JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 19, Number 9, 2012
© Mary Ann Liebert, Inc.
Pp. 1073–1088
DOI: 10.1089/cmb.2012.0008

Polynomial-Time Algorithms for Building a Consensus MUL-Tree

YUN CUI[1], JESPER JANSSON,[2] and WING-KIN SUNG[1,3]

ABSTRACT

A multi-labeled phylogenetic tree, or MUL-tree, is a generalization of a phylogenetic tree that allows each leaf label to be used many times. MUL-trees have applications in biogeography, the study of host–parasite cospeciation, gene evolution studies, and computer science. Here, we consider the problem of inferring a *consensus MUL-tree* that summarizes a given set of conflicting MUL-trees, and present the first polynomial-time algorithms for solving it. In particular, we give a straightforward, fast algorithm for building a *strict consensus MUL-tree* for any input set of MUL-trees with identical leaf label multisets, as well as a polynomial-time algorithm for building a *majority rule consensus MUL-tree* for the special case where every leaf label occurs at most twice. We also show that, although it is NP-hard to find a majority rule consensus MUL-tree in general, the variant that we call the *singular majority rule consensus MUL-tree* can be constructed efficiently whenever it exists.

Key words: algorithm, cluster, computational complexity, consensus tree, multi-labeled phylogenetic tree multiset, MUL-tree.

1. INTRODUCTION

To describe treelike evolutionary history, scientists often use a data structure known as the *phylogenetic tree* (Felsenstein, 2004; Sung, 2010). Over the years, many variants of phylogenetic trees (e.g., rooted or unrooted, with or without edge weights, bounded or unbounded degrees, ordered or unordered, etc.) have been introduced and successfully employed in various contexts. A *consensus tree* is a phylogenetic tree that summarizes the branching structure contained in a given set of (conflicting) phylogenetic trees. Different types of consensus trees, along with fast algorithms for constructing them, have been developed since the 1970s and are widely used by biologists today (see, for example, the surveys in Bryant, 2003; Felsenstein, 2004; and Sung, 2010).

In traditional applications, phylogenetic trees have usually been *distinctly* leaf labeled, and, in fact, the computational efficiency of most existing methods for constructing and comparing phylogenetic trees

Parts of this article appeared in preliminary form in *Proceedings of the 22nd International Symposium on Algorithms and Computation* (ISAAC 2011), volume 7074 of *Lecture Notes in Computer Science,* pages 744–753, Springer-Verlag, Berlin, 2011.

[1]School of Computing, National University of Singapore, Singapore.
[2]Laboratory of Mathematical Bioinformatics, Kyoto University, Gokasho, Uji, Kyoto, Japan.
[3]Genome Institute of Singapore, Genome, Singapore.

implicitly depends on this uniqueness property. The *multi-labeled phylogenetic tree*, or *MUL-tree* for short, is a natural generalization of the standard phylogenetic tree model that allows the same leaf label to be used more than once in a single tree structure. For some examples, see Figures 2, 3, 4, 5, 6, and 8. MUL-trees have a number of applications in different research fields, such as biogeography (see, e.g., Ganapathy et al., 2006; Minaka, 1990; and Chapter 6 of Nelson and Platnick, 1981); the study of host–parasite cospeciation (Page, 1993); gene evolution studies (Fellows et al., 2003; Lott et al., 2009b; Page, 1994; Scornavacca et al., 2011), and computer science (see the references in Huber et al., 2011).

Combining the concepts of a consensus tree and a MUL-tree leads to the computational problem of building a *consensus MUL-tree* from an input set of MUL-trees. It was first addressed by Lott et al. (2009b). Their motivation came from a more general problem related to reconstructing complex evolutionary scenarios involving so-called polyploid species. Here, the input is a collection of gene trees where the same species name can label more than one leaf (i.e., MUL-trees), and the output should be a leaf-labeled directed acyclic graph called a *phylogenetic network*, in which each species name appears once only. Lott et al. (2009a) suggested that rather than inferring a phylogenetic network directly, it may be easier to first reconcile the input into a single MUL-tree and then apply an algorithm from Huber et al. (2006) that is guaranteed to output a network with the minimum number of non tree nodes. For this purpose, Lott et al. (2009b) presented a method for constructing a greedy kind of consensus MUL-tree that uses the same basic strategy as the well-known *greedy consensus tree method* (Bryant, 2003; Sung, 2010) for single-labeled phylogenetic trees. A serious disadvantage of their method is that its running time is exponential in the size of the input, and indeed, according to the discussion in Lott et al. (2009a), the method probably needs to be improved to deal with datasets from new sequencing technologies in the near future. A recent article (Huber et al., 2012) incorporates a fixed-parameter algorithm from Section 5 of Huber et al. (2008) to obtain a faster and more practical method for building a greedy consensus MUL-tree, but its worst-case running time remains exponential.

It is an important open problem to identify alternative types of (informative) consensus MUL-trees that can be computed more efficiently than Huber et al. (2012) and Lott et al. (2009b). In this article, we investigate the computational complexity of inferring three types of consensus MUL-trees, which we call the *strict consensus MUL-tree*, the *majority rule consensus MUL-tree*, and the *singular majority rule consensus MUL-tree*, and derive a number of positive and negative results. To our knowledge, the new algorithms developed here are the first ever polynomial-time algorithms for building a consensus MUL-tree of any kind.

1.1. Organization of the article

This article is organized as follows. Section 2 provides the formal definitions and terminology used throughout the text, and Section 3 highlights some key properties of strict majority rule and singular majority rule consensus MUL-trees. Next, we explain how to construct a strict consensus MUL-tree in polynomial time in Section 4. Then, Section 5 proves that constructing a majority rule consensus MUL-tree is NP-hard, even if restricted to instances with three input MUL-trees in which every leaf label occurs at most three times. On the positive side, Section 6 gives a polynomial-time algorithm for constructing a majority rule consensus MUL-tree for the special case where every leaf label occurs at most twice. Although constructing a majority rule consensus MUL-tree is NP-hard in general, the variant, which we call the singular majority rule consensus MUL-tree, admits a polynomial-time algorithm, described in Section 7. Finally, open problems and possible extensions are discussed in Section 8.

From here on, $\mathcal{T}$ is assumed to be an input set of MUL-trees such that every $T_i \in \mathcal{T}$ has the same leaf label multiset L. We define $k = |\mathcal{T}|$ and $n = |L|$. Also, we let q equal the number of distinct elements in L. In other words, $q \leq n$. Furthermore, we define $m = \max_{\ell \in L} mult_L(\ell)$, where $mult_L(\ell)$ is the number of occurrences of ℓ in the multiset L, and call m the *multiplicity of L*. Our new results for consensus MUL-trees, along with previously known results for single-labeled phylogenetic trees (corresponding to the case $m = 1$), are summarized in Figure 1.

2. DEFINITIONS

2.1. MUL-trees

A MUL-tree is a rooted, unordered, leaf-labeled tree in which every internal node has at least two children. Importantly, in a MUL-tree the same label may be used for more than one leaf. Figure 2 shows an

Strict consensus

	$k \geq 2$
$m = 1$	Always exists; always unique; $O(nk)$ time. (Day, 1985)
$m \geq 2$	Always exists; may not be unique; $O(nqk)$ time. (Sections 3 and 4)

Majority rule consensus

	$k = 2$	$k \geq 3$
$m = 1$	Always exists; always unique; $O(n)$ time. (Day, 1985)	Always exists; always unique; $O(n^2 + nk^2)$ time. (Wareham, 1985)
$m = 2$	Always exists; may not be unique; $O(nq)$ time. (Sections 3 and 4)	May not exist; may not be unique; $O(n^2k + nk^2)$ time. (Section 6)
$m \geq 3$	Always exists; may not be unique; $O(nq)$ time. (Sections 3 and 4)	May not exist; may not be unique; NP-hard. (Sections 3 and 5)

Singular majority rule consensus

	$k = 2$	$k \geq 3$
$m = 1$	Always exists; always unique; $O(n)$ time. (Day, 1985)	Always exists; always unique; $O(n^2 + nk^2)$ time. (Wareham, 1985)
$m \geq 2$	Always exists; always unique; $O(n^3)$ time. (Sections 3 and 7)	May not exist; always unique; $O(n^3k + nk^2)$ time. (Sections 3 and 7)

FIG. 1. Summary of the computational complexity of building a strict consensus, a majority rule consensus, and a singular majority rule consensus, multi-labeled phylogenetic tree (MUL-tree) of $\mathcal{T}$. For $k = 2$, a strict consensus MUL-tree and a majority rule consensus MUL-tree are equivalent according to Lemma 2 in Section 3. For $m = 1$, a majority rule consensus MUL-tree and a singular majority rule consensus MUL-tree are equivalent because every cluster is singular.

example. The multiset of all leaf labels that occur in a MUL-tree T is denoted by $\Lambda(T)$. For any multiset X and element x, the *multiplicity of x in X* is the number of occurrences of x in X and is denoted by $mult_X(x)$. Below, the multiset union operation is expressed by the symbol $\uplus$.

