Fixed Parameter Polynomial Time Algorithms for Maximum Agreement and Compatible Supertrees

Viet Tung Hoang, Wing-Kin Sung

Symposium on Theoretical Aspects of Computer Science 2008 (Bordeaux), pp. 361-372
www.stacs-conf.org

FIXED PARAMETER POLYNOMIAL TIME ALGORITHMS FOR MAXIMUM AGREEMENT AND COMPATIBLE SUPERTREES

VIET TUNG HOANG [1,2] AND WING-KIN SUNG [1,2]

[1] Department of Computer Science, National University of Singapore
E-mail address: {hoangvi2,ksung}@comp.nus.edu.sg

[2] Genome Institute of Singapore

Abstract. Consider a set of labels L and a set of trees $\mathcal{T} = \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(k)}\}$ where each tree $\mathcal{T}^{(i)}$ is distinctly leaf-labeled by some subset of L. One fundamental problem is to find the biggest tree (denoted as supertree) to represent $\mathcal{T}$ which minimizes the disagreements with the trees in $\mathcal{T}$ under certain criteria. This problem finds applications in phylogenetics, database, and data mining. In this paper, we focus on two particular supertree problems, namely, the maximum agreement supertree problem (MASP) and the maximum compatible supertree problem (MCSP). These two problems are known to be NP-hard for $k \geq 3$. This paper gives the first polynomial time algorithms for both MASP and MCSP when both k and the maximum degree D of the trees are constant.

1. Introduction

Given a set of labels L and a set of unordered trees $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(k)}\}$ where each tree $\mathcal{T}^{(i)}$ is distinctly leaf-labeled by some subset of L. The supertree method tries to find a tree to represent all trees in $\mathcal{T}$ which minimizes the possible conflicts in the input trees. The supertree method finds applications in phylogenetics, database, and data mining. For instance, in the Tree of Life project [10], the supertree method is the basic tool to infer the phylogenetic tree of all species.

Many supertree methods have been proposed in the literature [2, 5, 6, 8]. This paper focuses on two particular supertree methods, namely the Maximum Agreement Supertree (MASP) [8] and the Maximum Compatible Supertree (MCSP) [2]. Both methods try to find a consensus tree with the largest number of leaves which can represent all the trees in $\mathcal{T}$ under certain criteria. (Please read Section 2 for the formal definition.)

MASP and MCSP are known to be NP-hard as they are the generalization of the Maximum Agreement Subtree problem (MAST) [1, 3, 9] and the Maximum Compatible Subtree problem (MCT) [7, 4] respectively. Jansson et al. [8] proved that MASP remains NP-hard even if every tree is a rooted triplet, i.e., a binary tree of 3 leaves. For $k = 2$, Jansson et al. [8] and Berry and Nicolas [2] proposed a linear time algorithm to transform MASP and MCSP for 2 input trees to MAST and MCT respectively. For $k \geq 3$, positive

1998 ACM Subject Classification: Algorithms, Biological computing.
Key words and phrases: maximum agreement supertree, maximum compatible supertree.

	Rooted		**Unrooted**	
MASP for k trees of max degree D	$O((kD)^{kD+3}(2n)^k)$	†	$O((kD)^{kD+3}(4n)^k)$	†
MCSP for k trees of max degree D	$O(2^{2kD}n^k)$	†	$O(2^{2kD}n^k)$	†
MASP/MCSP for k binary trees	$O\big(k(2n^2)^{3k^2}\big)$	[8]		
	$O(8^k n^k)$	[6]		
	$O(6^k n^k)$	†		

Table 1: Summary of previous and new results († stands for new result).

results for computing MASP/MCSP are reported only for rooted binary trees. Jansson et al. [8] gave an $O\big(k(2n)^{3k^2}\big)$ time solution to this problem. Recently, Guillemot and Berry [6] further improve the running time to $O(8^k n^k)$.

In general, the trees in $\mathcal{T}$ may not be binary nor rooted. Hence, Jansson et al. [8] posted an open problem and asked if MASP can be solved in polynomial time when k and the maximum degree of the trees in $\mathcal{T}$ are constant. This paper gives an affirmative answer to this question. We show that both MASP and MCSP can be solved in polynomial time when $\mathcal{T}$ contains constant number of bounded degree trees. For the special case where the trees in $\mathcal{T}$ are rooted binary trees, we show that both MASP and MCSP can be solved in $O(6^k n^k)$ time, which improves the previous best result. Table 1 summarizes the previous and new results.

The rest of the paper is organized as follows. Section 2 gives the formal definition of the problems. Then, Sections 3 and 4 describe the algorithms for solving MCSP for both rooted and unrooted cases. Finally, Sections 5 and 6 detail the algorithms for solving MASP for both rooted and unrooted cases. Proofs omitted due to space limitation will appear in the full version of this paper.

2. Preliminary

A *phylogenetic tree* is defined as an unordered and distinctly leaf-labeled tree. Given a phylogenetic tree T, the notation $L(T)$ denotes the leaf set of T, and the *size* of T refers to $|L(T)|$. For any label set S, the *restriction* of T to S, denoted $T|S$, is a phylogenetic tree obtained from T by removing all leaves in $L(T) - S$ and then suppressing all internal nodes of degree two. (See Figure 1 for an example of *restriction*.) For two phylogenetic trees T and T', we say that T *refines* T', denoted $T \trianglerighteq T'$, if T' can be obtained by contracting some edges of T. (See Figure 1 for an example of *refinement*.)

Maximum Compatible Supertree Problem: Consider a set of k phylogenetic trees $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(k)}\}$. A *compatible supertree* of $\mathcal{T}$ is a tree Y such that $Y|L(\mathcal{T}^{(i)}) \trianglerighteq \mathcal{T}^{(i)}|L(Y)$ for all $i \leq k$. The Maximum Compatible Supertree Problem (MCSP) is to find a compatible supertree with as many leaves as possible. Figure 2 shows an example of a compatible supertree Y of two rooted phylogenetic trees $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$. If all input trees have the same leaf sets, MCSP is referred as Maximum Compatible Subtree Problem (MCT).

