Accepted Manuscript

The complexity of comparing multiply-labelled trees by extending phylogenetic-tree metrics

M. Lafond, N. El-Mabrouk, K.T. Huber, V. Moulton

PII: S0304-3975(18)30522-X
DOI: https://doi.org/10.1016/j.tcs.2018.08.006
Reference: TCS 11703

To appear in: *Theoretical Computer Science*

Received date: 9 March 2018
Revised date: 10 July 2018
Accepted date: 2 August 2018

Please cite this article in press as: M. Lafond et al., The complexity of comparing multiply-labelled trees by extending phylogenetic-tree metrics, *Theoret. Comput. Sci.* (2018), https://doi.org/10.1016/j.tcs.2018.08.006

THE COMPLEXITY OF COMPARING MULTIPLY-LABELLED TREES BY EXTENDING PHYLOGENETIC-TREE METRICS

M. LAFOND, N. EL-MABROUK, K. T. HUBER, AND V. MOULTON

Abstract. A multilabeled tree (or MUL-tree) is a rooted tree in which every leaf is labelled by an element from some set, but in which more than one leaf may be labelled by the same element of that set. In phylogenetics, such trees are used in biogeographical studies, to study the evolution of gene families, and also within approaches to construct phylogenetic networks. A multilabelled tree in which no leaf-labels are repeated is called a phylogenetic tree, and one in which every label is the same is also known as a tree-shape. In this paper, we consider the complexity of computing metrics on MUL-trees that are obtained by extending metrics on phylogenetic trees. In particular, by restricting our attention to tree shapes, we show that computing the metric extension on MUL-trees is NP-complete for two well-known metrics on phylogenetic trees, namely, the path-difference and Robinson Foulds distances. We also show that the extension of the Robinson Foulds distance is fixed parameter tractable with respect to the distance parameter. The path distance complexity result allows us to also answer an open problem concerning the complexity of solving the quadratic assignment problem for two matrices that are a Robinson similarity and a Robinson dissimilarity, which we show to be NP-complete. We conclude by considering the maximum agreement subtree (MAST) distance on phylogenetic trees to MUL-trees. Although its extension to MUL-trees can be computed in polynomial time, we show that computing its natural generalization to more than two MUL-trees is NP-complete, although fixed-parameter tractable in the maximum degree when the number of given trees is bounded.

Keywords: tree shape, multilabelled tree, phylogenetic tree, NP-hardness, fixed-parameter tractability

1. Introduction

In phylogenetics, leaf-labelled, rooted trees are used to represent the vertical evolution of a collection X of species, genes or other units of heredity [22]. In this context, a multilabeled tree (or MUL-tree) [17] is a rooted tree in which every leaf is labelled by an element in the set X, but in which more than one leaf may be labelled by the same element of X. In case no leaf-labels are repeated, a MUL-tree is called a *phylogenetic tree*. At the other extreme, where the leaves of the MUL-tree are all labelled with the same element of X the tree is also known as a *tree-shape* (or simply a rooted tree). MUL-trees appear in biogeographical studies [14] where they are also known as area cladograms, in the study of the evolution of gene families [16] where multiple labels represent paralogous genes in the same genome, and also within approaches to construct phylogenetic networks [18]. MUL-trees and related structures also appear in areas such as data-mining [8] and string-matching [11].

As methods for constructing MUL-trees often results in multiple solutions or require searching through collections of MUL-trees, it is important to develop systematic methods to compare MUL-trees [14, 19]. One way to do this is to extend metrics on phylogenetic tree-metrics to MUL-trees. Phylogenetic trees are well understood and have been studied for several years [23]. Given a metric d on the set of phylogenetic trees with leaf-set X, we can define a metric d^* on the set of MUL-trees with leaf-set being a multiset M of size m and underlying set X with $|X| \geq m$ as follows. Given two MUL-trees T_1 and T_2 with leaf-set M, we bijectively assign labels to the leaves of T_1 and T_2 from the set $\{1, \ldots, m\}$ in such a way that the two subsets of $\{1, \ldots, m\}$ that are assigned to the leaves of T_1 and T_2 that are labelled by the same element in X are equal. This results in two phylogenetic trees T_1^* and T_2^* each with leaf-set $\{1, \ldots, m\}$. We then define the extension $d^*(T_1, T_2)$ to be the minimum value of $d(T_1^*, T_2^*)$ taken over all possible assignments of this kind. In [19, p.1031] it is shown that d^* is a metric on the set of MUL-trees with leaf-set M.

As pointed out in [19, p.1037] the complexity of computing d^* is not known for some tree-metrics d which are commonly used in phylogenetics. In this paper, we will therefore consider the complexity of computing d^* for three well-known metrics on phylogenetic trees: the path-difference distance d_{path} [23], the Robinson-Foulds distance d_{RF} [21] and the maximum agreement subtree (MAST) distance d_{MAST} [15]. Note that we do not consider the so-called nearest-neighbour interchange (NNI) distance and related operation-based tree metrics since, in contrast to d_{path}, d_{RF} and d_{MAST}, these are NP-complete to compute even for phylogenetic trees (see [19] and the references therein).

Before proceeding, we note that computing the distance d_{RF}^* is not equivalent to computing the Robinson-Foulds distance between two MUL-trees as defined in [6]. This is because in that paper the distance is defined between two trees with fixed labellings, whereas to compute d_{RF}^* we minimise d_{RF} over all possible assignments of labellings of the two trees. It should also be noted that it is not possible to compute d_{RF}^* between two MUL-trees by taking a consensus of the two trees as defined in [12] (which can be done in polynomial time) for the same reason.

To determine the complexity of computing d_{path}^* and d_{RF}^*, we shall restrict attention to the case where M consists of a single element, i.e. we shall reduce the problem to tree shapes. In this special case, for two tree shapes T_1, T_2 with m leaves, the problem of computing d^* reduces to finding a bijective labelling of the leaf-sets of T_1 and T_2 by the set $\{1, \ldots, m\}$ which minimises the value of the metric d between the resulting phylogenetic trees. Note that other approaches have been proposed for defining metrics on tree shapes – see e.g. [9] and the references therein – although such metrics are not more generally applicable to MUL-trees.

After presenting some preliminaries in Section 2, we begin by considering the complexity of computing d_{path}^*. The path distance between two phylogenetic trees is essentially the sum of the length-differences of the paths connecting two specified leaves in the two trees taken over every possible pair of leaves. In Section 3 we show that computing d_{path}^* is NP-complete (Theorem 3.2). Our proof is based on a previous result that was used to show that the so-called Gromov-Hausdorff distance between metric trees is NP-hard [1]. Interestingly, in Section 4 we are then able to use the fact that computing d_{path}^* is NP-complete to solve an open problem presented in [20]. In that paper, it is stated that the complexity is unknown for the

problem of finding a permutation which solves the quadratic assignment problem for two matrices P and Q, when P and Q are a Robinson similarity and Robinson dissimilarity, respectively (see Section 4 for definitions of these terms). Here we show that this problem is NP-complete (Theorem 4.1).

We then turn to considering the complexity of computing d^*_{RF}. The Robinson-Foulds distance d_{RF} between two phylogenetic trees on X is essentially the size of the symmetric difference of the two sets of clusters induced on X by the two trees. In Section 5 we show that computing d^*_{RF} is NP-complete (Theorem 5.6), even for two binary tree shapes. However, we shall also show that there is a fixed-parameter tractable algorithm for computing d^*_{RF} (Theorem 5.9). We shall present the proof for NP-completeness for the non-binary case; as the argument is quite long and technical, we present the proof for the binary case in an appendix.

In Section 6 we consider the MAST distance d_{MAST} between two phylogenetic trees, which is given by the size of the leaf-set of a maximum agreement subtree of the two trees. Interestingly, by results in [14], a maximum agreement sub-MUL-tree for two MUL-trees can be computed in polynomial time (even for two MUL-trees with different size leaf-sets), from which it follows that d^*_{MAST} can be computed in polynomial time [19]. Motivated by this fact, we consider the related problem where the aim is to find a maximal agreement sub-MUL-tree for 3 or more MUL-trees. Note that this is closely related to the largest common subtree problem [2]. By reducing again to tree shapes, in Section 6 we show that this more general problem is NP-complete for three tree shapes when the degree of the input trees is unbounded (Theorem 6.1). However, we also show that the problem is fixed parameter tractable with respect to the maximum degree if the number of trees is constant (Theorem 6.3). We conclude by stating some open problems in the final section.

2. Preliminaries

All graphs considered in this paper are *simple*, that is, they do not contain multiple edges or loops. We denote the edge set of a graph G by $E(G)$ and its vertex set by $V(G)$. If G is a tree, we will call its vertices *nodes*. A map $\omega : E(G) \to \mathbb{R}_{\geq 0}$ is called an *edge-weighting* for G. An *edge-weighted* graph is a pair (G, w) where G is a graph and ω is an edge-weighting for G.

Let T be a *rooted* tree, that is, a tree with a distinguished node $r(T)$ called its *root*. We denote by $L(T)$ the set of leaves of T and by $I(T)$ its set of internal nodes. We also view an isolated node as a rooted tree. For $u \in V(T)$ we denote by $T(u)$ the sub-tree of T rooted at u. The *size* $sz(u)$ of a node $u \in V(T)$ is $|L(T(u))|$. The *size of* T, denoted $sz(T)$, is the size of $r(T)$. Note that $sz(T) = |L(T)|$. A tree is *binary* if its root has degree 2, and every other internal node has degree 3.

A node $u \in V(T)$ is a *descendant* of a node $v \in V(T)$ if v is on the path between u and $r(T)$, and in this case v is an *ancestor* of u. Note that any node u is both a descendant and an ancestor of itself. If v is an ancestor of u and $u \neq v$, then v is called a *proper ancestor* of u and u is a *proper descendant* of v. If u is a descendant of v and $uv \in E(T)$, then u is a *child* of v. Two nodes u and v are *incomparable* if none is an ancestor of the other. A *common ancestor* of a subset $L' \subseteq L(T)$ is a node $v \in V(T)$ that lies, for all $x \in L'$, on the path from $r(T)$ to x. The *last common ancestor* of L' is the unique common ancestor $v \in V(T)$ of L' such that no proper descendant of v is also a common ancestor of L'.

Suppose X is a finite non-empty set. A *rooted phylogenetic tree on* X is a pair $\mathcal{T} = (T, \phi)$ where T is a rooted tree for which every internal node has at least 2 children, and ϕ is a bijective mapping from $L(T)$ into X assigning each leaf a label[1]. We may call ϕ a *leaf assignment* of T. In case the knowledge of X is of no relevance to the discussion then we refer to a rooted phylogenetic tree on X simply as a phylogenetic tree. For T a rooted tree with $|L(T)| = |X|$ we denote by $\mathcal{F}(T) = \{(T, \phi) : \phi \text{ is a bijection from } L(T) \text{ to } X\}$ the set of all possible rooted phylogenetic trees on X given by bijectively labeling the leaves of T by the elements of X. To improve clarity of our arguments and distinguish rooted trees from phylogenetic trees, we refer to a rooted tree as a *tree shape* in order to emphasize that their leaves are unlabelled.

Suppose that $n \geq 1$ and that d is a metric on the set of rooted phylogenetic trees on $[n] = \{1, 2, \ldots, n\}$. If T_1 and T_2 are both tree shapes with n leaves, then we put

$$d^*(T_1, T_2) = \min_{\mathcal{T}_1 \in \mathcal{F}(T_1), \mathcal{T}_2 \in \mathcal{F}(T_2)} d(\mathcal{T}_1, \mathcal{T}_2).$$

As remarked in [19, p.1031] for the more general case of MUL-trees, this is a metric on the set of rooted tree shapes with n leaves.

3. Path Distance

Given a weighted, rooted tree T and $i, j \in V(T)$, we let $\ell_T(i, j)$ denote the length of the shortest (undirected) path in T between i and j. The path distance d_{path} between two weighted, rooted phylogenetic trees $\mathcal{T}_1 = (T_1, \phi_1)$ and $\mathcal{T}_2 = (T_2, \phi_2)$ on $[n]$ is equal to

$$d_{path}(\mathcal{T}_1, \mathcal{T}_2) = \sum_{i,j \in [n]} |\ell_{T_1}(i, j) - \ell_{T_2}(i, j)|.$$

Note that d_{path} is a metric on the set of weighted, rooted phylogenetic trees on $[n]$ (see e. g. [23]). We shall show that computing d^*_{path} is NP-complete by reduction to the following problem, using a similar technique to that presented in [1, Section 3]

UNRESTRICTED PARTITION: Given a multiset $X = \{a_1, \ldots, a_n\}$ of positive integers where $n = 3m$ and $m \geq 1$, is it possible to partition X into m multisets $\{X_1, \ldots, X_m\}$ such that all the elements in each multiset X_i sum to $(\sum_{i=1}^n a_i)/m$?

Note that this problem has been proven to be strongly NP-complete, meaning that the hardness holds even if the size of the integers is polynomial in the input, by a reduction from 3-PARTITION [1]. In particular, we can assume that the size of the integers is polynomial in the input.

Given an integer $p > 0$, we let T_p denote the *rooted star shape* with $p \geq 1$ leaves that is, the rooted tree shape with p leaves, and such that every edge in T_p contains the root and a leaf (in particular, T_p has p edges).

Now, given an instance $X = \{a_1, \ldots, a_n\}$ of UNRESTRICTED PARTITION where $n = 3m$ and $m \geq 1$, we let T and T' be the two weighted, rooted tree shapes given in Figure 1, where for $S = (\sum_{i=1}^n a_i)/m$ and all $1 \leq i \leq m$, T^i_S denotes a copy of the rooted star shape T_S. Note that in the special case where $m = 1$, we replace T' with the star-tree T^1_S in which every edge has weight $10\frac{1}{2}$. Moreover, in case

[1]Note that this definition of a phylogenetic tree on X is different from the one given in e. g. [22] in that the roles of the domain and co-domain of ϕ are swapped.

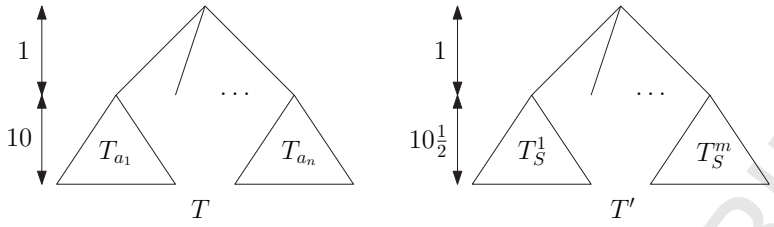

FIGURE 1. The weighted rooted tree shapes T and T'. For T, every edge of the form $r(T_{a_i})l$ where $l \in L(T_{a_i})$ and $1 \leq i \leq n$ has weight 10 and all remaining edges in that tree have weight one. Similarly, every edge of T' the form $r(T_S^i)l$ with $l \in L(T_S^i)$ where $1 \leq i \leq m$ has weight 10.5 and all remaining edges of that tree have weight one.

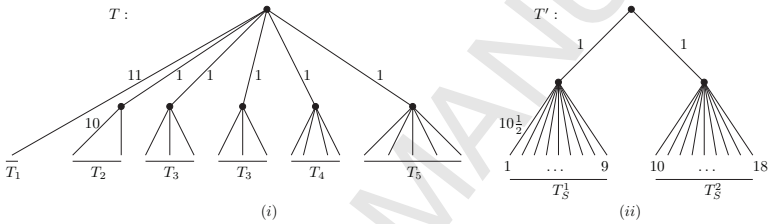

FIGURE 2. The weighted rooted tree shapes T and T' for $X = \{1, 2, 3, 3, 4, 5\}$.

$m > 1$, in the special case where T_{a_i} has a single leaf we suppress the corresponding node of degree 2 in T, and give the resulting edge weight equal to the sum of the weights of the two edges that contained that node.

To illustrate these definitions consider the multiset $X = \{1, 2, 3, 3, 4, 5\}$. Then $n = 6$, $m = 2$, and $S = (\sum_{i=1}^{n} a_i)/m = 9$. For $i = 1, 2$ we therefore have that T_S^i is the rooted star shape with nine leaves. The trees T and T' are depicted in Figure 2.

