

Approximating the Complement of the Maximum Compatible Subset of Leaves of k Trees

Ganeshkumar Ganapathy¹ and Tandy Warnow¹

Department of Computer Sciences, University of Texas, Austin, TX 78712;
{gsgk, tandy}@cs.utexas.edu

Abstract. We address a combinatorial problem which arises in computational phylogenetics. In this problem we are given a set of unrooted (not necessarily binary) trees each leaf-labelled by the same set S , and we wish to remove a minimum number of leaves so that the resultant trees share a common refinement (i.e. they are “compatible”). If we assume the input trees are all binary, then this is simply the Maximum Agreement Subtree problem (MAST), for which much is already known. However, if the input trees need not be binary, then the problem is much more computationally intensive: it is NP-hard for just two trees, and solvable in polynomial time for any number k of trees when all trees have bounded degree. In this paper we present an $O(k^2 n^2)$ 4-approximation algorithm and an $O(k^2 n^3)$ 3-approximation algorithm for the general case of this problem.

1 Introduction

Let \mathcal{T} be a set of unrooted, and not necessarily binary trees, each bijectively leaf-labelled by the same set S of objects. A tree T on the set S of leaves is said to *refine* another tree T' on the same set if T' can be obtained by contracting some selection of edges of T . If the trees share a common refinement, then the set is said to be “compatible”. In this case, the minimal common refinement is unique, and can be computed in $O(nk)$ time, where $k = |\mathcal{T}|$ and $|S| = n$ [6, 12]. Consider the following optimization problem:

Maximum Compatible Subset Problem:

Input: Given \mathcal{T} , a collection of trees, each leaf-labelled by the same set S ,

Output: Find a maximum cardinality $A \subseteq S$ so that the set $\{T|A : T \in \mathcal{T}\}$ (where $T|A$ denotes the tree obtained by restricting T to the leaves labelled by elements of A , and suppressing degree two nodes) has a common refinement.

This problem is motivated by applications to computational phylogenetics (and we describe the application in Section 5), and was initially posed by Hamel and Steel in [7]. It was proven NP-hard for six or more trees in [7], and later shown also NP-hard for two trees when at least one tree is of unbounded degree in [8]. It is also known to be solvable for any number of trees when all trees are of bounded degree [8].

We study the approximability of the complement of this problem, whereby we seek a minimum sized subset of the leaves so that restricted to the remaining

leaves, all the trees are compatible. We call this the *Complement of the Maximum Compatible Subset Problem*, or *CMCS*. We will show that we can transform the input to the CMCS problem into an instance of the well-known Hitting Set problem, so that we can obtain a 4-approximation algorithm to the problem. The explicit calculation of this transformation yields a polynomial time algorithm which has high degree. We then show that we can avoid this explicit calculation, and obtain a 4-approximation algorithm in $O(n^2k^2)$ time, where $n = |S|$ and $k = |\mathcal{T}|$. We also present a slower $O(n^3k^2)$ algorithm using similar techniques which has a guaranteed 3-approximation ratio.

The rest of the paper is organized as follows. In Section 2 we present the basic definitions and lemmas, and we outline a brute force 4-approximation algorithm which is computationally expensive. In Section 3 we describe our $O(k^2n^2)$ algorithm which also achieves the 4-approximation ratio, and its proof of correctness and running time analysis. We present the 3-approximation algorithm for the same problem which runs in $O(k^2n^3)$ in Section 4. In Section 5 we describe the context in which this problem arises in computational biology, and we discuss the uses of these algorithms for this application.

2 Basics

In this section we present the basic definitions and lemmas that motivate a brute force algorithm that forms the basis for our $O(k^2n^2)$ 4-approximation algorithm for the CMCS problem.

Definition 1. (*Tree compatibility*): A set of trees $\mathcal{T} = \{T_1, T_2, \dots, T_k\}$ on the set of leaves S is said to be compatible if there is a tree that is a common refinement of trees in \mathcal{T} .

Definition 2. (*Restriction of a tree to a set of leaves*): The restriction of a tree T to a set X of leaves, denoted $T|X$, is the tree obtained by deleting the leaves in X from T and then suppressing all internal nodes of degree 2.

In the CMCS problem, we seek a minimum cardinality subset X of S such that $\{T|(S - X) : T \in \mathcal{T}\}$ is a compatible set of trees.

Much of the theory we develop rests upon the concept of compatibility of bipartitions, which we now define.

Definition 3. (*The set $C(T)$, the bipartition encoding of T*): Removing an edge e from a leaf-labelled tree T induces a bipartition π_e on its set S of leaves. We denote by $C(T)$ the set $\{\pi_e : e \in E(T)\}$ and note that it uniquely identifies T , up to degree 2 nodes.