Maximum Agreement Supertree Problem: Consider a set of k phylogenetic trees $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(k)}\}$. An *agreement supertree* of $\mathcal{T}$ is a tree X such that $X|L(\mathcal{T}^{(i)}) = \mathcal{T}^{(i)}|L(X)$ for all $i \leq k$. The Maximum Agreement Supertree Problem (MASP) is to find an agreement supertree with as many leaves as possible. Figure 2 shows an example of an

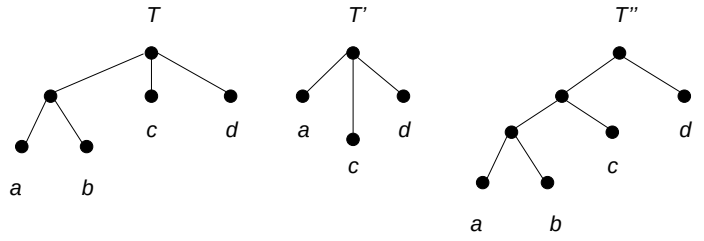

Figure 1: Three rooted trees. A tree T, a tree T' such that $T' = T \mid \{a, c, d\}$, and a tree T'' such that $T'' \trianglerighteq T$.

agreement supertree X of two rooted phylogenetic trees $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$. If all input trees have the same leaf sets, MASP is referred as Maximum Agreement Subtree Problem (MAST).

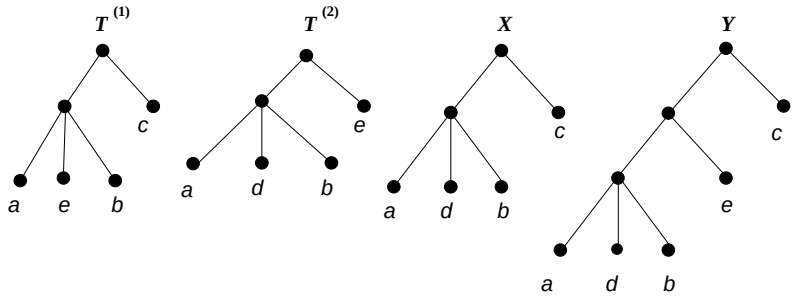

Figure 2: An agreement supertree X and a compatible supertree Y of 2 rooted phylogenetic trees $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$.

In the following discussion, for the set of phylogenetic trees $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(k)}\}$, we denote $n = |\bigcup_{i=1..k} L(\mathcal{T}^{(i)})|$, and D stands for the maximum degree of the trees in $\mathcal{T}$. We assume that none of the trees in $\mathcal{T}$ has an internal node of degree two, so that each tree contains at most $n-1$ internal nodes. (If a tree $\mathcal{T}^{(i)}$ has some internal nodes of degree two, we can replace it by $\mathcal{T}^{(i)} \mid L(\mathcal{T}^{(i)})$ in linear time.)

3. Algorithm for MCSP of rooted trees

Let $\mathcal{T}$ be a set of k rooted phylogenetic trees. This section presents a dynamic programming algorithm to compute the size of a maximum compatible supertree of $\mathcal{T}$ in $O\left(2^{2kD}n^k\right)$ time. The maximum compatible supertree can be obtained in the same asymptotic time bound by backtracking.

For every compatible supertree Y of $\mathcal{T}$, there exists a binary tree that refines Y. This binary tree is also a compatible supertree of $\mathcal{T}$, and is of the same size as Y. Hence in this section, every compatible supertree is implicitly assumed to be binary.

Definition 3.1 (Cut-subtree)**.** A *cut-subtree* of a tree T is either an empty tree or a tree obtained by first selecting some subtrees attached to the same internal node in T and then connecting those subtrees by a common root.

Definition 3.2 (Cut-subforest)**.** Given a set of k rooted (or unrooted) trees $\mathcal{T}$, a *cut-subforest* of $\mathcal{T}$ is a set $\mathcal{A} = \{\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k)}\}$, where $\mathcal{A}^{(i)}$ is a cut-subtree of $\mathcal{T}^{(i)}$ and at least one element of $\mathcal{A}$ is not an empty tree.

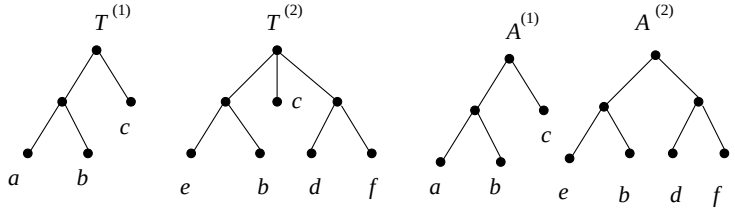

Figure 3: A cut-subforest $\mathcal{A}$ of $\mathcal{T}$.

For example, in Figure 3, $\{\mathcal{A}^{(1)}, \mathcal{A}^{(2)}\}$ is a cut-subforest of $\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}\}$. Let $\mathcal{O}$ denote the set of all possible cut-subforests of $\mathcal{T}$.

Lemma 3.3. *There are* $O\left(2^{kD}n^k\right)$ *different cut-subforests of* $\mathcal{T}$*.*

Proof. We claim that each tree $\mathcal{T}^{(i)}$ contributes $2^D n$ or fewer cut-subtrees; therefore there are $O\left(2^{kD}n^k\right)$ cut-subforests of $\mathcal{T}$. At each internal node v of $\mathcal{T}^{(i)}$, since the degree of v does not exceed D, we have at most 2^D ways of selecting the subtrees attached to v to form a cut-subtree. Including the empty tree, the number of cut-subtrees in $\mathcal{T}^{(i)}$ cannot go beyond $(n-1)2^D + 1 < 2^D n$. ∎

Figure 4 demonstrates that a compatible supertree of some cut-subforest $\mathcal{A}$ of $\mathcal{T}$ may not be a compatible supertree of $\mathcal{T}$. To circumvent this irregularity, we define *embedded supertree* as follows.

Definition 3.4 (Embedded supertree)**.** For any cut-subforest $\mathcal{A}$ of $\mathcal{T}$, a tree Y is called an *embedded supertree* of $\mathcal{A}$ if Y is a compatible supertree of $\mathcal{A}$, and $L(Y) \cap L(\mathcal{T}^{(i)}) \subseteq L(\mathcal{A}^{(i)})$ for all $i \leq k$.

