Algorithms for Building Consensus MUL-trees

Yun Cui[1], Jesper Jansson[2,⋆], and Wing-Kin Sung[1,3]

[1] National University of Singapore, 13 Computing Drive, Singapore 117417
`{yuncui01@gmail.com, ksung@comp.nus.edu.sg}`
[2] Ochanomizu University, 2-1-1 Otsuka, Bunkyo-ku, Tokyo 112-8610, Japan
`Jesper.Jansson@ocha.ac.jp`
[3] Genome Institute of Singapore, 60 Biopolis Street, Genome, Singapore 138672

Abstract. A MUL-tree is a generalization of a phylogenetic tree that allows the same leaf label to be used many times. Lott *et al.* [9,10] recently introduced the problem of inferring a so-called *consensus MUL-tree* from a set of conflicting MUL-trees and gave an exponential-time algorithm for a special greedy variant. Here, we study *strict* and *majority rule consensus MUL-trees*, and present the first ever polynomial-time algorithms for building a consensus MUL-tree. We give a simple, fast algorithm for building a strict consensus MUL-tree. We also show that although it is NP-hard to find a majority rule consensus MUL-tree, the variant which we call the *singular majority rule consensus MUL-tree* is unique and can be constructed efficiently.

1 Introduction

To describe tree-like evolutionary history, scientists often use a data structure known as the *phylogenetic tree* [3,17]. In traditional applications, phylogenetic trees were always distinctly leaf-labeled, and in fact, the computational efficiency of most existing methods for constructing and comparing phylogenetic trees implicitly depends on this uniqueness property. The *multi-labeled phylogenetic tree*, or *MUL-tree* for short, is a 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 Fig. 2 and 3. MUL-trees have applications in different research fields such as Molecular Systematics [10,14,15], Biogeography [4,12], the study of host-parasite cospeciation [13], and Computer Science [8].

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, given a multiset $\mathcal{S}$ of *splits* (bipartitions of a fixed multiset L of leaf labels), it is NP-hard to determine whether there exists an unrooted MUL-tree leaf labeled by L such that the multiset of all its splits is equal to $\mathcal{S}$,

---

[⋆] Funded by the Special Coordination Funds for Promoting Science and Technology and KAKENHI grant number 23700011.

T. Asano et al. (Eds.): ISAAC 2011, LNCS 7074, pp. 744–753, 2011.

whereas the corresponding problem for single-labeled trees is solvable in polynomial time [7]. As another example, given a set R of *rooted triplets* (single-labeled phylogenetic trees with exactly three leaves each), a classical algorithm by Aho *et al.* [1] can check if there exists a single-labeled phylogenetic tree that is consistent with all of the rooted triplets in R in polynomial time; on the other hand, it is NP-hard to decide if there exists a MUL-tree consistent with R having at most d leaf duplications, even if restricted to $d = 1$ [6]. In short, MUL-trees pose new and sometimes unexpected algorithmic challenges.

A *consensus tree* is a phylogenetic tree that summarizes the branching of a given set of (conflicting) phylogenetic trees. Different types of consensus trees for single-labeled trees, along with fast algorithms for constructing them, have been developed since the 1970's and are widely used today; see, e.g., [3,17]. The problem of constructing a *consensus MUL-tree* was introduced in [9,10], where an exponential-time algorithm was provided for a specific, greedy type of consensus MUL-tree.

1.1 Definitions

A rooted *multi-labeled phylogenetic tree*, or *MUL-tree* for short, 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. Fig. 2 and 3 show some examples. The multiset of all leaf labels that occur in a MUL-tree T is denoted by $\Lambda(T)$. For any multiset X and $x \in X$, 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 the multiset $\mathcal{C}(T) = \biguplus_{u \in V(T)} \{\Lambda(T[u])\}$. When a cluster C belongs to $\mathcal{C}(T)$, we say that T *contains* C or that C *occurs in* T. Thus, when a cluster C does not occur in a MUL-tree T, we have $mult_{\mathcal{C}(T)}(C) = 0$.