Definition 4. (*Bipartition (or edge) compatibility*): A set of bipartitions B is said to be compatible if and only if $B \subseteq C(T)$ for some tree T . We will call

a pair of edges (e, e') compatible if the corresponding bipartitions induced by these edges are compatible.

We now state the relationship between tree compatibility and bipartition compatibility.

Theorem 1. *A set \mathcal{T} of trees is compatible if and only if the set $\bigcup_{T \in \mathcal{T}} C(T)$ is compatible.*

Hence, if we wish to determine if a set of trees is compatible, we need only determine whether the set of bipartitions of all the trees in the set is compatible. This can be determined in $O(kn)$ time [6, 12], where \mathcal{T} has k trees, each on the same n leaves.

Our 4-approximation algorithm operates by eliminating taxa from the set of trees which cause the set of bipartitions to be incompatible. We therefore rely upon the following theory.

Lemma 1 (From Buneman [2]). *A set of bipartitions is compatible iff any two bipartitions in the set are pairwise compatible. Furthermore, two bipartitions $A = A_1 : A_2$ and $B = B_1 : B_2$ are compatible iff at least one of the four sets $A_1 \cap B_1$, $A_1 \cap B_2$, $A_2 \cap B_1$ and $A_2 \cap B_2$ is empty.*

Definition 5. (Quartet Tree): *A quartet tree is an unrooted tree on four leaves that does not have any internal node of degree two. We denote the quartet tree induced by a tree T on a set q of four leaves by $T|q$. We will call q the basis of the quartet tree $T|q$. Two quartet trees on the same leaf set are said to be incompatible if both are binary trees and they differ.*

Corollary 1. *A set \mathcal{T} of trees is incompatible if and only if there exists a quartet q of leaves and a pair of trees T_1 and T_2 in \mathcal{T} such that $T_1|q$ and $T_2|q$ are incompatible quartet trees.*

Proof. If the set is incompatible, then there is a pair of bipartitions $A = A_1 : A_2$ and $B = B_1 : B_2$ (coming from trees T and T' , respectively) and leaves a, b, c, d such that $a \in A_1 \cap B_1, b \in A_1 \cap B_2, c \in A_2 \cap B_1$, and $d \in A_2 \cap B_2$. Hence, T induces $ab|cd$ and T' induces $ac|bd$, so that $T|\{a, b, c, d\}$ and $T'|\{a, b, c, d\}$ are incompatible. The converse is trivial.

We now introduce the Hitting Set problem.

Definition 6. (Hitting Set:) *Let $\mathcal{X} \subseteq 2^A$ be a collection of subsets of a ground set A . A hitting set for \mathcal{X} is a subset $V \subset A$ such that for all $X \in \mathcal{X} \exists a \in V$ such that $a \in X$.*

The best known version of the Hitting Set problem is the *Vertex Cover* problem: given a graph $G = (V, E)$, find a minimum subset $A \subset V$ so that every edge in E has at least one of its endpoints in A . This problem is NP-Hard but can be 2-approximated in a simple algorithm which greedily accumulates a *maximal* matching E_0 within E (so that no two edges within E_0 share an endpoint). Then the vertex set $A = \{v : \exists w \in V : (v, w) \in E_0\}$ is a vertex cover for G which is at most twice the size of an optimal vertex cover. Thus, the NP-hard Vertex Cover problem can be 2-approximated in polynomial time.

More generally, the Hitting Set problem can be p -approximated in polynomial time using the same greedy technique, where $p = \max\{|S| : S \in \mathcal{X}\}$.

Our algorithms are based upon a connection between the Hitting Set problem and the CMCS problem. We begin by defining the set \mathcal{Q}_{inc} :

Definition 7. (\mathcal{Q}_{inc}): Let \mathcal{T} be a set of trees leaf-labelled by the same set S , and let \mathcal{Q}_{inc} denote the set of all quartets on which the trees in \mathcal{T} are incompatible.

The following lemma explains the connection between the Hitting Set problem and the CMCS problems.

Lemma 2. Let \mathcal{T} be a set of trees on a set of leaves S . Then, H is a hitting set for \mathcal{Q}_{inc} if and only if $S - H$ is a compatible set for \mathcal{T} .

Proof. Let H be a hitting set for \mathcal{Q}_{inc} . Hence, for every $q \in \mathcal{Q}_{inc}$, there is at least one $s \in H$ such that $s \in q$. Now consider any pair of trees T_1 and T_2 , when restricted to $S - H$. By Corollary 1 they are compatible, since they have no quartets on which they are incompatible. Hence the set of trees \mathcal{T} restricted to $S - H$ is compatible. For the converse, if $X \subseteq S$ is such that all trees in \mathcal{T} are compatible when restricted to $S - X$, then there are no incompatible quartets left; hence X is a hitting set for \mathcal{Q}_{inc} .