Note that a compatible supertree of $\mathcal{T}$ is also an embedded supertree of $\mathcal{T}$. For each cut-subforest $\mathcal{A}$ of $\mathcal{T}$, let $\texttt{mcsp}(\mathcal{A})$ denote the maximum size of embedded supertrees of $\mathcal{A}$. Our aim is to compute $\texttt{mcsp}(\mathcal{T})$. Below, we first define the recursive equation for computing $\texttt{mcsp}(\mathcal{A})$ for all cut-subforests $\mathcal{A} \in \mathcal{O}$. Then, we describe our dynamic programming algorithm.

We partition the cut-subforests in $\mathcal{O}$ into two classes. A cut-subforest $\mathcal{A}$ of $\mathcal{T}$ is *terminal* if each element $\mathcal{A}^{(i)}$ is either an empty tree or a leaf of $\mathcal{T}^{(i)}$; it is called *non-terminal*, otherwise.

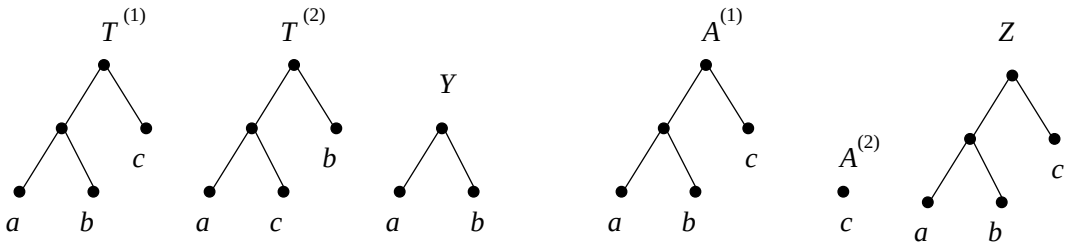

Figure 4: Consider $\mathcal{T} = \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}\}$ and its cut-subforest $\mathcal{A} = \{\mathcal{A}^{(1)}, \mathcal{A}^{(2)}\}$. Although Z is a compatible supertree of $\mathcal{A}$, it is not a compatible supertree of $\mathcal{T}$. The maximum compatible supertree of $\mathcal{T}$ is Y that contains only 2 leaves.

For each terminal cut-subforest $\mathcal{A}$, let

$$\Lambda(\mathcal{A}) = \Big\{ l \in \bigcup_{j=1..k} L(\mathcal{A}^{(j)}) \mid l \notin L(\mathcal{T}^{(i)}) - L(\mathcal{A}^{(i)}) \text{ for } i = 1, 2, \ldots, k \Big\} \ . \tag{3.1}$$

For example, with $\mathcal{T}$ in Figure 2, if $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ are leaves labeled by a and d respectively then $\Lambda(\mathcal{A}) = \{d\}$. In Lemma 3.5, we show that $\texttt{mcsp}(\mathcal{A}) = |\Lambda(\mathcal{A})|$.

Lemma 3.5. *If $\mathcal{A}$ is a terminal cut-subforest then* $\texttt{mcsp}(\mathcal{A}) = |\Lambda(\mathcal{A})|$.

Proof. Consider any embedded supertree Y of $\mathcal{A}$. By Definition 3.4, every leaf of Y belongs to $\Lambda(\mathcal{A})$. Hence the value $\texttt{mcsp}(\mathcal{A})$ does not exceed $|\Lambda(\mathcal{A})|$.

It remains to give an example of some embedded supertree of $\mathcal{A}$ whose leaf set is $\Lambda(\mathcal{A})$. Let C be a rooted caterpillar [1] whose leaf set is $\Lambda(\mathcal{A})$. The definition of $\Lambda(\mathcal{A})$ implies that $L(C) \cap L(\mathcal{T}^{(i)}) \subseteq L\big(\mathcal{A}^{(i)}\big)$ for every $i \leq k$. Since each $\mathcal{A}^{(i)}$ has at most one leaf, it is straightforward that C is a compatible supertree of $\mathcal{A}$. Hence C is the desired example. ∎

Definition 3.6 (Bipartite)**.** Let $\mathcal{A}$ be a cut-subforest of $\mathcal{T}$. We say that the cut-subforests $\mathcal{A}_L$ and $\mathcal{A}_R$ *bipartition* $\mathcal{A}$ if for every $i \leq k$, the trees $\mathcal{A}_L^{(i)}$ and $\mathcal{A}_R^{(i)}$ can be obtained by (1) partitioning the subtrees attached to the root of $\mathcal{A}^{(i)}$ into two sets $S_L^{(i)}$ and $S_R^{(i)}$; and (2) connecting the subtrees in $S_L^{(i)}$ (resp. $S_R^{(i)}$) by a common root to form $\mathcal{A}_L^{(i)}$ (resp. $\mathcal{A}_R^{(i)}$).

Figure 5 shows an example of the preceding definition. For each non-terminal cut-subforest $\mathcal{A}$, we compute $\texttt{mcsp}(\mathcal{A})$ based on the mcsp values of $\mathcal{A}_L$ and $\mathcal{A}_R$ for each bipartite $(\mathcal{A}_L, \mathcal{A}_R)$ of $\mathcal{A}$. More precisely, we prove that

$$\texttt{mcsp}(\mathcal{A}) = \max\{\texttt{mcsp}(\mathcal{A}_L) + \texttt{mcsp}(\mathcal{A}_R) \mid \mathcal{A}_L \text{ and } \mathcal{A}_R \text{ bipartition } \mathcal{A}\} \ . \tag{3.2}$$

The identity (3.2) is then established by Lemmas 3.8 and 3.10.

Lemma 3.7. *Consider a bipartite $(\mathcal{A}_L, \mathcal{A}_R)$ of some cut-subforest $\mathcal{A}$ of $\mathcal{T}$. If Y_L and Y_R are embedded supertrees of $\mathcal{A}_L$ and $\mathcal{A}_R$ respectively then Y is an embedded supertree of $\mathcal{A}$, where Y is formed by connecting Y_L and Y_R to a common root.*

[1] A rooted caterpillar is a rooted, unordered, and distinctly leaf-labeled binary tree where every internal node has at least one child that is a leaf.

