Constructing the R* Consensus Tree of Two Trees in Subcubic Time

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

[1] Ochanomizu University, 2-1-1 Otsuka, Bunkyo-ku, Tokyo 112-8610, Japan
Jesper.Jansson@ocha.ac.jp
[2] School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543
ksung@comp.nus.edu.sg
[3] Genome Institute of Singapore, 60 Biopolis Street, Genome, Singapore 138672

Abstract. The previously fastest algorithms for computing the R* consensus tree of two given (rooted) phylogenetic trees T_1 and T_2 with a leaf label set of cardinality n run in $\Theta(n^3)$ time [3,8]. In this manuscript, we describe a new $O(n^2 \log n)$-time algorithm to solve the problem. This is a significant improvement because the R* consensus tree is defined in terms of a set $\mathcal{R}_{maj}$ which may contain $\Omega(n^3)$ elements, so any direct approach which explicitly constructs $\mathcal{R}_{maj}$ requires $\Omega(n^3)$ time.

1 Introduction

Phylogenetic trees are leaf-labeled trees which are commonly used to describe the evolutionary history of a set of objects such as biological species or languages [5,6,10]. Typically, in a phylogenetic tree, each leaf represents one of the objects being studied and the branching structure of the tree indicates the assumed evolutionary relationships among the objects. A *consensus tree* is a single phylogenetic tree which summarizes the branching information contained in an input collection of phylogenetic trees with identical leaf label sets. Consensus trees are useful when different data sets or different tree inference methods have produced a set of trees with the same leaf label set and slightly conflicting structures, yet a single tree is required to represent all of them [5]. Also, by exclusion, a consensus tree indicates areas of conflict in the input trees [2]. Furthermore, consensus trees are sometimes used as a basis for new phylogenetic inferences [2].

Many types of consensus trees have been defined and studied in detail. For a survey, see, e.g., [2]. Different types of consensus trees use different criteria to resolve conflicts among the input trees, so their mathematical properties vary. Therefore, the most suitable type of consensus tree to use in practice depends on the particular application. In this paper, we consider the so-called *R* consensus tree*. One advantage of the R* consensus tree is that it provides a statistically consistent estimator of the species tree topology when combining a set of gene

⋆ Funded by the Special Coordination Funds for Promoting Science and Technology, Japan.

M. de Berg and U. Meyer (Eds.): ESA 2010, Part I, LNCS 6346, pp. 573–584, 2010.

trees, as recently demonstrated by Degnan *et al.* [4]; moreover, R* consensus outperformed other methods such as majority-rule consensus in the study conducted by [4]. For the case of two input trees, it is known that the R* consensus tree is equivalent to the *RV-III tree* introduced in [8].

The R* consensus tree is defined formally in Section 2 below. In short, given a set of input trees on a leaf label set L, if the rooted triplet $xy|z$ for any $\{x, y, z\} \subseteq L$ is consistent with more input trees than each of the two rooted triplets $xz|y$ and $yz|x$ is, then $xy|z$ belongs to a set named $\mathcal{R}_{maj}$. The R* consensus tree is the tree τ having the largest possible number of internal nodes such that every rooted triplet consistent with τ belongs to $\mathcal{R}_{maj}$.

The goal of this paper is to develop a fast algorithm for constructing the R* consensus tree for two input trees T_1 and T_2 with a leaf label set L of cardinality n. The previous algorithms for this problem require $\Theta(n^3)$ time [3,8]. Our main result is to reduce the time complexity to subcubic. When T_1 and T_2 have similar branching structures, $|\mathcal{R}_{maj}| = \Omega(n^3)$. Hence, in order to obtain a fast algorithm, we have to avoid explicitly constructing the set $\mathcal{R}_{maj}$. For this purpose, we use an alternative formulation based on distances from leaves to lowest common ancestors of pairs of leaves to compute the Apresjan clusters of a function $s_{\mathcal{R}_{maj}}$ associated to $\mathcal{R}_{maj}$. Then, we find the strong clusters of $\mathcal{R}_{maj}$ which are subsequently used to build the R* consensus tree. The running time of our new algorithm `R*_consensus_tree` (described in Section 3) is $O(n^2 \log n)$.

2 Preliminaries

2.1 Basic Definitions

A *phylogenetic tree* is a rooted, unordered, distinctly leaf-labeled tree in which every internal node has at least two children. From here on, "tree" means "phylogenetic tree", and every leaf in a tree is identified with its label.

We shall use the following terminology and notation. For any tree T and any internal node u of T, the subtree of T rooted at u is denoted by $T[u]$. The set of all leaves in a tree T is written as $\Lambda(T)$. For any two nodes u and v in a tree T, the *lowest common ancestor of u and v in T* is denoted by $lca^T(u, v)$. A *triplet* is a tree with exactly three leaves. Any *non-binary* tree with exactly three leaves $\{x, y, z\}$ is called a *fan triplet* and is written as $x|y|z$. On the other hand, any *binary* tree with exactly three leaves $\{x, y, z\}$ is called a *rooted triplet*, and is denoted by $xy|z$ if the lowest common ancestor of x and y is a proper descendant of the lowest common ancestor of x and z. Note that there are precisely four different triplets for any set of three leaf labels $\{x, y, z\}$, namely $x|y|z$, $xy|z$, $xz|y$, and $yz|x$ (see Fig. 1).