Continuing with this notation we obtain:

Lemma 3.1. *Suppose we are given an instance $\{a_1, \ldots, a_n\}$ of UNRESTRICTED PARTITION. Then*

$$d^*_{path}(T, T') \geq \binom{\sum_{i=1}^{n} a_i}{2}$$

with equality holding if and only if the given instance is a "yes" instance.

Proof. Suppose that ψ is a bijection from $L(T)$ to $L(T')$. Let $A = \binom{L(T)}{2}$ and put

$$\begin{aligned} B &= \{\{l, l'\} \in A : \{l, l'\} \subseteq L(T_{a_i}), \text{ some } 1 \leq i \leq n\}, \\ B_1 &= \{\{l, l'\} \in B : \{\psi(l), \psi(l')\} \subseteq L(T_S^j), \text{ some } 1 \leq j \leq m\}, \text{and} \\ B_2 &= \{\{l, l'\} \in B : \{\psi(l), \psi(l')\} \not\subseteq L(T_S^j), \text{ for any } 1 \leq j \leq m\}. \end{aligned}$$

Furthermore, put $C = A - B$ and

$$\begin{aligned} C_1 &= \{\{l, l'\} \in C : \{\psi(l), \psi(l')\} \subseteq L(T_S^j), \text{ some } 1 \leq j \leq m\}, \text{and} \\ C_2 &= \{\{l, l'\} \in C : \{\psi(l), \psi(l')\} \not\subseteq L(T_S^j), \text{ for any } 1 \leq j \leq m\}. \end{aligned}$$

To illustrate these definition consider again the previous example. Then A is the set of all leaf pairs of T, and B is the set of all leaf pairs that are either contained in T_S^1 or in T_S^2. Its subset B_2 comprises all leaf pairs in B which get mapped to different rooted star shapes in T' under ψ and B_1 comprises of the remaining leaf pairs of B. The set C comprises all leaf pairs of T where the two leaves that make up a pair are from different rooted star shapes that make up T. Its subset C_2 comprises all leaf pairs of C that are mapped to different rooted star shapes of T' and C_1 comprises all remaining leaf pairs of C.

Note that $|A| = \binom{\sum_{i=1}^n a_i}{2}$, B is the disjoint union of B_1 and B_2, and that C is the disjoint union of C_1 and C_2. Moreover, setting $f(l,l') = |\ell_T(l,l') - \ell_{T'}(\psi(l),\psi(l'))|$ for $\{l,l'\} \in A$, we have

$$\begin{aligned}\sum_{\{l,l'\}\in A} f(l,l') &= \sum_{\{l,l'\}\in B_1} f(l,l') + \sum_{\{l,l'\}\in B_2} f(l,l') + \sum_{\{l,l'\}\in C_1} f(l,l') + \sum_{\{l,l'\}\in C_2} f(l,l') \\ &= |B_1| + 3|B_2| + |C_1| + |C_2| \\ &= (|B_1| + |B_2| + |C_1| + |C_2|) + 2|B_2| \\ &= |A| + 2|B_2|.\end{aligned}$$

It follows that $B_2 = \emptyset$ if and only if ψ corresponds to a "yes" instance of UNRESTRICTED PARTITION.

Now note that any pair of bijective labellings of the leaf-sets of the tree shapes T and T' by the set $[mS]$ (giving rooted phylogenetic trees $\mathcal{T}$ and $\mathcal{T}'$ on $[mS]$, respectively) gives rise to a bijection between $L(T)$ and $L(T')$ in view of the definition of S. Moreover, all bijections between $L(T)$ and $L(T')$ can arise in this way, and if such a bijection is the map ψ given above, then the path-distance between $\mathcal{T}$ and $\mathcal{T}'$ is $|A| + 2|B_2|$.

Since $d^*_{path}(T,T')$ is given by taking the minimum over all pairs of bijective labellings of the leaf-sets of the tree shapes T and T' by $[mS]$, the lemma now follows immediately.

□

Using the Lemma 3.1, we now prove:

Theorem 3.2. *For two tree shapes T_1 and T_2 of the same size, computing $d^*_{path}(T_1,T_2)$ is NP-complete, even in the case that both of the tree shapes are binary.*

Proof. The problem of computing d^*_{path} is easily seen to be in NP, as a leaf assignment of T_1 and T_2 can serve as a certificate from which the distance can be computed in polynomial time. The NP-hardness of the problem follows immediately from Lemma 3.1, and therefore, computing d^*_{path} is NP-complete.

We now sketch the proof of the last statement. Given an instance $\{a_1,\ldots,a_n\}$ with $n = 3m$ and $m \geq 1$ of UNRESTRICTED PARTITION, we first turn the tree shapes T and T' used in Lemma 3.1 into binary tree shapes by resolving the nodes of outdegree 3 or more in both tree shapes through inserting a caterpillar (tree) shape of height ϵ, as indicated in Figure 3 (a caterpillar shape is a binary tree in which each internal node has exactly one child that is a leaf, except for one single internal node which has two leaf children). Note that in case either T or T' has an edge which contains both the root and a leaf, then we add ϵ to the weight of that edge before we insert the caterpillars, and that if either T or T' contains a node of outdegree 2 then we add ϵ to the weight of each of the edges below that node

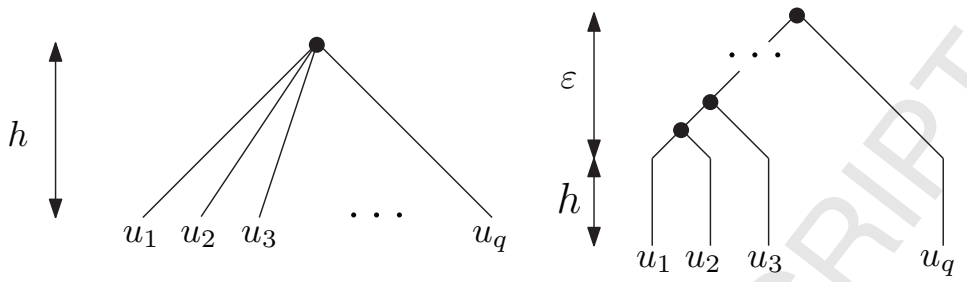

FIGURE 3. The replacement of an unresolved node of height h by a caterpillar tree shape of height $h + \epsilon$ employed in the proof of Theorem 3.2.

before we insert the caterpillars. Since at most two unresolved nodes can lie on a directed path from the root of T to a leaf of T or from the root of T' to a leaf of T' it follows that we obtain two weighted binary tree shapes T and T' of height $10 + 2\epsilon$ and $10.5 + 2\epsilon$, respectively, for some small $\epsilon > 0$.

Now, using the same type of argument as in Lemma 3.1, it can be seen that for a bijection ψ between $L(T)$ and $L(T')$, we have

$$\sum_{\{l,l'\}\in A} |\ell_T(l,l') - \ell_{T'}(\psi(l),\psi(l'))| = |A| + 2|B_2| + \epsilon g(a_1,\ldots,a_n)$$

where A and B_2 are defined just as in the proof of Lemma 3.1, and g is some function of $a_1,\ldots,a_n$. Hence, by considering all bijections from $L(T)$ to $L(T')$, we can take ϵ sufficiently small (for example, so that $\epsilon g(a_1,\ldots,a_n) < \frac{1}{100}$) so that the instance $\{a_1,\ldots,a_n\}$ corresponds to a "yes" instance of UNRESTRICTED PARTITION if and only if

$$|d^*_{path}(T,T') - \binom{\sum_{i=1}^n a_i}{2}| < 1.$$

The last statement of the theorem now follows immediately.

□

4. The Quadratic assignment problem for Robinson matrices

Given two $N \times N$ symmetric matrices P, Q with $N \geq 1$, the QUADRATIC ASSIGNMENT problem is to find a permutation ψ of $[N]$ which minimizes

$$\sum_{i,j=1}^{N} P_{ij} Q_{\psi(i)\psi(j)}. \tag{1}$$

Here for a matrix M, M_{ij} denotes the entry at row i and column j. For $N \geq 1$ a symmetric $N \times N$ matrix P is called a *Robinson similarity matrix* if its entries decrease monotonically in the rows and the columns when moving away from the main diagonal, i.e., if

$$P_{ik} \leq min\{P_{ij}, P_{jk}\} \text{ for all } 1 \leq i \leq j \leq k \leq N.$$

Similarly, an $N \times N$ symmetric matrix Q is called a *Robinson dissimilarity matrix* if its entries increase monotonically in the rows and the columns when moving away from the main diagonal.

In [20], it is stated that the complexity of finding a permutation which solves the QUADRATIC ASSIGNMENT problem for two symmetric $N \times N$ matrices P and Q when P is a Robinson similarity and Q is a Robinson dissimilarity, is not known. Here we shall show that this problem is NP-complete.

First, suppose we are given an instance $\{a_1, \ldots, a_n\}$ of UNRESTRICTED PARTITION where $n = 3m$ and $m \geq 1$. For T and T' the weighted rooted tree shapes depicted in Figure 1, define $N \times N$ matrices $P(T)$ and $Q(T')$ as follows. For $P(T)$, label the leaves of T by 1 up to $N = \sum_{i=1}^{n} a_i$ from left to right and, for all $i, j \in [N]$, set $P(T)_{ij} = -l_T(i, j)$. For $Q(T')$, also label the leaves of T' by 1 up to N from left to right and, for all $i, j \in [N]$, set $Q(T')_{ij} = l_{T'}(i, j)$. It is straight-forward to see that $-P(T)$ is a Robinson similarity matrix and that $Q(T')$ is a Robinson dissimilarity matrix. We also remark that it is easy to see that a permutation ψ of $[N]$ minimizes the quantity in Expression (1) for $P = P(T)$ and $Q = Q(T')$ if and only if ψ minimizes

$$\sum_{i,j \in [N]} (\ell_T(i, j) - \ell_{T'}(\psi(i), \psi(j)))^2.$$

Continuing with this notation, we obtain

Theorem 4.1. *The QUADRATIC ASSIGNMENT problem for P and Q is NP-complete for P the Robinson similarity $P(T)$ and Q the Robinson dissimilarity $Q(T')$.*

Proof. We claim that given an instance $\{a_1, \ldots, a_n\}$ of UNRESTRICTED PARTITION where $n = 3m$ and $m \geq 1$ and any permutation ψ of $[N]$ where $N = \sum_{i=1}^{n} a_i$, we have

$$\sum_{i,j \in [N]} (\ell_T(i, j) - \ell_{T'}(\psi(i), \psi(j)))^2 \geq \binom{N}{2},$$

with equality holding if and only if ψ corresponds to a "yes" instance. The proof of the theorem then follows immediately from the remark preceding Theorem 4.1.

The proof of the claim is very similar to that of Lemma 3.1 and so we only give a sketch proof. Suppose ψ is a permutation of $[N]$. Then ψ is a bijection between $L(T) = [N]$ and $L(T') = [N]$. Define the sets $A = \binom{L(T)}{2}$, B_1, B_2, C_1, and C_2 in an analogous way to the sets defined in the proof of Lemma 3.1. Setting $f(l, l') = (l_T(i, j) - l_{T'}(\psi(i), \psi(j))^2$ for $\{l, l'\} \in A$, it follows that

$$\sum_{\{l,l'\} \in A} f(l, l') = |B_1| + 9|B_2| + |C_1| + |C_2| = |A| + 8|B_2|.$$

The proof of the claim now easily follows using a similar argument to that used in the last part of the proof of Lemma 3.1. □

5. Robinson Foulds Distance

Suppose $\mathcal{T} = (T, \phi)$ is a rooted phylogenetic tree on X. For a node u in $\mathcal{T}$, we denote by $C_{\mathcal{T}}(u) = \{\phi(l) : l \in L(T(u))\}$ the *cluster* of $\mathcal{T}$ associated with u, that is, the set of labels assigned by ϕ to the leaves of $T(u)$. In case $\mathcal{T}$ is clear from the context, we may also write $C(u)$, for short. The set of clusters of $\mathcal{T}$ is $\mathcal{C}(\mathcal{T}) = \{C(u) : u \in V(T)\}$. For $u \in V(\mathcal{T})$ we call the cluster $C(u)$ *trivial* if u is either the root or a leaf of T, and *non-trivial* otherwise. The *Robinson Foulds (RF)*

distance $d_{RF}(\mathcal{T}_1, \mathcal{T}_2)$ between two rooted phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$ on X is the cardinality of the symmetric difference of $\mathcal{C}(\mathcal{T}_1)$ and $\mathcal{C}(\mathcal{T}_2)$, i.e.

$$d_{RF}(\mathcal{T}_1, \mathcal{T}_2) = |\mathcal{C}(\mathcal{T}_1) \setminus \mathcal{C}(\mathcal{T}_2)| + |\mathcal{C}(\mathcal{T}_2) \setminus \mathcal{C}(\mathcal{T}_1)|$$

Note that trivial clusters never contribute towards the RF distance between two rooted phylogenetic trees. Also observe that if both $\mathcal{T}_1$ and $\mathcal{T}_2$ are binary, then $d_{RF}(\mathcal{T}_1, \mathcal{T}_2) = 2|\mathcal{C}(\mathcal{T}_1) \setminus \mathcal{C}(\mathcal{T}_2)|$. Given two tree shapes T_1 and T_2, the d^*_{RF} distance asks for a leaf assignment of T_1 and T_2 that minimizes the d_{RF} distance. We will show that computing $d^*_{RF}(T_1, T_2)$ is NP-complete, but it is fixed-parameter tractable with respect to that distance. Beforehand, we establish a useful connection between d^*_{RF} and *cluster matchings*.

5.1. **Cluster matchings and the Robinson Foulds distance.** A *cluster matching* $\mathcal{M}$ between two tree shapes T_1 and T_2 is a set of pairs $(v_1, v_2) \in V(T_1) \times V(T_2)$, such that each node of $V(T_1) \cup V(T_2)$ appears in at most one pair of $\mathcal{M}$. We say that a node $v \in V(T_1) \cup V(T_2)$ is *matched in* $\mathcal{M}$ if there exists a pair in $\mathcal{M}$ that contains v, and *unmatched in* $\mathcal{M}$ otherwise. We may simply say that v is *matched* (or *unmatched*) if $\mathcal{M}$ is clear from the context. Also, we say that $\mathcal{M}$ is a *consistent* cluster matching between T_1 and T_2 if the two following properties hold:

(M1) $(v_1, v_2) \in \mathcal{M}$ implies $sz(v_1) = sz(v_2)$;
(M2) for any two pairs $(v_1, v_2), (v'_1, v'_2) \in \mathcal{M}$, v_1 and v'_1 are incomparable if and only if v_2 and v'_2 are incomparable.

Note that, if $\mathcal{M}$ is a consistent cluster matching and $(v_1, v_2), (v'_1, v'_2) \in \mathcal{M}$ then v_2 is an ancestor of v'_2 if and only if v_1 is an ancestor of v'_1.

We say that a consistent cluster matching $\mathcal{M}$ between two tree shapes T_1 and T_2 is *maximum* if for any consistent cluster matching $\mathcal{M}'$ between T_1 and T_2, we have $|\mathcal{M}| \geq |\mathcal{M}'|$. Also, we denote by $\mu(T_1, T_2)$ the cardinality of a maximum consistent cluster matching between T_1 and T_2.

To illustrate these definitions assume that T_1 and T_2 are two rooted tree shapes with leaf sets $\{v_1, v'_1\}$ and $\{v_2, v'_2\}$, respectively. Then the set $\mathcal{M} = \{(v_1, v_2), (v'_1, v'_2)\}$ is a consistent cluster matching for T_1 and T_2. However $\mathcal{M}$ is not maximum since the cluster matching $\mathcal{M}' = \mathcal{M} \cup \{(r(T_1), r(T_2))\}$ is also a consistent cluster matching for T_1 and T_2. Clearly $\mu(T_1, T_2) = 3$.

We next establish that in a maximum consistent cluster matching for two tree shapes T_1 and T_2 of the same size we can always assume that every leaf of T_1 is matched with a leaf of T_2.