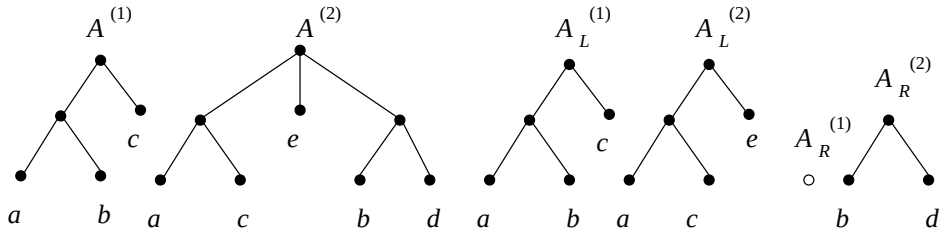

Figure 5: A bipartite $(\mathcal{A}_L, \mathcal{A}_R)$ of a cut-subforest $\mathcal{A}$. The empty tree is represented by a white circle.

Lemma 3.8. *Let $\mathcal{A}$ be a cut-subforest of $\mathcal{T}$. If $(\mathcal{A}_L, \mathcal{A}_R)$ is a bipartite of $\mathcal{A}$ then* $\texttt{mcsp}(\mathcal{A}) \geq \texttt{mcsp}(\mathcal{A}_L) + \texttt{mcsp}(\mathcal{A}_R)$.

Proof. Consider an embedded supertree Y_L of $\mathcal{A}_L$ such that $|L(Y_L)| = \texttt{mcsp}(\mathcal{A}_L)$. Define Y_R for $\mathcal{A}_R$ similarly. Let Y be a tree formed by connecting Y_L and Y_R with a common root. Note that Y is of size $\texttt{mcsp}(\mathcal{A}_L) + \texttt{mcsp}(\mathcal{A}_R)$. By Lemma 3.7, Y is an embedded supertree of $\mathcal{A}$ and hence the lemma follows. ∎

Lemma 3.9. *Given a cut-subforest $\mathcal{A}$ of $\mathcal{T}$, let Y be a binary embedded supertree of $\mathcal{A}$ with left subtree Y_L and right subtree Y_R. There exists a bipartite $(\mathcal{A}_L, \mathcal{A}_R)$ of $\mathcal{A}$ such that either (i) Y is an embedded supertree of $\mathcal{A}_L$; or (ii) Y_L and Y_R are embedded supertrees of $\mathcal{A}_L$ and $\mathcal{A}_R$ respectively.*

Lemma 3.10. *For each non-terminal cut-subforest $\mathcal{A}$ of $\mathcal{T}$, there exists a bipartite $(\mathcal{A}_L, \mathcal{A}_R)$ of $\mathcal{A}$ such that* $\texttt{mcsp}(\mathcal{A}) \leq \texttt{mcsp}(\mathcal{A}_L) + \texttt{mcsp}(\mathcal{A}_R)$.

Proof. Let Y be a binary embedded supertree of $\mathcal{A}$ such that $|L(Y)| = \texttt{mcsp}(\mathcal{A})$. By Lemma 3.9, there exists a bipartite $(\mathcal{A}_L, \mathcal{A}_R)$ of $\mathcal{A}$ such that either (1) Y is an embedded supertree of $\mathcal{A}_L$; or (2) Y_L and Y_R are embedded supertrees of $\mathcal{A}_L$ and $\mathcal{A}_R$ respectively, where Y_L is the left subtree and Y_R is the right subtree of Y. In both cases, $|L(Y)| \leq \texttt{mcsp}(\mathcal{A}_L) + \texttt{mcsp}(\mathcal{A}_R)$. Then the lemma follows. ∎

The above discussion then leads to Theorem 3.11.

Theorem 3.11. *For every cut-subforest $\mathcal{A}$ of $\mathcal{T}$, the value* $\texttt{mcsp}(\mathcal{A})$ *equals to*

$$\begin{cases} |\Lambda(\mathcal{A})|\,, & \text{if } \mathcal{A} \text{ is terminal,} \\ \max\{\texttt{mcsp}(\mathcal{A}_L) + \texttt{mcsp}(\mathcal{A}_R) \mid \mathcal{A}_L \text{ and } \mathcal{A}_R \text{ bipartition } \mathcal{A}\}, & \text{otherwise}\ . \end{cases}$$

We define an ordering of the cut-subforests in $\mathcal{O}$ as follows. For any cut-subforests $\mathcal{A}_1, \mathcal{A}_2$ in $\mathcal{O}$, we say that $\mathcal{A}_1$ is smaller than $\mathcal{A}_2$ if $\mathcal{A}_1^{(i)}$ is a cut-subtree of $\mathcal{A}_2^{(i)}$ for $i = 1, 2, \ldots, k$. Our algorithm enumerates $\mathcal{A} \in \mathcal{O}$ in topologically increasing order and computes $\texttt{mcsp}(\mathcal{A})$ based on Theorem 3.11. Theorem 3.12 states the complexity of our algorithm.

Theorem 3.12. *A maximum compatible supertree of k rooted phylogenetic trees can be obtained in $O\left(2^{2kD} n^k\right)$ time .*

Proof. Testing if a cut-subforest is terminal takes $O(k)$ times, and each terminal cut-subforest $\mathcal{A}$ then requires $O(k^2)$ time for the computation of $\Lambda(\mathcal{A})$. In view of Lemma 3.3, it suffices to show that each non-terminal cut-subforest $\mathcal{A}$ has $O(2^{kD})$ bipartites. This result follows from the fact that for each $i \leq k$, there are at most 2^D ways to partition the set of the subtrees attached to the root of $\mathcal{A}^{(i)}$. ∎

In the special case where every tree $\mathcal{T}^{(i)}$ is binary, Theorem 3.13 shows that our algorithm actually has a better time complexity. Note that the concepts of agreement supertree and compatible supertree will coincide for binary trees. Hence, our algorithm improves the $O\left(8^k n^k\right)$-time algorithm in [6] for computing maximum agreement supertree of k rooted binary trees.