For any tree T and $\{x, y, z\} \subseteq \Lambda(T)$, the fan triplet $x|y|z$ is said to be *consistent with* T if $lca^T(x, y) = lca^T(x, z) = lca^T(y, z)$. Similarly, the rooted triplet $xy|z$ is *consistent with* T if $lca^T(x, y)$ is a proper descendant of $lca^T(x, z) = lca^T(y, z)$. Let $T||_{\{x,y,z\}}$ denote the unique fan triplet or rooted triplet with leaf label set $\{x, y, z\}$ which is consistent with T. Finally, for any tree T, let $r(T)$ be the set of all rooted triplets which are consistent with T, i.e., define

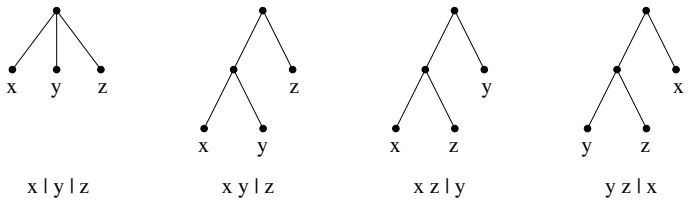

Fig. 1. The four different triplets $x|y|z$, $xy|z$, $xz|y$, and $yz|x$ leaf-labeled by $\{x, y, z\}$

$r(T) = \{T||_{\{x,y,z\}} : \{x, y, z\} \subseteq \Lambda(T)$ and $T||_{\{x,y,z\}}$ is not a fan triplet$\}$, and define $t(T)$ as the set of *all* triplets (rooted triplets as well as fan triplets) consistent with T, i.e., $t(T) = \{T||_{\{x,y,z\}} : \{x, y, z\} \subseteq \Lambda(T)\}$. It follows that $|t(T)| = \Theta(|\Lambda(T)|^3)$ for any tree T, and $|r(T)| = \Theta(|\Lambda(T)|^3)$ when T is a binary tree because $|r(T)| = |t(T)|$ in this case.

2.2 Strong Clusters and Apresjan Clusters

Below, let $\mathcal{R}$ be a given set of triplets over a leaf label set $L = \bigcup_{r\in\mathcal{R}} \Lambda(r)$ such that for each $\{x, y, z\} \subseteq L$, at most one of $x|y|z$, $xy|z$, $xz|y$, and $yz|x$ belongs to $\mathcal{R}$. A *cluster of* L is any subset of L. We define two special types of clusters:

- A cluster A of L is called a *strong cluster of* $\mathcal{R}$ if $aa'|x \in \mathcal{R}$ for all $a, a' \in A$ and $x \in L \setminus A$. Furthermore, L as well as every singleton set of L is also defined to be a strong cluster of $\mathcal{R}$.
- For each $a, b \in L$, define $s_{\mathcal{R}}(a, b) = |\{ab|y : ab|y \in \mathcal{R}\}|$. A cluster A of L is called an *Apresjan cluster of* $s_{\mathcal{R}}$ if $s_{\mathcal{R}}(a, a') > s_{\mathcal{R}}(a, x)$ for all $a, a' \in A$ and $x \in L \setminus A$.

Write $n = |L|$. By Theorem 2.3 and Corollary 2.1 of [3], the following holds:

Lemma 1. *(Bryant and Berry [3].)*

1. *There are $O(n)$ Apresjan clusters of $s_{\mathcal{R}}$.*
2. *Given the values of $s_{\mathcal{R}}(a, b)$ for all $a, b \in L$, the Apresjan clusters of $s_{\mathcal{R}}$ can be computed in $O(n^2)$ time.*

There is a relationship between the strong clusters of $\mathcal{R}$ and the Apresjan clusters of $s_{\mathcal{R}}$:

Lemma 2. *Every strong cluster of $\mathcal{R}$ is an Apresjan cluster of $s_{\mathcal{R}}$.*

Proof. Let C be a strong cluster of $\mathcal{R}$. Consider any fixed $a, a' \in C$ and $x \in L\setminus C$. For every $y \in L\setminus C$, we have $aa'|y \in \mathcal{R}$ by the definition of a strong cluster, which gives $s_{\mathcal{R}}(a, a') = |\{aa'|y : aa'|y \in \mathcal{R}\}| \geq |L \setminus C|$. In the same way, $ab|x \in \mathcal{R}$ holds for every $b \in C$, which (along with the requirement that for each $\{x, y, z\} \subseteq L$, at most one of $x|y|z$, $xy|z$, $xz|y$, and $yz|x$ belongs to $\mathcal{R}$) implies that $ax|b \notin \mathcal{R}$. Thus, $s_{\mathcal{R}}(a, x) = |\{ax|y : ax|y \in \mathcal{R}\}| \leq |(L \setminus C) \setminus \{x\}| < |L \setminus C|$. We have just shown that $s_{\mathcal{R}}(a, a') > s_{\mathcal{R}}(a, x)$, so C is an Apresjan cluster of $s_{\mathcal{R}}$. □

2.3 R* Consensus Trees

Let T_1 and T_2 be two given trees with $\Lambda(T_1) = \Lambda(T_2) = L$. For any $\{a, b, c\} \subseteq L$, define $\#ab|c$ as the number of trees T_i for which $ab|c \in r(T_i)$. The set of "majority rooted triplets" $\mathcal{R}_{maj}$ is defined as $\{ab|c : a, b, c \in L \text{ and } \#ab|c > \#ac|b, \#bc|a\}$. An *R* consensus tree of* T_1 *and* T_2 is a tree τ with $\Lambda(\tau) = L$ which satisfies $r(\tau) \subseteq \mathcal{R}_{maj}$ and which maximizes the number of internal nodes.