Let L be a multiset and let T be a MUL-tree with $\Lambda(T) = L$. If $mult_L(\ell) = 1$ for all $\ell \in L$, then T is a *single-labeled phylogenetic tree*. Next, any submultiset C of L is called a *cluster* of L, and if $|C| = 1$ then C is called *trivial*. Let $V(T)$ be the set of all nodes in T. For any $u \in V(T)$, the subtree of T rooted at u is written as $T[u]$, and $\Lambda(T[u])$ is referred to as *the cluster associated with u*. The *cluster collection of T* is defined as the multi-set $\mathcal{C}(T) = \biguplus_{u \in V(T)} \{\Lambda(T[u])\}$. When a cluster C belongs to $\mathcal{C}(T)$, we say that T *contains* C or that C

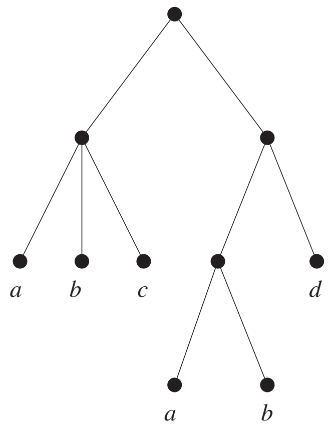

FIG. 2. A MUL-tree T with leaf label multiset $\Lambda(T) = \{a, a, b, b, c, d\} = L$ and cluster collection $\mathcal{C}(T) = \{\{a\}, \{a\}, \{b\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, b, c\}, \{a, b, d\}, L\}$.

occurs in T. Using the notation above, the multiplicity of a cluster C in a cluster collection $\mathcal{C}(T)$ is written as $mult_{\mathcal{C}(T)}(C)$. Thus, when a cluster C does not occur in a MUL-tree T, we have $mult_{\mathcal{C}(T)}(C)=0$.

2.2. Three types of consensus MUL-trees

Let $\mathcal{T}=\{T_1, T_2, \ldots, T_k\}$ be a given set of MUL-trees satisfying $\Lambda(T_1)=\Lambda(T_2)=\ldots=\Lambda(T_k)=L$. Two popular types of consensus trees for single-labeled phylogenetic trees are the *strict consensus tree* (Sokal and Rohlf, 1981) and the *majority rule consensus tree* (Margush and McMorris, 1981). We extend their definitions to MUL-trees as follows:

- A *strict consensus MUL-tree of* $\mathcal{T}$ is a MUL-tree T such that $\Lambda(T) = L$ and $\mathcal{C}(T)=\bigcap_{i=1}^{k}\mathcal{C}(T_i)$, where $\bigcap$ is the intersection of multisets. Formally, for every $C\in\mathcal{C}(T)$, $mult_{\mathcal{C}(T)}(C)=\min_{1\le i\le k} mult_{\mathcal{C}(T_i)}(C)$. (In other words, the number of times that a particular cluster C occurs in T equals the minimum number of times that C occurs in each of $T_1, T_2, \ldots, T_k$.)
- A cluster that occurs in more than $k/2$ of the MUL-trees in $\mathcal{T}$ is called a *majority cluster*. A *majority rule consensus MUL-tree of* $\mathcal{T}$ is a MUL-tree T such that $\Lambda(T) = L$ and $\mathcal{C}(T)$ consists of all majority clusters, and for any $C\in\mathcal{C}(T)$, $mult_{\mathcal{C}(T)}(C)$ equals the largest integer j such that the following condition holds: $|\{T_i : mult_{\mathcal{C}(T_i)}(C)\ge j\}| > k/2$.

Next, we introduce a new kind of consensus tree. For any MUL-tree T, a cluster C in $\mathcal{C}(T)$ is called *singular* if $C\uplus C\nsubseteq\Lambda(T)$. Note that if $\mathcal{C}\in\mathcal{C}(T)$ is singular then $mult_{\mathcal{C}(T)}(C)=1$ (but not the other way around; e.g., if $mult_{\mathcal{C}(T)}(\{a,b\})=1$ and $\Lambda(T)=\{a,a,b,b,\ldots\}$ then $\{a,b\}$ is not singular).

- A *singular majority rule consensus MUL-tree of* $\mathcal{T}$ is a MUL-tree T such that $\Lambda(T) = L$ and $\mathcal{C}(T)$ consists of: (1) every trivial cluster that occurs in all of $T_1, T_2, \ldots, T_k$; and (2) every singular cluster that occurs in more than $k/2$ of the MUL-trees in $\mathcal{T}$.

2.3. The delete operation

Define the *delete* operation on any nonroot, internal node u in a MUL-tree T as letting all children of u become children of the parent of u, and then removing u and the edge between u and its parent (Figure 3). Note that any delete operation on a node u in T removes one occurrence of a cluster from the cluster collection $\mathcal{C}(T)$, namely $\Lambda(T[u])$, without affecting the other clusters.

3. PRELIMINARIES

It is possible for two non-isomorphic MUL-trees to have identical cluster collections. See T_1 and T_2 in Figure 4 for an example. This property was first observed by Ganapathy et al. (2006) for unrooted MUL-trees, and their example was later simplified by Huber et al. (2008). (The example given here is the same as Fig. 1b,c in Huber et al., 2008, adapted to rooted MUL-trees.)

We immediately have:

Lemma 1. *Let* $\mathcal{T}=\{T_1, T_2, \ldots, T_k\}$ *be a set of MUL-trees with* $\Lambda(T_1)=\Lambda(T_2)=\ldots=\Lambda(T_k)=L$. *A strict consensus MUL-tree of* $\mathcal{T}$ *always exists but might not be unique.*

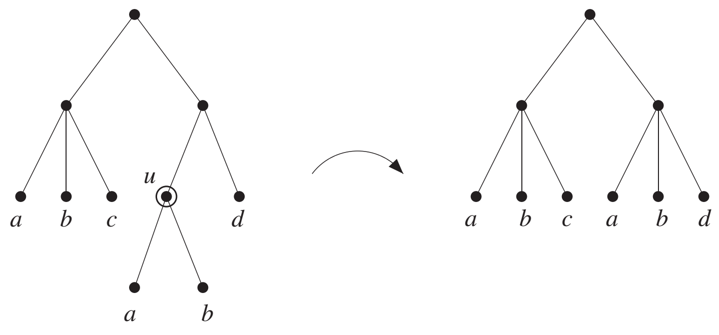

FIG. 3. Let T be the MUL-tree on the left and let u be the marked node in T. In this example, $\Lambda(T[u]) = \{a, b\}$ and applying the delete operation on node u removes the only occurrence of the cluster $\{a, b\}$ from $C(T)$.

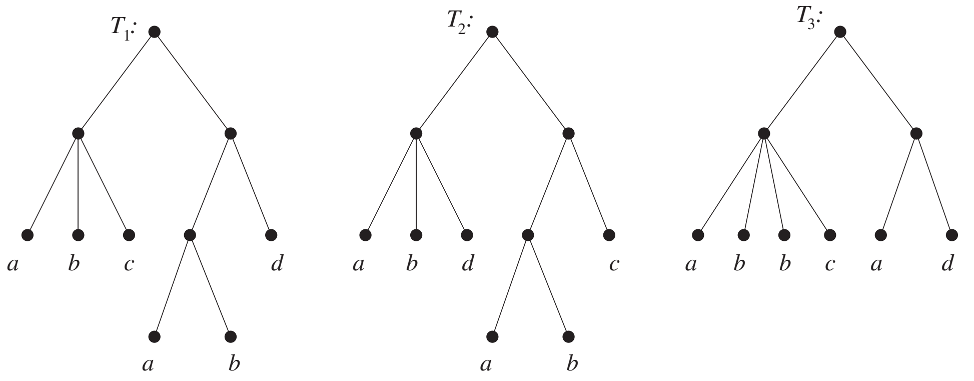

FIG. 4. Let T_1, T_2, T_3 be the three MUL-trees shown above with $\Lambda(T_1) = \Lambda(T_2) = \Lambda(T_3) = \{a, a, b, b, c, d\} = L$. Then $T_1 \neq T_2$ although $\mathcal{C}(T_1)=\mathcal{C}(T_2)=\{\{a\}, \{a\}, \{b\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, b, c\}, \{a, b, d\}, L\}$. Each of T_1 and T_2 is a strict consensus MUL-tree of $\{T_1, T_2\}$, and also a majority rule consensus MUL-tree of $\{T_1, T_2, T_3\}$. (However, neither T_1 nor T_2 is a singular majority rule consensus MUL-tree of $\{T_1, T_2\}$ or $\{T_1, T_2, T_3\}$ since the cluster $\{a, b\}$ is not singular.)

Proof. To prove the existence, let $Z = \bigcap_{i=1}^{k} \mathcal{C}(T_i)$ (using the intersection of multisets), and construct a MUL-tree T with $\Lambda(T) = L$ and $\mathcal{C}(T)=Z$ as follows. Set T equal to T_1. Since $Z \subseteq \mathcal{C}(T)$, we have $mult_Z(C) \leq mult_{\mathcal{C}(T)}(C)$ for every $C \in \mathcal{C}(T)$. For each $C \in \mathcal{C}(T)$, arbitrarily select $(mult_{\mathcal{C}(T)}(C) - mult_Z(C))$ nodes u in T with $\Lambda(T[u]) = C$ and perform the delete operation (see Section 2.3) on them. This yields a MUL-tree T with $mult_Z(C) = mult_{\mathcal{C}(T)}(C)$ for every $C \subseteq L$ and $\Lambda(T) = L$, so T is a strict consensus MUL-tree of $\mathcal{T}$.