Theorem 3.13. *If every tree in $\mathcal{T}$ is binary, a maximum compatible supertree (or a maximum agreement supertree) can be computed in $O\left(6^k n^k\right)$ time.*

Proof. We claim that the processing of non-terminal cut-subforests of $\mathcal{T}$ requires $O\left(6^k n^k\right)$ time. The argument in the proof of Theorem 3.12 tells that the remaining computation runs within the same asymptotic time bound. Consider an integer $r \in \{0, 1, \ldots, k\}$. We shall be dealing with a cut-subforest $\mathcal{A}$ such that there are exactly r cut-subtrees $\mathcal{A}^{(i)}$ whose roots are internal nodes of $\mathcal{T}^{(i)}$. The key of this proof is to show that the number of those cut-subforests does not exceed $\left(\begin{array}{c} k \\ r \end{array} \right) (n-1)^r (n+1)^{k-r}$, and the running time for each cut-subforest is $O\left(4^r 2^{k-r}\right)$. Hence, the total running time for all non-terminal cut-subforests is

$$\sum_{r=0}^{k} \left(\begin{array}{c} k \\ r \end{array} \right) (n-1)^r (n+1)^{k-r} O\left(4^r 2^{k-r}\right) = O\left(6^k n^k\right).$$

We can count the number of the specified cut-subforests $\mathcal{A}$ as follows. First there are $\left(\begin{array}{c} k \\ r \end{array} \right)$ options for r indices i such that the roots of cut-subtrees $\mathcal{A}^{(i)}$ are internal nodes of $\mathcal{T}^{(i)}$. For those cut-subtrees, we then appoint one of the $(n-1)$ or fewer internal nodes of $\mathcal{T}^{(i)}$ to be the root node of $\mathcal{A}^{(i)}$. Every other cut-subtree of $\mathcal{A}$ is a leaf or the empty tree, and then can be determined from at most $n+1$ alternatives. Multiplying those possibilities gives us the bound stipulated in the preceding paragraph.

It remains to estimate the running time for each specified cut-subforest $\mathcal{A}$. This task requires us to bound the number of bipartites of each cut-subforest. If the root v of $\mathcal{A}^{(i)}$ is an internal node of $\mathcal{T}^{(i)}$ then $\mathcal{A}^{(i)}$ contributes 4 or fewer ways of partitioning the set of the subtrees attached to v. Otherwise, we have at most 2 ways of partitioning this set. Hence $\mathcal{A}$ owns at most $4^r 2^{k-r}$ bipartites, and this completes the proof. ∎

4. Algorithm for MCSP of unrooted trees

Let $\mathcal{T}$ be a set of k unrooted phylogenetic trees. This section extends the algorithm in Section 3 to find the size of a maximum compatible supertree of $\mathcal{T}$. The maximum compatible supertree can be obtained by backtracking. Surprisingly, the extended algorithm for unrooted trees runs within the same asymptotic time bound as the original algorithm for rooted trees.

We will follow the same approach as Section 3, i.e., for each cut-subforest $\mathcal{A}$ of $\mathcal{T}$, we find an embedded supertree of $\mathcal{A}$ of maximum size. Definitions 3.1, 3.2, and 3.4 for cut-subforest and embedded supertree in the previous section are still valid for unrooted trees. Notice that although $\mathcal{T}$ is the set of unrooted trees, each cut-subforest $\mathcal{A}$ of $\mathcal{T}$ consists of rooted trees. (See Figure 6 for an example of cut-subforest for unrooted trees.) Hence we can use the algorithm in Section 3 to find the maximum embedded supertree of $\mathcal{A}$. We then select the biggest tree T among those maximum embedded supertrees for all cut-subforests of $\mathcal{T}$, and unroot T to obtain the maximum compatible supertree of $\mathcal{T}$.

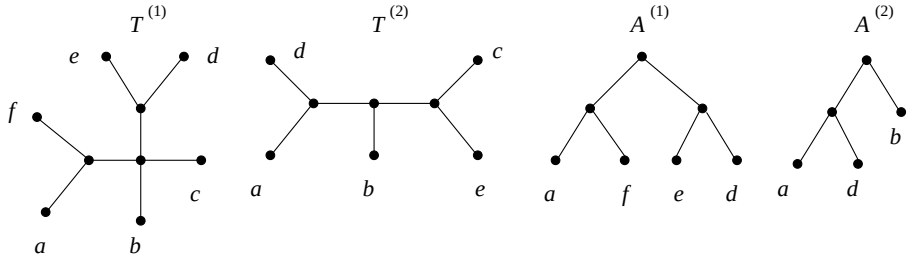

Figure 6: The set of rooted trees $\mathcal{A} = \{\mathcal{A}^{(1)}, \mathcal{A}^{(2)}\}$ is a cut-subforest of $\mathcal{T} = \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}\}$.

Theorem 4.1 shows that the extended algorithm has the same asymptotic time bound as the algorithm in Section 3.

Theorem 4.1. *We can find a maximum compatible supertree of k unrooted phylogenetic trees in $O\left(2^{2kD}n^k\right)$ time.*

Proof. Using a similar proof as Lemma 3.3, we can prove that there are $O\left(2^{kD}n^k\right)$ cut-subforests of $\mathcal{T}$. As given in the proof of Theorem 3.12, finding the maximum embedded supertrees of each cut-subforest takes $O(2^{kD})$ time. Hence the extended algorithm runs within the specified time bound. ∎

5. Algorithm for MASP of rooted trees

Let $\mathcal{T}$ be a set of k rooted phylogenetic trees. This section presents a dynamic programming algorithm to compute the size of a maximum agreement supertree of $\mathcal{T}$ in $O\left((kD)^{kD+3}(2n)^k\right)$ time. The maximum agreement supertree can be obtained in the same asymptotic time bound by backtracking.

The idea here is similar to that of Section 3. However, while we can assume that compatible supertrees are binary, the maximum degree of agreement supertrees can grow up to kD. It is the reason why we have the factor $O((kD)^{kD+3})$ in the complexity.

Definition 5.1 (Sub-forest)**.** Given a set of k rooted trees $\mathcal{T}$, a *sub-forest* of $\mathcal{T}$ is a set $\mathcal{A} = \{\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k)}\}$, where each $\mathcal{A}^{(i)}$ is either an empty tree or a complete subtree rooted at some node of $\mathcal{T}^{(i)}$, and at least one element of $\mathcal{A}$ is not an empty tree.

Notice that the definition of sub-forest does not coincide with the concept of cut-subforest in Definition 3.2 of Section 3. For example, the cut-subforest $\mathcal{A}$ in Figure 3 is not a sub-forest of $\mathcal{T}$, because $\mathcal{A}^{(2)}$ is not a complete subtree rooted at some node of $\mathcal{T}^{(2)}$. Let $\mathcal{O}$ denote the set of all possible sub-forests of $\mathcal{T}$. Then $|\mathcal{O}| = O\left((2n)^k\right)$.