The next two lemmas describe some useful properties of the strong clusters of $\mathcal{R}_{maj}$.

Lemma 3. *Let T be a tree with $\Lambda(T) = L$ and $r(T) \subseteq \mathcal{R}_{maj}$. For any node u of T, $\Lambda(T[u])$ is a strong cluster of $\mathcal{R}_{maj}$.*

Proof. If u is a leaf or the root of T then $\Lambda(T[u])$ is trivially a strong cluster of $\mathcal{R}_{maj}$. If u is an internal node then for any two $a, a' \in \Lambda(T[u])$ and any $x \notin \Lambda(T[u])$, the triplet $aa'|x$ belongs to $\mathcal{R}_{maj}$, so $\Lambda(T[u])$ is a strong cluster of $\mathcal{R}_{maj}$. □

Lemma 4. *There exists a tree τ such that the set $\{\Lambda(\tau[u]) : u$ is a node in $\tau\}$ equals the set of strong clusters of $\mathcal{R}_{maj}$.*

Proof. First observe that for any two strong clusters A and B of $\mathcal{R}$, either $A \subsetneq B$, $B \subsetneq A$, or $A \cap B = \emptyset$. To prove this claim, suppose for the sake of contradiction that there are $x, y, z \in L$ such that $x \in A \setminus B$, $y \in B \setminus A$, and $z \in A \cap B$. Since A is a strong cluster, $xz|y \in \mathcal{R}$. Since B is a strong cluster, $yz|x \in \mathcal{R}$. By the definition of $\mathcal{R}_{maj}$, we cannot have both of $xz|y$ and $yz|x$ in $\mathcal{R}_{maj}$. The claim follows. This implies that the strong clusters of $\mathcal{R}_{maj}$ form a hierarchy (laminar family) on L.

Next, if A and B are strong clusters of $\mathcal{R}_{maj}$ with $A \subsetneq B$ then there must exist some strong cluster C of $\mathcal{R}_{maj}$ such that $C \subsetneq B$ and $C \cap A = \emptyset$. (To see this, take any $c \in B \setminus A$ and recall that $\{c\}$ is by definition a strong cluster of $\mathcal{R}_{maj}$.)

By these two observations, there exists a tree τ leaf-labeled by L such that: (1) each strong cluster A of $\mathcal{R}_{maj}$ corresponds to a node in τ whose set of descendants is precisely A; (2) $A \subsetneq B$, where A and B are strong clusters of $\mathcal{R}_{maj}$, implies that the node corresponding to A is a proper descendant of the node corresponding to B; and (3) every internal node of τ has at least two children. Hence, the lemma follows. □

Say that a tree T *includes* a cluster A of L if T contains a node u such that $\Lambda(T[u]) = A$.

Theorem 1. *The R* consensus tree of T_1 and T_2 exists and is unique. In particular, it includes every strong cluster of $\mathcal{R}_{maj}$ and no other clusters of L.*

Proof. By Lemma 4, there exists a (unique) tree τ that includes all the strong clusters of $\mathcal{R}_{maj}$ and does not include any other clusters. For any $aa'|x \in r(\tau)$, there exists a node u in τ such that $a, a' \in \Lambda(\tau[u])$ and $x \notin \Lambda(\tau[u])$, i.e., $a, a' \in A$

and $x \notin A$ for some strong cluster A of $\mathcal{R}_{maj}$, which implies that $aa'|x \in \mathcal{R}_{maj}$. Thus, $r(\tau) \subseteq \mathcal{R}_{maj}$.

To prove the optimality of τ, suppose that there exists a tree τ' which satisfies $r(\tau') \subseteq \mathcal{R}_{maj}$ and has more internal nodes than τ. By Lemma 3, the set of leaves in each rooted subtree of τ' forms a strong cluster of $\mathcal{R}_{maj}$. Since τ' has more internal nodes than τ, it follows that τ' includes some strong cluster of $\mathcal{R}_{maj}$ which τ does not include. This contradicts Lemma 4. Hence, τ is an R* consensus tree of T_1 and T_2. □

3 Constructing the R* Consensus Tree

Based on the discussion in Sections 2.2 and 2.3, we can construct the R* consensus tree of two given trees T_1 and T_2 as follows: First compute $s_{\mathcal{R}_{maj}}$ and all the Apresjan clusters of $s_{\mathcal{R}_{maj}}$.[1] Then, check each Apresjan cluster to see if it is a strong cluster of $\mathcal{R}_{maj}$ (by Lemma 2, the set of strong clusters of $\mathcal{R}_{maj}$ is a subset of the set of Apresjan clusters of $s_{\mathcal{R}_{maj}}$). Finally, build a tree which includes all the strong clusters of $\mathcal{R}_{maj}$ in accordance with Theorem 1. The algorithm is named `R*_consensus_tree` and is outlined in Fig. 2.

Algorithm `R*_consensus_tree`
Input: Two trees T_1, T_2 with $\Lambda(T_1) = \Lambda(T_2)$.
Output: The R* consensus tree of T_1 and T_2.