Lemma 5.1. *Let T_1 and T_2 be two tree shapes of the same size, and let $\mathcal{M}$ be a consistent cluster matching between T_1 and T_2. Suppose that at least one leaf of T_1 is unmatched in $\mathcal{M}$. Then there exist $l_1 \in L(T_1)$ and $l_2 \in L(T_2)$ such that both l_1 and l_2 are unmatched in $\mathcal{M}$, and $\mathcal{M} \cup \{(l_1, l_2)\}$ is a consistent cluster matching.*

Proof. We first consider the case that $(r(T_1), r(T_2)) \in \mathcal{M}$. Then we can choose some $v \in V(T_1) \cup V(T_2)$ such that v is matched in $\mathcal{M}$, v has a descendant leaf l that is unmatched in $\mathcal{M}$ and $sz(v)$ is as small as possible (the assumption that $(r(T_1), r(T_2)) \in \mathcal{M}$ guarantees that such a node v exists). Suppose without loss of generality that $v \in V(T_1)$, and let $v' \in V(T_2)$ be such that $(v, v') \in \mathcal{M}$. Because of the remark following the definition of a consistent cluster matching, it follows that v' must have an unmatched descendant leaf l'. By the choice of v, no node on the $l - v$ path is matched in $\mathcal{M}$ except v, and no node on the $l' - v'$ path is matched

in $\mathcal{M}$ except v'. Since $\mathcal{M}$ is a consistent cluster matching and Property (M1) is satisfied for the pair $p = (l, l')$ and Property (M2) is satisfied for p and any pair of $\mathcal{M}$ it follows that $\mathcal{M} \cup \{(l, l')\}$ is a consistent cluster matching for T_1 and T_2.

So assume that $(r(T_1), r(T_2)) \notin \mathcal{M}$. Then it is straight-forward to see that $\mathcal{M}' = \mathcal{M} \cup \{(r(T_1), r(T_2))\}$ is a consistent cluster matching for T_1 and T_2. In view of the previous case, there exist some leaf l_1 in T_1 and some leaf l_2 in T_2 such that $\mathcal{M}' \cup \{(l, l')\}$ is a consistent cluster matching for T_1 and T_2. But then $\mathcal{M}' \cup \{(l, l')\} - \{(r(T_1), r(T_2))\} = \mathcal{M} \cup \{(l, l')\}$ is also a consistent cluster matching for T_1 and T_2.

□

The following result allows us to reformulate the problem of computing the d^*_{RF} distance between two tree shapes of the same size in terms of maximum consistent cluster matchings.

Lemma 5.2. *Suppose T_1 and T_2 are two tree shapes of the same size. Then $d^*_{RF}(T_1, T_2) = |V(T_1)| + |V(T_2)| - 2\mu(T_1, T_2)$.*

Furthermore, given a consistent cluster matching $\mathcal{M}$ of cardinality $\mu(T_1, T_2)$, let ϕ_1 and ϕ_2 be leaf assignments of T_1 and T_2, respectively, such that for all leaves $l_1 \in L(T_1), l_2 \in L(T_2)$, the property that $(l_1, l_2) \in \mathcal{M}$ implies that $\phi_1(l_1) = \phi_2(l_2)$. Then $d_{RF}((T_1, \phi_1), (T_2, \phi_2)) = d^*_{RF}(T_1, T_2)$.

Proof. We first show that $d^*_{RF}(T_1, T_2) \geq |V(T_1)| + |V(T_2)| - 2\mu(T_1, T_2)$. Let $\phi_1 : L(T_1) \to X$ and $\phi_2 : L(T_2) \to X$ denote two leaf assignments of T_1 and T_2, respectively, such that $d_{RF}(\mathcal{T}_1, \mathcal{T}_2) = d^*_{RF}(T_1, T_2)$ where $\mathcal{T}_1 = (T_1, \phi_1)$ and $\mathcal{T}_2 = (T_2, \phi_2)$. Let $\mathcal{M} = \{(v_1, v_2) \in V(T_1) \times V(T_2) : C_{\mathcal{T}_1}(v_1) = C_{\mathcal{T}_2}(v_2)\}$. Clearly, $\mathcal{M}$ is a consistent cluster matching. Moreover, $d_{RF}(\mathcal{T}_1, \mathcal{T}_2)$ is the number of nodes that are unmatched in $\mathcal{M}$ in each tree shape, and so $d^*_{RF}(T_1, T_2) = d_{RF}(\mathcal{T}_1, \mathcal{T}_2) = |V(T_1)| + |V(T_2)| - 2|\mathcal{M}| \geq |V(T_1)| + |V(T_2)| - 2\mu(T_1, T_2)$.

Conversely, let $\mathcal{M}$ be a consistent cluster matching of cardinality $\mu(T_1, T_2)$. By Lemma 5.1, every leaf $l \in L(T_1) \cup L(T_2)$ is matched in $\mathcal{M}$. Let ϕ_1 and ϕ_2 be leaf assignments of T_1 and T_2, respectively, such that for every $l_1 \in L(T_1)$ and $l_2 \in L(T_2)$, $\phi_1(l_1) = \phi_2(l_2)$ if and only if $(l_1, l_2) \in \mathcal{M}$. Now, let $(v_1, v_2) \in \mathcal{M}$. By Property (M2), every $l \in L(T_1(v_1))$ is matched with a leaf $l' \in L(T_2(v_2))$. This implies that $C_{(T_1,\phi_1)}(v_1) = C_{(T_2,\phi_2)}(v_2)$. That is, if a node u of T_1 or T_2 is matched in $\mathcal{M}$, then the u cluster does not contribute to the d^*_{RF} distance between T_1 and T_2 under ϕ_1 and ϕ_2. Thus, $d^*_{RF}(T_1, T_2) \leq d_{RF}((T_1, \phi_1), (T_2, \phi_2)) \leq |\{u \in V(T_1) \cup V(T_2) : u \text{ is not matched in } \mathcal{M}\}|$. The number of unmatched nodes is $V(T_1) + V(T_2) - 2|\mathcal{M}|$, proving the first claim of the Lemma.

The second statement of the lemma follows immediately, as we have just shown that $d_{RF}((T_1, \phi_1), (T_2, \phi_2)) \leq V(T_1) + V(T_2) - \mu(T_1, T_2)$, and that $V(T_1) + V(T_2) - \mu(T_1, T_2) = d^*_{RF}(T_1, T_2)$.

□

Note that in view of the arguments used in the proof of Lemma 5.2, the problem of minimizing the d^*_{RF} distance between two tree shapes is equivalent to finding a maximum consistent cluster matching between them. As we shall see it will sometimes be more convenient to formulate the d^*_{RF} minimization problem in this latter form.

We also observe that if both tree shapes T_1 and T_2 are binary then $|V(T_1)| = |V(T_2)| = 2sz(T_1) - 1$ (see e.g. [22]). Combined with Lemma 5.2, we obtain the following result

Corollary 5.3. *Suppose T_1 and T_2 are two binary tree shapes of the same size. Then $d^*_{RF}(T_1, T_2) = 4sz(T_1) - 2 - 2\mu(T_1, T_2))$.*

Furthermore, as d^*_{RF} is a metric and therefore satisfies the triangle inequality we also obtain the following result via substitution.

Corollary 5.4. *If T_1, T_2, T_3 are three tree shapes of the same size, then $|V(T_2)| + \mu(T_1, T_3) \geq \mu(T_1, T_2) + \mu(T_2, T_3)$.*

5.2. **NP-completeness for the non-binary case of Robinson Foulds.** Using a reduction from the DOMINATING SET problem, we establish in this section that computing $d^*_{RF}(T_1, T_2)$ is NP-complete, even in trees of height at most 3. In the DOMINATING SET problem, we are given a connected graph $G = (V, E)$ and an integer $k \geq 1$, and ask if G has a dominating set of size at most k, where a *dominating set* is a subset D of nodes of a graph H such that for every node $v \in V(H)$ we have that either $v \in D$, or $v \notin D$ and there exists $u \in D$ such that u and v are adjacent.

Let (G, k) be an instance of DOMINATING SET, with $n = |V(G)| \geq 3$. We next outline the construction of two tree shapes T_1 and T_2 from G the details of which we give below (see Figure 4 for an illustration of the case $n = 3$). Assume that $(v_1, \ldots, v_n)$ is an (arbitrary) ordering on $V(G)$. Then tree shape T_1 is built from two types of tree shapes. For $i \in [n]$ these are tree shapes rooted at a node w_i which corresponds to v_i, and tree shapes rooted at a node $d_{i,j}$ which corresponds to the edge v_iv_j of G. As we shall see, $sz(w_i) = n^2 + i$ holds for each $i \in [n]$ and $sz(d_{i,j} = j + 1$ for every $v_iv_j \in E(G)$.

To start with, the root of the tree shape T_1 has n children $w_1, \ldots, w_n$. Then apply the following procedure for each $i \in [n]$. Let $v_{j_1}, \ldots, v_{j_l}$ be the neighbors of v_i in G (noting that $1 \leq l \leq n - 1$). Then we add to w_i exactly $l + 1$ children $d_{i,j_1}, d_{i,j_2}, \ldots, d_{i,j_l}$ and $d_{i,i}$, each respectively of size $j_1 + 1, j_2 + 1, \ldots, j_l + 1$ and $i + 1$. These sizes are achieved by inserting $j_p + 1$ leaves as children of d_{i,j_p} for each $p \in [l]$, and $i + 1$ leaves as children of $d_{i,i}$. Observe that so far, w_i has size at most $\sum_{p=1}^{l}(j_p + 1) + i + 1 \leq (n - 1)n + i + 1 < n^2 + i$. We add leaf children to w_i until w_i has size $n^2 + i$ (as in Figure 4).

As for T_2, the root $r(T_2)$ has $2n$ children $d'_1, \ldots, d'_n, w'_1, \ldots, w'_n$. For each $i \in [n]$, we insert $i + 1$ leaves as children of d'_i and $n^2 + i$ leaves as children of w'_i. Hence $sz(d'_i) = i + 1$ and $sz(w'_i) = n^2 + i$ for each $i \in [n]$. To finish the construction of T_1 and T_2, we add leaf children to the smallest of $r(T_1)$ or $r(T_2)$ until $sz(T_1) = sz(T_2)$ (Figure 4 shows leaf insertions under $r(T_1)$).

Notice that in a consistent cluster matching for T_1 and T_2, the d'_j nodes of T_2 can only be matched with the nodes of T_1 of the form $d_{i,j}$, and the w'_i nodes of T_2 with the nodes of T_1 of the form w_i. Moreover, since all the w'_i and d'_j nodes are incomparable, all the nodes of T_1 that are matched with the w'_i and d'_j nodes must also be incomparable.

Theorem 5.5. *For two tree shapes T_1 and T_2 of the same size, finding $d^*_{RF}(T_1, T_2)$ is NP-complete even if both trees have height at most 3.*

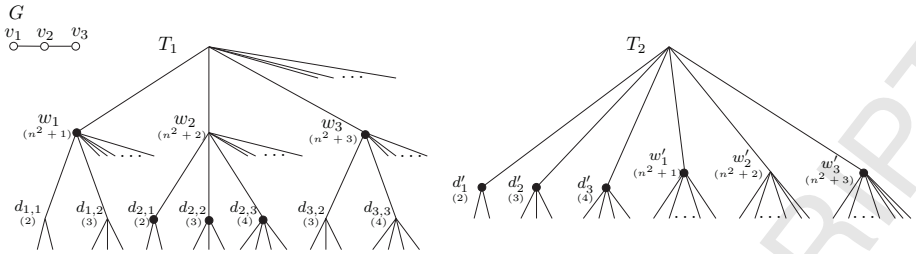

FIGURE 4. An example of the construction of T_1 and T_2 from G in the case that $n = 3$. The ordering of the vertices is (v_1, v_2, v_3). The size of a node is shown in parentheses so, for example, $sz(w_1) = (n^2 + 1)$. A maximum cluster matching which corresponds to the dominating set is $D = \{v_2\}$ depicted by black nodes (the vertices w_2 and w'_2 are unmatched). See the proof of Theorem 5.5 for details.

Proof. The problem is clearly in NP, since every bijective labelling of the leaf sets of T_1 and T_2 in terms of a set of size $sz(T_1)$ can serve as a certificate of the distance and can be verified easily. For hardness, we show that, given a DOMINATING SET instance (G, k), for the two tree shapes T_1 and T_2 constructed above, G has a dominating set of size at most k if and only if there exists a consistent cluster matching of size at least $sz(T_1) + 1 + 2n - k$ where n is again $|V(G)|$.

Using the same notation as in that construction, including our arbitrary ordering $(v_1, \ldots, v_n)$ on $V(G)$, assume first that $D = \{v_{i_1}, \ldots, v_{i_k}\}$ is a dominating set of G of size k.

We construct a consistent cluster matching $\mathcal{M}$ between T_1 and T_2 by first matching every d'_j node of T_2 to a node in T_1 as follows. Start with $\mathcal{M} = \emptyset$. For each $v_j \in V(G)$, let $v_i \in D$ be a vertex dominating v_j (with $i = j$ if $v_j \in D$). Then we add the pair $(d_{i,j}, d'_j)$ to $\mathcal{M}$ (note that $d_{i,j}$ must exist, as either $i = j$ or $v_i v_j \in E(G)$). Since the empty set is vacuously a consistent cluster matching between T_1 and T_2, it follows that the resulting set $\mathcal{M}$ is also a consistent cluster matching between T_1 and T_2 since all the d_j nodes are incomparable in T_2, and all the nodes of T_1 of the form $d_{i,j}$ are also incomparable in T_1. Since, by construction, $r(T_2)$ has n children of the form d'_j it follows that so far, $|\mathcal{M}| = n$.

Observe that since we have only matched children of w_i nodes corresponding to the D vertices, exactly $n-k$ of the child nodes of $r(T_1)$ are *free*, that is, do not have a descendant that is matched in $\mathcal{M}$. For all $i \in [n]$, we match the w'_i child nodes of $r(T_2)$ with the w_i child nodes of $r(T_1)$ that are free, i.e. we add $\{(w_i, w'_i) : v_i \notin D\}$ to $\mathcal{M}$. The resulting set $\mathcal{M}$ is clearly a consistent cluster matching for T_1 and T_2 and contains $2n-k$ pairs. Using Lemma 5.1, we may add $sz(T_1)$ leaf pairs to $\mathcal{M}$ as well as the pair $(r(T_1), r(T_2))$ to $\mathcal{M}$, which results in a consistent cluster matching of size $sz(T_1) + 1 + 2n - k$.

Conversely, assume that $\mathcal{M}$ is a consistent cluster matching between T_1 and T_2 that contains at least $sz(T_1) + 1 + 2n - k$ pairs. We show first that we may assume that in $\mathcal{M}$, every d'_j node is matched with a node in T_1 as otherwise we may transform $\mathcal{M}$ as follows. If d'_j is not matched with some node in T_1 for some $j \in [n]$, it must be that no node of T_1 of the form $d_{i,j}$ is available. This happens only if

for every w_i node of T_1 having a child $d_{i,j}$, we have that w_i is already matched (with w'_i). Furthermore, because w_i is matched, no non-leaf child $d_{i,p}$ of w'_i can be matched with a node in T_2, because w'_i does not have children that are available to match. Thus by replacing $(w_i, w'_i) \in \mathcal{M}$ by $(d_{i,j}, d'_j)$ (and rematching the leaves appropriately using Lemma 5.1), we obtain a consistent cluster matching of the same size. We may repeat this process for every j until every d'_j node is matched with a node in T_1, each time updating $\mathcal{M}$.