Thus, for us \mathcal{Q}_{inc} is the set \mathcal{X} in the definition of the Hitting Set problem. Hence, as argued before, we can 4-approximate the CMCS problem in polynomial time:

Theorem 2. Let Q be a maximal collection of pairwise disjoint quartets drawn from \mathcal{Q}_{inc} . Let $S_Q = \bigcup_{q \in Q} q$. Then, \mathcal{T} is compatible on $S - S_Q$, and $|S_Q| \leq 4|H^{opt}|$, where H^{opt} is an optimal solution to the CMCS problem on input \mathcal{T} .

(The proof follows from the previous lemma.)

As in all Hitting Set problems, this suggests a straightforward algorithm to obtain a 4-approximation for the CMCS problem. That is, compute \mathcal{Q}_{inc} , and then find a maximal collection of pairwise disjoint quartets within this set:

APPROX-HITTING-SET(\mathcal{Q}_{inc})

```
 $H \leftarrow \emptyset$   
 $\mathcal{P} \leftarrow \mathcal{Q}_{inc}$   
while  $\mathcal{P} \neq \emptyset$   
    pick an arbitrary set  $q \in \mathcal{P}$   
     $H \leftarrow H \cup q$   
    remove every  $X \in \mathcal{P}$  such that  $q \cap X \neq \emptyset$   
return  $H$ 
```

The above procedure would give us a hitting set for \mathcal{Q}_{inc} that is at most four times the size of an optimal hitting set. However, the above outlined procedure depends on explicitly enumerating all the incompatible quartets, and this is expensive. (There are $O(n^4)$ such quartets, and determining if a set of k trees is incompatible on a given quartet takes $O(nk)$ time.) This brute force approach for finding a 4-approximation to the hitting set problem is therefore not practical, as it uses $O(kn^5)$ time.

3 The efficient 4-approximation algorithm

3.1 Outline of the algorithm

Note that the input to the CMCS problem is a set of k unrooted trees, which *implicitly* codes for the set \mathcal{Q}_{inc} . The approach we use for obtaining a faster 4-approximation algorithm takes advantage of this implicit representation of \mathcal{Q}_{inc} , by never explicitly calculating \mathcal{Q}_{inc} .

The key to an efficient algorithm is efficient identification of incompatible quartets. In our algorithm we do not generate all the incompatible quartets - rather, we identify incompatible quartets through (on the fly) identification of incompatible *bipartitions*. That is, we show that we can efficiently identify a pair of incompatible bipartitions, and from that pair of bipartitions we can then efficiently obtain an incompatible quartet, so that in $O(k^2n^2)$ time we have obtained a hitting set of size at most four times the optimal hitting set. This is the basic idea of our algorithm. Here we describe this basic idea just on two trees, noting that the algorithm for k trees operates by repeatedly examining pairs of trees, finding incompatible quartets, and deleting each of the leaves in the incompatible quartets from all the trees in the input.

MODIFIED-APPROX-HITTING-SET(T_1, T_2)

```
/*  $T_1$  and  $T_2$  are two trees on the set of leaves  $S$  */  
1    $H \leftarrow \emptyset$ ;  $S' \leftarrow S$   
2   Mark all edges in  $T_1$  as BAD  
3   let  $e$  be the first BAD edge
```

```

4   while there are BAD edges
5       if  $e$  is compatible with all edges in  $T_2$ 
6           mark  $e$  as GOOD
7            $e \leftarrow$  next BAD edge, if there is one
8       else
9           find an edge  $e'$  in  $T_2$  that  $e$  and  $e'$  are incompatible
10          let  $q$  be an incompatible basis as in Corollary 1
11           $S' \leftarrow S' - q$ 
12           $H \leftarrow H \cup q$ 
13          remove leaves in  $Q$  from  $T_1$  and  $T_2$ 
            (but do not restrict  $T_1$  and  $T_2$  to  $S'$ )
14  return  $H$ 

```

It can be seen easily that MODIFIED-APPROX-HITTING-SET is correct. Note that since the while loop in line 4 terminates only when there are no BAD edges, all the edges remaining in T_1 at the end are compatible with the edges in T_2 . Hence, by Lemma 1 $T_1|(S - H)$ is compatible with $T_2|(S - H)$. Hence $S - H$ is indeed a compatible set for $\{T_1, T_2\}$, and thus, H is a hitting set for the collection of all incompatible bases for $\{T_1, T_2\}$. Moreover, since H is the union of *disjoint* bases (each of which is of cardinality four), $|H| \leq 4|H^{opt}|$, where $S - H^{opt}$ is an MCS for $\{T_1, T_2\}$.