1: Define $L = \Lambda(T_1) = \Lambda(T_2)$ and compute $s_{\mathcal{R}_{maj}}(a, b)$ for all $a, b \in L$ as described in Section 4.
2: Compute the Apresjan clusters of $s_{\mathcal{R}_{maj}}$.
3: **for** each Apresjan cluster A of $s_{\mathcal{R}_{maj}}$ **do**
4: Determine if A is a strong cluster of $\mathcal{R}_{maj}$ as described in Section 5.
5: **end for**
6: Construct the R* consensus tree using all the strong clusters of $\mathcal{R}_{maj}$.

Fig. 2. Algorithm `R*_consensus_tree`

We now analyze the time complexity of Algorithm `R*_consensus_tree`. In Section 4 below, we shall describe how to implement step 1 in $O(n^2 \log n)$ time. Next, in step 2, the Apresjan clusters of $s_{\mathcal{R}_{maj}}$ can be computed by running the algorithm of Bryant and Berry [3], which takes $O(n^2)$ time according to Lemma 1 (see also Corollary 2.1 in [3]). There are $O(n)$ Apresjan clusters in the loop of step 3 by Lemma 1, and each one is checked in $O(n)$ time in step 4, as detailed in Section 5 below. Finally, the last step builds the R* consensus tree from the strong clusters of $\mathcal{R}_{maj}$ in $O(n^2)$ time. In summary, we have the following theorem.

[1] Recall from Section 2.2 that for a set $\mathcal{R}$ of triplets over a leaf label set L, the function $s_{\mathcal{R}}$ is defined on each $a, b \in L$ by $s_{\mathcal{R}}(a, b) = |\{ab|y : ab|y \in \mathcal{R}\}|$.

Theorem 2. *Algorithm* `R*_consensus_tree` *constructs the R^* consensus tree of T_1 and T_2 in $O(n^2 \log n)$ time.*

The following two sections are devoted to implementing steps 1 and 4 of Algorithm `R*_consensus_tree` efficiently.

4 Computing $s_{\mathcal{R}_{maj}}(a, b)$ for all $a, b \in L$

From the definition of $\mathcal{R}_{maj}$, we have:

Lemma 5. *For any $a, b, c \in L$, $ab|c \in \mathcal{R}_{maj}$ if and only if either*

1. *$ab|c \in t(T_1) \cap t(T_2)$; or*
2. *$ab|c \in t(T_1)$ and $a|b|c \in t(T_2)$; or*
3. *$a|b|c \in t(T_1)$ and $ab|c \in t(T_2)$.*

We introduce the following three auxiliary functions:

$$\begin{cases} count_1(a,b) = |\{w \in L \setminus \{a,b\} \,:\, ab|w \in t(T_1) \cap t(T_2)\}| \\ count_2(a,b) = |\{w \in L \setminus \{a,b\} \,:\, ab|w \in t(T_1),\ a|b|w \in t(T_2)\}| \\ count_3(a,b) = |\{w \in L \setminus \{a,b\} \,:\, a|b|w \in t(T_1),\ ab|w \in t(T_2)\}| \end{cases}$$

Then, Lemma 5 immediately gives $s_{\mathcal{R}_{maj}}(a,b) = count_1(a,b) + count_2(a,b) + count_3(a,b)$ for any $a, b \in L$.

To compute $count_1$, $count_2$, and $count_3$, we could preprocess T_1 and T_2 in $O(n)$ time so that lowest common ancestor queries in T_1 and T_2 can be answered in $O(1)$ time [1,7]. Then, for any given $a, b \in L$, a brute-force solution would easily obtain all of $count_1(a,b)$, $count_2(a,b)$, and $count_3(a,b)$ in $O(n)$ time by checking $T_1||_{\{a,b,w\}}$ and $T_2||_{\{a,b,w\}}$ for each $w \in L \setminus \{a,b\}$. This approach would therefore compute $count_i(a,b)$ for all $a, b \in L$ and all $i = 1, 2, 3$ in $O(n^3)$ time. In the rest of this section, we show how to obtain $count_i(a,b)$ for all $a, b \in L$ and $i = 1, 2, 3$ more efficiently.

First, in Section 4.1, we reformulate the $count_i(a,b)$-functions in terms of distances from leaves to lowest common ancestors of leaves. All of these distances may be computed in $O(n^2)$ time. Then, based on this alternative formulation, for any fixed leaf label $a \in L$, Section 4.2 describes an $O(n \log n)$-time method for computing $count_1(a,b)$ for all $b \in L \setminus \{a\}$, and Section 4.3 describes an $O(n)$-time method for computing $count_2(a,b)$ for all $b \in L \setminus \{a\}$. (By symmetry, we can obtain $count_3(a,b)$ for all $b \in L \setminus \{a\}$ in $O(n)$ time with the same technique as in Section 4.3.) To summarize, after $O(n^2)$ time preprocessing, $O(n \log n)$ time is enough to compute $count_1(a,b)$, $count_2(a,b)$, and $count_3(a,b)$ for all $b \in L \setminus \{a\}$ for any fixed $a \in L$. Therefore, we only need $O(n^2 \log n)$ time in total to obtain $s_{\mathcal{R}_{maj}}(a,b)$ for all $a, b \in L$.