Definition 5.2 (Enclosed supertree)**.** For any sub-forest $\mathcal{A}$ of $\mathcal{T}$, a tree X is called an *enclosed supertree* of $\mathcal{A}$ if X is an agreement supertree of $\mathcal{A}$, and $L(X) \cap L(\mathcal{T}^{(i)}) \subseteq L(\mathcal{A}^{(i)})$ for all $i \leq k$.

For each sub-forest $\mathcal{A}$ of $\mathcal{T}$, let $\mathtt{masp}(\mathcal{A})$ denote the maximum size of enclosed supertrees of $\mathcal{A}$. We use a similar approach as Section 3, i.e., we compute $\mathtt{masp}(\mathcal{A})$ for all $\mathcal{A} \in \mathcal{O}$, and $\mathtt{masp}(\mathcal{T})$ is the size of a maximum agreement supertree of $\mathcal{T}$. We partition the sub-forests in $\mathcal{O}$ to two classes. A sub-forest $\mathcal{A}$ is *terminal* if each $\mathcal{A}^{(i)}$ is either an empty tree or a leaf. Otherwise, $\mathcal{A}$ is called *non-terminal*.

Notice that for terminal sub-forest, the definition of enclosed supertree coincides with the concept of embedded supertree in Definition 3.4 of Section 3. Then by Lemma 3.5, we have $\mathtt{masp}(\mathcal{A}) = |\Lambda(\mathcal{A})|$. (Please refer to the formula (3.1) in the paragraph preceding Lemma 3.5 for the definition of function Λ.)

Definition 5.3 (Decomposition)**.** Let $\mathcal{A}$ be a sub-forest of $\mathcal{T}$. We say that sub-forests $\mathcal{B}_1, \ldots, \mathcal{B}_d$ (with $d \geq 2$) *decompose* $\mathcal{A}$ if for all $i \leq k$, either (i) Exactly one of $\mathcal{B}_1^{(i)}, \ldots, \mathcal{B}_d^{(i)}$ is isomorphic to $\mathcal{A}^{(i)}$ while the others are empty trees; or (ii) There are at least 2 nonempty trees in $\mathcal{B}_1^{(i)}, \ldots, \mathcal{B}_d^{(i)}$, and all those nonempty trees are isomorphic to pairwise distinct subtrees attached to the root of $\mathcal{A}^{(i)}$.

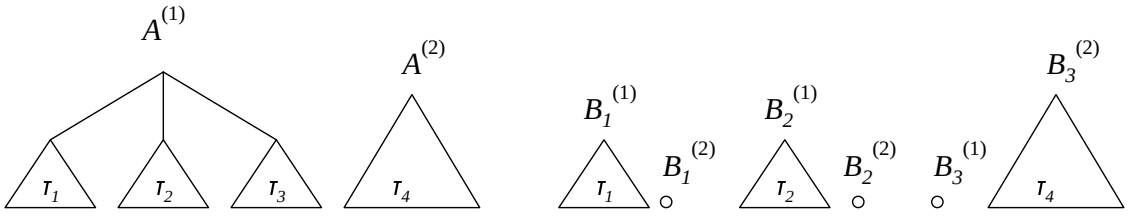

Figure 7: A decomposition $(\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$ of a sub-forest $\mathcal{A}$. The empty trees are represented by white circles.

Figure 7 illustrates the concept of decomposition. For each sub-forest $\mathcal{A}$ of $\mathcal{T}$, we will prove that

$$\mathtt{masp}(\mathcal{A}) = \max\{\mathtt{masp}(\mathcal{B}_1) + \ldots + \mathtt{masp}(\mathcal{B}_d) \mid \mathcal{B}_1, \ldots, \mathcal{B}_d \text{ decompose } \mathcal{A}\} \ . \tag{5.1}$$

The identity (5.1) is then established by Lemmas 5.5 and 5.7.

Lemma 5.4. *Suppose $(\mathcal{B}_1, \ldots, \mathcal{B}_d)$ is a decomposition of some sub-forest $\mathcal{A}$ of $\mathcal{T}$. Let $\tau_1, \ldots, \tau_d$ be some enclosed supertrees of $\mathcal{B}_1, \ldots, \mathcal{B}_d$ respectively, and let X be the tree obtained by connecting $\tau_1, \ldots, \tau_d$ to a common root. Then, X is an enclosed supertree of $\mathcal{A}$.*

Lemma 5.5. *If $(\mathcal{B}_1, \ldots, \mathcal{B}_d)$ is a decomposition of a sub-forest $\mathcal{A}$ of $\mathcal{T}$ then* $\mathtt{masp}(\mathcal{A}) \geq \mathtt{masp}(\mathcal{B}_1) + \ldots + \mathtt{masp}(\mathcal{B}_d)$.

Proof. For each $\mathcal{B}_j$, let τ_j be an enclosed supertree of $\mathcal{B}_j$ such that $|L(\tau_j)| = \mathtt{masp}(\mathcal{B}_j)$. Let X be the tree obtained by connecting $\tau_1, \ldots, \tau_d$ to a common root. By Lemma 5.4, X is an enclosed supertree of $\mathcal{A}$. Hence $|L(\tau_1)| + \ldots + |L(\tau_d)| = |L(X)| \leq \mathtt{masp}(\mathcal{A})$. ∎

Lemma 5.6. *Let X be an enclosed supertree of some sub-forest $\mathcal{A}$ of $\mathcal{T}$, and let $\tau_1, \ldots, \tau_d$ be all subtrees attached to the root of X. Then either (i) There is a decomposition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{A}$ such that X is an enclosed supertree of $\mathcal{B}_1$; or (ii) There is a decomposition $(\mathcal{B}_1, \ldots, \mathcal{B}_d)$ of $\mathcal{A}$ such that each τ_j is an enclosed supertree of $\mathcal{B}_j$.*

Lemma 5.7. *For each non-terminal sub-forest $\mathcal{A}$ of $\mathcal{T}$, there is a decomposition $(\mathcal{B}_1, \ldots, \mathcal{B}_d)$ of $\mathcal{A}$ such that $\texttt{masp}(\mathcal{A}) \leq \texttt{masp}(\mathcal{B}_1) + \ldots + \texttt{masp}(\mathcal{B}_d)$*

Proof. Let X be an enclosed supertree of $\mathcal{A}$ such that $|L(X)| = \texttt{masp}(\mathcal{A})$ and let $\tau_1, \ldots, \tau_d$ be all subtrees attached to the root of X. By Lemma 5.6, either (i) There exists a decomposition $(\mathcal{B}_1, \mathcal{B}_2)$ of $\mathcal{A}$ such that X is an enclosed supertree of $\mathcal{B}_1$; or (ii) There is a decomposition $(\mathcal{B}_1, \ldots, \mathcal{B}_d)$ of $\mathcal{A}$ such that each τ_j is an enclosed supertree of $\mathcal{B}_j$. In case (i), we have $|L(X)| \leq \texttt{masp}(\mathcal{B}_1) \leq \texttt{masp}(\mathcal{B}_1) + \texttt{masp}(\mathcal{B}_2)$. On the other hand, in case (ii), we have $|L(X)| = |L(\tau_1)| + \ldots + |L(\tau_d)| \leq \texttt{masp}(\mathcal{B}_1) + \ldots + \texttt{masp}(\mathcal{B}_d)$. ∎

The above discussion then leads to Theorem 5.8.