Note that the while loop terminates after $O(n)$ iterations, since in each iteration it eliminates four leaves or marks an edge as GOOD and there are $O(n)$ edges in T_1 . Hence, if we ensure that each iteration can be completed in $O(n)$ time, we would have an $O(n^2)$ time algorithm. In the next section we will describe how this running time can be achieved.

3.2 Achieving $O(n^2)$ running time for the CMCS of two trees

From the outline of the algorithm in the previous section, it can be observed that the key to achieving $O(n^2)$ time bound is identifying an edge $e \in E(T_1)$ in $O(n)$ time such that e is either compatible with all the edges in T_2 or is incompatible with some edge in T_2 . We will now show how the above objective can be achieved.

We modify and extend MODIFIED-APPROX-HITTING-SET thus:

1. Pick a leaf arbitrarily, and root both the trees on this leaf. Let the rooted trees be T'_1 and T'_2 .
2. For each node in T'_1 maintain a sorted list of all leaves below it. We denote the set of leaves below node v as L_v . We maintain the sorted set as an indexed (doubly) linked list, to facilitate fast deletions. Such a data structure suffices since we will only delete from the list and never insert anything.

3. Let (v, w) be an edge in T_1' with w below v . For each node x in T_2' , compute two quantities n_x and n'_x defined as follows: $n_x = |L_x \cap L_w|$ and $n'_x = |L_x \cap (S' - L_w)|$. Here, S' is the *current* set of leaves in T_1' and T_2' .

The following lemma allows us to determine if an edge in T_1 is compatible with all other edges in T_2 or not.

Lemma 3. *Let (v, w) be an edge in T_1 and let (y, x) be an edge in T_2 . Without loss of generality assume that v is the parent of w in T_1' and that y is the parent of x in T_2' . Then (v, w) is incompatible with (y, x) iff $0 < n_x < |L_w|$ and $0 < n'_x < |S' - L_w|$.*

Proof. The proof follows from the following four observations.

1. $n_x > 0$ iff there exists a leaf $l \in L_x$ in T_2' that is in L_w in T_1' , in other words if and only if $L_x \cap L_w \neq \emptyset$.
2. $n_x < |L_w|$ iff there exists a leaf $l \in L_w$ that is *not* in L_x . In other words iff $L_w \cap (S' - L_x) \neq \emptyset$.
3. $n'_x > 0$ iff there is a leaf that is not in L_w but is in L_x . In other words iff $(S' - L_w) \cap L_x \neq \emptyset$.
4. $n'_x < |S' - L_w|$ iff there is a leaf that is neither in L_x nor in L_w . In other words iff $(S' - L_x) \cap (S' - L_w) \neq \emptyset$.

We can compute n_x and n'_x for all nodes x in T_2' in $O(n)$ time by doing the computations bottom up. We then inspect all edges in T_2' to see if they are compatible with the edge (v, w) (which is in T_1'). This can again be accomplished in $O(n)$ time since there are at most $O(n)$ edges. Hence, we can identify if an edge (v, w) is GOOD or not in $O(n)$ time.

In case we determine it to be GOOD, we do no further processing. In case we find that the edge is incompatible with an edge (y, x) we have to do some further processing.

1. We compute a basis Q by computing $L_w \cap L_x$, $L_w \cap (S' - L_x)$, $(S' - L_w) \cap L_x$ and $(S' - L_w) \cap (S' - L_x)$ and picking one element from each set. Each of the four intersection computations can be carried out in $O(n)$ time since we maintain the sets sorted.
2. We remove the leaves in Q from T_1 and T_2 (but we do not eliminate nodes of degree two that may be created due to the removal of Q) and hence we have to update the sorted set of leaves maintained in each node. A deletion from this set can be carried out in $O(1)$ time since the set is indexed and doubly linked. Hence, the time taken for the removal of Q from the two trees is in $O(n)$.

The above discussion shows that both the *if* and *else* clause in the *while* loop in our algorithm can be accomplished in $O(n)$ time. We also noted that the loop terminates in $O(n)$ iterations. Hence the loop takes $O(n^2)$ time. The rest of the algorithm takes $O(n)$ time. Thus, the entire algorithm can be made to run in $O(n^2)$ time. The extension to k trees is straightforward, as we just greedily compute incompatible quartet trees, and delete the leaves from each of the trees in the input. Hence we have the following theorem.