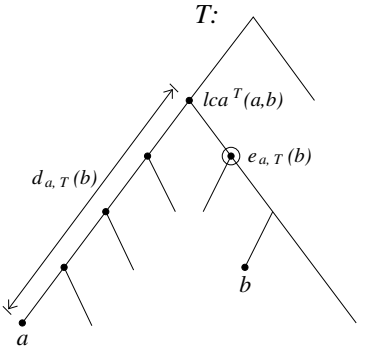

Fig. 3. Illustrating the definitions of $d_{a,T}(b)$ and $e_{a,T}(b)$

4.1 Distances from Leaves to Lowest Common Ancestors

For any tree T and any $a, b \in \Lambda(T)$, let $d_{a,T}(b)$ be the distance between a and $lca^T(a,b)$ in T, and let $e_{a,T}(b)$ be the child of $lca^T(a,b)$ which is an ancestor of b. (W.l.o.g., define $e_{a,T}(a) = \emptyset$.) See Fig. 3 for an example. Note that the values of $d_{a,T}(b)$ and $e_{a,T}(b)$ for all $a, b \in \Lambda(T)$ can be computed in $O(n^2)$ time by a bottom-up traversal. The next lemma states the connection between d and e and the triplets consistent with T.

Lemma 6. *For any tree T and any $a, x, w \in \Lambda(T)$, it holds that:*

- *If $d_{a,T}(x) < d_{a,T}(w)$ then $ax|w \in t(T)$.*
- *If $d_{a,T}(x) = d_{a,T}(w)$ and $e_{a,T}(x) \neq e_{a,T}(w)$ then $a|x|w \in t(T)$.*

Proof. If $d_{a,T}(x) < d_{a,T}(w)$, the $lca^T(a,w)$ is an ancestor of $lca^T(a,x)$. Thus, $ax|w \in t(T)$.

If $d_{a,T}(x) = d_{a,T}(w)$ then $v = lca^T(a,w) = lca^T(a,x)$. Since $e_{a,T}(x) \neq e_{a,T}(w)$, we know that $lca^T(x,w) = v$. Hence, $a|x|w \in t(T)$. □

We remark that the first part of Lemma 6 is a special case of Theorem 1 in [9], which was derived by Lee *et al.* [9] to solve a different problem known as *the maximum agreement subtree problem.*

Next, consider two trees T_1 and T_2 with $\Lambda(T_1) = \Lambda(T_2) = L$. We have:

Lemma 7. *For any $a, b, w \in L$:*

- $count_1(a,b) = |\{w : d_{a,T_1}(b) < d_{a,T_1}(w)$ *and* $d_{a,T_2}(b) < d_{a,T_2}(w)\}|$.
- $count_2(a,b) = |\{w : d_{a,T_1}(b) < d_{a,T_1}(w)$, $d_{a,T_2}(b) = d_{a,T_2}(w)$, *and* $e_{a,T_2}(b) \neq e_{a,T_2}(w)\}|$.
- $count_3(a,b) = |\{w : d_{a,T_1}(b) = d_{a,T_1}(w)$, $e_{a,T_1}(b) \neq e_{a,T_1}(w)$, *and* $d_{a,T_2}(b) < d_{a,T_2}(w)\}|$.

Proof. By Lemma 6, $d_{a,T_1}(b) < d_{a,T_1}(w)$ and $d_{a,T_2}(b) < d_{a,T_2}(w)$ mean that $ab|w \in t(T_1) \cap t(T_2)$. Hence, $count_1(a,b) = |\{w : d_{a,T_1}(b) < d_{a,T_1}(w)$ and $d_{a,T_2}(b) < d_{a,T_2}(w)\}|$.

By Lemma 6 again, $d_{a,T_1}(b) < d_{a,T_1}(w)$, $d_{a,T_2}(b) = d_{a,T_2}(w)$, and $e_{a,T_2}(b) \neq e_{a,T_2}(w)$ mean that $ab|w \in t(T_1)$ and $a|b|w \in t(T_2)$. Hence, $count_2(a,b) = |\{w : d_{a,T_1}(b) < d_{a,T_1}(w)$, $d_{a,T_2}(b) = d_{a,T_2}(w)$, and $e_{a,T_2}(b) \neq e_{a,T_2}(w)\}|$.

The formula for $count_3(a,b)$ follows by symmetry. □

4.2 Computing $count_1(a,b)$ for All $b \in L \setminus \{a\}$

This section describes how to compute $count_1(a,b)$ for all $b \in L \setminus \{a\}$ in $O(n \log n)$ time for any fixed $a \in L$, assuming the values of $d_{a,T_1}(b)$, $d_{a,T_2}(b)$ for all $b \in L \setminus \{a\}$ have been precomputed.

By applying a stable sorting, all elements in $L \setminus \{a\}$ can be ordered using $O(n)$ time so that x is before y if either: (1) $d_{a,T_1}(x) < d_{a,T_1}(y)$; or (2) $d_{a,T_1}(x) = d_{a,T_1}(y)$ and $d_{a,T_2}(x) < d_{a,T_2}(y)$. Suppose the resulting ordering of $L \setminus \{a\}$ is $x_{(1)}, x_{(2)}, \ldots, x_{(n-1)}$. For any $x_{(i)}$, let i' be the smallest integer which is larger than i such that $d_{a,T_1}(x_{(i')}) \neq d_{a,T_1}(x_{(i)})$. Then, $count_1$ obeys the following:

Lemma 8. *For any $i \in \{1, 2, \ldots, n-1\}$, $count_1(a, x_{(i)})$ equals the number of elements in $\{d_{a,T_2}(x_{(j)}) : j \geq i'\}$ which are strictly larger than $d_{a,T_2}(x_{(i)})$.*

