On the Approximation of Computing Evolutionary Trees

Vincent Berry*, Sylvain Guillemot, François Nicolas, and Christophe Paul

Département Informatique, L.I.R.M.M. - C.N.R.S.
161 rue Ada, 34392 Montpellier Cedex 5
{vberry,sguillem,nicolas,paul}@lirmm.fr

Abstract. Given a set of leaf-labelled trees with identical leaf sets, the well-known MAST problem consists of finding a subtree homeomorphically included in all input trees and with the largest number of leaves. MAST and its variant called MCT are of particular interest in computational biology. This paper presents positive and negative results on the approximation of MAST, MCT and their complement versions, denoted CMAST and CMCT.
For CMAST and CMCT on rooted trees we give 3-approximation algorithms achieving significantly lower running times than those previously known. In particular, the algorithm for CMAST runs in linear time. The approximation threshold for CMAST, resp. CMCT, is shown to be the same whenever collections of rooted trees or of unrooted trees are considered. Moreover, hardness of approximation results are stated for CMAST, CMCT and MCT on small number of trees, and for MCT on unbounded number of trees.

1 Introduction

Given a set of leaf-labelled trees with identical leaf sets, the well-known Maximum Agreement SubTree problem (MAST) consists of finding a subtree homeomorphically included in all input trees and with the largest number of leaves [2, 7, 10, 13, 21, 22]. In other words, this involves selecting a largest set of input leaves such that the input trees are isomorphic, i.e. *agree* with each other, when restricted to these leaves.

This problem arises in various areas including phylogenetics which is concerned with *evolutionary trees*, i.e. trees representing the evolutionary history of a set of species: the leaves of the tree are in one-to-one correspondence with species under study and the branching pattern of the tree describes the way in which speciation events lead from ancestral species to more recent ones. In phylogenetics, the MAST problem is used to reach different practical goals: to obtain a consensus of several trees inferred by different methods, or that are optimal for a given criteria; to measure the similarity between different evolutionary scenarii; to identify horizontal transfers of genes. Recently, MAST has

* Supported by the *Act. Incit. Inf.-Math.-Phys. en Biol. Mol.* [ACI IMP-Bio] and the *Act. Inter. Incit. Région.* [BIOSTIC-LR].

L. Wang (Ed.): COCOON 2005, LNCS 3595, pp. 115–125, 2005.
© Springer-Verlag Berlin Heidelberg 2005

been extended to the context of supertrees where input trees can have different sets of leaves [3].

The Maximum Compatible Tree problem (MCT) is a variant of MAST that is of particular interest in phylogenetics when the input trees are not binary [11, 12, 15, 17]. MCT requires that selected subtrees of the input trees are *compatible*, *i.e.* that groups of leaves they define can all be combined in a same tree. This is less strict than requiring the isomorphism of the subtrees, hence usually leads to selecting a larger set of leaves than allowed by MAST.

We give below a brief overview of the litterature, precising how the results presented in this paper relate to previously known results. The MAST problem is NP-hard on three rooted trees of unbounded degree [2], and MCT on two rooted trees if one of them is of unbounded degree [17]. Subquadratic algorithms have been proposed for MAST on two rooted n-leaf trees [7, 20, 21]. When dealing with k rooted trees, MAST can be solved in $O(n^d + kn^3)$ time provided that the degree of one of the input trees is bounded by d [2, 6, 10], and MCT can be solved in $O(2^{2kd}n^k)$ time provided that all input trees have degree bounded by d [12]. Both problems can be solved in $O(\min\{3^p kn, 2.27^p + kn^3\})$ time, i.e. are FPT in p, where p is the smallest number of leaves to be removed from the input set of leaves so that the input trees agree [3].

More generally, when the previously mentioned parameters are unbounded, several works (starting from [2]) propose 3-approximation algorithms for CMAST and CMCT, where CMAST, resp. CMCT, is the complement version of MAST, resp. MCT, i.e. aims at selecting the smallest number of leaves to be removed from the input trees in order to obtain their agreement. In practice, input trees usually agree on the position of most leaves, thus approximating CMAST and CMCT is more relevant than approximating MAST and MCT. For CMCT, [11] propose an $O(k^2n^2)$ time 3-approximation algorithm. We propose here an $O(n^2 + kn)$ time algorithm. For MAST, [3] propose an $O(kn^3)$ time algorithm. Here we improve on this result by providing a linear time, i.e. $O(kn)$, algorithm. We also state that rooted and unrooted versions of CMAST (and CMCT) have the same approximation threshold.