Now, let $D = \{v_i \in V(G) : (d_{i,j}, d'_j) \in \mathcal{M} \text{ and } i, j \in [n]\}$. For each $j \in [n]$, since $d_{i,j}$ exists in T_1 if and only if $v_i v_j$ is an edge of G or $i = j$, it follows that D must be a dominating set (because every d'_j node is matched). It remains only to show that $|D| \leq k$. To see this, observe that if a node of T_1 of the form $d_{i,j}$ is matched with a node of T_2, then this node must be d'_j. In this case, the parent w_i of $d_{i,j}$ cannot be matched, since d'_j does not have an ancestor of size $sz(w_i) = n^2 + i$. Thus if $|D| > k$ held then there would be fewer than $n - k$ of the w_i nodes of T_1 that can be matched with nodes in T_2. It follows that $|\mathcal{M}| < sz(T_1) + 1 + 2n - k$, a contradiction. Therefore, D is a dominating set of size at most k, as desired. □

Note that the arguments in the proof of Theorem 5.5 can be extended to show that computing d^*_{RF} between two *binary* trees shapes is NP-complete (Theorem 5.6). However, the proof requires a careful handling of the details, and is significantly more technical. We redirect the interested reader to the Appendix.

Theorem 5.6. *For two* binary *tree shapes T_1 and T_2 of the same size, finding $d^*_{RF}(T_1, T_2)$ is NP-complete.*

5.3. **Fixed-parameter tractability of the RF distance.** We show that deciding for two tree shapes T_1 and T_2 of the same size and a non-negative integer k if $d^*_{RF}(T_1, T_2) \leq k$ can be done in time $O(2^{k \log(2k+1)} n^3)$, where $n = sz(T_1) = sz(T_2)$. Thus computing d^*_{RF} for T_1 and T_2 is in FPT with respect to parameter k. This result holds independently of the maximum degree of T_1 and T_2. We need some more notation first.

If T is a tree shape and $u \in V(T) - L(T)$ is a child of $r(T)$, we denote by $T - u$ the tree shape obtained from T by collapsing the edge $r(T)u$ (i.e. removing u and passing its children to $r(T)$). We denote by $T - T(u)$ the tree shape obtained by removing the $T(u)$ subtree, ignoring the root if it is of degree one. More precisely, if $r(T)$ has at least 3 children, $T - T(u)$ is the result of deleting u, all of its descendants, and all their incident edges. If $r(T)$ has 2 children, $T - T(u)$ is obtained as in the previous case, plus deleting $r(T)$ and its incident edge. For the convenience of the reader, we illustrate the definition of $T - T(u)$ in terms of an example in Figure 5.

For a given integer $h \geq 1$, let $u_1, u_2, \ldots, u_l$ be the children of $r(T)$ of size h. Write $u_i \simeq u_j$ if $T(u_i)$ is isomorphic to $T(u_j)$. The $\simeq$ relation is clearly an equivalence relation. We denote by $ch_{\simeq}(r(T), h)$ the subset of children of $r(T)$ of size h obtained by choosing exactly one child (arbitrarily) for each equivalence class of this relation.

On a high level, our FPT algorithm is a top down recursive search tree algorithm in which the parameter k indicates how many nodes are allowed to contribute to the RF distance in $V(T_1) \cup V(T_2)$. The idea is that if a child u of $r(T_1)$ has size larger than any node of T_2, then u can never be matched in any consistent cluster matching, and will therefore contribute to the d^*_{RF} distance. We may thus remove u, decrease k by 1 and recurs. The same holds if a node u' of T_2 has the same

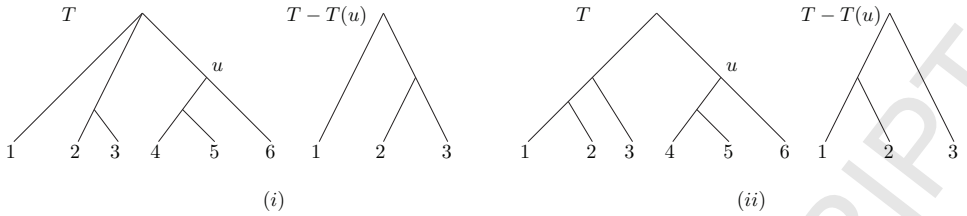

FIGURE 5. For each of (i) and (ii) we depict for the tree shape on the left and u as indicated the tree shape $T - T(u)$ on the right.

property. If otherwise u can be matched to some nodes of T_2, then we simply try every possibility (including *not* matching u), recursing in each case. The list of possibilities is given by $ch_{\simeq}(r(T_2), sz(u))$, which has bounded cardinality, provided that we eliminate pairs of isomorphic sub-tree shapes as we will show. We start with the following Lemma.

Lemma 5.7. *Let T_1 and T_2 be two tree shapes of the same size, and let u be a child of $r(T_1)$ such that $u \notin L(T_1)$. Then $d^*_{RF}(T_1, T_2) \leq d^*_{RF}(T_1 - u, T_2) + 1$.*

Proof. The lemma follows from the fact that in any leaf assignment, the set of clusters induced by $T_1 - u$ is the set of clusters induced by T_1 with the cluster induced by u removed.

□

The next result ensures that for computing $d^*_{RF}(T_1, T_2)$ we can essentially ignore pairs (A_1, A_2) of isomorphic sub-tree shapes where the root of A_i is a child of $r(T_i)$ for $i \in \{1, 2\}$, as the nodes of A_1 and A_2 can be made to have no contribution towards the d^*_{RF} distance. Although this is quite an intuitive statement, the proof is rather technical and for the sake of readability, we defer it to the end of this section.

Lemma 5.8. *Let T_1 and T_2 be two tree shapes of the same size, and suppose that $r(T_1)$ has a child u and $r(T_2)$ has a child u' such that $T_1(u)$ and $T_2(u')$ are isomorphic tree shapes. Let $\sigma : V(T_1(u)) \to V(T_2(u'))$ be the underlying bijection for an isomorphism between $T_1(u)$ and $T_2(u')$.*

Then there exists leaf assignments ϕ_1 of T_1 and ϕ_2 of T_2 such that the two following properties hold:

- *for every leaf $l \in T_1(u)$, we have $\phi_1(l) = \phi_2(\sigma(l))$;*
- *$d_{RF}((T_1, \phi_1), (T_2, \phi_2)) = d^*_{RF}(\mathcal{T}_1, \mathcal{T}_2)$.*

Algorithm 1 describes our FPT procedure in detail. In a first phase, we simplify T_1 and T_2 by eliminating children of $r(T_1)$ of size greater than those of $r(T_2)$, as they are nodes that are certain to contribute to the RF distance. The same applies for the large size children of $r(T_2)$. Each removal decrease k, the number of nodes that are allowed to contribute to the RF distance, by 1. If there are no such nodes to eliminate, then the roots of T_1 and T_2 both have a child of maximum size p. We then eliminate all pairs of isomorphic sub-tree shapes of T_1 and T_2 of size p - with a special case to handle when one tree has a degree 2 root and not the other. We repeat this process until T_1 and T_2 have non-isomorphic children of size p. This is where we have to try every possible matching between these children.

Algorithm 1 Algorithm for the RF distance. T_1 and T_2 are two trees of the same size, and k is the maximum allowed RF distance between T_1 and T_2. The Algorithm returns $d^*_{RF}(T_1, T_2)$ if it is at most k, or ∞ if it is above k. We assume that both trees shapes are to be assigned the same set of labels.

```
1: procedure RFDIST(T_1, T_2, k)
2:     if k < 0 then Return ∞
3:     done ← False
4:     while done = False do
5:         if T_1 and T_2 are isomorphic then Return 0
6:         Let u_1 (resp. u_2) be a child of r(T_1) (resp. of r(T_2)) of maximum size
7:         if sz(u_1) < sz(u_2) then
8:             Return RFDIST(T_1, T_2 − u_2, k − 1) + 1
9:         else if sz(u_2) < sz(u_1) then
10:            Return RFDIST(T_1 − u_1, T_2, k − 1) + 1
11:        else if sz(u) = sz(v) then
12:            p ← sz(u_1)
13:            while r(T_1) has a child u_1 of size p and r(T_2) has a child u_2 of size
    p
14:                such that T_1(u_1) and T_2(u_2) are isomorphic do
15:                if p = sz(T_1)/2, r(T_1) has 2 children {u_1, u'} and
16:                  r(T_2) has at least 3 children then
17:                    Return RFDIST(T_1 − u', T_2, k − 1) + 1
18:                else if p = sz(T_1)/2, r(T_2) has 2 children {u_2, u'} and
19:                      r(T_1) has at least 3 children then
20:                    Return RFDIST(T_1, T_2 − u', k − 1) + 1
21:                else
22:                    T_1 ← T_1 − T_1(u_1)
23:                    T_2 ← T_2 − T_2(u_2)
24:            if both r(T_1) and r(T_2) both still have at least one child of size p
    then
25:                done = True
26:    Let {a_1, ..., a_s} = ch_≃(r(T_1), p)
27:    Let {b_1, ..., b_t} = ch_≃(r(T_2), p)
28:    if s > k or t > k then
29:        Return ∞
30:    else
31:        best ← RFDIST(T_1 − a_1, T_2, k − 1)+1
32:        for i ∈ [t] do
33:            dist_1 ← RFDIST(T_1(a_1), T_2(b_i), k − 1)
34:            dist_2 ← RFDIST(T_1 − T_1(a_1), T_2 − T_2(b_i), k − 1)
35:            if dist_1 + dist_2 < best then best ← dist_1 + dist_2
36:        Return best
```

Theorem 5.9. *For a given pair of tree shapes T_1 and T_2 of size n and an integer k, Algorithm 1 correctly decides if $d^*_{RF}(T_1, T_2) \leq k$ and runs in time $O(2^{k \log(2k+1)} n^3)$.*

Proof. The algorithm creates a search tree of recursive calls, where the root is the initial call and the leaves are the terminal cases. We show by induction over the

height of a node in this search tree that if $d^*_{RF}(T_1, T_2) \leq k$, then Algorithm 1 returns $d^*_{RF}(T_1, T_2)$, and otherwise it returns ∞. This is clearly true for the terminal cases when $k = -1$ or when T_1 and T_2 are isomorphic.

Otherwise, note that if at any point the node u_2 is removed on line 8, it is because no node of T_1 has size $sz(u_2)$. Thus, u_2 will inevitably contribute to the d^*_{RF} distance, and by Lemma 5.7, $d^*_{RF}(T_1, T_2) = d^*_{RF}(T_1, T_2 - u_2) + 1$. This justifies removing u_2 and reducing k by 1. The same argument applies when removing u_1 on line 10.

Consider now the special case that occurs on line 17. Here, $r(T_1)$ has exactly two children u_1, u' both of size p and $r(T_2)$ has one child of size p, and other children which must all have size strictly smaller than p. Since $T_1(u_1)$ and $T_2(u_2)$ are isomorphic, by Lemma 5.8 we may assume that there is a leaf assignment of T_1 and T_2 that minimizes $d^*_{RF}(T_1, T_2)$ and that matches the leaves (and clusters) of $T_1(u_1)$ and $T_2(u_2)$. It follows that the u' cluster under this assignment must contribute to the RF distance (as u_2 is the only node of T_2 of its size, and it already matched with u_1). This justifies line 17 and, by symmetry, line 20.

Suppose that the algorithm reached lines 22-23. We wish to show that $d^*_{RF}(T_1, T_2)$ remains unchanged after executing these two lines, i.e. that $d^*_{RF}(T_1, T_2) = d^*_{RF}(T_1 - T_1(u_1), T_2 - T_2(u_2))$. Note that this latter statement is not true in general - we merely show it true for the T_1 and T_2 at this point in the algorithm. By Lemma 5.8, we may assume that there is a leaf assignment of T_1 and T_2 that minimizes $d^*_{RF}(T_1, T_2)$ in which none of the nodes of $V(T_1(u_1)) \cup V(T_2(u_2))$ contributes to the $d^*_{RF}(T_1, T_2)$ distance, as their leaves are matched together according to some isomorphism. It is therefore easy to see that $d^*_{RF}(T_1, T_2) = d^*_{RF}(T_1 - T_1(u_1), T_2 - T_2(u_2))$ if both $r(T_1)$ and $r(T_2)$ have at least 3 children, or if both have 2 children. This condition must hold at this point, since its negation was verified by the two special cases just above. Therefore, we have introduced no error by removing the two isomorphic sub-tree shapes.

It follows that the algorithm is correct if it returns when it is inside the main **while** loop. Assume that the algorithm exits the loop without returning.

Let $A = \{a_1, \ldots, a_s\}$ and $B = \{b_1, \ldots, b_t\}$ be the sets identified by the algorithm after this phase. Suppose that $s > k$ or $t > k$ and line 29 is executed. To establish that returning ∞ by our algorithm is justified, it suffices by symmetry to consider the case that $s > k$. First note that due to line 13, for any $a \in A$, no sub-tree shape of T_2 can be isomorphic to $T_1(a)$. Also observe that in any leaf assignment ϕ_1 of T_1 and ϕ_2 of T_2, the cluster $C_{(T_1,\phi_1)}(a)$ is either preserved (i.e. in $\mathcal{C}(T_2, \phi_2)$), or not. If $C_{(T_1,\phi_1)}(a)$ is not preserved, then it contributes to $d^*_{RF}(T_1, T_2)$. If it is preserved, then there exists some $b \in B$, such that $C_{(T_1,\phi_1)}(a) = C_{(T_2,\phi_2)}(b)$. Since $T_1(a)$ and $T_2(b)$ cannot be isomorphic, there must exist a node in one of them that contributes to $d^*_{RF}(T_1, T_2)$. In both cases, each $a_i \in A$ implies the existence of a distinct node of $V(T_1) \cup V(T_2)$ that contributes to the d^*_{RF} distance, implying $d^*_{RF}(T_1, T_2) > k$ under the assumption that $s > k$. The argument is the same if $t > k$. Thus line 29 is justified.

Suppose instead that the 'else' on line 30 is entered, and consider the a_1 node. Then again, under any leaf assignments ϕ_1 of T_1 and ϕ_2 of T_2, the cluster induced by a_1 is either not preserved, or it is preserved. In the latter case, there is some child b of $r(T_2)$ of size p such that $T_2(b)$ is isomorphic to $T_2(b_j)$, where $b_j \in B$, such that $C_{(T_1,\phi_1)}(a_1) = C_{(T_2,\phi_2)}(b)$ (this is because there are no other nodes of size

$p = sz(a_1)$). The algorithm tests all these cases (it is clearly only necessary to try matching a_1 to only one representative per equivalence class of $ch_{\simeq}(r(T_2), p)$). To justify reducing to $k-1$ in the recursive call of line 33 and 34, observe that both the pairs $\{T_1(a_1), T_2(b_i)\}$ and $\{T_1 - T_1(a_1), T_2 - T_2(b_i)\}$ have d^*_{RF} distance at least 1, since they are non-isomorphic. Thus if one of the tree pairs has distance strictly more than $k-1$, $d^*_{RF}(T_1, T_2) > k$. The correctness of the algorithm then follows by induction.

As for the complexity, first assume that we initially computed, for each pair of nodes $u \in V(T_1)$ and $v \in v(T_2)$, whether $T_1(u)$ and $T_2(v)$ are isomorphic. This can be done in time $O(n^3)$ (see e.g. [5]). The search tree created by the algorithm has depth at most k and maximum degree $1+2k$ (one recursive call for a_1 unmatched, plus 2 for each $i \leq t \leq k$), and it is straightforward to see that one pass through the algorithm takes $O(n^3)$ time (the first while loop is iterated at most $O(n)$ times since each iteration eliminates at least one vertex, and the inner while loop iterates over $O(n^2)$ pairs of vertices). The complexity is therefore $O((2k+1)^k n^3 + n^3) = O(2^{k \log(2k+1)} n^3)$.

□

We now present the proof of Lemma 5.8.

of Lemma 5.8. To prove the lemma, we claim that there exists a maximum consistent cluster matching $\mathcal{M}$ between T_1 and T_2 such that $(x, \sigma(x)) \in \mathcal{M}$ for every $x \in V(T_1(u))$. By Lemma 5.2, this is sufficient to prove our lemma, as $\mathcal{M}$ can be turned into leaf assignments ϕ_1 and ϕ_2 satisfying $\phi_1(l) = \phi_2(\sigma(l))$ for every leaf $l \in L(T_1(u))$.