Proof. From Lemma 7, $count_1(a, x_{(i)}) = |\{x_{(j)} : d_{a,T_1}(x_{(i)}) < d_{a,T_1}(x_{(j)})$ and $d_{a,T_2}(x_{(i)}) < d_{a,T_2}(x_{(j)})\}|$. By the definition of i', $count_1(a, x_{(i)}) = |\{x_{(j)} : j \geq i'$ and $d_{a,T_2}(x_{(i)}) < d_{a,T_2}(x_{(j)})\}|$. The lemma follows. □

We compute $count_1(a, x_{(i)})$ for all i in decreasing order from $n-1$ to 1 iteratively, as illustrated in Algorithm `Compute_`$count_1$ in Fig. 4. We maintain the invariant

Algorithm `Compute_`$count_1$
Input: $a \in L$.
Output: The values of $count_1(a,b)$ for all $b \in L \setminus \{a\}$.

1: Perform a stable sorting to rank the elements of $L \setminus \{a\}$ according to d_{a,T_1} and d_{a,T_2}, and obtain the ordering $x_{(1)}, x_{(2)}, \ldots, x_{(n-1)}$.
2: Let D be an empty binary search tree.
3: Let $i' = n$ and define $d_{a,T_1}(x_{(n)}) = d_{a,T_2}(x_{(n)}) = \infty$.
4: **for** $i = n-1$ downto 1 **do**
5: **if** $d_{a,T_1}(x_{(i)}) < d_{a,T_1}(x_{(i+1)})$ **then**
6: Insert $d_{a,T_2}(x_{(j)})$ into D for $j = i+1, \ldots, i'-1$.
7: Let $i' = i+1$.
8: **end if**
9: Set $count_1(a, x_{(i)})$ to the number of elements in D which are strictly greater than $d_{a,T_2}(x_{(i)})$.
10: **end for**

Fig. 4. Algorithm `Compute_`$count_1$

that in every iteration i, the value i' equals the smallest integer larger than i such that $d_{a,T_1}(x_{(i')}) \neq d_{a,T_1}(x_{(i)})$, and keep a binary search tree D for $\{d_{a,T_2}(x_{(j)}) : j \geq i'\}$ so that by Lemma 8, $count_1(a, x_{(i)})$ equals the number of elements in D strictly greater than $d_{a,T_2}(x_{(i)})$. Each operation on a binary search tree takes $O(\log n)$ time, so in total, `Compute_`$count_1$ runs in $O(n \log n)$ time.

4.3 Computing $count_2(a, b)$ for All $b \in L \setminus \{a\}$

Here, we show how to compute $count_2(a, b)$ (and by symmetry, $count_3(a, b)$) for all $b \in L \setminus \{a\}$ in $O(n)$ time for any fixed $a \in L$. We assume that the values of $d_{a,T_1}(b)$, $d_{a,T_2}(b)$, $e_{a,T_1}(b)$, $e_{a,T_2}(b)$ for all $b \in L \setminus \{a\}$ have been precomputed.

The algorithm is named `Compute_`$count_2$ and is listed in Fig. 5. It first partitions $L \setminus \{a\}$ into disjoint subsets $C_1, C_2, \ldots, C_p$ in such a way that for every pair of elements x, y in the same C_i, it holds that $d_{a,T_2}(x) = d_{a,T_2}(y)$. Then, the algorithm further partitions each C_i into $C_{i1}, \ldots, C_{iq}$ so that for every pair of elements x, y in the same C_{ij}, it holds that $e_{a,T_2}(x) = e_{a,T_2}(y)$. Finally, the algorithm obtains $count_2(a, x)$ for all $x \in L \setminus \{a\}$ in $O(n)$ time based on Lemma 9.

Lemma 9. *Let $b \in L \setminus \{a\}$ and suppose $b \in C_{ij}$. Then, $count_2(a, b) = |\{w \in C_i : d_{a,T_1}(w) > d_{a,T_1}(b)\}| - |\{w \in C_{ij} : d_{a,T_1}(w) > d_{a,T_1}(b)\}|$.*

Proof. By Lemma 7, $count_2(a, b) = |\{w : d_{a,T_1}(b) < d_{a,T_1}(w), d_{a,T_2}(b) = d_{a,T_2}(w), \text{ and } e_{a,T_2}(b) \neq e_{a,T_2}(w)\}|$. Since $b \in C_{ij}$, we have: (1) $w \in C_i$ means $d_{a,T_2}(b) = d_{a,T_2}(w)$; and (2) $w \in C_{ij}$ means $e_{a,T_2}(b) = e_{a,T_2}(w)$. Therefore, $count_2(a, b) = |\{w : d_{a,T_1}(b) < d_{a,T_1}(w), w \in C_i, \text{ and } w \notin C_{ij}\}|$. □

5 Determining If a Cluster Is a Strong Cluster

This section shows how to check whether a given cluster A of L is a strong cluster of $\mathcal{R}_{maj}$ in $O(n)$ time. Our solution depends on the following lemma.