To prove the nonuniqueness, consider $\mathcal{T} = \{T_1, T_2\}$ in Figure 4. Each of T_1 and T_2 is a strict consensus MUL-tree of the set $\mathcal{T} = \{T_1, T_2\}$. ∎

Next, we consider majority rule consensus MUL-trees.

Lemma 2. *Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a set of MUL-trees with $\Lambda(T_1) = \Lambda(T_2) = \ldots = \Lambda(T_k) = L$.*

- *If $k = 2$, then a majority rule consensus MUL-tree of $\mathcal{T}$ always exists but might not be unique.*
- *If $k \geq 3$, then a majority rule consensus MUL-tree of $\mathcal{T}$ might not exist and might not be unique.*

Proof. For the case $k = 2$, a cluster occurs in more than $k/2$ of the MUL-trees in $\mathcal{T}$ if and only if it occurs in both MUL-trees in $\mathcal{T}$. Hence, for $k = 2$, a majority rule consensus MUL-tree of $\mathcal{T}$ is equivalent to a strict consensus MUL-tree of $\mathcal{T}$, and the result follows from Lemma 1.

For $k \geq 3$, the nonuniqueness follows from the example in Figure 4, where each of T_1 and T_2 is a majority rule consensus MUL-tree of $\{T_1, T_2, T_3\}$. The nonexistence follows from the set $\{T_4, T_5, T_6\}$ in Figure 5. ∎

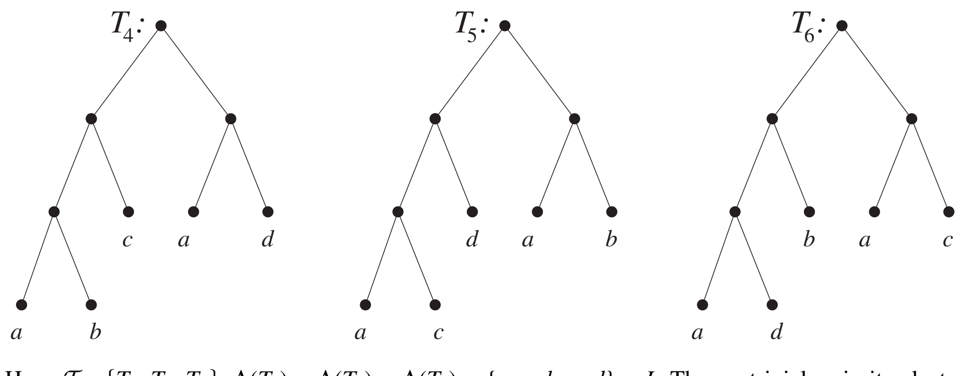

FIG. 5. Here, $\mathcal{T} = \{T_4, T_5, T_6\}$, $\Lambda(T_4) = \Lambda(T_5) = \Lambda(T_6) = \{a, a, b, c, d\} = L$. The nontrivial majority clusters are $\{\{a, b\}, \{a, c\}, \{a, d\}, \{a, a, b, c, d\}\}$. For any MUL-tree T that contains all these clusters, $mult_{\Lambda(T)}(a) \geq 3$ while $mult_L(a) = 2$, i.e., $\Lambda(T) \neq L$. Thus, a majority rule consensus MUL-tree of $\mathcal{T}$ does not exist. Also, all the nontrivial majority clusters are singular, so no singular majority rule consensus MUL-tree exists.

Finally, we consider singular majority rule consensus MUL-trees. Let S be the set of all singular, nontrivial clusters that occur in at least $k/2$ of the MUL-trees in $\mathcal{T}$. By definition, for any cluster $C \in S$ and any singular majority rule consensus MUL-tree T of $\mathcal{T}$, we have $mult_{\mathcal{C}(T)}(C) = 1$. Thus, for every $C \in S$, there is a unique node t_C in T such that $C = \Lambda(T[t_C])$. For any two clusters $C, C' \in S$, we say that C *is an ancestor (the parent) cluster of* C' *in* T if the node t_C is an ancestor (the parent) of the node $t_{C'}$.

Lemma 3. *Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a set of MUL-trees with $\Lambda(T_1) = \Lambda(T_2) = \ldots = \Lambda(T_k) = L$.*

- *If $k = 2$, then a singular majority rule consensus MUL-tree of $\mathcal{T}$ always exists and is always unique.*
- *If $k \geq 3$, then a singular majority rule consensus MUL-tree of $\mathcal{T}$ might not exist, but if it does, it is unique.*

Proof. First consider the case $k = 2$. Let X be the multiset of all trivial clusters of L and all singular clusters that occur in more than $k/2$ of the MUL-trees in $\mathcal{T}$, i.e., in both T_1 and T_2. Let T be a strict consensus MUL-tree of $\mathcal{T}$ and note that $X \subseteq \mathcal{C}(T)$. All nonsingular clusters in T can be removed as follows: For each $C \in \mathcal{C}(T) \setminus X$, perform a delete operation on any node u in T satisfying $\Lambda(T[u]) = C$. This yields a MUL-tree T' with $\mathcal{C}(T') = X$. Thus, a singular majority rule consensus MUL-tree always exists when $k = 2$.

For $k \geq 3$, the nonexistence follows from the example in Figure 5.

Lastly, we prove the uniqueness. For the sake of obtaining a contradiction, suppose there exists two different singular majority rule consensus MUL-trees A, B of $\mathcal{T}$. Since $A \neq B$ although $\mathcal{C}(A) = \mathcal{C}(B)$, there are two clusters $C, C' \in S$ such that C' is the parent cluster of C in A while C' is not the parent cluster of C in B. It follows from the definition of a singular cluster that C' must be an ancestor cluster of C in B. Thus, there exists another cluster C'' such that C' is an ancestor cluster of C'', and C'' is the parent cluster of C in B. This means that $C \subsetneq C'' \subsetneq C'$, so C'' cannot be an ancestor cluster of C' in A. Hence, C'' is not an ancestor cluster of C in A, and so A must contain at least two copies of all elements in C. But then $C \uplus C \subseteq L$, contradicting the definition of a singular cluster. ■

Observe that the nonexistence and nonuniqueness results in Lemmas 1, 2, and 3 hold even when restricted to instances with $m = 2$, i.e., when $mult_L(x) \leq 2$ for all $x \in L$.

4. BUILDING A STRICT CONSENSUS MUL-TREE

Recall from Lemma 1 that for any given set T of MUL-trees with identical leaf label multisets, a strict consensus MUL-tree always exists. This section describes a simple algorithm for constructing such a consensus MUL-tree. Intuitively, this problem is easier than constructing a MUL-tree consisting of all the clusters in a given multiset (see, e.g., Huber et al., 2008), because all the branching information that must appear in the final output is already contained in any one of the input MUL-trees, say T_1, and we just need to determine what parts of T_1 to ignore.

Our algorithm, named `Strict_consensus,` is essentially an implementation of the existence proof for Lemma 1. The basic strategy is to remove clusters from the cluster collection $\mathcal{C}(T_1)$ by performing delete operations on suitable internal nodes from T_1 until a strict consensus MUL-tree is obtained. To identify which clusters to remove, the algorithm uses vectors of integers to represent clusters in $\mathcal{T}$ and stores these vectors in tries, as explained next.

A *leaf label numbering function* is a bijection from the set of q distinct leaf labels in L to the set $\{1, 2, \ldots, q\}$. We fix an arbitrary leaf label numbering function f. For every $T_i \in \mathcal{T}$ and node $u \in V(T_i)$, define a vector D_i^u of length q in which for every $j \in \{1, 2, \ldots, q\}$, the jth element equals $mult_{\Lambda(Ti[u])}(f^{-1}(j))$ (Figure 6). In other words, each element of the vector D_i^u counts how many times the corresponding leaf label occurs in the subtree rooted at node u in T_i. Clearly, D_i^{ℓ} contains exactly one 1 for any leaf ℓ of T_i, and D_i^u for any internal node u equals the sum of its children's D_i-vectors.