Theorem 5.8. *For every sub-forest $\mathcal{A}$ of $\mathcal{T}$, the value $\texttt{masp}(\mathcal{A})$ equals to*

$$\begin{cases} |\Lambda(\mathcal{A})|\,, & \text{if } \mathcal{A} \text{ is terminal,} \\ \max\{\texttt{masp}(\mathcal{B}_1) + \ldots + \texttt{masp}(\mathcal{B}_d) \mid \mathcal{B}_1, \ldots, \mathcal{B}_d \text{ decompose } \mathcal{A}\}, & \text{otherwise}\ . \end{cases}$$

We define an ordering of the sub-forests in $\mathcal{O}$ as follows. For any sub-forests $\mathcal{A}_1, \mathcal{A}_2$ in $\mathcal{O}$, we say $\mathcal{A}_1$ is smaller than $\mathcal{A}_2$ if $\mathcal{A}_1^{(i)}$ is either an empty tree or a subtree of $\mathcal{A}_2^{(i)}$ for $i = 1, 2, \ldots, k$. Our algorithm enumerates $\mathcal{A} \in \mathcal{O}$ in topologically increasing order and computes $\texttt{masp}(\mathcal{A})$ based on Theorem 5.8.

In Lemma 5.9, we bound the number of decompositions of each sub-forest of $\mathcal{T}$. Theorem 5.10 states the complexity of the algorithm.

Lemma 5.9. *Each sub-forest of $\mathcal{T}$ has $O\left((kD)^{kD+1}\right)$ decompositions, and generating those decompositions takes $O\left(k^2D^2\right)$ time per decomposition.*

Proof. Let $\mathcal{A}$ be a sub-forest of $\mathcal{T}$. Since the maximum degree of any agreement supertree of $\mathcal{A}$ is bounded by kD, we consider only decompositions that consist of at most kD elements. We claim that for each $d \in \{2, \ldots, kD\}$, the sub-forest $\mathcal{A}$ owns $O\left((d+2)^{kD}\right)$ decompositions $(\mathcal{B}_1, \ldots, \mathcal{B}_d)$. Summing up those asymptotic terms gives us the specified bound.

The key of this proof is to prove that for each $s \in \{1, \ldots, k\}$, the tree $\mathcal{A}^{(s)}$ contributes at most $(d+1)^D + d < (d+2)^D$ sequences $\mathcal{B}_1^{(s)}, \ldots, \mathcal{B}_d^{(s)}$, and generating those sequences requires $O(d)$ time per sequence. We have two cases, each corresponds to a type of the above sequence.

Case 1: One term in the sequence is $\mathcal{A}^{(s)}$; therefore the other terms are empty trees. Then, we can generate this sequence by assigning $\mathcal{A}^{(s)}$ to exactly one term and setting the rest to be empty trees. This case provides exactly d sequences and enumerates them in $O(d)$ time per sequence.

Case 2: No term in the above sequence is $\mathcal{A}^{(s)}$. Consider an integer $r \in \{0, 1, \ldots, d\}$ and assume that the sequence consists of exactly r terms that are nonempty nodes. Then those r nonempty trees are isomorphic to pairwise distinct subtrees attached to the root of $\mathcal{A}^{(s)}$. Let δ be the degree of the root of $\mathcal{A}^{(s)}$. We generate the sequence as follows. First we draw r pairwise distinct subtrees attached to the root of $\mathcal{A}^{(s)}$. Next, we select r terms

in the sequence and distribute the above subtrees to them. Finally we set the remaining terms to be empty trees. Hence this case gives at most

$$\sum_{r\leq\min\{\delta,d\}} \left(\begin{array}{c} \delta \\ r \end{array} \right) \frac{d!}{(d-r)!} < \sum_{r=0}^{D} \left(\begin{array}{c} D \\ r \end{array} \right) d^r = (d+1)^D$$

sequences, and generates them in $O(d)$ time per sequence. ∎

Theorem 5.10. *A maximum agreement supertree of k rooted phylogenetic trees can be obtained in $O\left((kD)^{kD+3}(2n)^k\right)$ time.*

Proof. Testing if a sub-forest is terminal takes $O(k)$ times, and each terminal sub-forest $\mathcal{A}$ then requires $O(k^2)$ time for computing $\Lambda(\mathcal{A})$. By Lemma 5.9, each non-terminal sub-forest requires $O\left((kD)^{kD+3}\right)$ running time. Summing up those asymptotic terms for $O\left((2n)^k\right)$ sub-forests of $\mathcal{T}$ gives us the specified time bound. ∎

6. Algorithm for MASP of unrooted trees

Let $\mathcal{T}$ be a set of k unrooted phylogenetic trees. This section extends the algorithm in Section 5 to find the size of a maximum agreement supertree of $\mathcal{T}$ in $O\left((kD)^{kD+3}(4n)^k\right)$ time. The maximum agreement supertree can be obtained by backtracking.

We say that a set of k rooted trees $\mathcal{F} = \{\mathcal{F}^{(1)}, \ldots, \mathcal{F}^{(k)}\}$ is a *rooted variant* of $\mathcal{T}$ if we can obtain each $\mathcal{F}^{(i)}$ by rooting $\mathcal{T}^{(i)}$ at some internal node. One naive approach is to use the algorithm in the previous section to solve MASP for each rooted variant of $\mathcal{T}$. Each rooted variant then gives us a solution, and the maximum of those solutions is the size of a maximum agreement supertree of $\mathcal{T}$. Because there are $O\left(n^k\right)$ rooted variants of $\mathcal{T}$, this approach adds an $O\left(n^k\right)$ factor to the complexity of the algorithm for rooted trees.

