NP-complete when restricted to instances with at most two MUL-trees with multiplicity 2, as long as all trees are binary.

Recall that in general any consistent set of trees is also a compatible set; in the case where all trees are binary the inverse also holds, as a binary tree cannot be refined further. This proves Remark 17.

5 An algorithm for MULSETPC instances with k binary MUL-trees where every label is unique in at least one tree

In this section, we consider the instances of MULSETPC in which we are given k binary MUL-trees $T_1, \ldots, T_k$ such that every label appears uniquely in at least one tree, that is, $\bigcup_{i=1}^{k} D(T_i) = \mathcal{X}$, where $\mathcal{X} = \bigcup_{i=1}^{k} \mathcal{X}(T_i)$.

We adapt a technique using dynamic programming over k-tuples of nodes previously used for the MAST problem [13, 22, 32]. For all k-tuples of nodes $(a_1, \ldots, a_k) \in \prod_{i=1}^{k} V(T_i)$, let $S(a_1, \ldots, a_k) = \bigcup_{i=1}^{k} D(T_i^{a_i})$ denote the set of unique leaf labels that occur in the subtrees $T_1^{a_1}, \ldots, T_k^{a_k}$, rooted at $a_1, \ldots, a_k$, respectively.

We aim to find a binary tree T that is leaf-labelled by $S(a_1, \ldots, a_k)$ such that each of $T_i^{a_i}$ for $i = 1, \ldots, k$ displays $T{\restriction}_{\mathcal{X}(T_i)}$. Lemma 18, below, shows that the necessary condition of the existence of such a tree is that $S(a_1, \ldots, a_k) \cap \mathcal{X}(T_i) \subseteq \mathcal{X}(T_i^{a_i})$ for $i = 1, \ldots, k$.

▶ **Lemma 18.** *Let T be a binary tree leaf-labelled by $S(a_1, \ldots, a_k)$. If $S(a_1, \ldots, a_k) \cap \mathcal{X}(T_i) \nsubseteq \mathcal{X}(T_i^{a_i})$ for some i, $T_i^{a_i}$ does not display $T{\restriction}_{\mathcal{X}(T_i)}$.*

Proof. Suppose that $S(a_1, \ldots, a_k) \cap \mathcal{X}(T_i) \nsubseteq \mathcal{X}(T_i^{a_i})$, then there exists $x \in S(a_1, \ldots, a_k)$ such that $x \notin \mathcal{X}(T_i^{a_i})$. Since $x \in T{\restriction}_{\mathcal{X}(T_i)}$, we have that $T_i^{a_i}$ cannot display $T{\restriction}_{\mathcal{X}(T_i)}$. ◀

Next, for each $a_i \in V(T_i)$, let $P(a_i) = \{\epsilon, a_i, a_i^l, a_i^r\}$, for each $i = 1, \ldots, k$, where a_i^ℓ and a_i^r denotes the two (unordered) children of vertex a_i, and with $a_i^l = \epsilon$ and $a_i^r = a_i$ in the case that $a_i \in L$ is a leaf vertex w.l.o.g. Let $c_i : P(a_i) \to P(x_i)$ be the involution given by

$$c_i(\epsilon) = a_i, \qquad c_i(a_i) = \epsilon, \qquad c_i(a_i^r) = a_i^\ell, \qquad \text{and} \qquad c_i(a_i^\ell) = a_i^r,$$

associating each $x \in P(a_i)$ with a complement. Let $\Pi(a_1, \ldots, a_k) = \prod_{i=1}^{k} P(a_i) - \{(\epsilon, \ldots, \epsilon), (a_1, \ldots, a_k)\}$, and note that $|\Pi(a_1, \ldots, a_k)| \leq 4^k$, for all $a_i \in V(T_i)$, for $i = 1, \ldots, k$.