The pseudocode of Algorithm Strict_consensus is given in Figure 7. Step 1 considers each MUL-tree T_i in $\mathcal{T}$ separately. It first computes the D_i^u-vectors for all nodes in T_i by one bottom-up traversal of T_i. Then it initializes a trie A_i and stores the cluster collection $\mathcal{C}(T_i)$ in A_i by taking the q elements of each D_i^u-vector, concatenating them into a string of length q, and inserting the string into A_i. To keep track of multiple occurrences of strings in A_i, every created leaf ℓ in A_i is augmented with a variable $count_i(\ell)$ that stores the number of times that its string has been inserted. Next, in Step 2, for each distinct cluster in T_1 (i.e., for each

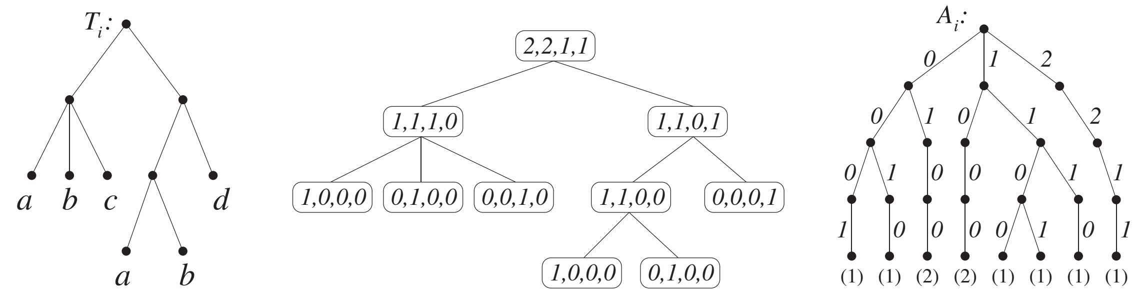

FIG. 6. A MUL-tree T_i, the D_i^u-vectors for its nodes under the leaf label numbering function $f(a) = 1, f(b) = 2, f(c) = 3, f(d) = 4$, and the trie A_i for storing $\mathcal{C}(T_i)$ are shown here. The value of $count_i(\ell)$ for each leaf ℓ in A_i is written in parentheses.

leaf ℓ in the trie A_1), the algorithm calculates how many occurrences need to be removed from T_1 to obtain a strict consensus MUL-tree by subtracting its minimum number of occurrences among $T_2, \ldots, T_k$ from the number of occurrences in T_1. The tries $A_1, A_2, \ldots, A_k$ and the variables $count_i(\ell)$ are used to retrieve these numbers efficiently, and the result is denoted by $excess(\ell)$. Finally, Steps 3 and 4 perform the necessary node deletions in top-down order, and Step 5 outputs the answer.

Theorem 1. *Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a set of MUL-trees with $\Lambda(T_1) = \Lambda(T_2) = \ldots = \Lambda(T_k)$. Algorithm* `Strict_consensus` *constructs a strict consensus MUL-tree of $\mathcal{T}$ in $O(nqk)$ time.*

Proof. The correctness of the algorithm follows from the proof of Lemma 1.

To analyze the time complexity, each of the k MUL-trees in $\mathcal{T}$ contains $O(n)$ nodes, so $O(nk)$ D_i-vectors need to be computed. Moreover, every D_i-vector is of length q. Therefore, Step 1 takes $O(nqk)$ time. Step 2 spends $O(qk)$ time for each of the $O(n)$ leaves in A_1, i.e., $O(nqk)$ time in total. Steps 3–5 require $O(nq)$ time

```
Algorithm  Strict_consensus
Input:  A set $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of MUL-trees with $\Lambda(T_1) = \Lambda(T_2) = \ldots = \Lambda(T_k)$.
Output: A strict consensus MUL-tree of $\mathcal{T}$.

1  for every $T_i \in \mathcal{T}$ do
       Initialize $D_i^\ell$ for every leaf $\ell$ in $T_i$.
       Do one bottom-up traversal of $T_i$ to compute $D_i^u$ for all internal nodes in $T_i$.
       Initialize an empty trie $A_i$ over the alphabet $\Sigma = \{0, 1, \ldots, n\}$.
       For every node $u \in V(T_i)$, concatenate the $q$ consecutive elements of $D_i^u$ to obtain a string $S(u)$
       of length $q$ over $\Sigma$. If $S(u)$ does not already exist in $A_i$ then insert $S(u)$ into $A_i$ and define
       $count_i(\ell) = 1$; otherwise, increment $count_i(\ell)$ by one.
   endfor
2  for each leaf $\ell$ of the trie $A_1$ do
       Let $S(\ell)$ be the string encoded by the path in $A_1$ from the root to $\ell$.
       Set $d_{min}(\ell) = count_1(\ell)$.
       For each $i \in \{2, \ldots, k\}$ do: Look for $S(\ell)$ in $A_i$. If $S(\ell)$ belongs to $A_i$ then let $d = count_i(\ell)$;
       otherwise, let $d = 0$. If $d < d_{min}(\ell)$ then let $d_{min}(\ell) = d$.
       Define $excess(\ell) = count_1(\ell) - d_{min}(\ell)$.
   endfor
3  Let $T$ be a copy of $T_1$.
4  for each node $u \in V(T_1)$ in top-down order do
       Locate the leaf $\ell$ in $A_1$ for $D_1^u$. If $excess(\ell) > 0$ then do a delete operation on the node corresponding
       to $u$ in $T$ and let $excess(\ell) = excess(\ell) - 1$.
   endfor
5  return $T$
End Strict_consensus
```

FIG. 7. Algorithm Strict_consensus.

because: (1) Locating a leaf ℓ in A_1 takes $O(q)$ time and this is done $O(n)$ times; and (2) in total, all delete operations take $O(n)$ time.

To prove (2), first observe that for any node u in $V(T_1)$ considered by the **for**-loop in Step 4, if u is deleted then the children of u will become children of the parent of u instead; conveniently, these nodes will never need to be moved again due to the top-down ordering used in the **for**-loop. Consequently, the delete operation on a single node u in T always takes (at most) time proportional to the number of children of its corresponding node in T_1. Finally, the sum of the number of children of all nodes in T_1 is $O(n)$. ■

5. BUILDING A MAJORITY RULE CONSENSUS MUL-TREE IS NP-HARD

Here, we show that the following decision problem is NP-hard:

Majority Rule Consensus MUL-Tree (MCMT):
Input: A set $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of MUL-trees and a multiset L of leaf labels such that $\Lambda(T_i) = L$ for every $T_i \in \mathcal{T}$.
Question: Is there a majority rule consensus MUL-tree of $\mathcal{T}$?

To prove the result, we will reduce the 1-IN-3 3SAT problem to MCMT. 1-IN-3 3SAT is known to be NP-hard (Garey and Johnson, 1979) and is defined as:

One-in-Three 3-Satisfiability (1-IN-3 3SAT):
Input: A Boolean formula F in conjunctive normal form where every clause contains at most 3 literals (3-CNF).
Question: Does there exist a truth assignment for F such that each clause contains exactly one true literal?

We first define an operation called *non-mono-replace* on any Boolean formula F in 3-CNF as:

- For every clause C_u in F that consists of three positive literals, arbitrarily select one of its three literals x_k and replace $C_u = (x_i \vee x_j \vee x_k)$ by two clauses $(x_i \vee x_j \vee \bar{y_u}) \wedge (y_u \vee x_k)$, where y_u is a newly added Boolean variable. Similarly, for every clause C_u in F that consists of three negative literals, arbitrarily select one of its three literals $\bar{x_k}$ and replace $C_u = (\bar{x_i} \vee \bar{x_j} \vee \bar{x_k})$ by two clauses $(\bar{x_i} \vee \bar{x_j} \vee y_u) \wedge (\bar{y_u} \vee \bar{x_k})$, where y_u is a newly added Boolean variable.

Below, we will use the non-mono-replace operation to ensure that the Boolean formula we reduce from has a special restricted structure. Denote the result of applying the non-mono-replace operation on F by F'. The next lemma establishes the relationship between F and F'.

Lemma 4. *Let F be a Boolean formula in 3-CNF and let F′ be the 3-CNF Boolean formula obtained by applying the non-mono-replace operation on F. There exists a truth assignment for F such that every clause contains exactly one true literal if and only if there exists a truth assignment for F′ such that every clause contains exactly one true literal.*

Proof. ($\rightarrow$) Suppose F has a truth assignment σ in which every clause contains exactly one true literal. Let σ' be the following truth assignment for F': For every variable x_i that appears in both F and F', set $\sigma'(x_i) = \sigma(x_i)$. For variables y_u that only appear in F', set $\sigma'(y_u) \neq \sigma(x_k)$, where $(y_u \vee x_k) \in F'$.

To see that every clause in F' contains exactly one true literal under σ', consider any clause C_u in F. By the assumptions, C_u has exactly one true literal under σ. There are three possibilities:

- If C_u contains both positive and negative literals, then C_u also belongs to F' and has exactly one true literal under σ'.
- If C_u contains positive literals only, write $C_u = (x_i \vee x_j \vee x_k)$, where its two corresponding clauses in F' are $C'_u = (x_i \vee x_j \vee \bar{y_u}) \wedge (y_u \vee x_k)$. In case $\sigma'(x_k)$ is false, then either $\sigma'(x_i)$ or $\sigma'(x_j)$ must be true and $\sigma'(y_u)$ is true. On the other hand, in case $\sigma'(x_k)$ is true, then both $\sigma'(x_i)$ and $\sigma'(x_j)$ must be false and $\sigma'(y_u)$ is false. In both cases, C'_u is true and each of its two clauses contains exactly one true literal.

- If C_u contains negative literals only, it can be verified in the same way that each of its two corresponding clauses in F' is true and contains exactly one true literal.