Theorem 3. *Given a set \mathcal{T} of k unrooted trees on a set of leaves S , let $S - H^{opt}$ be a maximum compatible set for \mathcal{T} . We can compute a compatible set $S - H$ such that $|H| \leq 4|H^{opt}|$ in $O(k^2n^2)$ time.*

4 A slower algorithm with a better approximation ratio

In this section we describe how to modify the previously described algorithm to achieve a 3-approximation ratio, albeit using more time. The algorithm operates by approximating the solution to the rooted version of the CMCS problem, but must apply it to n subproblems, where the i^{th} subproblem considers all the trees in \mathcal{T} rooted at leaf i . By taking the minimum of all the solutions obtained, we can compute the overall solution with the desired approximation bound. (Later we give an example of two unrooted trees which shows why it does not suffice to just look at a single rooting.) Although we can show that each individual rooted problem can be solved again in $O(k^2n^2)$ problem, because we look at n subproblems this approach is $O(n)$ slower than the 4-approximation algorithm we showed earlier. Hence, this is a slower algorithm, but it achieves a better guaranteed approximation ratio than our first algorithm.

We begin by making some definitions.

Definition 8. (*Triplet Tree*): A triplet tree is a rooted tree on three leaves that does not have any internal node of degree two. We use the notation $T|q$ to denote the triplet tree induced by a rooted tree T on a set $q = \{a, b, c\}$ of three leaves. The star triplet tree (where all three leaves are children of the root) is denoted by (a, b, c) , whereas $((a, b), c)$ (or its equivalent form, $(c, (a, b))$) denotes the triplet tree in which a and b are siblings, and have a least common ancestor below the root. We will call q the basis of the triplet tree $T|q$.

Definition 9. (*Triplet Compatibility*): Two triplet trees, each on the same leaf set, are said to be compatible if at least one of them is a star or they are the same. Otherwise they are said to be incompatible.

As in unrooted trees, we can talk about the subtree of a rooted tree induced by a set of leaves. Here, however, we do not suppress a node of degree two if it is the root. Corresponding to the set Q_{inc} of incompatible quartet trees, we can define the set $Trip_{inc}$ of incompatible triplet trees:

Definition 10. ($Trip_{inc}$): Given a set \mathcal{T} of rooted trees, each leaf-labelled by the same set S of leaves, we define the set $Trip_{inc} = \{\{a, b, c\} \subseteq S : \exists \{A, B\} \subseteq \mathcal{T} : A| \{a, b, c\} \text{ and } B| \{a, b, c\} \text{ induce incompatible triplet trees}\}$.

Now let A be an unrooted tree on the set S of leaves $S = \{1, 2, \dots, n\}$, and let $i \in S$ be arbitrary.

Definition 11. (A^i): A^i refers to the rooted tree on the leaf set $S - \{i\}$, obtained by rooting A at the leaf i , and deleting i and its incident edge.

As before, we let L_v represent the leaves below the node v in a tree (whose identity will be known in context).

We now show through the following lemma that the incompatibility question for unrooted trees can be rephrased in terms of incompatibility of induced triplets, an idea crucial to improving the approximation ratio. We state the lemma in terms of two trees, but the result obviously extends to any number of trees.

Lemma 4. Let A and B be two trees on leaf set S , and let $i \in S$ be a fixed leaf. Then A^i and B^i induce no incompatible triplet trees if and only if the unrooted trees A and B are compatible.

Proof. First we prove that if the rooted pair of trees A^i and B^i have a pair of incompatible triplets, then A and B are incompatible. Suppose that the set of three leaves $\{a, b, c\}$ is an incompatible triplet, so that A^i induces the triplet tree $(a, (b, c))$ and B^i induces the triplet tree $((a, b), c)$. Then we can see that on $\{a, b, c, i\}$, tree A induces the split $\{a, i\} | \{b, c\}$ whereas tree B induces the split $\{a, b\} | \{c, i\}$. Hence, A and B are incompatible.

Suppose now that A and B are incompatible trees. Hence, there is a set $q = \{a, b, c, d\}$ on which the two trees induce two different splits. Recall that i is an arbitrary fixed leaf. We will show no matter how i is chosen, we can select three leaves from q so that A^i and B^i induce incompatible triplet trees for that subset. Let A induce $\{a, b\} | \{c, d\}$ and B' induce $\{a, c\} | \{b, d\}$ on q . We first address the case where $i \in \{a, b, c, d\}$. In this case, without loss of generality assume $i = d$. Then $A^i (= A^d)$ induces $((a, b), c)$ on leaf set a, b, c , whereas $B^i (= B^d)$ induces $((a, c), b)$. Hence, A^i and B^i induce incompatible triplet trees. Now we consider the case where $i \notin \{a, b, c, d\}$. Consider the restriction of the rooted trees A^i and B^i to the leaves a, b, c, d . In each of these rooted trees there is at least one sibling pair of leaves. Since A induces the quartet $ab|cd$, the only possible sibling pairs in $A^i| \{a, b, c, d\}$ are a, b and c, d . Assume, without loss of generality, that a, b are siblings in $A^i| \{a, b, c, d\}$. Similarly, the only possible sibling pairs in $B^i| \{a, b, c, d\}$ are a, c and b, d . Again, without loss of generality, assume a, c are siblings in $B^i| \{a, b, c, d\}$. Then on the set a, b, c of leaves the

rooted trees A^i and B^i differ: $A^i|_{\{a, b, c\}} = ((a, b), c)$ whereas $B^i|_{\{a, b, c\}} = ((a, c), b)$. Hence, a, b, c induces incompatible quartet trees in A^i and B^i .