We define a function $W : \prod_{i=1}^{k} V(T_i) \to \{\texttt{true}, \texttt{false}\}$ recursively, as follows:

- If $|S(a_1, \ldots, a_k)| \leq 1$, $W(a_1, \ldots, a_k) = \texttt{true}$.
- If $S(a_1, \ldots, a_k) \cap \mathcal{X}(T_i) \nsubseteq \mathcal{X}(T_i^{a_i})$ for some i, $W(a_1, \ldots, a_k) = \texttt{false}$.
- Otherwise,

$$W(\vec{a}) = \bigvee_{\vec{x} \in \Pi(\vec{a})} \Big(W(\vec{x}) \wedge W(\vec{c}(\vec{x})) \wedge Q(\vec{a}, \vec{x}) \wedge Q(\vec{a}, \vec{c}(\vec{x})) \Big)$$

where $\vec{x} = (x_1, \dots, x_k)$, $\vec{c}(\vec{x}) = (c_1(x_i), \dots, c_k(x_k))$ and

$$\begin{aligned} Q(\vec{a}, \vec{x}) \quad = \quad & \bigwedge_{i \neq j} \Big(x_i = a_i \wedge x_j = \epsilon \;\rightarrow\; L(a_i) \cap L(a_j^\ell) = \emptyset \;\vee\; L(a_i) \cap L(a_j^r) = \emptyset \Big) \\ \wedge \quad & \bigwedge_{i \neq j} \Big(x_i = a_i \wedge x_j = a_j^\ell; \rightarrow\; L(a_i) \cap L(a_j^r) = \emptyset \Big) \\ \wedge \quad & \bigwedge_{i \neq j} \Big(x_i = a_i \wedge x_j = a_j^r; \rightarrow\; L(a_i) \cap L(a_j^\ell) = \emptyset \Big). \end{aligned}$$

We may define a partial ordering $\lhd$ on $\prod_{i=1}^k (V(T_i) \cup \{\epsilon\})$ by taking

$$(a_1, \dots, a_k) \lhd (a_1', \dots, a_k') \quad \Longleftrightarrow \quad a_i = \epsilon \text{ or } a_i \prec_i a_i', \quad \text{for all } i = 1, \dots, k$$

where $\prec_i$ is the successor relation in T_i, with the unique $\lhd$-minimum element $(\epsilon, \dots, \epsilon)$. An example computation of the function W is described in Figure 12.

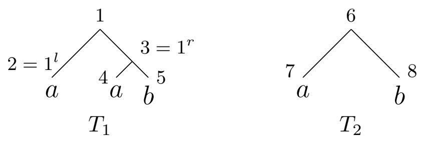


Figure 12 Given the trees T_1 and T_2, $W(1,6) = \texttt{true}$ since $W(2,7) \wedge W(3,8) \leftarrow W(2,7) \wedge W(4,\epsilon) \wedge W(5,8) = \texttt{true}$. Note that $W(1,6)$ would also compute $W(2,8) \wedge W(3,7)$, $W(2,6) \wedge W(3,\epsilon)$, $W(2,\epsilon) \wedge W(3,6)$, $W(1,7) \wedge W(\epsilon,8)$, and $W(1,8) \wedge W(\epsilon,7)$.

▶ **Lemma 19.** *Let $T_1, T_2, \dots, T_k$ be a collection of k binary MUL-trees such that $\bigcup_i \mathcal{X}(T_i) = \bigcup_i D(T_i)$. Then $W(a_1, \dots, a_k) = \texttt{true}$ if and only if there exists a single-labelled tree T leaf-labelled by $S(a_1, \dots, a_k)$ such that $T_i^{a_i}$ displays $T{\restriction}_{\mathcal{X}(T_i)}$, under a mapping that maps $r_i(T) \mapsto x_i$, for all $i = 1, \dots, k$.*