($\leftarrow$) Suppose F' has a truth assignment σ' in which every clause contains exactly one true literal. Then we directly obtain a truth assignment σ for F simply by setting $\sigma(x_i) = \sigma'(x_i)$ for every variable x_i in F. Moreover, each clause C_u in F contains exactly one true literal under σ, as shown next. If $C_u \in F'$, then C_u has exactly one true literal under σ' by the assumptions, and hence under σ as well. On the other hand, if $C_u \notin F'$, then C_u must consist of three positive literals or three negative literals. In the former case, write $C_u = (x_i \vee x_j \vee x_k)$ with $C'_u = (x_i \vee x_j \vee \bar{y_u}) \wedge (y_u \vee x_k) \in F'$. There are two subcases: (a) If $\sigma'(y_u)$ is false, then $\sigma(x_k)$ is true while both $\sigma(x_i)$ and $\sigma(x_j)$ are false; thus, precisely one literal, namely x_k, in C_u is true. (b) If $\sigma'(y_u)$ is true, then $\sigma(x_k)$ is false while either $\sigma(x_i)$ or $\sigma(x_j)$ is true; thus, precisely one literal (either x_i or x_j) in C_u is true. The final case where C_u consists of three negative literals is symmetric. ■

We now describe the reduction from 1-IN-3 3SAT to MCMT. Let F be any given Boolean formula in 3-CNF. As in the proof of Theorem 3.1 in Huber et al. (2008), assume without loss of generality that:

(i) No single clause in F contains a variable x_i as well as its negation $\bar{x}_i$ as literals; and
(ii) For every variable x_i in F, both x_i and its negation $\bar{x}_i$ appear somewhere in F as literals.

Then, apply the non-mono-replace operation on F to obtain a Boolean formula F' with s variables and t clauses, for some positive integers s and t [this does not affect properties (i) and (ii) above]. Lastly, construct three MUL-trees T_1, T_2, T_3 based on F' as follows. Let $X = \{x_1, \ldots, x_s\}$ and $Z = \{z_1, \ldots, z_t\}$ be two sets in one-to-one correspondence with the variables and clauses of F', respectively. Say that x_i *is positive (negative) in* z_j if x_i corresponds to a variable in F' that occurs positively (negatively) in the jth clause. Define the leaf label multiset L for T_1, T_2, T_3 as $L = \{x, x : x \in X\} \cup \{z, z, z : z \in Z\}$. (Observe that L contains two copies of every element in X and three copies of every element in Z.) Next, for each $x \in X$, define two subsets Z_x, $\tilde{Z}_x$ of Z by $Z_x = \{z \in Z : x \text{ is positive in } z\}$ and $\tilde{Z}_x = \{z \in Z : x \text{ is negative in } z\}$. Let $\mathcal{W} = \{Z_x \cup \{x\} : x \in X\}$ and $\tilde{\mathcal{W}} = \{\tilde{Z}_x \cup \{x\} : x \in X\}$. From $\mathcal{W}$ and $\tilde{\mathcal{W}}$, construct three MUL-trees T_1, T_2, T_3 with $\Lambda(T_1) = \Lambda(T_2) = \Lambda(T_3) = L$, whose sets of nontrivial clusters are: $\mathcal{W} \cup \tilde{\mathcal{W}}$, $\mathcal{W} \cup \{X \cup Z\}$, and $\tilde{\mathcal{W}} \cup \{X \cup Z\}$, respectively. Then, the set of nontrivial majority clusters for $\{T_1, T_2, T_3\}$ is: $\mathcal{W} \cup \tilde{\mathcal{W}} \cup \{X \cup Z\}$. The next lemma shows that each of T_1, T_2, T_3 is indeed a valid MUL-tree.

Lemma 5. *The MUL-trees T_1, T_2, and T_3 defined above always exist.*

Proof. By definition, each $x \in X$ occurs exactly once in $\mathcal{W}$ and exactly once in $\tilde{\mathcal{W}}$. Moreover, every clause in F' contains at most three literals, so every $z \in Z$ occurs at most three times in $\mathcal{W} \cup \tilde{\mathcal{W}}$. Hence, $\biguplus_{S \in \mathcal{W} \cup \tilde{\mathcal{W}}} S$ is a submultiset of L and there exists a tree with a root whose children are associated with the clusters in $\mathcal{W} \cup \tilde{\mathcal{W}}$. Thus, T_1 always exists.

Because of the non-mono-replace operation, every clause in F' contains at most two positive (and at most two negative) literals. This means that every $z \in Z$ occurs at most two times in $\mathcal{W}$ (and at most two times in $\tilde{\mathcal{W}}$). Hence, $\biguplus_{S \in \mathcal{W} \cup \{X \cup Z\}} S$ is a submultiset of L, and T_2 always exists. Similarly, $\biguplus_{S \in \tilde{\mathcal{W}} \cup \{X \cup Z\}} S$ is a submultiset of L, so T_3 always exists. ■

An example of how the three MUL-trees T_1, T_2, T_3 are constructed from a given Boolean formula F is shown in Figure 8.

The reduction's correctness is guaranteed by:

Lemma 6. *A majority rule consensus MUL-tree for T_1, T_2, T_3 exists if and only if there exists a truth assignment for F' such that every clause contains exactly one true literal.*

Proof. ($\rightarrow$) Suppose there exists a majority rule consensus MUL-tree T_{123} for $\{T_1, T_2, T_3\}$. By definition, its set of nontrivial clusters is $\mathcal{W} \cup \tilde{\mathcal{W}} \cup \{X \cup Z\}$.

Consider any two sets S_1 and S_2 in $\mathcal{W} \cup \tilde{\mathcal{W}}$ that contain a common element from X, i.e., $S_1 = Z_x \cup \{x\}$ and $S_2 = \tilde{Z}_x \cup \{x\}$ for the same $x \in X$. According to assumption (ii) above, both Z_x and $\tilde{Z}_x$ are nonempty, so each of S_1 and S_2 contains one or more elements from Z. Furthermore, according to assumption (i) above, every $z \in Z$ may appear in at most one of Z_x and $\tilde{Z}_x$. Thus, by (i) and (ii), the following crucial observation holds:

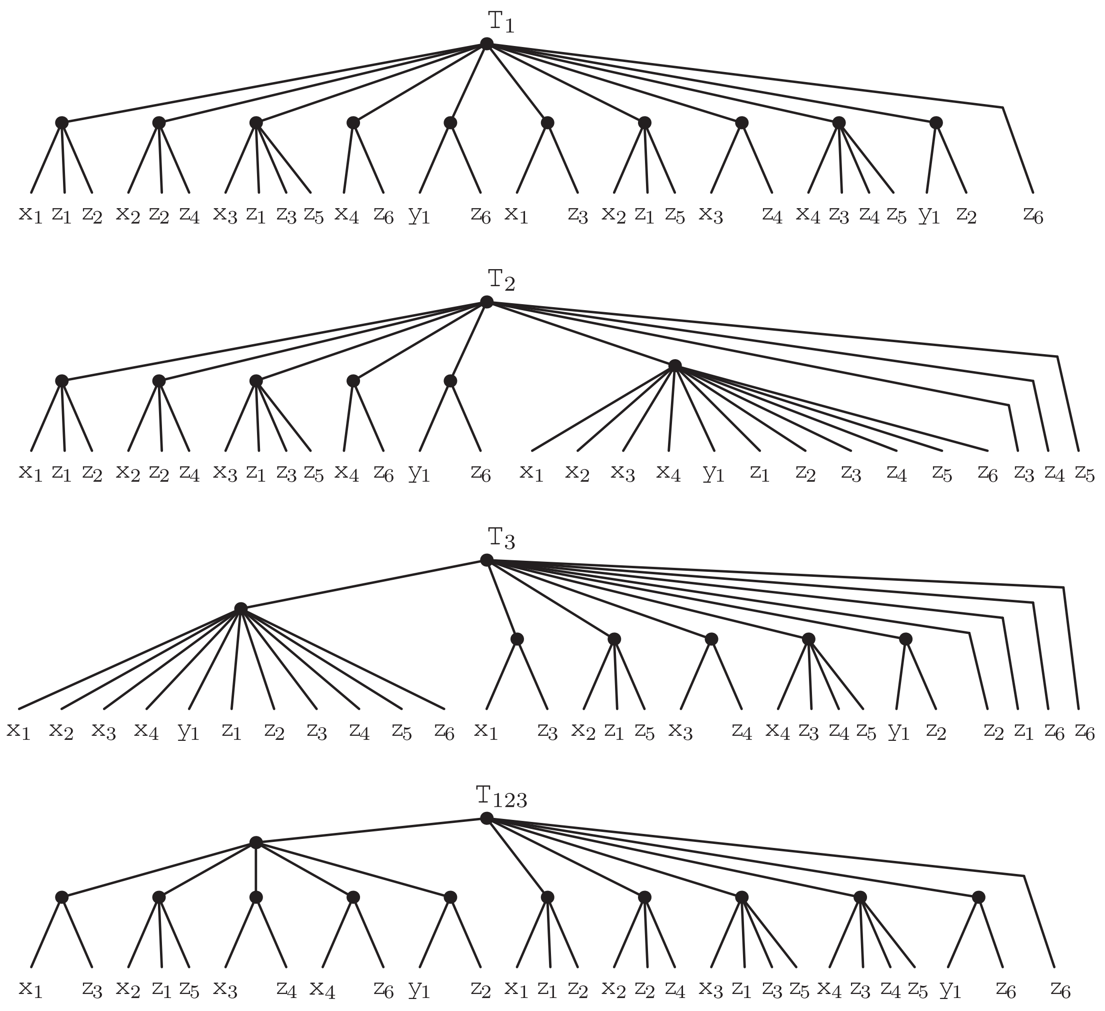
T1
x1 z1 z2 x2 z2 z4 x3 z1 z3 z5 x4 z6 y1 z6 x1 z3 x2 z1 z5 x3 z4 x4 z3 z4 z5 y1 z2 z6
T2
x1 z1 z2 x2 z2 z4 x3 z1 z3 z5 x4 z6 y1 z6 x1 x2 x3 x4 y1 z1 z2 z3 z4 z5 z6 z3 z4 z5
T3
x1 x2 x3 x4 y1 z1 z2 z3 z4 z5 z6 x1 z3 x2 z1 z5 x3 z4 x4 z3 z4 z5 y1 z2 z2 z1 z6 z6
T123
x1 z3 x2 z1 z5 x3 z4 x4 z6 y1 z2 x1 z1 z2 x2 z2 z4 x3 z1 z3 z5 x4 z3 z4 z5 y1 z6 z6