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 trees are the *strict consensus tree* [16] and the *majority rule consensus tree* [11]. We extend their definitions 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 $C \in \mathcal{C}(T)$, $mult_{\mathcal{C}(T)}(C) = \min_{1 \le i \le k} mult_{\mathcal{C}(T_i)}(C)$.
- A cluster that occurs in more than $k/2$ of the MUL-trees in $\mathcal{T}$ is 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 $C \uplus C \not\subseteq \Lambda(T)$. Note that if $C \in \mathcal{C}(T)$ is singular then $mult_{\mathcal{C}(T)}(C) = 1$ (but not the other way around).

- 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) all trivial clusters in $T_1, T_2, \ldots, T_k$; and (2) all singular clusters that occur in more than $k/2$ of the MUL-trees in $\mathcal{T}$.

1.2 Our Results and Organization of the Paper

From here on, let $\mathcal{T}$ be a given set of MUL-trees and L a fixed multiset of leaf labels with $\Lambda(T_i) = L$ for every $T_i \in \mathcal{T}$. Define $k = |\mathcal{T}|$ and $n = |L|$. Also, let q equal the number of distinct elements in L. In other words, $q \leq n$. We define $m = \max_{\ell \in L} mult_L(\ell)$ and call m the *multiplicity of* L.

The paper is organized as follows. Section 2 highlights some key properties of consensus MUL-trees. Next, we explain how to construct a strict consensus MUL-tree in $O(nqk)$ time in Section 3. Section 4 shows that constructing a majority rule consensus MUL-tree is NP-hard, even if restricted to instances where $k = 3$ and $m = 3$. However, the singular majority rule consensus MUL-tree admits an efficient algorithm running in $O(n^3 k)$ time, described in Section 5. To our knowledge, these are the first ever polynomial-time algorithms for building a consensus MUL-tree.

Our new results for strict and majority rule consensus MUL-trees, along with previously known results for single-labeled phylogenetic trees (corresponding to the case $m = 1$), are summarized in Fig. 1. Our results also hold for the analogous *unrooted* MUL-tree versions of the problems, with the same computational complexities. Due to space constraints, most proofs have been omitted from this conference version of the paper.

Strict consensus

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

Majority rule consensus

	$k = 2$	$k \geq 3$
$m = 1$	Always exists; always unique; $O(n)$ time. (Day [2])	Always exists; always unique; $O(n^2 + nk^2)$ time. (Wareham [18])
$m = 2$	Always exists; may not be unique; $O(nq)$ time. Sections 2 and 3	May not exist; may not be unique; unknown complexity.
$m \geq 3$	Always exists; may not be unique; $O(nq)$ time. Sections 2 and 3	May not exist; may not be unique; NP-hard. Sections 2 and 4

Fig. 1. The complexity of building strict and majority rule consensus MUL-trees. For $k = 2$, a strict consensus and a majority rule consensus MUL-tree are equivalent.

2 Preliminaries

It is possible for two non-isomorphic MUL-trees to have identical cluster collections. See T_1 and T_2 in Fig. 2 for an example. This property was first observed by Ganapathy *et al.* [4] for unrooted MUL-trees, and their example was simplified by Holm *et al.* [7]. (The example given here is the same as Fig. 1 (b)–(c) in [7], adapted to rooted MUL-trees.)

Define the *delete* operation on any non-root, 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. Note that any delete operation on a node u in T effectively removes one occurrence of a cluster from $\mathcal{C}(T)$, namely $\Lambda(T[u])$, without affecting the other clusters.