Proof. We prove this by induction on k. For the base case, suppose that each of $(a_1, \dots a_k) = (\epsilon, \dots, \epsilon)$ is the $\prec$-minimum, so that $T_i^{a_i} = T_i^\epsilon = T_\emptyset$ is the empty tree, for $i = 1, \dots, k$. In which case $S(\epsilon, \dots, \epsilon) = \emptyset$, and hence $W(a_1, \dots a_k) = \texttt{true}$ by definition, while (trivially) the empty tree $T = T_\emptyset$ is such that $T_i^{a_i} = T_\emptyset$ displays $T_\emptyset{\restriction}_{\mathcal{X}(T_i)} = T_\emptyset$. Next, suppose that the result holds for all $(u_1, \dots, u_k) \lhd (a_1, \dots, a_k)$ for some tuple $(a_1, \dots, a_k)$. We claim that the result holds too for $(a_1, \dots, a_k)$.

($\Leftarrow$) Suppose that T is as described. Then for each $i = 1, \dots, k$ there is some subtree $S_i \subseteq T^{a_i}$ and some label-preserving isomorphism $f_i : V(S_i) \to V(T{\restriction}_{\mathcal{X}(T_i)})$.
Let T^ℓ and T^r denote the left and right subtrees attached at $r(T)$. As T is a single-labelled tree, it follows that $L(T^\ell) \cap L(T^r) = \emptyset$, as each label occurs exactly once.
We can partition each $V(S_i)$ into three parts $L_i = \{v \in V(S_i) : f_i(v) \in T^\ell\}$ and $R_i = \{v \in V(S_i) : f_i(v) \in T^r\}$ and $C_i = \{v \in V(S_i) : f_i(v) = r(T)\}$. Note that, since f_i is an isomorphism, if $u \in X$ and $u <_i v$ then $v \in X$, for $X \in \{L_i, R_i\}$.
We have three cases depending on which of these three sets lies the root node a_i of the subtree $T_i^{a_i}$:

- If $a_i \in C_i$ then it follows that either $a_i^\ell \in L_i$ and $a_i^r \in R_i$ or $a_i^r \in L_i$ and $a_i^\ell \in R_i$.
  - If $a_i^\ell \in L_i$ and $a_i^r \in R_i$ then $T_i^{a_i^\ell}$ displays $T^\ell{\restriction}_{\mathcal{X}(a_i)}$ and $T_i^{a_i^r}$ displays $T^r{\restriction}_{\mathcal{X}(a_i)}$.
  - If $a_i^r \in L_i$ and $a_i^\ell \in R_i$ then $T_i^{a_i^r}$ displays $T^\ell{\restriction}_{\mathcal{X}(a_i)}$ and $T_i^{a_i^\ell}$ displays $T^r{\restriction}_{\mathcal{X}(a_i)}$.

In each case, there is some $x_i \in \{a_i^\ell, a_i^r\}$ such that $T_i^{x_i}$ displays $T^\ell\restriction_{\mathcal{X}(a_i)}$ and $T_i^{c_i(x_i)}$ displays $T^\ell\restriction_{\mathcal{X}(a_i)}$.

- If $a_i \in L_i$ then it follows that $L_i = V(S_i)$ and $C_i = R_i = \emptyset$. Hence we have that $T_i^{a_i}$ displays $T^\ell\restriction_{\mathcal{X}(a_i)}$, while $T_i^\epsilon = T_\emptyset$ (trivially) displays the empty tree $T^r\restriction_{\mathcal{X}(a_i)} = T_\emptyset$.
- Symmetrically, if $a_i \in R_i$ then $T_i^{a_i}$ displays $T^r\restriction_{\mathcal{X}(a_i)}$ and T_i^ϵ displays $T^\ell\restriction_{\mathcal{X}(a_i)}$.