FIG. 8. An illustration of the reduction in Section 5. In this example, $F=(x_1 \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x_1} \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4})$ and $F'=(x_1 \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{y_1}) \wedge (\bar{x_1} \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4}) \wedge (y_1 \vee x_4)$. According to the definitions, $Z_{x_1}=\{z_1, z_2\}$ and $\tilde{Z}_{x_1}=\{z_3\}$, etc., so that $\mathcal{W}=\{\{x_1, z_1, z_2\}, \ldots\}$ and $\widetilde{\mathcal{W}}=\{\{x_1, z_3\}, \ldots\}$, yielding three MUL-trees T_1, T_2, T_3. Here, T_1, T_2, T_3 have a majority rule consensus MUL-tree T_{123}. As explained in the proof of Lemma 6, the non leaf children of the root of T_{123} are: (1) the roots of the trees in a set Γ_f; and (2) an internal node whose children are the roots of the trees in a set Γ_t that encode, for each variable x_i, which of the clauses are satisfied by x_i. The corresponding truth assignment for F and F' in this example is: x_1 = false, x_2 = false, x_3 = false, x_4 = true, y_1 = false.

- For any two sets $S_1, S_2 \in \mathcal{W} \cup \widetilde{\mathcal{W}}$, S_1 and S_2 are not subsets of each other.

Let u be the internal node in T_{123} to which the cluster $X \cup Z$ is associated. Note that u must be a child of the root r of T_{123}. Also note that since $\biguplus_{S\in\mathcal{W}\cup\widetilde{\mathcal{W}}} S$ contains both copies of every $x \in X$ from L, there are no copies of x left to create any trivial clusters consisting of elements from X directly attached to r or u. This means that for every child v of u, the cluster associated with v must be a cluster from $\mathcal{W} \cup \widetilde{\mathcal{W}}$ or a trivial cluster $\{z\}$ where $z \in Z$. In addition, the clusters associated with the children of u form a partition of $X \cup Z$, so for each $x \in X$, exactly one of $Z_x \cup \{x\}$ and $\tilde{Z}_x \cup \{x\}$ is associated to a descendant of u; from the crucial observation above it follows that this descendant must, in fact, be a child of u.

Now, we obtain a truth assignment for F': For each $x \in X$, in case $Z_x \cup \{x\}$ is associated with a child of u, then let x be true; otherwise (i.e., $\tilde{Z}_x \cup \{x\}$ is associated with a child of u), let x be false. Since $\{Z_x \cup \{x\} : x \text{ is true in } F'\} \cup \{\tilde{Z}_x \cup \{x\} : x \text{ is false in } F'\} : x \text{ is false in } F'\}$ forms a partition of $X \cup Z$, it is easy to check that $\{Z_x : x \text{ is true in } F'\} \cup \{\tilde{Z}_x : x \text{ is false in } F'\}$ equals Z. Therefore, with this truth assignment, every clause in F' has exactly one true literal.

($\leftarrow$) Suppose F' has a truth assignment σ' in which every clause contains exactly one true literal. We show how to build a number of small trees and connect them to obtain a MUL-tree T_{123} with $\Lambda(T_{123}) = L$ such that the set of all nontrivial clusters in the cluster collection of T_{123} equals $\mathcal{W} \cup \widetilde{\mathcal{W}} \cup \{X \cup Z\}$. Then, by definition, T_{123} is a majority rule consensus MUL-tree of $\{T_1, T_2, T_3\}$.

First, for each $x \in X$, let R_x be a tree consisting of a root node attached to leaves labeled by $Z_x \cup \{x\}$. Similarly, for each $x \in X$, let $R_{\bar{x}}$ be a tree consisting of a root node attached to leaves labeled by $\tilde{Z}_x \cup \{x\}$. Note that each R_x-tree contains the leaf labels from one element in $\mathcal{W}$, and each $R_{\bar{x}}$-tree contains the leaf labels from one element in $\widetilde{\mathcal{W}}$. Partition the R_x- and $R_{\bar{x}}$-trees into two sets Γ_t and Γ_f:

- $\Gamma_t = \{R_x : \sigma'(x) = \text{true}, x \in X\} \cup \{R_{\bar{x}} : \sigma'(x) = \text{false}, x \in X\}$
- $\Gamma_f = \{R_x : \sigma'(x) = \text{false}, x \in X\} \cup \{R_{\bar{x}} : \sigma'(x) = \text{true}, x \in X\}$

Build a MUL-tree T_{123} with a root node whose children are: (1) the roots of the trees in Γ_f; (2) an internal node u whose children are the roots of the trees in Γ_t; and (3) leaves labeled by $L \setminus (\bigcup_{R \in \Gamma_f \cup \Gamma_t} \Lambda(R))$ (Figure 8). By the construction, every cluster in $\mathcal{W} \cup \widetilde{\mathcal{W}}$ occurs somewhere in T_{123}. Also, the cluster $X \cup Z$ occurs in T_{123} because it is associated with the internal node u in (2) above; to see this, note that every clause in F' contains exactly one true literal in the truth assignment σ', so each $x \in X$ and each $z \in Z$ occurs exactly once as leaf labels in the trees in Γ_t. Hence, T_{123} is a majority rule consensus MUL-tree for $\{T_1, T_2, T_3\}$. ■

In summary, the reduction above, together with Lemmas 4 and 6, yields the main theorem of this section:

Theorem 2. *The MCMT problem is NP-hard, even if restricted to inputs where* $k = 3$ *and* $m = 3$, *where* m *is the multiplicity of the leaf label multiset.*

Remark: In a related problem studied in Huber et al. (2008), named Multiset Split Compatibility (MSC), the input is a multiset $\mathcal{S}$ of bipartitions (so-called *splits*) of a multiset L of leaf labels, and the objective is to decide if there exists an unrooted MUL-tree leaf labeled by L whose set of edges induces a multiset of bipartitions of L that is equal to $\mathcal{S}$. It is easy to reduce MCMT to MSC: Given an instance of MCMT, count the occurrences of all clusters in $\mathcal{T}$ to identify the majority clusters and their multiplicities, and let $\mathcal{S}$ be the multiset of bipartitions corresponding to those clusters, in which an additional leaf label is used to represent the root node. Since MCMT is a special case of MSC, Theorem 2 above can be viewed as a technical strengthening of Theorem 3.1 in Huber et al. (2008), which states that MSC is NP-hard.

6. BUILDING A MAJORITY RULE CONSENSUS MUL-TREE WHEN $M = 2$

Section 5 above proves that, in general, constructing a majority rule consensus MUL-tree is a computationally hard problem. However, when the parameter m (the multiplicity of the leaf label multiset L) is restricted to be at most two, the problem can be solved in polynomial time, as demonstrated in this section.

At the end of Section 1 in Huber et al. (2008), briefly mentioned that for the special case where every leaf label has exactly two occurrences in L (i.e., when $mult_L(\ell) = 2$ for every $\ell \in L$), the problem of checking if there exists a MUL-tree that is compatible with a given set $\mathcal{S}$ of bipartitions on L can be reduced to a problem known as the Perfect Phylogeny Haplotyping problem (PPH) (Gusfield, 2002). Here, we work out the missing technical details to obtain an $O(n^2k + nk^2)$-time algorithm for constructing a majority rule consensus MUL-tree when $mult_L(\ell) \leq 2$ for every $\ell \in L$.

The Perfect Phylogeny Haplotyping problem (PPH) was introduced by Gusfield (2002) for the purpose of inferring *haplotypes* that resolve a given set of *genotypes* under the coalescent model of haplotype evolution (see Gusfield, 2002 for the biological motivation behind this problem). PPH is defined as follows. Given an $(n \times t)$-matrix M where each entry belongs to $\{0, 1, 2\}$, output a $(2n \times t)$-matrix M' such that: (1) Every entry of M' belongs to $\{0, 1\}$; (2) if $M[i, j] \in \{0, 1\}$, then $M'[2i - 1, j] = M'[2i, j] = M[i, j]$; (3) if $M[i, j] = 2$, then $\{M'[2i - 1, j], M'[2i, j]\} = \{0, 1\}$; and (4) M' admits a perfect phylogeny (i.e., the columns in M' are pairwise compatible (see, e.g., Section 17.3.3 in Gusfield, 1997). PPH has been well studied, and the fastest algorithm for solving it runs in $O(nt)$ time (Ding et al., 2006).

Now, we describe the method for building a majority rule consensus MUL-tree. It consists of three steps:

1. Identify all clusters that appear in a majority rule consensus MUL-tree of $\mathcal{T}$.
2. Construct an input matrix M to the PPH problem, apply the algorithm of Ding *et al.* (2006) for PPH to M, and let M' be the output.
3. Based on M', construct a majority rule consensus MUL-tree of $\mathcal{T}$, if one exists; otherwise, FAIL.