Lemma 10. *Let u_1 and u_2 be the lowest common ancestor of A in T_1 and T_2, respectively. A is a strong cluster of $\mathcal{R}_{maj}$ if and only if:*

(1) For $i = 1, 2$, each subtree B attached to u_i in T_i satisfies either $\Lambda(B) \subseteq A$ or $\Lambda(B) \subseteq L \setminus A$; and
(2) $X_1 \cap X_2 = \emptyset$, where $X_i = (L \setminus A) \cap \Lambda(T_i[u_i])$.

Proof. ($\Rightarrow$) Let A be a strong cluster of $\mathcal{R}_{maj}$. We need to prove that both conditions (1) and (2) hold.

For the sake of obtaining a contradiction, suppose that for some $i \in \{1, 2\}$, there exist $a \in A$, $x \in L \setminus A$ where both a and x are in the same subtree attached to u_i. Then, $lca^{T_i}(a, x)$ is a proper descendant of u_i. The node u_i is defined as the lowest common ancestor of the leaves in A, so $lca^{T_i}(a, a') = u_i$ for some $a' \in A$. However, then $lca^{T_i}(a, x)$ is a proper descendant of $lca^{T_i}(a, a')$, giving $ax|a' \in t(T_i)$. This implies that $aa'|x$ cannot belong to $\mathcal{R}_{maj}$, which contradicts the fact that A is a strong cluster of $\mathcal{R}_{maj}$. Thus, condition (1) must hold.

Algorithm Compute_$count_2$
Input: $a \in L$.
Output: The values of $count_2(a, b)$ for all $b \in L \setminus \{a\}$.

1: Partition $L \setminus \{a\}$ into $C_1, \ldots, C_p$ so that for every pair of elements x, y in the same C_i, it holds that $d_{a,T_2}(x) = d_{a,T_2}(y)$.
2: **for** every C_i **do**
3: Perform a stable sorting to rank the elements of C_i and obtain the ordering $b_1, \ldots, b_{|C_i|}$ such that $d_{a,T_1}(b_1) \leq d_{a,T_2}(b_2) \leq \ldots \leq d_{a,T_2}(b_{|C_i|})$.
4: { Note that $s_i(b) = |\{w \in C_i : d_{a,T_1}(w) > d_{a,T_1}(b)\}|$. }
5: $s_i(b_1) = 0$.
6: **for** $j = 2, 3, \ldots, |C_i|$ **do**
7: **if** $d_{a,T_1}(b_j) > d_{a,T_1}(b_{j-1})$ **then**
8: $s_i(b_j) = s_i(b_{j-1}) + 1$.
9: **else**
10: $s_i(b_j) = s_i(b_{j-1})$.
11: **end if**
12: **end for**
13: Partition C_i into $C_{i1}, \ldots, C_{iq}$ so that for every pair of elements x, y in the same C_{ij}, it holds that $e_{a,T_2}(x) = e_{a,T_2}(y)$.
14: **for** every C_{ij} **do**
15: Perform a stable sorting to rank the elements of C_{ij} and obtain the ordering $b_1, \ldots, b_{|C_{ij}|}$ such that $d_{a,T_1}(b_1) \leq d_{a,T_2}(b_2) \leq \ldots \leq d_{a,T_2}(b_{|C_{ij}|})$.
16: { Note that $s_{ij}(b) = |\{w \in C_{ij} : d_{a,T_1}(w) > d_{a,T_1}(b)\}|$.}
17: $s_{ij}(b_1) = 0$.
18: **for** $k = 2, 3, \ldots, |C_{ij}|$ **do**
19: **if** $d_{a,T_1}(b_k) > d_{a,T_1}(b_{k-1})$ **then**
20: $s_{ij}(b_k) = s_{ij}(b_{k-1}) + 1$.
21: **else**
22: $s_{ij}(b_k) = s_{ij}(b_{k-1})$.
23: **end if**
24: **end for**
25: **for** every $b \in C_{ij}$ **do**
26: Let $count_2(a, b)$ equal $s_i(b) - s_{ij}(b)$.
27: **end for**
28: **end for**
29: **end for**

Fig. 5. Algorithm Compute_$count_2$

Next, we prove by contradiction that condition (2) must hold. If $X_1 \cap X_2 \neq \emptyset$, where $X_i = (L \setminus A) \cap \Lambda(T_i[u_i])$, is true then there exists an $x \in X_1 \cap X_2$. We claim that there exist $a, a' \in A$ with $lca^{T_1}(a, a') = u_1$ and $lca^{T_2}(a, a') = u_2$. This yields $x|a|a' \in t(T_1)$ and $x|a|a' \in t(T_2)$ because the subtree attached to u_i for $i = 1, 2$ which contains x cannot contain a or a' by condition (1) above. Thus, $aa'|x \notin \mathcal{R}_{maj}$, contradicting the fact that A is a strong cluster of $\mathcal{R}_{maj}$.