Let k-MAST, resp. k-MCT, resp. k-CMAST, resp. k-CMCT, denote the particular case of MAST, resp. MCT, resp. CMAST, resp. CMCT, dealing with k rooted trees. Negative results for these problems are as follows:

- For all $\epsilon > 0$, the general MAST problem is not approximable within $n^{1-\epsilon}$ unless NP = ZPP [5]. A similar result is obtained here for MCT.
- It also stated here that 3-CMAST and 2-CMCT are APX-hard, i.e. that they do not admit a PTAS unless P = NP.
- For all $\delta < 1$, 3-MAST is not approximable within $2^{\log^\delta n}$ unless NP $\subseteq$ DTIME$\left(2^{\text{polylog}\, n}\right)$ [17]. The same result is obtained here for 2-MCT.

2 Definitions and Preliminaries

A rooted *evolutionary tree* is a tree whose leaf set $L(T)$ is in bijection with a label set, and whose internal nodes have at least two children. Hereafter, we only

consider such trees and identify leaves with their respective labels. The *size* of a tree T (denoted $\#T$) is the number of its leaves: $\#T = \#L(T)$.

Let u be a node of a tree T, $S(u)$ stands for the subtree rooted at u, $L(u)$ for the leaves of this subtree, and $d^+(u)$ for the number of children of u. For a set of leaves $L \subseteq L(T)$, $lca_T(L)$ denotes the lowest common ancestor of leaves L in T. Given a set L of labels and a tree T, the *restriction* of T to L, denoted $T|L$, is the tree homeomorphic to the smallest subtree of T connecting leaves of L.

Lemma 1. *Let T_1 and T_2 be two isomorphic trees with leaf set L, and let $L' \subseteq L$, then $T_1|L'$ is isomorphic to $T_2|L'$.*

Given a collection $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of trees on a same leaf set L of cardinality n, an *agreement subtree* of $\mathcal{T}$ is any tree T with leaves in L s.t. $\forall T_i \in \mathcal{T}$, $T = T_i|L(T)$. The MAST problem consists in finding an agreement subtree of $\mathcal{T}$ with the largest number of leaves. We denote $MAST(\mathcal{T})$ such a tree.

A tree T *refines* a tree T', if T' can be obtained by collapsing certain edges of T, (i.e. merging their extremities). More generally, a tree T refines a collection $\mathcal{T}$, whenever T refines all T_i's in $\mathcal{T}$. Given a collection $\mathcal{T}$ of k trees with identical leaf set L of cardinality n, a tree T with leaves in L is *compatible with* $\mathcal{T}$ iff $\forall T_i \in \mathcal{T}$, T refines $T_i|L(T)$. If there is a tree T compatible with $\mathcal{T}$ s.t. $L(T) = L$, i.e. that is a common refinement of all trees in $\mathcal{T}$, then the collection $\mathcal{T}$ is *compatible*. In this case, a *minimum refinement* T of $\mathcal{T}$ (*i.e.* collapsing a minimum number of edges) is a tree s.t. any tree T' refining $\mathcal{T}$ also refines T. Collections of trees considered in practice are usually not compatible, motivating the MCT problem which aims at finding a tree, denoted $MCT(\mathcal{T})$, compatible with $\mathcal{T}$ and having the largest number of leaves. Remark that MCT is equivalent to MAST when input trees are binary.

For any three leaves a, b, c in a tree T, there are only three possible *binary* shapes for $T|\{a, b, c\}$, denoted $a|bc$, resp. $b|ac$, resp. $c|ab$, depending on their innermost grouping of leaves (bc, resp. ac, resp. ab). These binary trees on 3 leaves are called *rooted triples*. Alternatively $T|\{a, b, c\}$ can be a *fan*, *i.e.* a unique internal node connected to the three leaves. A fan is denoted $\{a, b, c\}$.

We define $rt(T)$, resp. $f(T)$, as the set of rooted triples, resp. fans, induced by the leaves of a tree T. Given a collection $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of trees with leaf set L, a set $\{a, b, c\} \subseteq L$ is a *hard conflict* between (trees of) $\mathcal{T}$ whenever $\exists T_i, T_j \in \mathcal{T}$ s.t. $a|bc \in rt(T_i)$ and $b|ac \in rt(T_j)$. The set $\{a, b, c\}$ is a *soft conflict* between (trees of) $\mathcal{T}$ whenever $a|bc \in rt(T_i)$ and $\{a, b, c\} \in f(T_j)$.