We now show how to improve the above naive algorithm. As mentioned in the previous section, the computation of each rooted variant of $\mathcal{T}$ consists of $O\left((2n)^k\right)$ sub-problems which correspond to its sub-forests. (Please refer to Definition 5.1 for the concept of sub-forest.) Since different rooted variants may have some common sub-forests, the total number of sub-problems we have to run is much smaller than $O(2^k n^{2k})$. More precisely, we will show that the total number of sub-problems is only $O\left((4n)^k\right)$.

A (rooted or unrooted) tree is *trivial* if it is a leaf or an empty tree. A *maximal subtree* of an unrooted tree T is a rooted tree obtained by first rooting T at some internal node v and then removing at most one nontrivial subtree attached to v. Let $\mathcal{O}$ denote the set of sub-forests of all rooted variants of $\mathcal{T}$.

Lemma 6.1. *Let $\mathcal{A} = \{\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k)}\}$ be a set of rooted trees. Then $\mathcal{A} \in \mathcal{O}$ if and only if each $\mathcal{A}^{(i)}$ is either a trivial subtree or a maximal subtree of $\mathcal{T}^{(i)}$.*

Proof. Let $\mathcal{F}$ be a rooted variant of $\mathcal{T}$ such that $\mathcal{A}$ is a sub-forest of $\mathcal{F}$. Fix an index $s \in \{1, \ldots, k\}$ and let v be the root node of $\mathcal{A}^{(s)}$. Our claim is straightforward if either $\mathcal{A}^{(s)}$ is trivial or v is the root node of $\mathcal{F}^{(s)}$. Otherwise, let u be the parent of v in $\mathcal{F}^{(s)}$. Hence $\mathcal{A}^{(s)}$ is the maximal subtree of $\mathcal{T}^{(s)}$ obtained by first rooting $\mathcal{T}^{(s)}$ at v and then removing the complete subtree rooted at u.

Conversely, we construct a rooted variant $\mathcal{F}$ of $\mathcal{T}$ such that $\mathcal{A}$ is a sub-forest of $\mathcal{F}$ as follows. For each $i \leq k$, if $\mathcal{A}^{(i)}$ is trivial or $\mathcal{A}^{(i)}$ is a tree obtained by rooting $\mathcal{T}^{(i)}$ at some internal node then constructing $\mathcal{F}^{(i)}$ is straightforward. Otherwise $\mathcal{A}^{(i)}$ is a maximal subtree

of $\mathcal{T}^{(i)}$ obtained by first rooting $\mathcal{T}^{(i)}$ at some internal node v and then removing exactly one nontrivial subtree τ attached to v. Hence $\mathcal{F}^{(i)}$ is the tree obtained by rooting $\mathcal{T}^{(i)}$ at u, where u is the root of τ. ∎

Theorem 6.2. *We can find a maximum agreement supertree of k unrooted phylogenetic trees in $O\left((kD)^{kD+3}(4n)^k\right)$ time.*

Proof. The key of this proof is to show that each tree $\mathcal{T}^{(i)}$ contributes at most $(3n-1)$ maximal subtrees. It follows that $|\mathcal{O}| \leq (4n)^k$. The specified running time of our algorithm is then straightforward because each subproblem requires $O\left((kD)^{kD+3}\right)$ time as given in the proof of Theorem 5.10. Assume that the tree $\mathcal{T}^{(i)}$ has exactly L leaves, with $L \leq n$. We now count the number of maximal subtrees T of $\mathcal{T}^{(i)}$ in two cases.

Case 1: T is obtained by rooting $\mathcal{T}^{(i)}$ at some internal node. Hence this case provides at most $L - 1 < n$ maximal subtrees.

Case 2: T is obtained by first rooting $\mathcal{T}^{(i)}$ at some internal node v and then removing a nontrivial subtree τ attached to v. Notice that there is a one-to-one correspondence between the tree T and the directed edge (v, u) of $\mathcal{T}^{(i)}$, where u is the root node of τ. There are $2L - 2$ or fewer undirected edges in $\mathcal{T}^{(i)}$ but exactly L of them are adjacent to the leaves. Hence this case gives us at most $2(2L - 2 - L) < 2n - 1$ maximal subtrees. ∎

References

[1] A. Amir and D. Keselman. Maximum Agreement Subtree in a set of Evolutionary Trees: Metrics and Efficient Algorithms. *SIAM Journal on Computing*, 26(6):1656–1669, 1997.

[2] V. Berry and F. Nicolas. Maximum Agreement and Compatible Supertrees. In *Proc. 15th Symposium on Combinatorial Pattern Matching* (CPM 2004), *Lect. Notes in Comp. Science* **3109**, pp. 205–219. Springer, 2004.

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

[4] G. Ganapathysaravanabavan and T. Warnow. Finding a maximum compatible tree for a bounded number of trees with bounded degree is solvable in polynomial time. In *Proc. 1st Workshop on Algorithms in Bioinformatics* (WABI 2001), *Lect. Notes in Comp. Science* **2149**, pp. 156–163. Springer, 2001.

[5] A. G. Gordon. Consensus supertrees: the synthesis of rooted trees containing overlapping sets of labelled leaves. *Journal of Classification*, 3:335–348, 1986.

[6] Sylvain Guillemot and Vincent Berry. Fixed-Parameter Tractability of the Maximum Agreement Supertree Problem. In *Proc. 18th Symposium on Combinatorial Pattern Matching* (CPM 2007), *Lect. Notes in Comp. Science* **4580**, pp. 274–285. Springer, 2007.

[7] J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71:153–169, 1996.

[8] Jesper Jansson, Joseph H.-K. Ng, Kunihiko Sadakane, and Wing-King Sung. Rooted Maximum Agreement Supertrees. *Algorithmica*, 43:293–307, 2005.

[9] M.-Y. Kao, T.-W. Lam, W.-K. Sung, and H.-F. Ting. An Even Faster and More Unifying Algorithm for Comparing Trees via Unbalanced Bipartite Matchings. *Journal of Algorithms*, 40(2):212–233, 2001.

[10] Maddison, D.R., and K.-S. Schulz (eds.). The Tree of Life Web Project. `http://tolweb.org`, 1996-2006.