In Step 1, we compute all majority clusters in $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ and the number of times each cluster must occur in a solution (recall that, according to the definition of a majority rule consensus MUL-tree T of $\mathcal{T}$, for any $C \in \mathcal{C}(T)$, $mult_{\mathcal{C}(T)}(C)$ equals the largest integer j such that $|\{T_i : mult_{\mathcal{C}(T_i)}(C) \geq j\}| > k/2$). Let S be the resulting multiset and denote $S = \{s_1, s_2, \ldots, s_{|S|}\}$.

In Step 2, construct a $(q \times |S|)$-matrix M, where q is the number of distinct elements in L and $|S|$ is the total number of occurrences of all majority clusters found in Step 1. Each element $M[i,j] \in \{0, 1, 2\}$ specifies the relationship between the leaf label i and the cluster s_j. To be precise, for every $1 \leq i \leq q$ and $1 \leq j \leq |S|$, let:

$$M[i,j] = \begin{cases} 0, & \text{if leaf label } i \text{ does not occur in cluster } s_j \\ 2, & \text{if leaf label } i \text{ occurs once in cluster } s_j \text{ and } mult_L(i) = 2 \\ 1, & \text{if leaf label } i \text{ occurs once in cluster } s_j \text{ and } mult_L(i) = 1 \\ 1, & \text{if leaf label } i \text{ occurs twice in cluster } s_j \end{cases}$$

Apply the algorithm of Ding et al. (2006) to M and let M' be the output $(2q \times |S|)$-matrix.

In Step 3, if PPH does not admit a solution for M, we return FAIL. Otherwise, we use M' to recover a majority rule consensus MUL-tree T for $\mathcal{T}$. First construct a perfect phylogeny P for M', and note that P has the following property.

Lemma 7. *For any leaf label i in L with $mult_L(i) = 1$, its two corresponding leaves ℓ_{2i-1} and ℓ_{2i} in P have the same parent.*

Proof. By definition, the $(2i - 1)$-th and $(2i)$-th rows of M' are identical. Hence, in P, both leaves ℓ_{2i-1} and ℓ_{2i} are attached to the same internal node. ■

Next, for every leaf label i in L with $mult_L(i) = 2$, we replace its two corresponding leaves l_{2i-1} and l_{2i} in P by two i's. For every leaf label i in L with $mult_L(i) = 1$, Lemma 7 states that its two corresponding leaves l_{2i-1} and l_{2i} in P have the same parent; we simply replace these two leaves by a single leaf labeled by i. Let T be the resulting MUL-tree. The next lemma shows that T contains all of the clusters in S.

Lemma 8. *For every cluster $s_j \in S$, T contains s_j.*

Proof. By the properties of a perfect phylogeny P for M', the cluster s_j can be associated with exactly one node $P(j)$ in P so that for any row x of M', it holds that $M'[x, j] = 1$ if and only if the leaf x is a descendant of the node $P(j)$. In the tree T, for any leaf label i with $mult_L(i) = 1$, it still holds that $\Lambda(T[P(j)]) = s_j$ by Lemma 7 and the definition of T. On the other hand, for any leaf label i with $mult_L(i) = 2$, there are two cases. Firstly, if s_j contains two occurrences of i, then they will both be descendants of the node $P(j)$ in T. Secondly, if s_j contains one occurrence of i, then exactly one of $M'[2i - 1, j]$ and $M'[2i, j]$ equals 1, and by the above construction, there will only be one occurrence of leaf label i in the subtree $T[P(j)]$. This shows that there always exists a node u in T (namely $u = P(j)$) such that $\Lambda(T[u]) = s_j$. ■

Lemma 8 implies that T is a majority rule consensus MUL-tree of $\mathcal{T}$. This gives:

Theorem 3. *Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a set of MUL-trees with $\Lambda(T_1) = \Lambda(T_2) = \ldots = \Lambda(T_k)$. If $m = 2$, where m is the multiplicity of the leaf label multiset, then a majority rule consensus MUL-tree of $\mathcal{T}$ (if one exists) can be constructed in $O(n^2k + nk^2)$ time.*

Proof. Step 1 of the method can be carried out in $O(n^2k + nk^2)$ time by first applying the technique described in Section 4 to compute the D_i^u-vectors for every node u in every MUL-tree T_i and concatenating each such D_i^u-vector to a string of length at most n over the alphabet $\{0, 1, 2\}$. Then, all the $O(nk)$ strings are inserted into a single trie A, while for each leaf ℓ of A, k variables $count_1(\ell), count_2(\ell), \ldots, count_k(\ell)$ store the number of occurrences of the cluster represented by ℓ in each MUL-tree $T_i \in \mathcal{T}$. Next, for each leaf ℓ in A, compute the median of the values $count_i(\ell)$ for all $i \in \{1, 2, \ldots, k\}$ in $O(k)$ time to determine whether the cluster represented by ℓ is a majority cluster and, if so, its correct multiplicity in the set S. In total, Step 1 takes $O(n^2k + nk^2)$ time.

In Step 2, applying the algorithm of Ding et al. (2006) to M takes $O(q \cdot |S|)$ time. Each input MUL-tree T_i contains $O(n)$ nodes, so $|S| = O(nk)$ and Step 2 therefore takes $O(n^2k)$ time.

In Step 3, constructing a perfect phylogeny P for M' takes $O(2q \cdot |S|) = O(n^2k)$ time by the algorithm in Section 17.3.4 in Gusfield (1997), and the modifications to obtain T from P do not affect the asymptotic time complexity. ■

7. BUILDING A SINGULAR MAJORITY RULE CONSENSUS MUL-TREE

In this section, we present a polynomial-time algorithm for building a singular majority rule consensus MUL-tree or determining that such a tree does not exist. According to Lemma 3 in Section 3, when a singular majority rule consensus MUL-tree of $\mathcal{T}$ exists, it is unique.

Our algorithm consists of two phases. Phase 1 constructs the set S of all singular, nontrivial clusters that occur in at least $k/2$ of the MUL-trees in $\mathcal{T}$. To implement Phase 1, first enumerate all nontrivial clusters that occur in $\mathcal{T}$ and count their occurrences in the same way, as in the first part of the proof of Theorem 3 in Section 6. Then, let S be the set of those clusters that occur in more than $k/2$ of the MUL-trees in $\mathcal{T}$ and that are singular.

Phase 2 builds the singular majority rule consensus tree of $\mathcal{T}$ by calling a top-down, recursive procedure `Build_MUL-tree`(L, S), listed in Figure 9, which returns the singular majority rule consensus MUL-tree T for $\mathcal{T}$, if it exists. The cluster associated with the root of T is L, and the clusters associated with the children of the root of T belong to a set $\mathcal{F} \subseteq S$ of maximal elements in S. More precisely, we let $\mathcal{F} = \{C \in S : C$ is not a submultiset of any cluster $C' \in S\}$. Lemma 10 below ensures that $\mathcal{F}$ defined in this way equals the set of all clusters associated with children of the root of the (unique) singular majority rule consensus MUL-tree of $\mathcal{T}$, so that we may apply recursion to compute T.

Steps 1 and 2 of `Build_MUL-tree` compute $\mathcal{F}$ in a greedy fashion. After each update to $\mathcal{F}$ in Step 2, if L is a proper submultiset of $\biguplus_{C \in \mathcal{F}} C$, then no MUL-tree leaf-labeled by L containing all clusters in S exists, and the algorithm reports FAIL. Step 3 builds a sub-MUL-tree T_C for each cluster C in $\mathcal{F}$ by recursively calling `Build_MUL-tree`(C, $S|C$), where $S|C = \{X \in S : X \subsetneq C\}$. The base case of the recursion is given by the condition $S = \emptyset$, as this implies that $\mathcal{F} = \emptyset$, and no further recursive calls will be made. Then, in Step 4, the T_C-trees and all "leftover leaves" not in $\biguplus_{C \in \mathcal{F}} C$ are assembled into the final consensus MUL-tree T, which is returned in Step 5.