Lemma 2 ([2, 3, 12]). *Two trees with the same leaf set are isomorphic iff there is no* hard nor any soft *conflict between them. A collection $\mathcal{T}$ of trees with the same leaf set is compatible iff there is no* hard *conflict between $\mathcal{T}$.*

Definition 1. *Given a set of conflicts $\mathcal{C}$, let $L(\mathcal{C})$ denote the leaves appearing in $\mathcal{C}$. Given a collection $\mathcal{T}$ with conflicts, an* hs-peacemaker*, resp.* h-peacemaker*, of $\mathcal{T}$ is any set $\mathcal{C}$ of disjoint hard and soft, resp. only hard, conflicts between $\mathcal{T}$ s.t. $\{T_i|(L - L(\mathcal{C})) : T_i \in \mathcal{T}\}$ is a collection of isomorphic trees, resp. compatible*

trees. In other words, removing $L(\mathcal{C})$ from the input trees removes all conflicts, resp. all hard conflicts, between them.

3 Approximation Algorithms

3.1 An $O(n^2 + kn)$ Time 3-Approximation Algorithm for CMCT

Let $\mathcal{T}$ be a collection of trees on an n-leaf set L. It is well-known that $\mathcal{T}$ is compatible iff every pair of trees in $\mathcal{T}$ is compatible [8]. Moreover,

Lemma 3 ([4]). *$\mathcal{T}$ is a compatible collection of trees iff there exists a minimum refinement T of $\mathcal{T}$ and $rt(T) = \bigcup_{T_i \in \mathcal{T}} rt(T_i)$.*

If $\mathcal{T}$ is compatible, a minimum refinement T of $\mathcal{T}$ is a solution for MCT, as $L(T) = L$. From Lemma 2, one can obtain T by first computing a minimum refinement $T_{1,2}$ of two trees $T_1, T_2 \in \mathcal{T}$, and then iterating on $\mathcal{T} - \{T_1, T_2\} \cup \{T_{1,2}\}$ until only one tree remains that is the sought tree T.
If $\mathcal{T}$ is not compatible, then we apply the following:

Lemma 4 ([2, 3, 11]). *Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a collection of trees on a leaf set L and let $\mathcal{C}$ be an hs-peacemaker, resp. an h-peacemaker, of $\mathcal{T}$. Then any tree in $\mathcal{T}|(L - L(\mathcal{C}))$ is a 3-approximation for* CMAST*, resp. any refinement of $\mathcal{T}|(L - L(\mathcal{C}))$ is a 3-approximation for* CMCT*, on $\mathcal{T}$.*

Given a pair of trees, [4] give an $O(n)$ time algorithm that either returns a minimum refinement when the trees are compatible, or otherwise identifies a hard conflict C between them. Thus, from Lemma 4, the procedure sketched above for a compatible collection, can be adapted to obtain a 3-approximation of CMCT for a non-compatible collection $\mathcal{T}$. Apply the algorithm of [4] to a pair of trees $\{T_1, T_2\} \subseteq \mathcal{T}$ to obtain either their minimum refinement $T_{1,2}$ or a hard conflict C. In the latter case, remove C from all input trees and iterate. In the former case, iterate on $\mathcal{T} - \{T_1, T_2\} \cup \{T_{1,2}\}$. When $\mathcal{T}$ is reduced to a single tree, $O(k+n)$ calls to the algorithm of [4] have been issued and the resulting set $\mathcal{C}$ of removed conflicts is an h-peacemaker. Hence:

Theorem 1. *The* CMCT *problem on a collection of k rooted trees on a same n-leaf set can be 3-approximated in $O(n^2 + kn)$ time.*

3.2 A Linear Time 3-Approximation Algorithm for CMAST

W.l.o.g., this section considers input trees on a same n-leaf set labelled by positive integers 1, 2, ..., n. First consider collections $\mathcal{T}$ of two trees. The following characterization of isomorphic trees is the basis of our algorithm.

Lemma 5 ([4]). *Two trees T_i and T_j are isomorphic iff $rt(T_i) = rt(T_j)$ and $f(T_i) = f(T_j)$.*

The definition of $MAST(\mathcal{T})$ is independent of the order of the children of nodes in trees. However, to efficiently compute an approximation of $MAST(\mathcal{T})$, we considered that T_1 and T_2 are *ordered*. Ordering a tree T consists in totally ordering the children of every node in T. Thereby, this uniquely defines a left-right order π_T on the leaves L of T.