In all cases there is some $x_i \in P(a_i)$ such that $T_i^{x_i}$ displays $T^\ell\restriction_{\mathcal{X}(a_i)}$ and $T_i^{c_i(x_i)}$ displays $T^\ell\restriction_{\mathcal{X}(a_i)}$, for all $i = 1, \dots, k$. Since both $(x_1, \dots, x_k), (c_1(x_1), \dots, c_k(x_k)) \lhd (a_1, \dots, a_k)$, it follows from the induction hypothesis that $W(\vec{x}) \wedge W(\vec{c}(\vec{x})) = \texttt{true}$.
For all $i \neq j$, if $x_i = a_i$ and $x_j = \epsilon$ then by definition $a_i \in L_i \subseteq L(T^\ell)$ while $a_j \in C_j$. If $a_j^\ell \in R_i$ then $L(a_i) \cap L(a_j^\ell) = \emptyset$, otherwise $a_j^r \in R_i$ and so $L(a_i) \cap L(a_j^r) = \emptyset$, since $R_i \subseteq L(T^r)$ and $L(T^\ell) \cap L(T^r) = \emptyset$. If $x_i = a_i$ and $x_j = a_j^l$ then $a_i \in L_i \subseteq L(T^l)$ while $a_j^\ell \in L_j$ and $a_j^r \in R_j \subseteq L(T^r)$. From which it follows that $L(a_i) \cap L(a_j^r) = \emptyset$.
Similarly, if $x_i = a_i$ and $x_j = a_j^r$ then it follows $L(a_i) \cap L(a_j^l) = \emptyset$. This is to say that $Q(\vec{a}, \vec{x}) = \texttt{true}$, and by the same argument, so too that $Q(\vec{a}, \vec{c}(\vec{x})) = \texttt{true}$.
Hence, by definition, $W(a_1, \dots, a_k) = \texttt{true}$, as required.

(⇒) Suppose that $W(a_1, \dots, a_k) = \texttt{true}$, then there are two possible cases: *(i)* $|S(a_i, \dots, a_k)| \leq 1$; *(ii)* there is some $(x_1, \dots, x_k) \in \Pi(a_1, \dots, a_k)$ such that $W(\vec{x}) \wedge W(\vec{c}(\vec{x})) \wedge Q(\vec{x}) = \texttt{true}$:

(i) If $S(a_1, \dots, a_k) = \emptyset$ then we may take $T_i^{a_i}$ to display $T = T_\emptyset$ as the empty tree for all $i = 1, \dots, k$. Otherwise, if $S(a_1, \dots, a_k) = \{x\}$ is a singleton then we may take T to be the tree with a single leaf-labelled by x, where it is straightforward to check that $T_i^{a_i}$ can display $T\restriction_{\mathcal{X}(a_i)}$ for all $i = 1, \dots, k$.

(ii) It follows from the induction hypothesis that there exist single-labelled trees T_x and T_y, leaf-labelled by $S(x_1, \dots x_k)$ and $S(y_1, \dots y_k)$, respectively, such that $T_i^{x_i}$ displays $T_x\restriction_{\mathcal{X}(T_i)}$ and $T_i^{y_i}$ under a mapping that maps $r_i(T_x) \mapsto x_i$, and displays $T_y y\restriction_{\mathcal{X}(T_i)}$ under a mapping that maps $r_i(T_y) \mapsto y_i$, for all $i = 1, \dots, k$, where $y_i = c(x_i)$. If $L(T_x) \cap L(T_y) = \emptyset$ then we construct a new tree T by connecting the roots $r(T_x)$ and $r(T_y)$ of T_x and T_y to a common (new) root node r.
Otherwise since $Q(\vec{a}, \vec{x}) \wedge Q(\vec{a}, \vec{c}(\vec{x})) = \texttt{true}$, it follows that either $L(T_x) \cap L(T_y^r) = \emptyset$ or $L(T_x) \cap L(T_y^\ell) = \emptyset$. In the first case we can construct a new tree T by merging T_x with the left sub-tree of T_y, while in the latter case we can construct T by merging T_x with the left sub-tree of T_y.
In all cases, we have that T is a single-labelled tree, as required, as it remains to show that $T_i^{a_i}$ displays $T\restriction_{\mathcal{X}(T_i)}$, for each $i = 1, \dots, k$:

- If $x_i = a_i^r$ then $T_i^{a_i}$ displays $T\restriction_{\mathcal{X}(T_i)}$, mapping $r(T^x) \mapsto a_i^r$, $r(T^y) \mapsto a_i^\ell$, and $r \mapsto a_i$.
- If $x_i = a_i^\ell$ then $T_i^{a_i}$ displays $T\restriction_{\mathcal{X}(T_i)}$, mapping $r(T^x) \mapsto a_i^\ell$, $r(T^y) \mapsto a_i^r$, and $r \mapsto a_i$.
- If $x_i = a_i$ then $T_i^{a_i}$ displays $T\restriction_{\mathcal{X}(T_i)}$ under the mapping that maps $r(T^x) \mapsto a_i$.
- If $x_i = \epsilon$ then $T_i^{a_i}$ displays $T\restriction_{\mathcal{X}(T_i)}$ under the mapping that maps $r(T^y) \mapsto a_i$.

Hence, it follows from induction that $W(a_1, \dots, a_k) = \texttt{true}$ if and only if there is some tree T leaf-labelled by $S(a_1, \dots, a_k)$ such that $T_i^{a_i}$ displays $T\restriction_{\mathcal{X}(T_i)}$ under a mapping that maps $r_i(T) \mapsto x_i$, for each $i = 1, \dots, k$, as required. ◀

Lemma 19 provides us with a criterion for deciding the MULSETPC problem for a given collection of binary MUL-trees, that can be computed in polynomial-time in the size of the trees, for any fixed number of trees, and scales exponentially with the number of trees.

▶ **Theorem 20.** *Let $T_1, T_2, \dots, T_k$ be a collection of k binary MUL-trees such that $\bigcup_i \mathcal{X}(T_i) = \bigcup_i D(T_i)$. Then* MULSETPC *for this instance can be solved in $O(k^2 \cdot 4^k \prod_{i=1}^k (|T_i| + 1)) = O(4^k \prod_{i=1}^k |T_i|)$ time.*

Proof. Based on Lemma 19, it is sufficient to compute $W(r(T_1), \dots, r(T_k))$, since $T_i^{r(T_i)} = T_i$, by definition. We can compute W via dynamic programming as outlined in Algorithm 1, which will return `true` if $T_1, \dots, T_k$ display a single labelled tree.

Algorithm 1 Recursive dynamic programming algorithm for $W(a_1, \dots, a_k)$.

```
1: let $S = S(a_1, \dots, a_k)$
2: if $|S| \leq 1$ then return true
3: else if $S \cap \mathcal{X}(T_i) \not\subseteq \mathcal{X}(T_i^{a_i})$ for some $i = 1, \dots, k$ then return false
4: else
5:     for $(x_1, \dots, x_k) \in \Pi(a_1, \dots, a_k)$ do
6:         if $W(\vec{x}) \wedge W(\vec{c}(\vec{x})) \wedge Q(\vec{a}, \vec{x}) \wedge Q(\vec{a}, \vec{c}(\vec{x}))$ then return true
7:     return false
```

For the time complexity, we can use memoization to store the values of W in a table with at most $O(\prod_{i=1}^k |V(T_i) \cup \{\epsilon\}|) = O(\prod_{i=1}^k (|T_i| + 1))$ entries. Furthermore, we require at most $O(k^2 \cdot 4^k)$ time to compute the value of each entry $W(a_1, \dots, a_k)$, since $|\Pi(a_1, \dots, a_k)| = |P(a_i)| \times \dots \times |P(a_k)| \leq 4^k$, while $Q(\vec{a}, \vec{x})$ and $Q(\vec{a}, \vec{c}(\vec{x}))$ can each be computed in quadratic time. Hence, the running time is $O(k^2 \cdot 4^k \prod_{i=1}^k (|T_i| + 1)) = O(4^k \prod_{i=1}^k |T_i|)$, as required. ◀