We now show the correctness of using the set $\mathcal{F}$ to build the MUL-tree T:

Lemma 9. *Let T be the singular majority rule consensus MUL-tree of $\mathcal{T}$. If C_1 and C_2 are two clusters associated with two internal nodes u and v of T such that u is not an ancestor of v and v is not an ancestor of u, then neither of C_1 and C_2 is a submultiset of the other.*

Proof. If $C_1 \subseteq C_2$, then T contains at least two copies of all elements in C_1, and thus $C_1 \uplus C_1 \subseteq L$. This contradicts the fact that C_1 is singular. The case $C_2 \subseteq C_1$ is symmetric. The lemma follows. ■

Lemma 10. *$\mathcal{F} = \{C \in S : C$ is not a submultiset of any cluster $C' \in S\}$ equals the set of all clusters associated with children of the root of the unique singular majority rule consensus MUL-tree of $\mathcal{T}$.*

Proof. First, consider any cluster $X \in S \setminus \mathcal{F}$. According to the definition of $\mathcal{F}$, X must be a submultiset of some cluster $C \in \mathcal{F}$. Let x and c be the two nodes in T to which X and C are associated, respectively. By Lemma 9, node c is an ancestor of node x, so the subtree represented by X is contained in the subtree represented by C.

```
Algorithm    Build_MUL-tree
Input:    A multiset $L$ and a set $S$ of singular, non-trivial clusters of $L$.
Output: A MUL-tree leaf-labeled by $L$ that contains all clusters in $S$, if one exists; otherwise, FAIL.

1   Let $\mathcal{F}$ be the empty set.
2   for every $X \in S$ do
2.1     if $X$ is not a submultiset of any cluster currently in $\mathcal{F}$ then
            Delete every cluster from $\mathcal{F}$ that is a submultiset of $X$.
            Insert $X$ into $\mathcal{F}$.
            If $L \subsetneq \biguplus_{C \in \mathcal{F}} C$ then return FAIL.
        endif
    endfor
3   for every $C \in \mathcal{F}$ do
        Compute $S|C = \{X \in S : X \subsetneq C\}$.
        Let $T_C$ = Build_MUL-tree($C$, $S|C$).
    endfor
4   Let $T$ be a MUL-tree whose root is attached to: (1) the root of $T_C$ for each $C \in \mathcal{F}$; and (2) all leaves
    labeled by $L \setminus (\biguplus_{C \in \mathcal{F}} C)$.
5   return $T$
End Build_MUL-tree
```

FIG. 9. Algorithm Build_MUL-tree.

Next, consider any cluster $C \in \mathcal{F}$. Let c be the node in T to which C is associated. Suppose the parent of c is a node x and that x is not the root of T. But then $C \subsetneq X$, where X is the cluster associated to x, which contradicts the maximality of the clusters in $\mathcal{F}$. Therefore, c must be a child of the root of T, i.e., the subtree represented by C is attached directly to the root of T. ■

Theorem 4. *Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a set of MUL-trees with $\Lambda(T_1) = \Lambda(T_2) = \ldots = \Lambda(T_k)$. The algorithm constructs the singular majority rule consensus MUL-tree of $\mathcal{T}$ (if it exists) in $O(n^3k + nk^2)$ time.*

Proof. As in the proof of Theorem 3, the time complexity of Phase 1 is $O(n^2k + nk^2)$. Phase 2 calls `Build_MUL-tree`(L, S), which constructs a MUL-tree with at most $|L|$ internal nodes, i.e., $O(|L|)$ clusters. For each such cluster, it may need to execute all the steps of the procedure, which takes $O(|L||S|)$ time because $|\biguplus_{C \in \mathcal{F}} C| \leq |L|$. Since $|L| = n$ and $|S| = O(nk)$, the total running time of Phase 2 is $O(|L|^2|S|) = O(n^3k)$. ■

8. CONCLUDING REMARKS

Ideally, one would like to generalize tools and concepts that have been demonstrated to be useful for single-labeled phylogenetic trees to MUL-trees. Unfortunately, certain basic problems become NP-hard when extended to MUL-trees. For example, the MSC problem mentioned at the end of Section 5 is NP-hard, whereas the corresponding problem for single-labeled phylogenetic trees is solvable in polynomial time Huber et al., (2008). As another example, given a set of *rooted triplets* (single-labeled binary phylogenetic trees with exactly three leaves each), a classical algorithm by Aho et al. (1981) can check if there exists a single-labeled phylogenetic tree that is consistent with all of the rooted triplets in in polynomial time; on the other hand, it is NP-hard to decide if there exists a MUL-tree consistent with having at most d leaf duplications, even if restricted to $d = 1$ Guillemot et al., (2011). In short, MUL-trees pose new and sometimes unexpected algorithmic challenges for researchers.

In this article, we have shown that the problem of building a consensus MUL-tree can be solved in polynomial time for certain types of consensus MUL-trees, thus significantly improving on the previously existing, exponential-time methods of Huber et al. (2012) and Lott et al. (2009b). We have also presented a number of negative results regarding the existence, uniqueness, and time complexity of consensus MUL-trees. The main open problem is to identify other variants than the ones studied here with even better properties and to study their performance in practice. For example, is there some way to combine the advantages of the strict consensus MUL-tree and the singular majority rule consensus MUL-tree?

Our algorithm `Strict_consensus` in Section 4 runs in $O(nqk)$ time according to Theorem 1. We note that this is not optimal when applied to single-labeled phylogenetic trees, i.e., when $q = n$, because it gives a time complexity of $O(n^2k)$ while Day's algorithm (Day, 1985) solves the problem in $O(nk)$ time. However, it seems difficult to extend Day's algorithm to MUL-trees directly since its efficiency relies on the fact that, after relabeling the leaves by the positive integers $\{1, 2, \ldots, n\}$ according to the order in which they are visited by a depth-first traversal of T_1, every cluster contained in T_1 forms an interval. When T_1 is allowed to be a MUL-tree, such relabeling does not necessarily exist.

For inputs where a majority rule consensus MUL-tree does not exist, one might try to introduce additional occurrences of the leaf labels until it is possible to construct a MUL-tree that contains all majority clusters. Here, minimizing the number of leaf duplications appears to be a hard problem, and the computational complexity will probably not be polynomial. Another obvious disadvantage of this approach is that the output MUL-tree will no longer have the same leaf label multiset as the input MUL-trees.

ACKNOWLEDGMENTS

Jesper Jansson was funded by The Hakubi Project at Kyoto University and KAKENHI grant number 23700011.

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Aho, A.V., Sagiv, Y., Szymanski, T.G., et al. 1981. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing.* 10, 405–421.

Bryant, D. 2003. A classification of consensus methods for phylogenetics, 163–184. *In* Janowitz, M.F., Lapointe, F.-J., McMorris, F.R., et al., eds. Bioconsensus, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 61. American Mathematical Society, Providence, RI.

Day, W.H.E. 1985. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification.* 2, 7–28.

Ding, Z., Filkov, V., and Gusfield, D. 2006. A linear-time algorithm for the perfect phylogeny haplotyping (PPH) problem. *J. Comput. Biol.* 13, 522–553.

Fellows, M., Hallett, M., and Stege, U. 2003. Analogs & duals of the MAST problem for sequences & trees. *Journal of Algorithms.* 49, 192–216.

Felsenstein, J. 2004. *Inferring Phylogenies*. Sinauer Associates, Sunderland, Massachusetts.

Ganapathy, G., Goodson, B., Jansen, R., et al. 2006. Pattern identification in biogeography. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 3, 334–346.

Garey, M., and Johnson, D. 1979. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Guillemot, S., Jansson, J., and Sung, W.-K. 2011. Computing a smallest multilabeled phylogenetic tree from rooted triplets. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 8, 1141–1147.

Gusfield, D. 1997. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, New York.

Gusfield, D. 2002. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. *Proceedings of the 6th Annual International Conference on Computational Biology (RECOMB 2002)*. 166–175.

Huber, K.T., Lott, M., Moulton, V., et al. 2008. The complexity of deriving multi-labeled trees from bipartitions. *J. Comput. Biol.* 15, 639–651.

Huber, K.T., Moulton, V., Spillner, A. 2012. Computing a consensus of multilabeled trees. *Proceedings of the 14th Workshop on Algorithm Engineering and Experiments (ALENEX 2012)*. 84–92.

Huber, K.T., Oxelman, B., Lott, M., et al. 2006. Reconstructing the evolutionary history of polyploids from multi-labeled trees. *Mol. Biol. Evol.* 23, 1784–1791.

Huber, K.T., Spillner, A., Suchecki, R., et al. 2011. Metrics on multilabeled trees: Interrelationships and diameter bounds. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 8, 1029–1040.

Lott, M., Spillner, A., Huber, K.T. 2009a. PADRE: a package for analyzing and displaying reticulate evolution. *Bioinformatics.* 25, 1199–1200.

Lott, M., Spillner, A., Huber, K.T., et al. 2009b. Inferring polyploid phylogenies from multiply-labeled gene trees. *BMC Evol. Biol.* 9, 216.

Margush, T., and McMorris, F.R. 1981. Consensus *n*-Trees. Bull. Math. Biol. 43, 239–244.

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

Nelson, G., and Platnick, N. 1981. *Systematics and Biogeography: Cladistics and Vicariance*. Columbia University Press, New York.

Page, R.D.M. 1993. Parasites, phylogeny and cospeciation. *Int. J. Parasitol.* 23, 499–506.

Page, R.D.M. 1994. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst. Biol.* 43, 58–77.

Scornavacca, C., Berry, V. and Ranwez, V., 2011. Building species trees from larger parts of phylogenomic databases. *Information and Computation*. 209, 590–605.

Sokal, R.R., and Rohlf, F.J., 1981. Taxonomic congruence in the Leptopodomorpha re-examined. *Systematic Zoology*. 30, 309–325.

Sung, W.-K. 2010. *Algorithms in Bioinformatics: A Practical Introduction*. Chapman & Hall/CRC, Boca Raton, FL.

Wareham, H.T. 1985. An efficient algorithm for computing M_l consensus trees [B.Sc. Honours thesis]. Memorial University of Newfoundland, Newfoundland and Labrador. Canada.

Address correspondence to:
Wing-Kin Sung
School of Computing
National University of Singapore
13 Computing Drive
Singapore 117417
Singapore

E-mail: ksung@comp.nus.edu.sg