Given an arbitrary ordering of T_1, the approximation algorithm first tries to order T_2 accordingly. In the following, π_1, resp. π_2, stands for π_{T_1}, resp. π_{T_2}; and $\pi_2(i)$ stands for the i-th leaf in π_2. W.l.o.g., we also assume $\pi_1 = 1 \dots n$.

Definition 2. *Let π be an order on a set L. A subset S of L is an* interval *of π whenever the elements of S occur consecutively in π (but not necessarily in the same order). A tree T with leaf set L is* embeddable *in an order π on L whenever T can be ordered s.t. $\pi_T = \pi$.*

Lemma 6. *Let T be a tree with leaf set L and π be an arbitrary order of L. Then, T is* embeddable *in π iff for any node u of T, $L(u)$ is an interval of π.*

Proposition 1. *Let T be a tree and π be an order on its leaves. Testing whether T is embeddable in π costs $O(n)$ time. In the positive, ordering T such that $\pi_T = \pi$ can be done in $O(n)$ time.*

The running time stated in this proposition is achieved by performing bottom-up walks on disjoint paths in T, as described by Algorithm 1. For a node u in a tree, let $m(u)$ and $M(u)$ resp. denote the smallest and largest leaf of $L(u)$ in π. Assume the children of any non-leaf node $v \in T$ are originally stored in a doubly-linked list $l_c(v)$ which has to be ordered into a list $l'_c(v)$ so that $\pi_T|L(v) = \pi|L(v)$.

Algorithm 1: TreeOrder(T, π)

```
for any node u in T do l'_c(u) ← ∅ ;
for i = 1 to n do
    let u be the leaf s.t. u = π^{-1}(i);
    repeat
        Let v be the parent node of u in T;
        Remove u from l_c(v) and put it at the end of l'_c(v);
        u ← v;
    until i ≠ m(u) or u is the root;
```

Due to the existence of conflicting triples, two arbitrary trees T_1 and T_2 with same leaf set L may not be embeddable in a common order of L. If so, we can however show the following:

Proposition 2. *Let T_1, T_2 be trees with leaf set $L = \{1, \dots, n\}$. In time $O(n)$ it is possible to identify a set $\mathcal{C}$ of disjoint conflicts between T_1 and T_2 s.t. $T_2|\big(L - L(\mathcal{C})\big)$ is embeddable in $\pi_1|\big(L - L(\mathcal{C})\big)$.*

Below is given a sketch of the proof for this proposition. Let u be a node in a tree T with leaf set L and π be an arbitrary order on L. If an element $x \in L - L(u)$ is s.t. $m(u) <_\pi x <_\pi M(u)$, then $prev_\pi(x,u)$, resp. $next_\pi(x,u)$, stands for the maximum, resp. minimum, element of $L(u)$ w.r.t. π that is smaller, resp. larger, than x.

Lemma 7. *Let T_1, T_2 be trees on a leaf set $L \subseteq \{1,\ldots,n\}$ and let $\{a,b,c\} \subseteq L$. If both $a <_{\pi_1} b <_{\pi_1} c$ and $ac \mid b \in rt(T_2)$, then $\{a,b,c\}$ is a conflict between T_1 and T_2. In particular, for a node u of T_2 and a leaf $x \notin L(u)$ s.t. $m(u) <_{\pi_1} x <_{\pi_1} M(u)$ then $\{prev_{\pi_1}(x,u), x, next_{\pi_1}(x,u)\}$ is a conflict between T_1 and T_2.*

This lemma guides the search of T_2 to remove leaves (in T_2 and T_1) forming a set of disjoint conflicts $\mathcal{C}$ s.t. for any node u of $T_2|(L - L(\mathcal{C}))$, $L(u)$ is an interval of leaves in $\pi_1|(L - L(\mathcal{C}))$. Such a node u is then said to be *full*. When all nodes of the resulting T_2 are full, Lemma 6 ensures that T_2 is embeddable in the left-right order of the tree $T_1|(L - L(\mathcal{C}))$.

Nodes of T_2 are processed in post-order, such that the children of a node u are known to be full when u is processed. For efficiency reasons, a list L_I of disjoint intervals of π_1 is also maintained sorted w.r.t. to π_1. L_I is initially composed of unit intervals $(\{1\},\ldots,\{n\})$ corresponding to leaves of T_2. Then intervals of L_I are merged or removed while processing nodes of T_2 so as to maintain the following invariant:

Invariant 1. *Any interval of the list L_I contains the leaf set $L(u)$ of some node u of T_2 that is full w.r.t. $\pi_1|\big(L - L(\mathcal{C})\big)$.*

When a non-full node u is processed in the traversal of T_2, this invariant together with pointers from each children of u to the corresponding elements ordered in L_I enables us (according to Lemma 7) to efficiently identify conflicts whose removal turns u into a full node. Note that Invariant 1 is robust under the removal of a leaf in $L(v)$ for any processed node v.