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.*

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 delete 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 non-uniqueness, consider $\mathcal{T} = \{T_1, T_2\}$ in Fig. 2. 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. For $k = 2$, a majority rule consensus MUL-tree of $\mathcal{T}$ is equivalent to a strict consensus MUL-tree of $\mathcal{T}$. If $k \geq 3$, the non-uniqueness and non-existence follow from the examples in Fig. 2 and 3. Hence:

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

Finally, we consider singular majority rule consensus MUL-trees. Let S be the set of all singular, non-trivial clusters that occur in at least $k/2$ of the MUL-trees in $\mathcal{T}$. 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 $t_{C'}$.

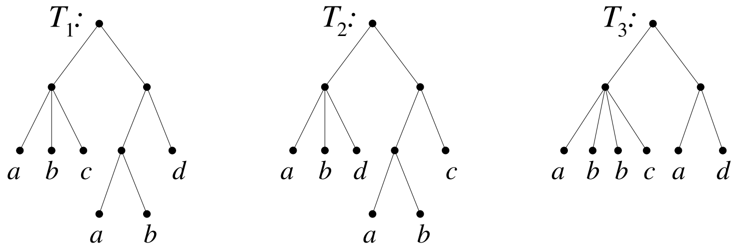

Fig. 2. Let T_1, T_2, T_3 be the three MUL-trees shown above with $\Lambda(T_1) = \Lambda(T_2) = \Lambda(T_3) = \{a, a, b, b, c, d\} = L$. Then $T_1 \neq T_2$ although $\mathcal{C}(T_1) = \mathcal{C}(T_2) = \{\{a\}, \{a\}, \{b\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, b, c\}, \{a, b, d\}, L\}$. Each of 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 $\{T_1, T_2, T_3\}$.

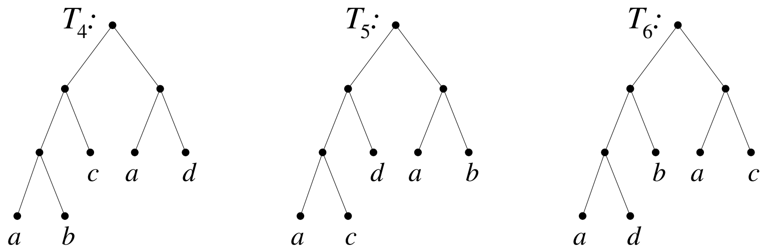

Fig. 3. Here, $\mathcal{T} = \{T_4, T_5, T_6\}$, $\Lambda(T_4) = \Lambda(T_5) = \Lambda(T_6) = \{a, a, b, c, d\} = L$. The non-trivial majority clusters are $\{\{a, b\}, \{a, c\}, \{a, d\}, \{a, a, b, c, d\}\}$. For any MUL-tree T that contains all these clusters, $mult_{\Lambda(T)}(a) \geq 3$ while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

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

Proof. The non-existence follows from the example in Fig. 3.

Next, we prove the uniqueness. For the sake of obtaining a contradiction, suppose there exist two different singular majority rule consensus MUL-trees A, B of $\mathcal{T}$. Since $A \neq B$, 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 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 $x \in L$.

3 Building a Strict Consensus MUL-tree

This section describes a simple algorithm for constructing a strict consensus MUL-tree. Our algorithm, named `Strict_consensus`, is essentially an implementation of the existence proof in Lemma 1. The basic strategy is to remove clusters from the cluster collection $\mathcal{C}(T_1)$ by 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 $\mathcal{T}$, as explained next. A *leaf label numbering function* is a bijection from the set of 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 such that, for every $j \in \{1, 2, \ldots, q\}$, the jth element equals $mult_{\Lambda(T_i[u])}(f^{-1}(j))$. 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 and $q-1$ 0's 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.

For each MUL-tree T_i in $\mathcal{T}$, `Strict_consensus` first computes all D_i^u-vectors by one bottom-up traversal of T_i and initializes a trie A_i augmented with leaf counters to store the cluster collection $\mathcal{C}(T_i)$. More precisely, the q elements of each D_i^u-vector are concatenated into a string of length q which is inserted into A_i. Next, for each cluster in T_1 (i.e., for each leaf in the trie A_1), the algorithm calculates how many of its occurrences to remove 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 $A_1, \ldots, A_k$ are used to retrieve these numbers efficiently. Finally, the necessary delete operations are performed on T_1.