We now state the **Maximum Compatible Subset of Rooted Trees (MCSR)** problem:

- **Input:** Set \mathcal{T} of rooted trees, each leaf-labelled by the same set L of leaves.
- **Output:** Subset $L' \subseteq L$ so that the trees in \mathcal{T} restricted to the leaves in L' do not induce any incompatible triplet trees.

The next observation follows directly from Lemma 4, and we state it without proof:

Observation 1 *Let \mathcal{T} be a set of unrooted trees leaf-labelled by the same set S , and assume that $|S| \geq 1$. Let $i \in S$ be arbitrary. For every subset $S_0 \subset S - \{i\}$, if S_0 is a feasible solution to the MCSR problem on set $\{T^i : T \in \mathcal{T}\}$ of trees, then $S_0 \cup \{i\}$ is a feasible solution to the MCS problem on \mathcal{T} .*

This suggests the following general strategy for the MCS problem:

- For each $i \in S$, let $L(i)$ denote the MCSR of the set of rooted trees $\{T^i : T \in \mathcal{T}\}$.
- Let i^* be selected so that $|L(i^*)|$ is maximum among $\{|L(i)| : 1 \leq i \leq n = |S|\}$. Return $L(i^*) \cup \{i^*\}$.

Comments:

(1) Observation 1 implies that this algorithm exactly solves the MCS problem, but that it requires n iterations of an exact solution to the MCSR problem. However, a little arithmetic and Observation 1 also imply that an r -approximation algorithm for the CMCSR problem (that is, for the complement of the MCSR problem) would yield an r -approximation algorithm for the CMCS problem. Hence, although CMCSR is also NP-hard, if we can approximate CMCSR we can use that algorithm to approximate CMCS, and with the same guaranteed performance ratio. This is the approach we will take.

(2) Note also that we cannot simply pick a single leaf and root the trees at this leaf, and use the solution obtained for the resultant rooted trees. This approach can be arbitrarily bad. Consider the two caterpillar trees T and T' on leaf set $1, 2, 3, \dots, 100, x$, as shown in Figure 2. These two trees clearly have a very large compatible subset: namely, the set $1, 2, \dots, 100$ of leaves. However, if we root T and T' at leaf x , then they have very small compatible subsets as rooted trees. Similarly, arbitrary rootings of the trees T and T' will greatly affect the size of the compatible subset of the leaves that is obtained.

Approximating the CMCSR problem: We now show how we can 3-approximate the complement of the MCSR problem. The technique is essentially the same as for the complement of the MCS problem, but rather than 4-approximating the Hitting Set for Q_{inc} we 3-approximate the Hitting Set for Tri_{inc} . Here, too, we can use the same techniques we used in the unrooted version (the CMCS problem) in order to get an $O(n^2)$ algorithm for the CMCSR of two trees. Since we do this for each possible leaf at which to root each of the trees, this gives us an $O(n^3)$ 3-approximation algorithm for the CMCSR problem on 2 trees, and hence, an $O(k^2n^3)$ 3-approximation algorithm for k trees.

We now show how we do this. Let A and B be two unrooted trees on leaf set L , and root each at leaf i thus producing A^i and B^i . Let $L' = L - \{i\}$. As before, let L_x denote the leaves below node x . The only variant is how we calculate incompatible triplets “on the fly”. As before, we look for “bad” edges. Suppose we find a bad edge (v, w) in A^i that is incompatible with edge (x, y) in B^i . As before, we identify one leaf from each of the four sets $L_w \cap L_y$, $L_w \cap (L' - L_y)$ and $(L' - L_w) \cap L_y$ and $(L' - L_y) \cap (L' - L_w)$. Let these leaves be a, b, c and d respectively. Earlier, we eliminated all the four leaves, whereas now we eliminate only a, b and c from A^i and B^i leaving d untouched.