Lemma 8. *Let T_1, T_2 be two trees with leaf set in L and u be the current node of T_2 to be processed by the bottom-up algorithm (*i.e. *the children of u are full w.r.t. π_1). Then a set $hs(u)$ of disjoint conflicts between $\{T_1, T_2\}$ s.t. u is full w.r.t. $\pi_1|\big(L - L(hs(u))\big)$ is found in time $O(d^+(u) + |hs(u)|)$.*

Proposition 2 follows from Lemma 7, Invariant 1 and Lemma 8. Given two arbitrary trees T_1, T_2, propositions 1 and 2 show that, in linear time, disjoint conflicts can be removed and children of nodes in T_2 ordered s.t. the two resulting trees have the same left-right order on their leaves. Thus, from now on, assume that $\pi_1 = \pi_2$. For convenience, even if some leaves have been removed, we note $\pi_1 = 1 \ldots n$. Even if T_1 and T_2 have the same left-right order on their leaves, they may still host conflicting triples. However, let us show that a post-order search of T_1 (or equiv. T_2) is sufficient to remove such conflicts.

Definition 3. *Let u be a node in a tree T, then $rt(u)$ is the subset of triples $x|yz \subseteq rt(T)$ s.t. $\#\big(\{x,y,z\} \cap L(u)\big) \geq 2$, and $f(u)$ is the set of fans $\{x,y,z\} \subseteq f(T)$ s.t. $\{x,y,z\} \subseteq L(u)$. Define a node u in tree T_1 to be* valid *w.r.t. tree T_2 if both $rt(u) \subseteq rt(T_2)$ and $f(u) \subseteq f(T_2)$ hold.*

Note that if r_1 is the root node of tree T_1, then $rt(r_1) = rt(T_1)$ and $f(r_1) = f(T_1)$. Moreover, given a tree T_2 s.t. $L(T_2) = L(T_1)$, the validity of r_1 w.r.t. T_2 implies that T_1 and T_2 are isomorphic, as any 3-leaf set is either a rooted triple or a fan of both trees. Next lemma is the basis of a recursive process to obtain the validity of r_1 w.r.t. T_2.

Lemma 9. *Let u be a node of T_1 whose children, denoted $c_1, \ldots, c_{d^+(u)}$, are all valid. Let $p(m(u))$, resp. $s(M(u))$, be the leaf preceding $m(u)$, resp. succeeding $M(u)$, in π_2 if it exists.*

1. *if $\{p(m(u))|m(u)M(u), s(M(u))|m(u)M(u)\} \subseteq rt(T_2)$ then $rt(u) \subseteq rt(T_2)$*
2. *if u has only two children then $f(u) \subseteq f(T_2)$*
3. *if u has at least three children and for any $i \in \{1, 2, \ldots, d^+(u) - 2\}$, $\{m(c_i), m(c_{i+1}), m(c_{i+2})\} \in f(T_2)$, then $f(u) \subseteq f(T_2)$.*

Lemma 9 implies that if every node $u \in T_1$ is processed after its children, examining only $O(d^+(u))$ 3-leaf sets is enough to know whether a node $u \in T_1$ is already valid. When a conflict is encountered during this examination, its leaves are removed from the trees.

Indeed, thanks to Lemma 1, removing a leaf in $S(u)$ does not change the pre-established validity of inner nodes of $S(u)$. Thus, if $c(u)$ denotes the number of such encountered conflicts, ensuring the validity of u involves looking at $O(d^+(u) + c(u))$ 3-leaf sets. See Algorithm 2 for a complete description of the procedure. Note that persistent dummy leaves can be artificially added at the beginning and end of π_1 and π_2 s.t. $p(m(u))$ and $s(M(u))$ always exist for any processed node u. Processing the whole tree T_1 globally involves $O(n)$ 3-leaf sets as $\sum_{u \in T_1} c(u) = O(n)$ and $\sum_{u \in T_1} d^+(u) = O(n)$.