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

4 Building a Majority Rule Consensus MUL-tree

This section demonstrates that constructing a majority rule consensus MUL-tree is computationally hard. Define the following decision problem:

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 SAT problem to MCMT. 1-IN-3 SAT is known to be NP-hard [5] and is defined as:

1-in-3 Satisfiability (1-IN-3 SAT):

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 to F such that each clause contains exactly one true literal?

First, define *non-mono-replace* on any Boolean formula F in 3-CNF as:

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

Below, we use the non-mono-replace operation to ensure that the Boolean formula we reduce from has a special structure. The relationship between F and the result of applying non-mono-replace on F is given by:

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.*

We now describe the reduction. Let F be any given Boolean formula in 3-CNF. As in the proof of Theorem 3.1 in [7], assume w.l.o.g. 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 non-mono-replace on F to obtain a Boolean formula F' with s variables and t clauses, for some positive integers s, 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'. Say that *x_i is positive (negative) in z_j* if x_i corresponds to a variable in F' that occurs positively (negatively) in jth clause. Define the leaf label multiset L for T_1, T_2, T_3 as $L = \{x, x : x \in X\} \cup \{z, z, z : z \in Z\}$. (In other words, 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 $\widetilde{\mathcal{W}} = \{\tilde{Z}_x \cup \{x\} : x \in X\}$. From $\mathcal{W}, \widetilde{\mathcal{W}}$, construct three MUL-trees T_1, T_2, T_3 with $\Lambda(T_1) = \Lambda(T_2) = \Lambda(T_3) = L$ whose sets of non-trivial clusters are: $\mathcal{W} \cup \widetilde{\mathcal{W}}$, $\mathcal{W} \cup \{X \cup Z\}$, and $\widetilde{\mathcal{W}} \cup \{X \cup Z\}$, respectively. Then, the set of non-trivial majority clusters for $\{T_1, T_2, T_3\}$ is: $\mathcal{W} \cup \widetilde{\mathcal{W}} \cup \{X \cup Z\}$. It is straightforward to show that T_1, T_2, T_3 are valid MUL-trees. Because of the non-mono-replace operation, for every $z_j \in Z$, there is exactly one or two subtrees attached to the root of T_2 (T_3) that contains an occurrence of z_j. The reduction's correctness follows from:

Lemma 5. *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.*

Theorem 2. *The MCMT problem is NP-hard, even if restricted to inputs where* $k = 3$ *and each leaf label occurs at most* 3 *times.*

5 Building a Singular Majority Rule Consensus MUL-tree

Here, we present a polynomial-time algorithm for building a singular majority rule consensus MUL-tree. By Lemma 3 in Section 2, when a singular majority rule consensus MUL-tree of $\mathcal{T}$ exists, it is unique.

Our algorithm consists of two phases. The first phase constructs the set S of all singular, non-trivial clusters that occur in at least $k/2$ of the MUL-trees in $\mathcal{T}$. To implement Phase 1, we enumerate all non-trivial clusters that occur in $\mathcal{T}$ and count their occurrences using the technique described in Section 3. The second phase builds the singular majority rule consensus tree of $\mathcal{T}$ by calling a top-down, recursive procedure `Build_MUL-tree`(L, S), listed in Fig. 4. 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 $C' \in S\}$. Then:

```
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 $S|C = \{X \in S : X \subseteq C\}$.
        If $S|C \neq \emptyset$, let $T_C$ = Build_MUL-tree($C$, $S|C$); otherwise, let $T_C = null$.
    endfor
4   Let $T$ be a MUL-tree whose root is attached to: (1) the root of $T_C$ for each
    $C \in \mathcal{F}$ with $T_C \neq null$; and (2) all leaves labeled by $L \setminus (\biguplus_{C \in \mathcal{F}} C)$.
5   return $T$
End Build_MUL-tree
```