To prove our claim, let $\mathcal{M}$ be a maximum consistent cluster matching that maximizes the number of vertices x of $T_1(u)$ such that $(x, \sigma(x)) \in \mathcal{M}$. Assume for contradiction that $\mathcal{M}$ does not satisfy our claim. Note that since $\mathcal{M}$ is maximum, by Lemma 5.1 we may assume that every leaf in T_1 is matched in $\mathcal{M}$ with a leaf in T_2. If we have $(l, \sigma(l)) \in \mathcal{M}$ for every leaf $l \in L(T_1(u))$, then it is not hard to see that $\mathcal{M}$ can be modified to satisfy $(x, \sigma(x)) \in \mathcal{M}$ for all $x \in V(T_1(u))$ without decreasing its cardinality. So assume that this is not the case.

Let $(x, y) \in \mathcal{M}$ be chosen so that (i) either $x \in V(T_1(u))$ and $y \neq \sigma(x)$ or $y \in V(T_2(u'))$ and $x \neq \sigma^{-1}(y)$, and (ii) that $sz(x) = sz(y)$ is maximum among all possible choices. Note that (x, y) exists since, in particular, some leaf l of $T_1(u)$ is not matched with $\sigma(l)$. Assume w.l.o.g. that the former case holds, i.e. $x \in V(T_1(u))$ but $y \neq \sigma(x)$. Let $x' = \sigma(x)$ and let $\mathcal{M}_{x'} = \{(z, z') \in \mathcal{M} : z' \text{ is a descendant of } x'\}$. Call $(z, z') \in \mathcal{M}_{x'}$ *maximal* if, for every $(w, w') \in \mathcal{M}_{x'}$, w' is not a proper ancestor of z'. Let $(z_1, z'_1), \ldots, (z_k, z'_k)$, $k \geq 1$, be the maximal elements of $\mathcal{M}_{x'}$. In particular, if x' is matched then there is only one maximal element. Note that $\sum_{i \in [k]} sz(z_i) = \sum_{i \in [k]} sz(z'_i) = sz(x') = sz(x) = sz(y)$.

We next argue that by matching the nodes of $T_1(x)$ with those of $T_2(x')$, and the nodes of $T_1(z_1), \ldots, T_1(z_k)$ with those of $T_2(y)$, we obtain another matching $\mathcal{M}'$ between T_1 and T_2 in which more nodes v of $T_1(u)$ are matched with their images $\sigma(v)$ in $T_2(u')$ which contradicts the choice of $\mathcal{M}$.

We start with claiming that except for $r(T_1)$ and $r(T_2)$, no proper ancestor of a node in $\{x, y, z_1, \ldots, z_k, z'_1, \ldots, z'_k\}$ is matched in $\mathcal{M}$. For x, assume that a non-root proper ancestor w of x is matched. Then $w \in V(T_1(u))$ as u is a child of $r(T_1)$. By the choice of x, we must have $(w, \sigma(w)) \in \mathcal{M}$. Let $M_1 = \{(v, v') \in \mathcal{M} : v$

is a descendant of $w\}$ and $M_2 = \{(v, \sigma(v)) : v \in V(T_1(w))\}$. Then as $T_1(w)$ and $T_2(\sigma(w))$ are isomorphic, $\mathcal{M}' := (\mathcal{M} \setminus M_1) \cup M_2$ is a consistent cluster matching, and it is maximum since $|M_2| \geq |M_1|$. Moreover, by our choice of $\mathcal{M}$, we must then have $M_1 = M_2$, as otherwise $\mathcal{M}'$ would have more pairs of the form $(v, \sigma(v))$ than $\mathcal{M}$. In particular, $(x, \sigma(x)) \in M_2 = M_1 \subseteq \mathcal{M}$, a contradiction to our choice of x.

For y, if it has a matched non-root proper ancestor, then by consistency this ancestor would be matched to a non-root proper ancestor of x, which cannot be the case as we just argued.

To see that no proper ancestor of z'_i, $i \in [k]$, can be matched except $r(T_2)$, and, therefore, that no proper ancestor of z_i, $i \in [k]$, can be matched except $r(T_1)$ it suffices to show in view of the maximality of the z'_i's that no proper ancestor w' of x' except $r(T_2)$ can be matched. If w' is matched with $\sigma^{-1}(w')$ then since $\mathcal{M}$ is a consistent cluster matching, and w' is an ancestor of x' it follows that $x = \sigma^{-1}(x')$ has a matched ancestor which is impossible. If w' is not matched with $\sigma^{-1}(w')$ then we obtain a contradiction to the choice of (x, y) since $sz(w') > sz(x') = sz(x)$. This completes the proof of the claim.

These facts allows us to rearrange the matchings of the nodes in the set $\{x, y, z_1, \ldots, z_k, z'_1, \ldots, z'_k\}$ without breaking consistency, as we now describe. Consider the cluster matching $\mathcal{M}'$ obtained by removing from $\mathcal{M}$ any pair containing a descendant of a node in $\{x, y, z_1, \ldots, z_k, z'_1, \ldots, z'_k\}$ (noting that no pair of the form $(v, \sigma(v))$ was removed), and then adding to $\mathcal{M}$ the following pairs. Let $T_z = T_1(z_1)$ if $k = 1$, and otherwise T_z is the tree obtained by joining the roots of $T_1(z_1), \ldots, T_1(z_k)$ under a common parent, creating an "artificial root". Let $\delta = 1$ if $k > 1$ and $\delta = 0$ if $k = 1$ (δ indicates whether T_z contains an artificial root). Add to $\mathcal{M}'$ all the pairs $(v, \sigma(v))$ for each $v \in V(T_1(x))$, and add all the pairs of a maximum cluster matching between T_z and $T_2(y)$, excluding $(r(T_z), y)$ if $k > 1$. Then $\mathcal{M}'$ is a consistent cluster matching between T_1 and T_2, since no non-root ancestor of the nodes in $\{x, y, z_1, \ldots, z_k, z'_1, \ldots, z'_k\}$ is matched in $\mathcal{M}$ (and hence property (M2) is preserved). Moreover, $\mathcal{M}'$ has more pairs of the form $(v, \sigma(v))$ than $\mathcal{M}$ since $y \neq \sigma(x)$ and $(x, y) \in \mathcal{M}$.

It remains to show that $\mathcal{M}'$ is a *maximum* consistent cluster matching between T_1 and T_2, which proves the lemma as this contradicts our choice of $\mathcal{M}$. Put $T_x = T_1(x), T_{x'} = T_2(x')$ and $T_y = T_2(y)$. Notice that

$$|\mathcal{M}'| = |\mathcal{M}| - \mu(T_x, T_y) - (\mu(T_z, T_{x'}) - \delta) + \mu(T_x, T_{x'}) + (\mu(T_z, T_y) - \delta).$$

Using Corollary 5.4, we have

$$\begin{aligned} \mu(T_x, T_{x'}) + \mu(T_z, T_y) &= |V(T_x)| + \mu(T_z, T_y) \\ &\geq \mu(T_z, T_x) + \mu(T_x, T_y) \\ &= \mu(T_z, T_{x'}) + \mu(T_x, T_y) \end{aligned}$$

where we use T_x and $T_{x'}$ interchangeably since they are isomorphic tree shapes. This implies $|\mathcal{M}'| \geq |\mathcal{M}|$, which concludes the proof.

□

6. The Maximum Agreement Subtree (MAST) on tree shapes

In this section, we study the Maximum Agreement Subtree (MAST) between two or more tree shapes. Unlike the previous sections, here we do not require that the given tree shapes have the same size. This prompts some additional definitions.

For a finite set X, a *rooted partial X-tree* is a pair $\mathcal{T} = (T, \phi)$ where T is a rooted tree shape and ϕ is an injection from $L(T)$ into X (hence $sz(T) < |X|$ is possible). Put $\chi(\mathcal{T}) = \{\phi(l) : l \in L(T)\}$, i.e. $\chi(\mathcal{T})$ is the set of labels that appear at the leaves of $\mathcal{T}$. If T is a tree shape, let $\mathcal{F}_X(T) = \{(T, \phi) : \phi \text{ is an injection from } L(T) \text{ into } X\}$ be the set of all possible rooted partial X-trees that have T as their underlying tree shape.

Let T be a rooted tree shape, let $L \subseteq L(T)$ and let x be the last common ancestor of L in T. The *restriction* of T to L, denoted $T|_L$, is the rooted tree shape obtained from $T(x)$ by first deleting every node of $T(x)$ that does not have a descendant in L and then contracting the nodes in the resulting tree with a single child until no such node remains. If $\mathcal{T} = (T, \phi)$ is a rooted partial X-tree and $X' \subseteq \chi(\mathcal{T})$, the *restriction* $\mathcal{T}|_{X'}$ of $\mathcal{T}$ to X' is the pair $(T|_L, \phi|_L)$ where $L = \{l \in L(T) : \phi(l) \in X'\}$. Clearly, $\mathcal{T}|_{X'}$ is a rooted phylogenetic tree on X'.

Suppose $\mathcal{T}_1, \ldots, \mathcal{T}_k$, $k \geq 2$, are rooted partial X-trees and $X' \subseteq \bigcap_{i\in[k]} \chi(\mathcal{T}_i)$. Then we say that the trees in $\tau = \{\mathcal{T}_1, \ldots, \mathcal{T}_k\}$ *agree* on X' if the induced rooted partial X-trees in the (multi)set $\tau|_{X'} = \{\mathcal{T}_1|_{X'}, \ldots, \mathcal{T}_k|_{X'}\}$ are pairwise isomorphic such that the underlying bijections preserve the elements in X'. In this case, we call a tree in $\tau|_{X'}$ an *agreement subtree* of τ. We denote by $MAST(\tau)$ the maximum size of a subset $X' \subseteq X$ such that the trees in τ agree on X', and we call a tree in $\tau|_{X'}$ a *maximum agreement subtree (MAST)* for τ. Note that if M is a MAST for τ then for all $\mathcal{T}_i \in \tau$ with underlying tree shape T_i, there exists an injection $f_i : V(M) \to V(T_i)$ from $V(M)$ into the node set of T_i such that if x is a descendant of y in M (resp. x and y are incomparable in M), then $f_i(x)$ is a descendant of $f_i(y)$ in T_i (resp. $f_i(x)$ and $f_i(y)$ are incomparable in T_i). Note that f_i is sometimes called an *embedding* of M into T_i. We say that the set of injections $f_1, \ldots, f_k$ *witness* the fact that M is a MAST for τ.

For a given set τ of tree shapes $T_1, \ldots, T_k$, possibly of different sizes, we define the *unlabelled MAST of* τ, denoted $uMAST(\tau)$ by putting $X = [\max_{i\in[k]}\{sz(T_i)\}]$ and

$$uMAST(\tau) = \max_{\mathcal{T}_1\in\mathcal{F}_X(T_1),\ldots,\mathcal{T}_k\in\mathcal{F}_X(T_k)} MAST(\{\mathcal{T}_1, \ldots, \mathcal{T}_k\}).$$

It is known that computing $uMAST(T_1, T_2)$ for two tree shapes (and even more generally for two MUL-trees) T_1 and T_2 can be done in quadratic time [14]. As noted in the introduction, it follows that the extension d^*_{MAST} of the MAST distance d_{MAST} on phylogenetic trees can be computing in polynomial time [19, p.1033].

6.1. **MAST on three tree-shapes.** In this section, we show that the problem of computing an unlabelled MAST is NP-complete on three tree shapes of the same size when the degree of the input tree shapes is unbounded. However, in the next section we shall also show that the problem is FPT with respect to the maximum degree if the number of tree shapes is constant.

Our proof of NP-completeness is an adaptation of [4]. We reduce from the RESTRICTED 3D-MATCHING problem, shown to be hard in [7]. In this problem, we are given an integer $k \geq 0$ and three pairwise disjoint sets V_1, V_2, V_3 each with $n \geq 2$ elements, and a set $E \subseteq V_1 \times V_2 \times V_3$ of triplets such that every $v \in V_1 \cup V_2 \cup V_3$ occurs in exactly 2 triplets. We ask if there exists a 3D-Matching of size k, i.e. a subset $E' \subseteq E$ of size at least k such that no two elements of E' intersect (when thought of as 3-sets). To present our reduction, we define a *caterpillar shape* to be a rooted binary tree shape in which every internal node has at least one leaf child.

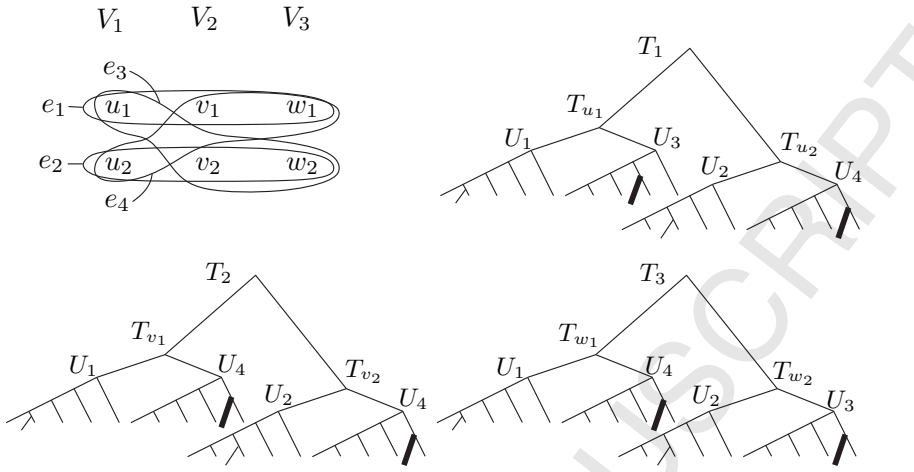

FIGURE 6. An example of the reduction with $n = 2$ and $m = 2n = 4$. On the left, the RESTRICTED-3D-MATCHING instance. The triplets are $e_1 = (u_1, v_1, w_1), e_2 = (u_2, v_2, w_2), e_3 = (u_1, v_2, w_2), e_4 = (u_2, v_1, w_1)$. There is a 3D-matching of size $k = 2$ comprising of e_1 and e_2. The corresponding MAST of size $n(2m + 2) + k = 22$ constructed in the proof of hardness of Theorem 6.1 can be obtained after removing the heavy edges along with their incident leaf.

For our reduction we next construct three rooted tree shapes T_1, T_2, and T_3 of the same size as follows (Figure 6 illustrates the reduction). Put $\mathcal{V} := V_1 \cup V_2 \cup V_3$ and $E = \{e_1, \ldots, e_m\}$ (note that $m = 2n$). To each triplet e_i we associate a rooted tree shape U_i on $m + 2$ leaves as follows: start with a caterpillar shape on $m + 1$ leaves, and list the leaves $l_1, l_2, \ldots, l_{m+1}$ in non-increasing order of depth. Obtain U_i by grafting a new leaf x_i on the edge between l_{i+1} and its parent. Observe that in this manner, no two tree shapes U_i and U_j are isomorphic, but become so when removing x_i and x_j (and their incident edges) from U_i and U_j (suppressing the resulting degree two node in each).

To each element $v \in \mathcal{V}$ we then associate a tree shape T_v as follows. Let e_{i_1}, e_{i_2} be the triplets containing v, let T_v be the tree shape obtained by taking a copy of U_{i_1} and U_{i_2}, then joining their roots under a common parent. Observe that the T_v sub-tree shapes differ only by the placement of their x_{i_1} and x_{i_2} leaves. Then for each $i \in \{1, 2, 3\}$, the tree shape T_i is obtained by joining the roots of $\{T_v : v \in V_i\}$ under a common parent. Note that for all $i = 1, 2, 3$, the tree shape T_i has $n(2m+4)$ leaves.

Theorem 6.1. *For three tree shapes T_1, T_2 and T_3 of the same size, computing $uMAST(\{T_1, T_2, T_3\})$ is NP-complete.*

Proof. To see that the problem is in NP, observe that a MAST can serve as a certificate, since subtree isomorphism can be checked in polynomial time [5]. As for hardness, let V_1, V_2, V_3, E and k form an instance of RESTRICTED 3D-MATCHING, and let T_1, T_2 and T_3 be tree shapes constructed as above. We claim that the given RESTRICTED 3D-MATCHING instance admits a 3D-matching of size at least k

if and only if the associated tree shapes T_1, T_2 and T_3 agree on a MAST of size at least $n(2m+2)+k$.