Provided π_1 is stored in a doubly-linked list; symmetric pointers are maintained between a node $u \in T_1$ to be processed, and the two elements of π_1 that are the leftmost and rightmost leaves of $S(u)$; and T_2 is preprocessed so as to identify in $O(1)$ the least common ancestor of any two of its nodes; then Algorithm 2 runs in linear time. Hence,

Theorem 2. *The* CMAST *problem on a collection of k rooted trees with same n-leaf set can be 3-approximated in $O(kn)$ time.*

The reader should notice that the above algorithms can be realized simultaneously by a single search of the tree. According to Proposition 1, Proposition 2 and Algorithm 2, the case $k = 2$ is solved in $O(n)$ time. Handling a collection $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of $k > 2$ trees is done as for the MCT problem (see Section 3.1), i.e. by successively considering pairs of trees in $\mathcal{T}$. This procedure runs in $O(nk)$ and, from Lemma 4, provides a 3-approximation of CMAST for $\mathcal{T}$.

4 Inapproximability Results for MAST and MCT

In this section, we first state that the rooted and unrooted versions of CMAST (equiv. CMCT) have the same approximation threshold. Then we detail new negative results concerning the approximation of MCT, CMAST and CMCT.

Algorithm 2: AGREEMENTSUBTREE (T_1, T_2)

Input: Two rooted trees s.t. $\pi_1 = \pi_2$
for *each node* u *in a post order traversal of* T_1 **do**
  /* *Ensures that* $rt(u) \subseteq rt(T_2)$ */
  repeat
    $m(u) \leftarrow$ leftmost leaf of $S(u)$; $M(u) \leftarrow$ rightmost leaf of $S(u)$
    $p(m(u)) \leftarrow$ leaf preceding $m(u)$ in π_1 ; $f(M(u)) \leftarrow$ leaf following $M(u)$ in π_1
    if $p(m(u))|m(u)M(u) \notin rt(T_2)$ **then** remove $p(m(u))$, $m(u)$, $M(u)$ from T_1 and T_2
    else if $f(M(u))|m(u)M(u) \notin rt(T_2)$ **then** remove $f(M(u))$, $m(u)$, $M(u)$ from T_1 and T_2
  until $\{p(m(u))|m(u)M(u), f(M(u))|m(u)M(u)\} \subseteq rt(T_2)$ *or* $d^+(u) < 2$
  /* *Ensures that* $f(u) \subseteq f(T_2)$ */
  $i \leftarrow 1$
  while $d^+(u) > 2$ *and* $i \leq d^+(u) - 2$ **do**
    let $c_1, c_2, \ldots, c_{d^+(u)}$ be the children of u
    if $\{m(c_i), m(c_{i+1}), m(c_{i+2})\} \in f(T_2)$ **then** $i \leftarrow i + 1$
    else remove $m(c_i)$, $m(c_{i+1})$, $m(c_{i+2})$ from T_1 and T_2
return T_1

4.1 Rooted and Unrooted Versions of CMAST (equiv. CMCT) Share the Same Approximation Threshold

Let $\varphi(n, k)$ be a function in $\Omega(n \times k)$.

Proposition 3. *Let* $\rho \geq 1$ *be a real constant. Assume there exists a* ρ*-approximation algorithm for* CMAST*, resp.* CMCT*, on rooted trees with* $O(\varphi(n, k))$ *running time. Then, there exists a* ρ*-approximation algorithm for* CMAST*, resp.* CMCT*, on unrooted trees with* $O(n \times \varphi(n - 1, k))$ *running time.*

Proposition 3 is implicitely used in [11] and is proved in the following way. Let $\mathcal{U}$ be a collection of unrooted trees. To ρ-approximate CMAST, resp. CMCT, on instance $\mathcal{U}$, apply the hypothetical ρ-approximation algorithm to each collection obtained by rooting all trees in $\mathcal{U}$ at a same leaf. Then, return the best of the n computed solutions. Combining Theorem 2 and Proposition 3, resp. Theorem 1 and Proposition 3, we obtain that the unrooted version of CMAST, resp. CMCT, is 3-approximable in $O(kn^2)$, resp. $O(n^3 + kn^2)$, time. Using a simple padding argument yields the converse of Proposition 3:

Proposition 4. *Let* $\rho \geq 1$ *be a rational constant. Assume there exists a* ρ*-approximation algorithm for* CMAST*, resp.* CMCT*, on unrooted trees with* $O(\varphi(n, k))$ *running time. Then, there exists a* ρ*-approximation algorithm for* CMAST*, resp.* CMCT*, on rooted trees with* $O(\varphi(n + \lceil \rho n \rceil, k))$ *running time.*