Observe that the set $\{a, b, c\}$ can not be part of any compatible set for the rooted trees A^i and B^i since when restricted to $\{a, b, c\}$, A^i induces $((a, b), c)$ and B^i induces $(a, (b, c))$. Hence, at least one leaf from $\{a, b, c\}$ has to be eliminated. Moreover observe that $\{a, b, c, i\}$ forms an incompatible quartet for the unrooted trees A and B . Hence, all incompatible induced triplets can be identified in the above manner. Hence we have the following theorem:

Theorem 4. *Given a set \mathcal{T} of k unrooted trees on a set of leaves L , let $L - H^{opt}(\mathcal{T})$ be a maximum compatible set for the trees. We can compute a compatible set $L - H$ such that $|H| \leq 3|H^{opt}|$ in $O(k^2n^3)$ time.*

5 Application in Computational Biology

A “phylogeny” (or evolutionary tree) represents the evolutionary history of a set of species S by a rooted (typically binary) tree in which the leaves are labelled with elements from S , and with internal nodes unlabelled.

A standard approach for inferring phylogenies is to attempt to solve a hard optimization problem (such as Maximum Parsimony or Maximum Likelihood), and this can result in hundreds or thousands of equally good trees obtained. A consensus tree is then used to represent the entire set of trees. However, standard consensus methods, such as the majority consensus and strict consensus, can return highly unresolved trees. For example, the strict consensus tree is that tree which is the most resolved common contraction of all the trees in the input

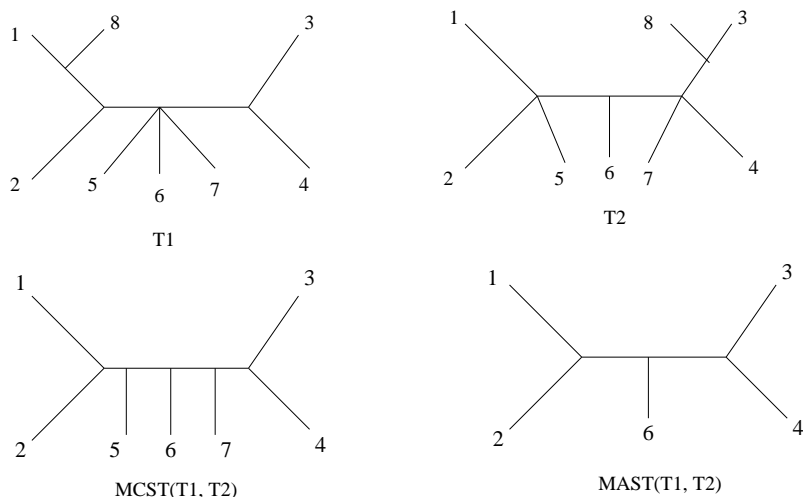


Fig. 1. We calculate the maximum agreement subtree (MAST) and the maximum compatible subtree (MCST) of two trees, and show they are different.

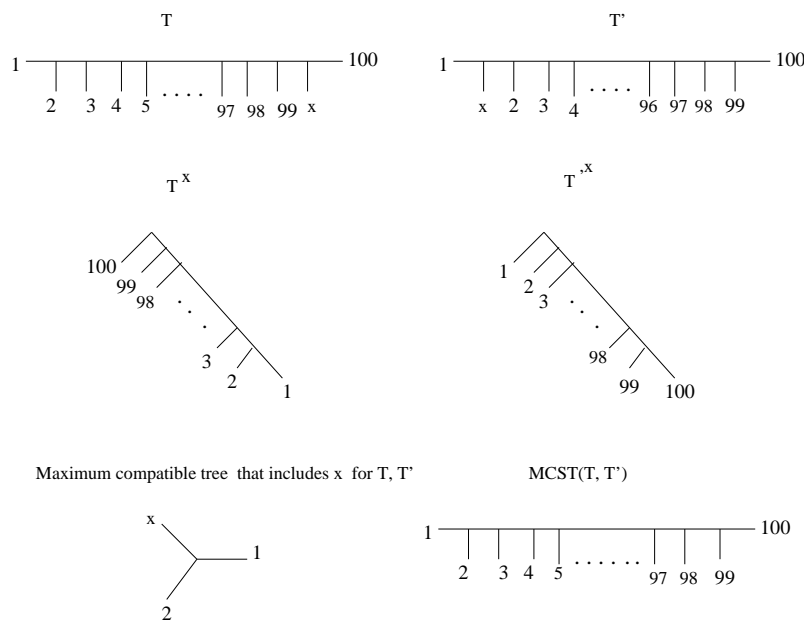


Fig. 2. Why we cannot root trees arbitrarily.

set. This consensus tree is highly unresolved when one species is sufficiently distantly related to the other species, so that its location in the tree is not well localized. Such a species is called a “rogue taxon”, and is known to cause prob-

lems in phylogenetic analysis. The majority consensus tree is similarly poorly affected. (See [11] for a discussion of phylogenetic inference in practice.)