It remains to prove the claim that there exist $a, a' \in A$ such that $lca^{T_1}(a, a') = u_1$ and $lca^{T_2}(a, a') = u_2$. Assume on the contrary that for each pair $a, a' \in A$,

Algorithm `Check_if_strong_cluster`
Input: A cluster A of L.
Output: "Yes", if A is a strong cluster of $\mathcal{R}_{maj}$; "no", otherwise.

```
1: Let u_1 and u_2 be the lowest common ancestor of A in T_1 and T_2, respectively.
2: for i = 1, 2 do
3:    if there exists a subtree in T_i attached to u_i containing leaves from A as well
      as from L \ A then
4:       return "no"
5:    end if
6: end for
7: Let X_i = (L \ A) ∩ Λ(T_i[u_i]) for i = 1, 2.
8: if X_1 ∩ X_2 = ∅ then
9:    return "yes"
10: else
11:   return "no"
12: end if
```

Fig. 6. Algorithm `Check_if_strong_cluster`

there exists an $i \in \{1, 2\}$ such that $lca^{T_i}(a, a')$ is a proper descendant of u_i. Note that A is located in at least two subtrees attached to u_i for $i = 1, 2$. For every pair of subtrees attached to u_1 that contain elements from A, suppose the sets of leaves in the two subtrees are A^1 and A^2. To ensure that $lca^{T_2}(x, y)$ is a proper descendant of u_2 for every $x \in A^1$ and $y \in A^2$, we need $A^1 \cup A^2$ to appear in one subtree attached to u_2. By this argument, all elements of A appear in one subtree attached to u_2. Then, u_2 cannot be the lowest common ancestor of A in T_2, and we arrive at a contradiction.

($\Leftarrow$) Suppose A satisfies the two conditions stated in the lemma. We need to prove that for every $x, y \in A$ and $z \in L \setminus A$, it holds that $xy|z \in \mathcal{R}_{maj}$. Note that z belongs to either X_1, X_2, or $((L \setminus A) \setminus X_1) \setminus X_2$, which gives three cases:

- If $z \in ((L \setminus A) \setminus X_1) \setminus X_2$: Then $lca^{T_i}(x, z)$ is a proper ancestor of $lca^{T_i}(x, y)$ for $i = 1, 2$. Thus, $xy|z \in \mathcal{R}_{maj}$.
- If $z \in X_1$: Then $xy|z \in t(T_2)$ and there are two cases depending on whether or not $lca^{T_1}(x, y)$ is a proper descendant of u_1. If yes, then $xy|z \in t(T_1)$, and thus $xy|z \in \mathcal{R}_{maj}$. If no, then $lca^{T_1}(x, y) = u_1$, and thus $x|y|z \in t(T_1)$, which also yields $xy|z \in \mathcal{R}_{maj}$.
- If $z \in X_2$: Then it follows that $xy|z \in \mathcal{R}_{maj}$ in the same way as for the case $z \in X_1$ above. □

We can use Lemma 10 to check if a given cluster A of L is a strong cluster of $\mathcal{R}_{maj}$ in $O(n)$ time, as shown in Algorithm `Check_if_strong_cluster` in Fig. 6.

6 Concluding Remarks

In this paper, we have described how to compute the R* consensus tree of two input trees in $O(n^2 \log n)$ time. The definition of the set of majority rooted

triplets $\mathcal{R}_{maj}$ as well as the definition of an R* consensus tree can be naturally extended to the case of $k > 2$ input trees; see [2]. The currently fastest algorithm for the case $k > 2$ runs in $O(kn^3)$ time and is outlined in [2]. It can be implemented by explicitly constructing the sets $r(T_i)$ for every input tree T_i using $O(kn^3)$ total time, then obtaining $\mathcal{R}_{maj}$ in $O(kn^3)$ time by finding the most frequently occurring rooted triplet (if it exists) for each $\{x, y, z\}$ in the leaf label set in $O(k)$ time, and finally applying the $O(n^3)$-time strong clustering algorithm from Corollary 2.2 in [3] to $\mathcal{R}_{maj}$. An open problem is to reduce the time complexity to $o(kn^3)$. It seems difficult to extend the approach used in this paper because it would yield an exponential number (in k) of cases in Lemma 5 and thus express $s_{\mathcal{R}_{maj}}(a, b)$ as the sum of an exponential number of auxiliary *count*-functions, causing the total running time to be exponential in k.

Acknowledgments

The authors would like to thank David Bryant for some clarifications and the anonymous referees for their helpful comments.

References

1. Bender, M.A., Farach-Colton, M.: The LCA problem revisited. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 88–94. Springer, Heidelberg (2000)
2. Bryant, D.: A classification of consensus methods for phylogenetics. In: Janowitz, M.F., Lapointe, F.-J., McMorris, F.R., Mirkin, B., Roberts, F.S. (eds.) Bioconsensus. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 61, pp. 163–184. American Mathematical Society, Providence (2003)
3. Bryant, D., Berry, V.: A structured family of clustering and tree construction methods. Advances in Applied Mathematics 27(4), 705–732 (2001)
4. Degnan, J.H., DeGiorgio, M., Bryant, D., Rosenberg, N.A.: Properties of consensus methods for inferring species trees from gene trees. Systematic Biology 58(1), 35–54 (2009)
5. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Inc., Sunderland (2004)
6. Gusfield, D.: Algorithms on Strings, Trees, and Sequences. Cambridge University Press, New York (1997)
7. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM Journal on Computing 13(2), 338–355 (1984)
8. Kannan, S., Warnow, T., Yooseph, S.: Computing the local consensus of trees. SIAM Journal on Computing 27(6), 1695–1724 (1998)
9. Lee, C.-M., Hung, L.-J., Chang, M.-S., Shen, C.-B., Tang, C.-Y.: An improved algorithm for the maximum agreement subtree problem. Information Processing Letters 94(5), 211–216 (2005)
10. Nakhleh, L., Warnow, T., Ringe, D., Evans, S.N.: A comparison of phylogenetic reconstruction methods on an Indo-European dataset. Transactions of the Philological Society 103(2), 171–192 (2005)