4.2 Hardness of Approximating CMAST on Three Trees

Theorem 3. *The* 3-CMAST *problem is* APX*-hard.*

Since 2-MAST (and thus, 2-CMAST) can be exactly solved in polynomial time [21], Theorem 3 is somehow tight. Its proof relies on a careful reading of [17] which states that the *general* 3-MAST problem is APX-hard. In fact [17] proves that a *restriction* of 3-MAST to a certain set of instances is APX-hard. CMAST is not considered in [17], but it is easy to see that for this particular set of instances, 3-MAST L-reduces to 3-CMAST

4.3 Hardness of Approximating MCT and CMCT on Two Trees

In order to prove Theorems 5 (APX-hardness of 2-CMCT) and 6 (inapproximability of 2-CMCT), we define an intermediate problem, called MAXIMUM STAR-FOREST (MSF). Let $G = (V, E)$ be a graph. A *star-forest* of G is a subset of E which does not contain any path of length 3. The MSF problem is: *"given a graph G, find a star-forest of G that is of maximum cardinality"* For each integer $\Delta \geq 1$, we denote by Δ-MSFB the restriction of MSF to *bipartite* input graphs having *maximum degree* at most Δ. The restriction of the MAXIMUM INDEPENDENT SET (shortly MIS) to input graphs having maximum degree at most 3 is denoted 3-MIS. Note that 3-MIS is APX-complete [1].

Theorem 4. *The* 4-MSFB *problem is* APX*-hard.*

Proof (sketch). We use an L-reduction from 3-MIS to 4-MSFB relying on the following transformation. Let $G = (V, E)$ be an instance of 3-MIS (*i.e.* a graph with maximum degree at most 3), we construct an instance $G' = (V', E')$ of 4-MSFB as follows.

$$V' := V \cup \{\gamma_e : e \in E\} \cup \{\sigma_v, \tau_v : v \in V\},$$
$$E' := \big\{\{u, \gamma_e\}, \{\gamma_e, v\} : e = \{u, v\} \in E\big\} \cup \big\{\{v, \sigma_v\}, \{\sigma_v, \tau_v\}, : v \in V\big\}.$$

Clearly, G' can be obtained from G in polynomial time, and $\#V' = m + 3n$ and $\#E' = 2m + 2n$, where n and m denote the cardinality of V and E resp.

□

Theorem 4 leads to the following result:

Proposition 5. 2-MCT *is* APX*-hard even if it is restricted to collections $\mathcal{T}$ of two rooted trees satisfying $\#MCT(\mathcal{T}) \geq \frac{1}{4} \times n$, where n denotes the size of each tree in $\mathcal{T}$.*

Proof (sketch). We use an L-reduction from 4-MSFB to 2-MCT relying on the following transformation. Let $G = (V, E)$ be an instance of 4-MSFB. Since G is bipartite there exists two independent sets I_1 and I_2 of G partitioning V. W.l.o.g., we can assume that G has no isolated vertex. We construct a collection $\mathcal{T} = \{T_1, T_2\}$ of two rooted trees with leaf set E. The root of T_i is denoted r_i. For each $v \in I_i$, let X_v be the non-empty *star-tree* whose leaf set is the set of all edges of E admitting v as an extremity (a *star-tree*, is a fan with an arbitrary number of leaves). The child subtrees of r_i, are trees X_v with $v \in I_i$.

The transformation requires polynomial time and the size of the instance of 2-MCT in linear in the size of the instance of 4-MSFB. The correctness of the reduction follows by proving that for each subset $F \subseteq E$, F is a star-forest of G iff $T_1|\,F$ and $T_2|\,F$ are compatible. □

Proposition 5 yields the two main results of this section. On the first hand, we obtain:

Theorem 5. *The* 2-CMCT *problem is* APX*-hard.*

On the other hand, using the "self-improvement" technique of [17, 19] we deduce from Proposition 5 that 2-MCT is hard to approximate within constant ratio:

Theorem 6. *For any real constant* $\delta < 1$*, the* 2-MCT *problem cannot be approximated within ratio* $2^{\log^{\delta} n}$*, unless* $\mathrm{NP} \subseteq \mathrm{DTIME}\big(2^{\mathrm{polylog}\, n}\big)$.

The analoguous to Theorem 6 for 3-MAST is [17, Theorem 3].

4.4 Hardness of Approximating MCT on Unbounded Number of Trees

For the general MCT problem we can find non-approximability results stronger than Theorem 6. Approximating MCT on collections of n-leaf trees is at least as hard as approximating MIS on n-vertex graphs. The proof consists in an approximation preserving reduction from MIS to MCT, similar to the reduction from MIS to MAST described in [5]. Since MIS is very hard to approximate [16] (see also [9]), we obtain:

Theorem 7. *For all real* $\epsilon > 0$, MCT *is not approximable within ratio* $(1 + \epsilon)n^{1-\epsilon}$ *unless* $\mathrm{NP} = \mathrm{ZPP}$*, resp. within ratio* $(1 + \epsilon)n^{0.5-\epsilon}$ *unless* $\mathrm{P} = \mathrm{NP}$.