6 Conclusions

The above results resolve an open problem posed in [15, 16] as to whether the MULSETPC problem remains NP-complete when the number of MUL-trees is constant. According to Theorems 9 and 10, two MUL-trees are sufficient for NP-completeness, even with each label appearing at most twice in any tree and either the height or the degree constant. Theorems 11 and 12 extend this result and show that the more general MULSETPComp problem also remains NP-complete even when the number of MUL-trees is constant. Theorem 9 is tight in the sense that, if we restrict our attention to MUL-trees in which each label appears uniquely in at least one tree, we obtain a polynomial-upper bound for a fixed number of trees (Theorem 20). However, Theorem 12 suggests the algorithm presented in Theorem 20 for MULSETPC cannot be directly generalised to solve equivalent MULSETPComp instances in polynomial time, unless $P = NP$.

The above results also suggest two new open problems. Firstly, is it possible to improve Theorem 11 to show that MULSETPComp is still NP-complete when restricted to MUL-trees with multiplicity 2? Secondly, what can be said about the complexity of MULSETPC and MULSETPComp for instances in which the multiplicity is not restricted but the number of leaf labels that may appear more than once is restricted? That is, for each MUL-tree, k leaf labels may appear an unbounded number of times in the tree, whereas all other labels appear at most once. For which values of k are these subproblems NP-complete? This question is interesting because of its connection to the instances investigated in Section 5.

References

1 Alfred V. Aho, Yehoshua Sagiv, Thomas G. Szymanski, and Jeffrey D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.

2 Amihood Amir and Dmitry Keselman. Maximum Agreement Subtree in a Set of Evolutionary Trees: Metrics and Efficient Algorithms. *SIAM Journal on Computing*, 26(6):1656–1669, 1997.

3 Mukul S Bansal. Linear-time algorithms for some phylogenetic tree completion problems under Robinson-Foulds distance. In *RECOMB International conference on Comparative Genomics*, pages 209–226. Springer, 2018.

4 Mukul S Bansal, J Gordon Burleigh, Oliver Eulenstein, and David Fernández-Baca. Robinson-Foulds supertrees. *Algorithms for molecular biology*, 5(1):1–12, 2010.

5 Magnus Bordewich and Charles Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of combinatorics*, 8(4):409–423, 2005.

6 David Bryant. A classification of consensus methods for phylogenetics. In M. F. Janowitz, F.-J. Lapointe, F. R. McMorris, B. Mirkin, and F. S. Roberts, editors, *Bioconsensus*, volume 61 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 163–184. American Mathematical Society, 2003.

7 Richard Cole, Martin Farach-Colton, Ramesh Hariharan, Teresa Przytycka, and Mikkel Thorup. An $O(n \log n)$ Algorithm for the Maximum Agreement Subtree Problem for Binary Trees. *SIAM Journal on Computing*, 30(5):1385–1404, 2000.

8 Yun Cui, Jesper Jansson, and Wing-Kin Sung. Polynomial-time Algorithms for Building a Consensus MUL-Tree. *Journal of Computational Biology*, 19(9):1073–1088, 2012.

9 Yun Deng and David Fernández-Baca. Fast compatibility testing for rooted phylogenetic trees. *Algorithmica*, 80(8):2453–2477, 2018.

10 Zhihong Ding, Vladimir Filkov, and Dan Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping (PPH) problem. *Journal of Computational Biology*, 13(2):522–553, 2006.