The identification, and elimination, of rogue taxa from phylogenetic analyses has led to alternative consensus approaches which reduce the set of species (by eliminating a small set of species) so that the resultant trees have a well resolved consensus tree. One such approach, originally posed in [5], is the “Maximum Agreement Subtree Problem”, or MAST: this consensus method eliminates the smallest subset of the species so that all the trees agree on the remaining species. (The largest common subtree in all the trees is the maximum agreement subtree, or MAST, of the input.) The MAST problem is NP-hard for three or more trees [1], but polynomial time for two trees [4, 9] or for any number of trees if at least one tree has bounded degree [3]. Software for the MAST problem exists in the PAUP [10] software package, but is not very useful since the size of the MAST is often not large enough to be interesting (David Swofford, personal communication). The problem seems to be that requiring complete agreement between the trees, even when restricted to a smaller set of taxa, is too strong a requirement.

A potential explanation for this phenomenon lies in the difficulty in correctly resolving “short edges” in the true tree. In this case, optimal solutions for problems such as Maximum Parsimony or Maximum Likelihood may differ significantly in terms of how they resolve around these short edges. This results in having the MAST of the set of trees being very small. On the other hand, the identification and contraction of short edges in each of the ML or MP trees could result in a collection of trees that, while not binary, have a common refinement on some large subset of leaves. This suggests the following approach:

- First, contract all short enough edges, and
- Second, identify and delete a minimum set of leaves so that after these leaves are deleted and degree two nodes suppressed, the resultant trees share a common refinement (i.e. are compatible). If a consensus tree is desired, the minimum common refinement (i.e. the maximum compatibility subtree, or MCST) of the resultant trees is returned.

The first step is straightforward (though establishing how short is short enough needs to be studied), but the second step is essentially the CMCS problem that we studied in this paper.

(Note that there is a relationship between the number of leaves in the MAST (maximum agreement subtree) of a set of trees and its MCST (maximum compatibility subtree). By definition, the number of leaves in a MCST is at least as big as that of the MAST, and this can often be significantly larger. See Figure 1 for an example of two trees and their MAST and MCST, for which there is

a difference in sizes. Thus, the MCST problem is a better model for consensus trees, as it retains more information and is less affected by noise in the input.)

Thus, our approximation algorithm for the CMCS problem allows us to achieve two goals: obtain meaningful and informative consensus trees and identify rogue taxa. Because our algorithms are approximations, they are only useful when the trees do share a large common subset of the taxa on which they are compatible. This is an assumption which seems to be believed by many systematic biologists, but one which we can now test through the use of these algorithms. Thus, these algorithms can serve an additional purpose: enabling us to validate (or prove incorrect) the assumptions of systematic biologists about the level of phylogenetic signal in their datasets, and the performance of their methods for estimating phylogenetic trees.

6 Acknowledgments

This research was supported by a Fellowship in Science and Engineering from the David and Lucile Packard Foundation to Warnow, and by NSF grant EIA 01-21680 to Warnow.

References

- [1] A. Amir and D. Keselman. Maximum agreement subtrees in a set of evolutionary evolutionary trees: Metrics and efficient algorithms. *SIAM Journal of Computing*, 26(6):1656–1669, 1997. A preliminary version of this paper appeared in FOCS '94.
- [2] P. Buneman. The recovery of trees from measures of dissimilarity. *Mathematics in the Archaeological and historical Sciences*, pages 387–395, 1971.
- [3] M. F. Colton, T.M. Przytycka, and M. Thorup. On the agreement of many trees. *Information Processing Letters*, 55:297–301, 1995.
- [4] M. F. Colton and M. Thorup. Fast comparison of evolutionary trees. In *Proc. of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 481–488, 1994.
- [5] C.R. Finden and A.D. Gordon. Obtaining common pruned trees. *Journal of Classification*, 2:255–276, 1985.
- [6] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28, 1991.
- [7] A. Hamel and M. A. Steel. Finding a maximum compatible tree is NP-hard for sequences and trees. *Applied Mathematics Letters*, 9(2):55–60, 1996.
- [8] J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Discrete and Applied Mathematics*, 71:153–169, 1996.
- [9] M. Steel and T. Warnow. Kaikoura tree theorems: computing the maximum agreement subtree. *Information Processing Letters*, 48:77–82, 1993.
- [10] D. Swofford. PAUP*: Phylogenetic analysis using parsimony (and other methods), version 4.0. 1996.
- [11] D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis. Phylogenetic inference. In D. M. Hillis, B. K. Mable, and C. Moritz, editors, *Molecular Systematics*, pages 407–514. Sinauer Assoc., 1996.
- [12] T. Warnow. Tree compatibility and inferring evolutionary history. *Journal of Algorithms*, 16:388–407, 1994.