Assume first that $E' \subseteq E$ is a 3D-matching of size k. The proof works by "matching" for $(u,v,w) \in E'$ the T_u, T_v and T_w sub-tree shapes of T_1, T_2 and T_3. To be more precise, for each $e_i = (u,v,w) \in E'$, we assign to each of T_u, T_v and T_w the same set Y of leaf labels so that the resulting rooted phylogenetic trees on Y agree on a MAST of size $2(m+2)-1 = 2m+3$ (note that this can easily be done since each of the tree shapes T_u, T_v and T_w contains a copy of U_i as a subtree, as in e.g. the $T_{u_1}, T_{v_1}, T_{w_1}$ sub-tree shapes in Figure 6). Let $F \subseteq V_1 \times V_2 \times V_3$ be a subset of triples such that each element of $\mathcal{V}$ not contained in an element of E' occurs exactly once. Note that $F = \emptyset$ is possible if E' is a perfect 3D-matching, as in Figure 6. In that particular case, we clearly have $n = k$, and T_1, T_2 and T_3 agree on a set X' of size at least $n(2m+3) = n(2m+2)+n$, as desired. Otherwise, assume $F \neq \emptyset$. For each $(u,v,w) \in F$, assign the same set Y' of leaf labels to T_u, T_v and T_w so that the resulting rooted phylogenetic trees on Y' agree on a MAST M' of size $2(m+2)-2 = 2m+2$ (this can be achieved by removing the x_i leaves in the U_i sub-tree shapes). Taking the disjoint union of the leaf sets of the MASTs constructed above, we obtain a bijective labelling of the leaf sets for the tree shapes T_1, T_2 and T_3 such that they agree on a MAST of size $k(2m+3)+(n-k)(2m+2) = n(2m+2)+k$.

Conversely, assume that there exists a leaf assignment of the tree shapes T_1, T_2, and T_3 in terms of a set X such that the resulting phylogenetic trees $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ on X agree on a MAST M of size $n(2m+2)+k$. Then for all $i \in [3]$ there exists an injection $f_i : V(M) \to V(T_i)$ witnessing that M is a MAST for $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$. First note for all $i \in [3]$ that $f_i(r(M)) = r(T_i)$ since $r(T_i)$ is the only node of T_i that has size at least $n(2m+2)+k$. Note next that for each $i \in [3]$, each child of $r(M)$ must be mapped to a child of $r(T_i)$ under f_i. Indeed, suppose for contradiction that there exists some $i \in [3]$ and some child z of $r(M)$ such that $f_i(z)$ is not a child of $r(T_i)$. Then $f_i(z)$ belongs to some U_j sub-tree shape, and $sz(z) \leq m+2$. Let w be the child of $r(T_i)$ that is an ancestor of $f_i(z)$. Then $f_i^{-1}(w) = \emptyset$, and so M has at most $m+2$ leaves mapped to the $T_i(w)$ subtree under f_i. As the other children of $r(T_i)$ each have $2m+4$ children, it follows that the size of M can be at most $m+2+(n-1)(2m+4) < n(2m+2)+k$ when $m = 2n$ which is impossible.

Now, consider a child z of $r(M)$. Let $u' = f_1(z), v' = f_2(z)$ and $w' = f_3(z)$. By the above argument, u', v' and w' are the roots of T_u, T_v and T_w, respectively, where $u \in V_1, v \in V_2, w \in V_3$. Put $\tau = \{T_u, T_v, T_w\}$. Then $sz(z) = sz(uMAST(\tau))$. It is not hard to see that the tree shapes in τ have at most one U_i sub-tree shape in common, which happens if and only if $(u,v,w) \in E$. In this case, $sz(uMAST(\tau)) = 2(m+2)+1$ and if there is no such triplet, $sz(uMAST(\tau)) = 2(m+2)$. Therefore, M has size $n(2m+2)+k$ only if $r(M)$ has k children that each correspond to a triplet of E. This correspondence yields a 3D-matching of size k.

□

6.2. **An algorithm for few tree shapes and bounded degree.** We present an algorithm that computes $uMAST(\tau)$ of $k \geq 2$ tree shapes, possibly of different sizes, in time $O((2n)^k \cdot d^3 \cdot 2^{d \log d \cdot (k+1/2)})$, where $d \geq 2$ is the maximum degree of a node of a tree shape in τ and $n \geq 2$ is the maximum size of a tree in τ. The algorithm is inspired by the algorithm for the labeled version presented in [13].

Let $\tau = \{T_1, \ldots, T_k\}$ be a set of tree shapes of size at most n in which each node of a tree shape has at most d children. A *node vector* $\vec{v} = (v_1, v_2, \ldots, v_k)$ is a

sequence of nodes where $v_i \in V(T_i)$ for each $i \in [k]$. From now on, we denote the i-th node in a node vector $\vec{v}$ for τ by v_i. We write $\vec{u} \leq \vec{v}$ if u_i is a descendant of v_i for each $i \in [k]$, and $\vec{u} < \vec{v}$ if additionally $\vec{u} \neq \vec{v}$. We say $\vec{u}$ is a *direct predecessor* of $\vec{v}$ if $u_i \neq v_i$ for exactly one $i \in [k]$, and u_i is a child v_i. Thus $\vec{u}$ is obtained from $\vec{v}$ by replacing for some $i \in [k]$ the node v_i by one of its children in T_i. Denote by $DP(\vec{v})$ the set of direct predecessors of $\vec{v}$. Since the degree of a tree shape in τ is at most d, it is clear that $DP(\vec{v})$ has at most kd elements.

For a node vector $\vec{v}$, a *child vector* for $\vec{v}$ is a node vector $\vec{v'} = (v'_1, \ldots, v'_k)$ where, for each $i \in [k]$, v'_i is a child of v_i. We call two child vectors $\vec{v'}$ and $\vec{v''}$ of $\vec{v}$ *compatible* if $v'_i \neq v''_i$ for every $i \in [k]$.

For a node vector $\vec{v}$ for τ, denote by $uMAST(\vec{v})$ the size of the unlabelled MAST for the set of tree shapes obtained by restricting each tree shape in T_i, $i \in [k]$ to the tree shape $T_i(v_i)$ rooted at v_i. Put differently, $uMAST(\vec{v}) = uMAST(T_1(v_1), \ldots, T_k(v_k))$. Clearly, our goal is to obtain $uMAST(\vec{v})$ where $\vec{v} = (r(T_1), \ldots, r(T_k))$. We achieve this by computing $uMAST(\vec{v})$ for each one of the possible $O((2n)^k)$ node vectors $\vec{v}$ for τ by dynamic programming.

Suppose $\vec{v}$ is a node vector for τ. Let $C(\vec{v})$ be the set of all possible child vectors for $\vec{v}$. Note that since the maximum degree of a tree shape in τ is d, $C(\vec{v})$ has at most d^k elements. The *compatibility graph* $G_{\vec{v}} = (V, E, w)$ for $\vec{v}$ is a weighted graph with vertex set $V = C(\vec{v})$ and edge set $E = \{\vec{v_1}\vec{v_2} : \vec{v_1}$ and $\vec{v_2}$ are compatible$\}$. Each vertex $\vec{v'}$ is weighted by $w(\vec{v'}) = uMAST(\vec{v'})$. We denote by $MWC(G_{\vec{v}})$ the maximum weight of a clique in $G_{\vec{v}}$. Moreover, we put $bestDP(\vec{v}) = \max_{\vec{u} \in DP(\vec{v})}(uMAST(\vec{u}))$ in case $DP(\vec{v}) \neq \emptyset$.

We can now state our dynamic programming recurrence.

Lemma 6.2. *Suppose $\tau = \{T_1, \ldots, T_k\}$ is a set of k tree shapes such that each node of a tree shape has at most d children. Then*

$$uMAST(\vec{v}) = \begin{cases} 1 & \text{if } v_i \in L(T_i), \text{ for all } i \in [k]; \\ \max(bestDP(\vec{v}), MWC(G_{\vec{v}})) & \text{otherwise.} \end{cases}$$

Proof. In the case when all members of $\vec{v}$ are leaves, the lemma is easily verified. Suppose that some member of $\vec{v}$ is not a leaf. Let $K := \max(bestDP(\vec{v}), MWC(G_{\vec{v}}))$. We first show that $uMAST(\vec{v}) \leq K$. Label the leaves of $T_1(v_1), \ldots, T_k(v_k)$ in terms of a set X so that the resulting set τ' of rooted partial X-trees have a $MAST$ M of maximum size. For all $i \in [k]$, let $f_i : V(M) \to V(T_i)$ be the injection that witnesses that M is a $MAST$ for τ'. Let $\vec{q} = (f_1(r(M)), \ldots, f_k(r(M))$. Note that since only the leaves that descend from a node in $\vec{q}$ contribute to $uMAST(\vec{v})$, it follows that for any $\vec{w}$ such that $\vec{q} \leq \vec{w} \leq \vec{v}$, we have $uMAST(\vec{w}) = uMAST(\vec{q})$.

If $\vec{q} < \vec{v}$, then there must be a direct predecessor $\vec{u}$ of $\vec{v}$ such that $\vec{q} \leq \vec{u}$. In this case, $uMAST(\vec{v}) = uMAST(\vec{q}) = uMAST(\vec{u}) \leq bestDP(\vec{v}) \leq K$.

If $\vec{q} = \vec{v}$, let $z_1, \ldots, z_j$ be the children of $r(M)$. For each $i \in [j]$, let $\vec{z^i} = (f_1(z_i), \ldots, f_k(z_i))$. Then although $\vec{z^i}$ need not be a child vector of $\vec{v}$ there must exist for each $\vec{z^i}$ a child vector $\vec{v^i}$ of $\vec{v}$ such that $\vec{z^i} \leq \vec{v^i}$. Moreover, since f_i is an injection for all $i \in [k]$ any two distinct child vectors in $\xi = \{\vec{v^1}, \ldots, \vec{v^j}\}$ must be compatible. Hence, the subgraph of $G_{\vec{v}}$ induced by ξ is a clique. Since the size of M is at most $\sum_{i=1}^{j} uMAST(\vec{v^i})$, it follows that $uMAST(\vec{v}) \leq MWC(G_{\vec{v}}) \leq K$.

To see that $uMAST(\vec{v}) \geq K$ holds too, suppose first that $K = bestDP(\vec{v})$. Then $uMAST(\vec{v}) \geq K$ follows easily, as any MAST for a direct predecessor of $\vec{v}$

is also a MAST for $\vec{v}$. So assume that $K = MWC(G_{\vec{v}})$. Let $\vec{v^1}, \ldots, \vec{v^j}$, $j \geq 2$, be the child vectors of $\vec{v}$ in a maximum weight clique of $G_{\vec{v}}$. Consider some $i \in [j]$. Then since $\vec{v^i} < \vec{v}$, we may assign labels to the leaf set of the sub-tree shapes $T_i(v^i_l)$, $l \in [k]$, of the elements in τ so that the resulting labelled trees agree on a MAST of size $uMAST(\vec{v^i})$. Since the child vectors of $\vec{v}$ are compatible, we may apply this operation to each $\vec{v^i}$ vector separately so that the resulting labelled trees agree on a a MAST of size at least $\sum_{i=1}^{j} uMAST(\vec{v^i}) = MWC(G_{\vec{v}})$, implying $uMAST(\vec{v}) \geq K$.

□

Theorem 6.3. *The unlabelled-MAST problem can be solved in time* $O((2n)^k \cdot d^3 \cdot 2^{d \log d \cdot (k+1/2)})$.

Proof. The algorithm simply traverses all possible node vectors in increasing order w.r.t. $<$, then computes $uMAST(\vec{v})$ in order using Lemma 6.2. Consider the time taken for a specific node vector $\vec{v}$, assuming that $uMAST(\vec{u})$ is known for every $\vec{u} < \vec{v}$. The value $bestDP(\vec{v})$ can be computed in time $O(kd)$. As for $MWC(\vec{v})$, we first need to construct $G_{\vec{v}}$. It has at most d^k vertices and d^{2k} edges. The weights take no time to compute, as they were computed by dynamic programming. Verifying compatibility of two child vectors can be done in time $O(k)$, and so the construction time is $O(kd^{2k})$. To find $MWC(G_{\vec{v}})$, we can simply enumerate each clique, and since no clique has size more than d, we can check the subsets of at most d vertices only. This takes time at most $\sum_{l=1}^{d} l^2 \binom{d^k}{l} = O(d^3 \binom{d^k}{d}) = O(d^3 (d^k \cdot e/d)^d)$ (using the $\binom{x}{y} \leq (xe/y)^y$ inequality). Since $d \geq 2$, $e \leq d^{3/2}$, and the above expression is thus $O(d^3 d^{d(k+1/2)}) = O(d^{dk+d/2+3})$, which dominates the complexity for $\vec{v}$. Since these operations are computed for all the $O((2n)^k)$ possible node vectors, the time of the algorithm is $O((2n)^k d^{dk+d/2+3}) = O((2n)^k d^3 2^{d \log d \cdot (k+1/2)})$.

□

7. Open problems

We conclude with some open problems:

- Is there is a fixed parameter tractable algorithm for computing d^*_{path}?
- Is there is a constant factor approximation algorithm for computing d^*_{path} or d^*_{RF}?
- Is it NP-hard to compute the extension d^* of the triplet distance d [10] on phylogenetic trees? Note that this is not equivalent to computing the triplet distance between two MUL-trees with fixed labels, which can be done in polynomial time. In this context it is also interesting to note that the so-called rooted triplet consistency problem for MUL-trees as defined in [12] is NP-complete, even though for phylogenetic trees in can be solved in polynomial time using the BUILD algorithm [22].
- Is the computation of $uMAST(\{T_1, T_2, T_3\})$ NP-hard for T_1, T_2, T_3 three binary tree shapes?
- For a metric d, it would be interesting to explore the hardness of computing d^* between MUL-two trees both with leaves labelled by a multi-set M in case the multiplicity of the elements in M is bounded by some constant (see e.g. [14, Theorem 4]). Note that even if each element in M has multiplicity 2 and $|M| = n$ with n even, then the number of possible assignments between

the leaves of the two trees is $2^{\frac{n}{2}}$. Hence, it is not possible to compute d^* in polynomial time in this case by simply checking all possible assignments.

Acknowledgment. The authors that Jesper Jansson and the annonymous referees for their helpful comments. K.T.H. and V.M. thank the Université de Montréal for hosting them. N.E.-M. and M.L. acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC) for the financial support of this project.

References

[1] P. Agarwal, K. Fox, A. Nath, A. Sidiropoulos, and Y. Wang. Computing the Gromov-Hausdorff distance for metric trees. In *International Symposium on Algorithms and Computation 2015*, pages 529–540. Springer, Berlin, Heidelberg, 2015.

[2] T. Akutsu and M. M. Halldórsson. On the approximation of largest common subtrees and largest common point sets. *Theoretical Computer Science*, 233(1-2):33–50, 2000.

[3] P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. *Algorithms and Complexity*, pages 288–298, 1997.

[4] A. Amir and D. Keselman. Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. *SIAM Journal on Computing*, 26(6):1656–1669, 1997.

[5] S. R. Buss. A log time algorithms for tree isomorphism, comparison, and canonization. In *Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pages 18–33. Springer, 1997.

[6] R. Chaudhary, J.G. Burleigh, and D. Fernandez-Beca. Inferring species trees from incongruent multi-copy gene trees using the robinson-foulds distance. *Algorithms for Molecular Biology*, 8:28, 2013.

[7] M. Chlebík and J. Chlebíková. Approximation hardness for small occurrence instances of NP-hard problems. *Algorithms and Complexity*, pages 631–631, 2003.

[8] S. Chou and C.-L. Hsu. Mmdt: a multi-valued and multi-labeled decision tree classifier for data mining. *Expert Systems with Applications*, 28(4):799–812, 2005.