11 Joseph Felsenstein. *Inferring Phylogenies.* Sinauer Associates, Inc., Sunderland, Massachusetts, 2004.

12 CR Finden and AD Gordon. Obtaining common pruned trees. *Journal of Classification*, 2(1):255–276, 1985.

13 Ganeshkumar Ganapathy, Barbara Goodson, Robert Jansen, Hai-son Le, Vijaya Ramachandran, and Tandy Warnow. Pattern identification in biogeography. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):334–346, 2006.

14 Michael R Garey and David S Johnson. *Computers and intractability.* A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness.

15 Mathieu Gascon, Riccardo Dondi, and Nadia El-Mabrouk. Complexity and Algorithms for MUL-Tree Pruning. In Paola Flocchini and Lucia Moura, editors, *Combinatorial Algorithms*, pages 324–339, Cham, 2021. Springer International Publishing.

16 Mathieu Gascon, Riccardo Dondi, and Nadia El-Mabrouk. MUL-tree pruning for consistency and optimal reconciliation - complexity and algorithms. *Theoret. Comput. Sci.*, 937:22–38, 2022.

17 Katharina T Huber and Vincent Moulton. Phylogenetic networks from multi-labelled trees. *Journal of Mathematical Biology*, 52(5):613–632, 2006.

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

19 Katharina T Huber, Bengt Oxelman, Martin Lott, and Vincent Moulton. Reconstructing the evolutionary history of polyploids from multilabeled trees. *Molecular Biology and Evolution*, 23(9):1784–1791, 2006.

20 Leo van Iersel, Steven Kelk, Nela Lekić, and Celine Scornavacca. A practical approximation algorithm for solving massive instances of hybridization number for binary and nonbinary trees. *BMC bioinformatics*, 15(1):1–12, 2014.

21 Jesper Jansson, Chuanqi Shen, and Wing-Kin Sung. Improved algorithms for constructing consensus trees. *Journal of the ACM*, 63(3), 2016. Article 28.

22 Manuel Lafond, Nadia El-Mabrouk, Katharina T Huber, and Vincent Moulton. The complexity of comparing multiply-labelled trees by extending phylogenetic-tree metrics. *Theoretical Computer Science*, 760:15–34, 2019.

23 Martin Lott, Andreas Spillner, Katharina T Huber, Anna Petri, Bengt Oxelman, and Vincent Moulton. Inferring polyploid phylogenies from multiply-labeled gene trees. *BMC Evolutionary Biology*, 9(1):1–11, 2009.

24 Nobuhiro Minaka. Cladograms and reticulated graphs: A proposal for graphic representation of cladistic structures. *Bulletin of the Biogeographical Society of Japan*, 45(1):1–10, 1990.

25 Gordon L Nelson and Norman I Platnick. *Systematics and Biogeography: Cladistics and Vicariance*. Columbia University Press, 1981.

26 Roderic D M Page. Parasites, phylogeny and cospeciation. *International Journal for Parasitology*, 23(4):499–506, 1993.

27 Roderic D M Page. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Systematic Biology*, 43(1):58–77, 1994.

28 Itsik Pe'er, Tal Pupko, Ron Shamir, and Roded Sharan. Incomplete directed perfect phylogeny. *SIAM J. Comput.*, 33(3):590–607, 2004.

29 David F Robinson and Leslie R Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1-2):131–147, 1981.

30 Celine Scornavacca, Vincent Berry, and Vincent Ranwez. Building species trees from larger parts of phylogenomic databases. *Information and Computation*, 209(3):590–605, 2011.

31 Mike Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *J. Classification*, 9(1):91–116, 1992.

32 Mike Steel and Tandy Warnow. Kaikoura tree theorems: Computing the maximum agreement subtree. *Information Processing Letters*, 48:77–82, 1993.

33 Christopher Whidden, Norbert Zeh, and Robert G Beiko. Supertrees Based on the Subtree Prune-and-Regraft Distance. *Systematic biology*, 63(4):566–581, 2014.

34 Yufeng Wu. A practical method for exact computation of subtree prune and regraft distance. *Bioinformatics*, 25(2):190–196, 2009.