Fig. 4. Algorithm `Build_MUL-tree`

Lemma 6. *$\mathcal{F} = \{C \in S : C$ 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 consensus MUL-tree of $\mathcal{T}$.*

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 C in $\mathcal{F}$ by recursively calling `Build_MUL-tree`$(C, S|C)$, where $S|C = \{X \in S : X \subseteq C\}$; the base case of the recursion is when $S|C = \emptyset$. 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.

`Build_MUL-tree` constructs a MUL-tree with $O(|L|)$ internal nodes. For each such node, it may need to execute all the steps of the procedure, which takes $O(|L||S|)$ time because $|\biguplus_{C \in \mathcal{F}} C| \leq |L|$. The total running time of Phase 2 is $O(|L|^2|S|) = O(n^3k)$ since $|L| = n$ and $|S| = O(nk)$.

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

References

1. Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM Journal on Computing 10, 405–421 (1981)
2. Day, W.H.E.: Optimal algorithms for comparing trees with labeled leaves. Journal of Classification 2(1), 7–28 (1985)
3. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Inc., Sunderland (2004)
4. Ganapathy, G., Goodson, B., Jansen, R., Le, H.-S., Ramachandran, V., Warnow, T.: Pattern identification in biogeography. IEEE/ACM Transactions on Computational Biology and Bioinformatics 3(4), 334–346 (2006)
5. Garey, M., Johnson, D.: Computers and Intractability – A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
6. Guillemot, S., Jansson, J., Sung, W.-K.: Computing a smallest multilabeled phylogenetic tree from rooted triplets. IEEE/ACM Transactions on Computational Biology and Bioinformatics 8(4), 1141–1147 (2011)
7. Huber, K.T., Lott, M., Moulton, V., Spillner, A.: The complexity of deriving multilabeled trees from bipartitions. J. of Comp. Biology 15(6), 639–651 (2008)
8. Huber, K.T., Spillner, A., Suchecki, R., Moulton, V.: Metrics on multilabeled trees: Interrelationships and diameter bounds. IEEE/ACM Transactions on Computational Biology and Bioinformatics 8(4), 1029–1040 (2011)
9. Lott, M., Spillner, A., Huber, K.T., Moulton, V.: PADRE: a package for analyzing and displaying reticulate evolution. Bioinformatics 25(9), 1199–1200 (2009)
10. Lott, M., Spillner, A., Huber, K.T., Petri, A., Oxelman, B., Moulton, V.: Inferring polyploid phylogenies from multiply-labeled gene trees. BMC Evolutionary Biology 9, 216 (2009)

11. Margush, T., McMorris, F.R.: Consensus n-Trees. Bulletin of Mathematical Biology 43(2), 239–244 (1981)
12. Nelson, G., Platnick, N.: Systematics and Biogeography: Cladistics and Vicariance. Columbia University Press (1981)
13. Page, R.D.M.: Parasites, phylogeny and cospeciation. International Journal for Parasitology 23, 499–506 (1993)
14. Page, R.D.M.: Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. Systematic Biology 43(1), 58–77 (1994)
15. Scornavacca, C., Berry, V., Ranwez, V.: Building species trees from larger parts of phylogenomic databases. Information and Computation 209(3), 590–605 (2011)
16. Sokal, R.R., Rohlf, F.J.: Taxonomic congruence in the Leptopodomorpha reexamined. Systematic Zoology 30(3), 309–325 (1981)
17. Sung, W.-K.: Algorithms in Bioinformatics: A Practical Introduction. Chapman & Hall/CRC (2010)
18. Wareham, H.T.: An efficient algorithm for computing Mi consensus trees. B.Sc. Honours thesis, Memorial University of Newfoundland, Canada (1985)