[9] C. Colijn and G. Plazzotta. A metric on phylogenetic tree shapes. *Systematic Biology*, 2017.

[10] D. E. Critchlow, D. K. Pearl, and C. Qian. The triples distance for rooted bifurcating phylogenetic trees. *Systematic Biology*, 45(3):323–334, 1996.

[11] M. Crochemore and R. Vérin. Direct construction of compact directed acyclic word graphs. In *Annual Symposium on Combinatorial Pattern Matching*, pages 116–129. Springer, 1997.

[12] Y. Cui, J. Jansson, and W.-K. Sung. Polynomial-time algorithms for building a consensus mul-tree. *Journal of Computational Biology*, 19(9):1073–1088, 2012.

[13] M. Farach, T. M. Przytycka, and M. Thorup. On the agreement of many trees. *Information Processing Letters*, 55(6):297–301, 1995.

[14] G. Ganapathy, B. Goodson, R. Jansen, H-S Le, V. Ramachandran, and T. Warnow. Pattern identification in biogeography. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4), 2006.

[15] W. Goddard, E. Kubicka, G. Kubicki, and F.R. McMorris. The agreement metric for labeled binary trees. *Mathematical Biosciences*, 123(2):215–226, 1994.

[16] WC T. Gregg, S. H. Ather, and M. W. Hahn. Gene-tree reconciliation with mul-trees to resolve polyploidy events. *Systematic Biology*, 66(6):1007–1018, 2017.

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

[18] K. T. Huber, V. Moulton, M. Steel, and T. Wu. Folding and unfolding phylogenetic trees and networks. *Journal of Mathematical Biology*, 73(6-7):1761–1780, 2016.

[19] K. T. Huber, A. Spillner, R. Suchecki, and V. Moulton. Metrics on multilabeled trees: interrelationships and diameter bounds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(4):1029-1040, 2011.

[20] M. Laurent and M. Seminaroti. The quadratic assignment problem is easy for Robinsonian matrices with Toeplitz structure. *Operations Research Letters*, 43(1):103-109, 2015.

[21] D. F. Robinson. Comparison of labeled trees with valency three. *Journal of Combinatorial Theory, Series B*, 11(2):105–119, 1971.

[22] C. Semple and M. A. Steel. *Phylogenetics*, volume 24. Oxford University Press, 2003.

[23] M. Steel and D. Penny. Distributions of tree comparison metrics—some new results. *Systematic Biology*, 42(2):126–141, 1993.

7.1. **Appendix: Hardness of the RF binary case.** We show here that computing d^*_{RF} is NP-hard even when both tree shapes are binary. Some more notation is needed beforehand. The set of nodes of a tree shape T that have size k is denoted $V_k(T)$. If S is a set of integers, $even(S)$ is the set of even integers in S, whereas $odd(S)$ is the set of odd integers in S. A *cherry* is a the tree shape on 2 leaves, and a *double-cherry* is a tree shape on four leaves that has two cherries. We denote by $chr_1(T)$ the number of cherries in a tree shape T, and by $chr_2(T)$ the number of double-cherries in T. Note that these are not counted separately: each double-cherry in a tree increases $chr_1(T)$ by two. For two tree shapes T_1 and T_2, *joining* T_1 and T_2 consists in creating a new node x and making $r(T_1)$ and $r(T_2)$ children of x, resulting in a new tree shape T'. We say that we *append* T_2 to T_1 when joining T_1 and T_2, and letting the resulting tree be the new T_1.

In this section, for brevity by a matching we mean a consistent cluster matching. Given two tree shapes T_1 and T_2 of the same size, We first find an upper bound UB_μ on $\mu(T_1, T_2)$. Ideally, we would like to be able to find a matching that attains this bound. However, this is not always feasible and, as we will show, it is NP-hard to decide if there is a matching that attains it. Let us proceed with the details.

Clearly, if there is a node $u \in V(T_1)$ such that $sz(u) \neq sz(v)$ for every $v \in V(T_2)$, then u can never be matched in any matching between T_1 and T_2. More generally, let $k \in [n]$ and suppose that $|V_k(T_1)| < |V_k(T_2)|$. Then a matching $\mathcal{M}$ can contain at most $|V_k(T_1)|$ pairs of nodes of size k.

The ideas above imply an upper bound on $\mu(T_1, T_2)$ that can be computed easily: we define

$$UB_\mu(T_1, T_2) = \sum_{i=1}^{n} \min(|V_i(T_1)|, |V_i(T_2)|)$$

It is easy to see that $\mu(T_1, T_2) \leq UB_\mu(T_1, T_2)$. We show that deciding if $\mu(T_1, T_2) = UB_\mu(T_1, T_2)$ is NP-hard, by a reduction from the problem DOMINATING SET IN CUBIC GRAPHS. Given a connected graph G in which every vertex has at most 3 neighbors and an integer k, this problem asks if G contains a dominating set of cardinality at most k. This problem is known to be NP-hard (and in fact, APX-hard) [3].

Our reduction constructs two binary tree shapes T_1 and T_2 from G. Roughly speaking, we would like to re-use the ideas of Theorem 5.5 for the non-binary case. Ideally, we would be able to simply take the $w_i, d_{i,j}, w'_i$ and d'_j nodes from this construction, and simply resolve these non-binary nodes. This, however, will create many new internal nodes, which we may or may not be able to match. The hardness proof presented in that theorem relied on the fact that d'_j nodes could only be matched with $d_{i,j}$ nodes, and w'_i nodes with w_i nodes. Maintaining such a fine-grained control over which nodes can be matched together is much more difficult in the binary case.

We will therefore need to construct sub-tree shapes that contain and avoid prescribed cluster sizes, and that have a certain number of cherries and of double-cherries, and hence we develop some constructive tools before proceeding with our reduction. An important idea behind the reduction is that the nodes of T_1 will mostly be of odd size and those of T_2 mostly of even size, with the exception of some *special* node sizes that are common to both trees. One technical part of the

reduction is that both trees must contain cherries and double-cherries, and we need to keep track of their counts to ensure that the reduction works as desired.

We call a pair of sets of integers (H, M) *well-behaved* if $H \cap M = \emptyset$, $\max(H \cup M) \geq 4$, and $|i - j| \geq 4$ for any distinct $i, j \in H \cup M$. In what follows, H will usually be used for node sizes to "hit", and M for node sizes to "miss".

More precisely, we say that a tree shape T *hits* an integer i if T has a *unique* node v such that $sz(v) = i$, and T *misses* i if $sz(v) \neq i$ for every $v \in V(T)$. For some well-behaved sets (H, M), we say that T is an (H, M)*-tree shape* if T hits h for every $h \in H$ and misses m for every $m \in M$.

An (H, M)-tree shape T is *odd* if, in addition to the above, it also satisfies the condition that for every $v \in V(T)$ such that $sz(v) \notin H$ and $sz(v) > 4$, the size of v is odd. Likewise, T is *even* if for every $v \in V(T)$ such that $sz(v) \notin H$ and $sz(v) > 4$, the size of v is even. The next technical lemma allows us to construct (H, M)-tree shapes while having some control over the cherries and double-cherries. The particular conditions of the lemma will all be of use later on.

Lemma 7.1. *Let (H, M) be well-behaved sets such that $c := \max(H) > \max(M)$ is an odd integer, and let q be an integer with $|M| \leq q \leq \max(|M|, c/4 - 5(|H| + |M| + 1))$.*

Then if $odd(M) = \emptyset$, there exists an even (H, M)-tree shape T_e of size c with $(c + 1)/2 - |odd(H)|$ cherries and q double-cherries. Similarly, if $even(M) = \emptyset$, there exists an odd (H, M)-tree shape T_o of size c with $(c-1)/2-|even(H)|$ cherries and q double-cherries. Moreover, both T_e and T_o can be constructed in polynomial time.

Proof. We make the construction explicit in Algorithm 2, which shows how to construct T_o and T_e (if the *isOdd* input is true, T_o is built, and otherwise T_e is built). Roughly speaking, we start with T a tree shape on two leaves for T_e and three leaves for T_o. While T does not have size c or $c - 1$, we join T with either a cherry or a double-cherry in order to maintain the odd/even parity requirement of the node sizes. Exceptionally, we may append a single leaf when an element of H requires us to hit a particular size, and this is followed by appending another single leaf to return to the desired parity. The values of M are skipped by appending a double-cherry, enforcing at least $|M|$ double-cherries in T. The remaining $q - |M|$ double-cherries are appended whenever H and M allow it during the construction. See Algorithm 2 for details.

Clearly, Algorithm 2 takes polynomial time. We now show the correctness of the procedure, i.e. that the output tree T satisfies the conditions of T_e and T_o. Let w be the number of times the algorithm enters the "while" loop on line 5, and for $i \in [w]$, let s_i be the value of s at the start of the i-th iteration. The lemma's statement is easy to verify if $w = 1$, so we assume $w > 1$. Denote $s_{w+1} := sz(T)$, i.e. the final size of T, and let $S = \{s_1, \ldots, s_w, s_{w+1}\}$. We have $s_1 = 2$ for T_e and $s_1 = 3$ for T_o $s_{i+1} - s_i \in \{2, 4\}$ for any $i \in [w - 1]$. Moreover, each $s_i \in S$ must be even for T_e (except s_{w+1}) and odd for T_o.

We first show that T hits every $h \in H$. Suppose instead that T does not hit some $h \in H$. Let $i \in [w]$ such that $s_i < h$ and $s_{i+1} > h$. Then $s_i < h < s_i + 4$, since $s_{i+1} \leq s_i + 4$. But all cases $s_i + 1, s_i + 2, s_i + 3 \in H$ are explicitly checked by the algorithm: in the first two cases, h gets hit by T, and in the $s_i + 3 \in H$ case, $s_{i+1} = s_i + 2 < h$, contradicting $s_{i+1} > h$ (note that the 'else if' on line 13 cannot

Algorithm 2 Algorithm to construct T_e or T_o

```
1: procedure BUILDTREE(H, M, q, isOdd)
2:     Let T be a cherry
3:     if isOdd = True then Append a single leaf to T
4:     r ← q − |M|   ▷ r is the number of double-cherries that are not enforced by M
5:     while sz(T) ≠ c do
6:         Let s := sz(T)
7:         if s + 1 = c then
8:             Append a single leaf to T
9:         else if s + 1 ∈ H then
10:            Append a single leaf to T, then append another single leaf to T
11:        else if s + 2 ∈ H then
12:            Append a cherry to T
13:        else if s + 2 ∈ M then
14:            Append a double-cherry to T
15:        else if s + 3 ∈ H or s + 4 ∈ M then
16:            Append a cherry to T
17:        else
18:            if r > 0 then
19:                Append a double-cherry to T and decrease r by 1
20:            else
21:                Append a cherry to T
       return T
```

be entered since (H, M) is well-behaved). Observe that since $c \in H$, this implies that $sz(T) = c$ as desired.

We next show that T misses every size in M. Let $m \in M$, and let $i \in [w]$ such that $s_i < m \leq s_{i+1}$. In the case of T_e (resp. T_o), m must be even (resp. odd), since by assumption we have $odd(M) = \emptyset$ (resp. $even(M) = \emptyset$). As s_i has the same parity as m, we must have $m = s_i + 2$ or $m = s_i + 4$. If $m = s_i + 2$, line 13 ensures that T misses m (no other "if" case can be entered by the well-behaved property). If $m = s_i + 4$, none of the cases that append a double-cherry apply (lines 13 and 17), and so $s_{i+1} \leq s_i + 2 = m - 2$, contradicting our assumption that $s_{i+1} \geq m$.

Next, we show that for the T_e case, $chr_1(T) = (c+1)/2 - |odd(H)|$. If suffices to observe that in general, if a binary rooted tree shape T' contains l leaves that do not belong to a cherry, which we will call *single* leaves, then $chr_1(T') = (sz(T') - l)/2$. When constructing T_e, the algorithm appends two single leaves for each element of $odd(H) \setminus \{c\}$, plus a single leaf at the step when the size of T is $c - 1$. The number of appended single leaves is therefore $2(|odd(H)| - 1) + 1$. All other cases append a cherry or a double-cherry, and so $chr_1(T_e) = (c - (2|odd(H)| - 1))/2 = (c+1)/2 - |odd(H)|$, as desired. For the T_o case, two single leaves are appended for each element of $even(H)$ (notice that c is odd, and so $c \notin even(H)$) and a single leaf is appended before the **while** loop, leading to $chr_1(T_o) = (c-(2|even(H)|+1))/2 = (c-1)/2 - |even(H)|$ in the same manner.

It only remains to show that $chr_2(T) = q$. First note that in both the T_e and T_o cases, the "if" case on line 13 is entered exactly $|M|$ times, and so $|M|$ double-cherries are due to line 14. If $q = |M|$, we are done, so assume $q > |M|$. Let us count

the number of times t that the "else" statement is entered on line 17. We show that $t \geq c/4 - 5(|H| + |M| + 1)$, which proves the desired result since r ensures that the exact number of double-cherries are appended. Observe first that $s_1 \geq 3$, $s_{|S|} = c$ and $s_i \leq s_{i-1} + 4$ for each $2 \leq i \leq |S|$. Hence, $|S| \geq \lfloor (c-3)/4 \rfloor$. Also, the case on line 17 is entered for each $s_i \in S$ such that $\{s_i, s_i + 1, \ldots, s_i + 4\} \cap H \cup M = \emptyset$. For each $h \in H \cup M$, there are at most 5 members of $s_i \in S$ such that $s_i + j \in H \cup M$, $0 \leq j \leq 4$. Therefore, the number of members of S that do satisfy the condition for entering line 17 is at least $|S| - 5(|H| + |M|) \geq \lfloor (c-3)/4 \rfloor - 5(|H| + |M|) \geq c/4 - 5(|H| + |M| + 1)$. We have verified every required property for T_e and T_o, concluding the proof.

□

Note that in the tree shape T constructed in Lemma 7.1, every node of size 4 is the root of a double-cherry, unless $T = T_o$ and $4 \in H$. The reader should bear in mind that this particular case will never occur, and we will assume that in what follows, every subtree of size 4 is a double-cherry.

We are now ready to describe our reduction. Let (G, k) be an instance of DOMINATING SET IN CUBIC GRAPHS, and let $(v_1, \ldots, v_n)$ be an (arbitrary) ordering of $V(G)$. Since a vertex can only dominate four vertices (its neighbors plus itself), we may assume that $k \geq n/4$.

For each $i \in [n]$, let $r_i = 4i + 1$ and $w_i = 80n + 4i + 5$. We also let $S = \{r_1, \ldots, r_n, w_1, \ldots, w_n\}$, and call an element $s \in S$ a *special size*. Moreover, let $\hat{r} = 20n + 1$. For $i \in [n]$, put $R_i = \{r_1, r_2, \ldots, r_i\}$ and $R_0 = \emptyset$. Note that for each $i \in [n]$, $(\{r_i\}, R_{i-1})$ is well-behaved. Our goal is to construct T_1 and T_2 from (G, k) such that the only possible nodes that can be matched are either cherries, double-cherries, or have a special size. We will have, with hindsight, $sz(T_1) = sz(T_2) = 200n^8 + 100n^4 + 82n^2 + 7n + k(84n+5)$. We start by constructing the T_1 tree shape, for which we need to define five types of sub-tree shapes, as follows. Figure 7 provides an illustration of the D_i and W_i trees, and Figure 8 of the T_1 and T_2 trees.

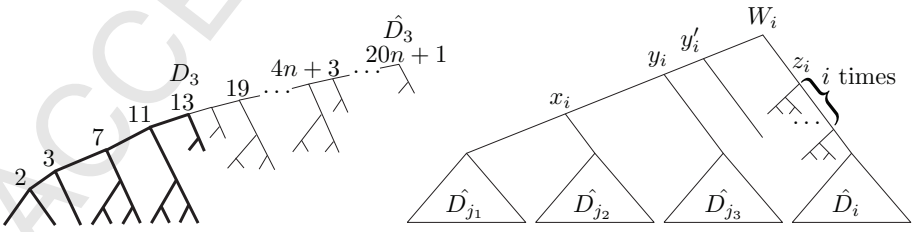

FIGURE 7. On the left, an illustration of D_i (with heavy edges) and $\hat{D}_i$ with $i = 3$. The labels correspond to the sizes of the internal nodes. Here, D_3 is an odd $(\{13\}, \{5, 9\})$-tree shape. Then $\hat{D}_3$ is obtained from D_3 by first appending a cherry, then $n - 3$ double-cherries, and finally $8n - 1$ cherries. Note that $\hat{D}_3$ hits $r_3 = 13$ and misses $\{5, 9, 17, 21, \ldots, 4n + 1\}$. On the right, an illustration of a W_i tree.