Note that Theorem 7 still holds if MCT is restricted to collections of trees containing at least a binary tree. Remark that using the *approximating via partitioning* paradigm [14], one can approximate MAST within $n/\log n$ [18]. This also holds for MCT.

References

1. P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.*, 237(1–2):123–134, 2000.
2. A. Amir and D. Keselman. Maximum agreement subtree in a set of evolutionary trees: metrics and efficient algorithm. *SIAM J. on Comput.*, 26(6):1656–1669, 1997.
3. V. Berry and F. Nicolas. Maximum agreement and compatible supertrees. In *15th Annual Symposium on Combinatorial Pattern Matching (CPM'04)*, volume 3109 of *LNCS*, pages 205–219, 2004.
4. V. Berry and F. Nicolas. Improved parametrized complexity of maximum agreement subtree and maximum compatible tree problems. *IEEE Trans. on Comput. Biology and Bioinf.*, (to appear).

5. P. Bonizzoni, G. Della Vedova, and G. Mauri. Approximating the maximum isomorphic agreement subtree is hard. *Int. J. of Found. of Comput. Sci.*, 11(4):579–590, 2000.
6. D. Bryant. *Building trees, hunting for trees and comparing trees: theory and method in phylogenetic analysis.* PhD thesis, University of Canterbury, Department of Mathemathics, 1997.
7. R. Cole, M. Farach-Colton, R. Hariharan, T. M. Przytycka, and M. Thorup. An $O(n \log n)$ algorithm for the Maximum Agreement SubTree problem for binary trees. *SIAM J. on Comput.*, 30(5):1385–1404, 2001.
8. G. F. Eastabrook and F. R. McMorris. When is one estimate of evolutionary relationships a refinement of another? *J. of Math. Biol.*, 10:367–373, 1980.
9. L. Engebretsen and J. Holmerin. Towards optimal lower bounds for clique and chromatic number. *Theor. Comput. Sci.*, 299(1–3):537–584, 2003.
10. M. Farach, T. M. Przytycka, and M. Thorup. On the agreement of many trees. *Inf. Proces. Letters*, 55(6):297–301, 1995.
11. G. Ganapathy and T. J. Warnow. Approximating the complement of the maximum compatible subset of leaves of k trees. In *5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'02)*, volume 2462 of *LNCS*, pages 122–134, 2002.
12. G. Ganapathysaravanabavan and T. J. Warnow. Finding a maximum compatible tree for a bounded number of trees with bounded degree is solvable in polynomial time. In *1st Int. Workshop on Algorithms in Bioinformatics (WABI'01)*, volume 2149 of *LNCS*, pages 156–163, 2001.
13. A. Gupta and N. Nishimura. Finding largest subtrees and smallest supertrees. *Algorithmica*, 21(2):183–210, 1998.
14. M. M. Halldòrsson. Approximations of weighted independent set and hereditary subset problems. *J. of Graph Algor. and Appl.*, 4(1), 2000.
15. A. M. Hamel and M. A. Steel. Finding a maximum compatible tree is NP-hard for sequences and trees. *Appl. Math. Letters*, 9(2):55–59, 1996.
16. J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182:105–142, 1999.
17. J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Disc. Appl. Math.*, 71(1–3):153–169, 1996.
18. J. Jansson, J. H.-K. Ng, K. Sadakane, and W.-K. Sung. Rooted maximum agreement supertrees. In *6th Latin American Symposium on Theoretical Informatics (LATIN'04)*, volume 2976 of *LNCS*, pages 499–508, 2004.
19. T. Jiang and M. Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM J. on Comput.*, 24(5):1122–1139, 1995.
20. M.-Y. Kao, T. W. Lam, W.-K. Sung, and H.-F. Ting. A decomposition theorem for maximum weight bipartite matchings with applications to evolutionary trees. In *7th Annual European Symposium on Algorithms (ESA'99)*, volume 1643 of *LNCS*, pages 438–449, 1999.
21. 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. *J. of Algor.*, 40(2):212–233, 2001.
22. M. A. Steel and T. J. Warnow. Kaikoura tree theorems: Computing the maximum agreement subtree. *Inf. Proces. Letters*, 48(2):77–82, 1993.