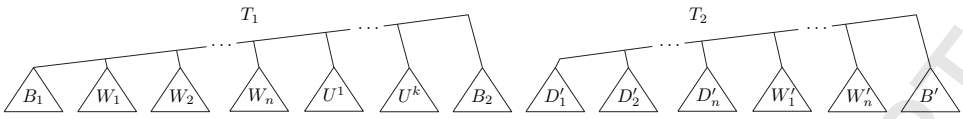

FIGURE 8. The T_1 and T_2 tree shapes.

The D_i and $\hat{D}_i$ tree shapes: For each $i \in [n]$, let D_i be an odd $(\{r_i\}, R_{i-1})$-tree shape of size $r_i = 4i + 1$ with $chr_1(D_i) = (r_i - 1)/2 = 2i$ and $chr_2(D_i) = |R_{i-1}| = i - 1$ (which can be constructed in polynomial time, by Lemma 7.1).

Then, obtain the tree shape $\hat{D}_i$ of size $\hat{r}$ from D_i by first joining D_i with a cherry, then successively appending a double-cherry $n - i$ times, resulting in a tree shape of size $4i + 1 + 2 + 4(n - i) = 4n + 3$. Then join this tree shape with a cherry $8n - 1$ times, resulting in $\hat{D}_i$ (of size $20n + 1 = \hat{r}$). See Figure 7. Note that $\hat{D}_i$ is an odd $(\{r_i\}, R_n \setminus \{r_i\})$-tree shape of size $\hat{r}$. Also, $chr_1(\hat{D}_i) = 10n$ and $chr_2(\hat{D}_i) = i - 1 + (n - i) = n - 1$.

The W_i tree shapes: for each $i \in [n]$, we construct the tree shape W_i as follows. Let $v_{j_1}, v_{j_2}, v_{j_3}$ be the neighbors of v_i in G. First take a copy of $\hat{D}_{j_1}$ and a copy of $\hat{D}_{j_2}$, and join their roots under a common parent x_i. Take a copy of $\hat{D}_{j_3}$ and join its root and x_i under a common parent y_i. Then join y_i and a single leaf under a common parent y'_i. Finally take a copy of $\hat{D}_i$, append a double-cherry i times to it to obtain a new tree shape rooted at a node z_i, and connect y'_i and z_i under a common parent, which is the root of W_i (see Figure 7).

For later use, we will need two sizes that are missed by every W_i. Let $w^* := 3\hat{r}+2$ and $w^{**} := 3\hat{r} + 4$. It is straightforward to verify that W_i misses w^* and w^{**}.

The following properties hold for each W_i tree:

(1) $sz(W_i) = 4\hat{r} + 4i + 1 = 80n + 4i + 5 = w_i$;
(2) $sz(x_i) = 2\hat{r}, sz(y_i) = 3\hat{r}, sz(y'_i) = 3\hat{r} + 1$ and $sz(z_i) = \hat{r} + 4i$. Moreover, x_i and y'_i are the only two nodes of even size in W_i, except cherries and double-cherries;
(3) $chr_1(W_i) = 4 \cdot 10n + 2i = 40n + 2i$;
(4) $chr_2(W_i) = 4(n - 1) + i = 4n + i - 4$;
(5) W_i hits $r_{j_1}, r_{j_2}, r_{j_3}, r_i$ and misses every other element of R_n;

The U sub-tree shape: we construct a tree shape U inductively as follows: first, U_1 is an odd $(\{2\hat{r}, 3\hat{r} + 1\}, R_n)$-tree of size $sz(W_1) = 80n + 9$ with $chr_1(U_1) = chr_1(W_1) = 40n + 2$ and $chr_2(U_1) = chr_2(W_1) = 4n - 3$. Notice that U_1 has the same two even node sizes as every W_i tree shape. One can check that U_1 satisfies all the conditions of Lemma 7.1, and hence can be constructed.

Then, for $1 < i \leq n$, U_i is obtained by joining U_{i-1} with a double-cherry. Since $sz(W_i) = sz(W_{i-1})+4$, $chr_1(W_i) = chr_1(W_{i-1})+2$ and $chr_2(W_i) = chr_2(W_{i-1})+1$, U_i has the same size, number of cherries and double-cherries as W_i. We let $U = U_n$, noting that $sz(U) = sz(W_n)$.

The B_1 and B_2 sub-tree shapes: the B_1 tree shape is simply an odd $(\emptyset, S)$-tree of size $100n^4$ (resp. of size $200n^8$). We are not concerned with their number of cherries or double-cherries.

Finally, T_1 is obtained by taking a caterpillar shape on $n+k+2$ leaves $\{\ell_1, \ldots, \ell_{n+k+2}\}$, and replacing ℓ_1 by the B_1 tree shape, replacing each ℓ_{i+1} by the W_i tree shape for $i \in [n]$, each ℓ_{n+1+j} by a copy U^j of U for $j \in [k]$, and ℓ_{n+k+2} by the B_2 sub-tree shape (see Figure 8). We have $sz(T_1) = 200n^8 + 100n^4 + k \cdot sz(U) + \sum_{i=1}^{n} sz(W_i) = 200n^8 + 100n^4 + 82n^2 + 7n + k(84n + 5)$, as predicted. Denote by I the set of even sizes of nodes in T_1 greater than 4, i.e. $I = \{sz(u) : u \in V(T), sz(u) > 4 \text{ and } sz(u) \text{ is even}\}$. Note that the W_i and U tree shapes, together, contribute to only two sizes in I (namely $2\hat{r}$ and $3\hat{r}+1$), and the only other even size nodes must be ancestors of $r(B_1)$. Hence if $i \in I \setminus \{2\hat{r}, 3\hat{r}+1\}$, then $i = \theta(n^4)$ (unless $i = sz(r(T_1))$, in which case $i = \theta(n^8)$). Moreover, it is not hard to see that $|i_1 - i_2| \geq 4$ for any distinct $i_1, i_2 \in I$.

Now we may construct T_2. Recall that $S = \{r_1, \ldots, r_n, w_1, \ldots, w_n\}$ is the set of special sizes. For each $s \in S$, we want T_2 to contain exactly one node of size s, such that any pair of nodes having a special size are incomparable.

For each $i \in [n]$, let $R'_i = \{4, 8, \ldots, 4i\}$ and $R'_0 = \emptyset$. Then let D'_i be an even $(\{r_i\}, R'_{i-1})$-tree of size r_i with $chr_1(D'_i) = chr_1(D_i) = 2i$ and $chr_2(D'_i) = i - 1$ (Lemma 7.1 ensures that D'_i can be constructed). Then let W'_i be an even $(\{w_i, w^*, w^{**}\}, \{2\hat{r}, 3\hat{r}+1\})$-tree of size $sz(W_i) = w_i = 80n+4i+5$ with $chr_1(W'_i) = chr_1(W_i) = 40n + 2i$ and $chr_2(W'_i) = chr_2(W_i) = 4n + i - 4$. Again, we invoke Lemma 7.1 for W'_i (which was in fact the sole purpose of w^* and w^{**}, as they control $chr_1(W'_i)$). We next build a sub-tree shape B' so that T_2 attains the same size as T_1. That is, let $s_{B'} := sz(T_1) - \sum_{i=1}^{n}(sz(D'_i) + sz(W'_i))$ (observe that $s_{B'}$ is clearly above 0). Letting $I' = \{i \in I : i < s_{B'}\}$, we let B' be an even $(\{s_{B'}\}, I')$-tree of size $s_{B'}$ (to see that $(\{s_{B'}\}, I')$ is well-behaved, we have argued that elements of I differ by at least 4, and for $s_{B'}$, we note that $s_{B'} = \theta(n^8)$ whereas $i = \theta(n^4)$ for each $i \in I'$). The tree shape T_2 is obtained by taking a caterpillar shape on $2n+1$ leaves $\{\ell'_1, \ldots, \ell'_{2n+1}\}$, replacing ℓ'_i by the D'_i sub-tree shape and ℓ'_{n+i} by the W'_i sub-tree shape for each $i \in [n]$, then replacing ℓ_{2n+1} by the B' sub-tree shape.

We are finally done with the construction. First, we calculate the upper bound on $\mu(T_1, T_2)$.

Lemma 7.2. $UB_\mu(T_1, T_2) = sz(T_1)+2n+\min(chr_1(T_1), chr_1(T_2))+\min(chr_2(T_1), chr_2(T_2))$.

Proof. The $sz(T_1)$ term in UB_μ is due to the fact that each leaf of T_1 can be matched. Now, by construction, there are $2n$ special sizes in S and for each $s \in S$, $|V_s(T_2)| = 1$ and $|V_s(T_1)| \geq 1$. This implies

$$\begin{aligned} UB_\mu(T_1, T_2) &= \sum_{i=1}^{n} \min(|V_i(T_1)|, |V_i(T_2)|) \\ &\geq sz(T_1) + 2n + \min(chr_1(T_1), chr_1(T_2)) + \min(chr_2(T_1), chr_2(T_2)) \end{aligned}$$

To see that this is also an upper bound on $UB_\mu(T_1, T_2)$, we must argue that T_1 and T_2 share no nodes of the same size greater than 1, except the cherries, double-cherries and the nodes having a special size. First observe that every node of size 4 in T_1 and T_2 is the root of a double-cherry. Thus we may restrict our attention to the sizes greater than 4. The $\hat{D}_i, W_i, U^i, B_1$ and B_2 sub-tree shapes of T_1 are odd, whereas the D'_i, W'_i and B' sub-tree shapes of T_2 are even, and are constructed so that the sets of sizes of their nodes intersect only on S. Therefore, if $sz(x_1) = sz(x_2)$ for some non-root $x_1 \in V(T_1)$ and non-root $x_2 \in V(T_2)$, x_1 must be an ancestor

of $r(B_1)$ and x_2 an ancestor of $r(D'_1)$. But by the placement of B_1 and B', every ancestor of $r(B_1)$ has size at least $100n^4$, whereas the non-root ancestors of $r(D'_1)$ have size $O(n^2)$. Hence $sz(x_1) = sz(x_2)$ is not possible.

□

Theorem 7.3. *Deciding if $\mu(T_1, T_2) = UB_\mu(T_1, T_2)$ is NP-complete.*

Proof. The problem is in NP, since UB_μ is easy to compute and a matching can be provided as a certificate, which can easily be verified in polynomial time. As for hardness, let (G, k) be an instance of dominating set on cubic graphs, and let T_1 and T_2 be the corresponding tree shapes constructed as above. We show that G has a dominating set of size k if and only if $\mu(T_1, T_2) = UB_\mu(T_1, T_2)$.

($\Rightarrow$): Let $X = \{v_{d_1}, \ldots, v_{d_k}\}$ be a dominating set of G (we may assume that $|X| = k$, as if $|X| < k$ we may add arbitrary vertices into X). We construct a matching $\mathcal{M}$ of the desired size, the main idea being to match the W'_i and D'_i roots to a set of incomparable nodes in T_1. For each $v_{d_i} \in X$, $i \in [k]$, match the $r(W'_{d_i})$ node of T_2 with the root of the $U^i_{d_i}$ sub-tree shape in T_1, which is the unique sub-tree shape of size w_{d_i} in the U^i sub-tree shape of T_1 (since there are k copies of U, each v_{d_i} can be matched in this manner). Note that by construction, $chr_1(U^i_{d_i}) = chr_1(W'_{d_i})$ and $chr_2(U^i_{d_i}) = chr_2(W'_{d_i})$. It is straightforward to see that the cherries and double-cherries of the two sub-tree shapes can all be matched in a consistent manner. Then for each $v_i \notin X$, we match $r(W'_i)$ with $r(W_i)$. As before, $chr_1(W_i) = chr_1(W'_i)$ and $chr_2(W_i) = chr_2(W'_i)$, and so all the cherries and double-cherries of the two sub-tree shapes can be matched. So far, all the W'_i roots are matched in a consistent manner, and the nodes $\{r(W_i) : v_i \in X\}$ of T_1 are unmatched, leaving their D_j sub-tree shapes available. Thus, for each $v_j \in V(G)$, let $v_h \in X$ be a vertex dominating v_j (if $v_j \in X$, it dominates itself and we let $v_j = v_h$). We match $r(D'_j)$ with the root of the D_j sub-tree shape of T_1 that lies within the W_h tree shape. Once again, $chr_1(D_j) = chr_1(D'_j)$ and $chr_2(D_j) = chr_2(D'_j)$, and their cherries/double-cherries can all be matched. So far, all nodes of T_1 that have been matched are incomparable, ensuring consistency.

In order to attain $UB_\mu(T_1, T_2)$, it only remains to match the roots of cherries and double-cherries that do not lie under a matched node. For these, it is not hard to see that we can compute a maximum matching between the double-cherries first (and match their descending cherries together), then a maximum matching between the cherries that have not been matched in the preceding step. In this manner, we find a cluster matching of size $sz(T_1) + 2n + \min(chr_1(T_1), chr_1(T_2)) + \min(chr_2(T_1), chr_2(T_2))$.

($\Leftarrow$): Suppose that $\mu(T_1, T_2) = UB_\mu(T_1, T_2)$, and let $\mathcal{M}$ be a matching between T_1 and T_2. It follows from Lemma 7.2 that every root of the D'_i and W'_i sub-tree shapes of T_2 must be matched, as matching the special size nodes of T_2 is necessary to attain $UB_\mu(T_1, T_2)$. Each D'_i root can only be matched with a D_i root in T_1, with $r(D_i)$ being a descendant of $r(W_j)$ for some $j \in [n]$. For $i \in [n]$, let d_i be the node of T_1 matched with $r(D'_i)$ in $\mathcal{M}$, and let $w(D'_i)$ be the unique index j such that $r(W_j)$ is an ancestor of d_i in T_1. Let $X = \{v_j \in V(G) : j = w(D'_i)$ for some $i \in [n]\}$. We claim that X is a dominating set of size at most k. Since, for $i \in [n]$, $r(W_j)$ is an ancestor of d_i if and only if $v_i v_j \in E(G)$ or $i = j$, each $v_i \in V(G)$ must have a neighbor in X or must itself be in X, and thus X is indeed a dominating set. Now suppose that $|X| > k$. Note that if some D'_i root is matched with d_i with

ancestor $r(W_j)$, then W'_j cannot be matched with W_j (because $r(D'_i)$ and $r(W'_j)$ are incomparable, whereas d_i and $r(W_j)$ are not). Then the only matching options for the root of W'_j are in the U^h sub-tree shapes in T_1, $h \in [k]$. In other words, for each $v_i \in X$, $r(W'_i)$ is matched with a node in the U^h copy for some $h \in [k]$. Since $|X| > k$, there must be two distinct $v_{i_1}, v_{i_2} \in X$ such that both $r(W'_{i_1})$ and $r(W'_{i_2})$ are matched with a sub-tree shape of the same U^h copy. But this is not possible since the roots of these sub-tree shapes are not incomparable. This shows that $|X| \leq k$, concluding the proof. □

Department of Mathematics and Statistics, University of Ottawa, Ottawa, Canada
E-mail address: mlafond2@uOttawa.ca

Département d'informatique et de recherche opérationnelle, Université de Montréal, Québec, Canada
E-mail address: mabrouk@iro.umontreal.ca

School of Computing Sciences, University of East Anglia, Norwich, UK
E-mail address: K.Huber@uea.ac.uk

School of Computing Sciences, University of East Anglia, Norwich, UK
E-mail address: v.moulton@uea.ac.uk