On the complexity of computing evolutionary trees

Leszek Gąsieniec *
Max-Planck Institut für Informatik

Jesper Jansson†
Lund University

Andrzej Lingas †
Lund University

Anna Östlin †
Lund University

Abstract

In this paper we study a few important tree optimization problems with applications to computational biology. These problems ask for trees that are consistent with an as large part of the given data as possible.

We show that the maximum homeomorphic agreement subtree problem cannot be approximated within a factor of N^ϵ, where N is the input size, for any $0 \leq \epsilon < \frac{1}{18}$ in polynomial time, unless P=NP. On the other hand, we present an $O(N \log N)$-time heuristic for the restriction of this problem to instances with $O(1)$ trees of height $O(1)$, yielding solutions within a constant factor of the optimum.

We prove that the maximum inferred consensus tree problem is NP-complete and we provide a simple fast heuristic for it, yielding solutions within one third of the optimum. We also present a more specialized polynomial-time heuristic for the maximum inferred local consensus tree problem.

1 Introduction

An evolutionary tree models how different species in a given set have evolved. The leaves in an evolutionary tree correspond to species and internal nodes represent the species' ancestors.

The problem of finding an evolutionary tree has been studied a lot recently [3, 4, 5, 8, 12, 14, 15, 17, 18]. There are many different approaches, depending on among other things what kind of data that is available. Therefore, various versions of this problem arise in, for example, computational biology when one wants to find out how different species are related, and comparative linguistics, where it is central to find out how different languages have evolved. In this paper, we look at some of these problems.

*Max-Planck Institut für Informatik, Im Stadtwald, D–66123, Saarbrücken, Germany, email:leszek@mpi-sb.mpg.de

†Department of Computer Science, Lund University, Box 118, S-221 00 Lund, Sweden, email: {Jesper.Jansson, Andrzej.Lingas, Anna.Ostlin}@dna.lth.se

Given a set of evolutionary trees dealing with a fixed set of species, one might want to identify a subtree contained within every given tree such that the number of leaves labeled by species is maximized. This problem is known as *the maximum homeomorphic agreement subtree problem* (MHT) [14]. More formally, it is defined as follows. Given k rooted trees T_1, $T_2, ...,$ T_k, each with n leaves labeled distinctly with elements chosen from a set A of cardinality n, find a maximum cardinality subset B of A such that the minimal homeomorphic subtrees of T_1, $T_2, ...,$ T_k (i.e., with all degree 2 nodes except for the root contracted) containing exactly the leaves labeled by B are isomorphic. To measure the input size of an instance of MHT, we let N denote the total number of nodes contained in the trees. MHT restricted to instances with two trees is frequently called MAST; algorithms for MAST have been developed since 1985 [6]. MAST has been shown to be solvable in polynomial time, both for rooted trees [4, 5] and for UMAST, a variant of MAST with unrooted trees [15, 18]. In practice, however, the number of trees is often much larger than two [14]. For the special case of MHT in which at least one of the given trees has bounded degree, there exist polynomial-time algorithms [3, 14]. In contrast, MHT is known to be NP-complete even for instances with three trees of unbounded degree [14]. Here we take one step further to prove that, unless P=NP, MHT cannot be approximated within a factor of N^{ϵ}, where N is the input size, for any $0 \leq \epsilon < \frac{1}{18}$ in polynomial time; see Section 2. Our result on the non-approximability of MHT holds even for instances containing solely trees of height 2. On the other hand, in Section 3, we show that MHT for instances with $O(1)$ number of trees of height $O(1)$ can be approximated within a constant factor in time $O(N \log N)$. Similar results also hold for the unrooted version of MHT which is at least as hard as MHT (this can be seen by an argument analogous to that in [4] for MAST and UMAST).

Usually MHT instances do not admit a solution containing all the members of the species set. Therefore, in some applications, other methods may be preferred. One alternative approach is to attempt to construct an evolutionary tree from a set of constraints that relate the species to each other. Already during the early eighties, Aho *et al.* [1] studied the problem of inferring a tree from constraints on its lowest common ancestors in the context of data bases and biological applications. They defined it as follows: Given a set of constraints of the form $\{i, j\} < \{k, l\}$ where $\{i, j, k, l\} \subset \{1, 2, ..., n\}$, if possible construct a tree on the set of leaves $\{1, 2, ..., n\}$ such that for each constraint of the aforementioned form, the lowest common ancestor of i and j is a proper descendant of the lowest common ancestor of k and l. Aho *et al.* showed how to decide whether an instance of this problem admits a solution, and if so, how to construct it, both in time $O(mn \log n)$, where m denotes the number of constraints.

Recently, many authors have studied the related problem of constructing the so called consensus tree or local consensus tree [8, 12, 17]. For a set of binary rooted trees $\{T_1, ..., T_k\}$, each one leaf-labeled by a subset $L(T_i)$ of $\{1, 2, ..., n\}$, the consensus tree problem asks whether or not there is a tree T such that for $i = 1, ..., k$, T_i is homeomorphic to the subtree of T induced by the nodes in $L(T_i)$ and their ancestors. If the input trees are of constant size it is termed the local consensus tree problem. A constraint of the form $\{i, j\} < \{i, k\}$ (

denoted $(\{i,j\},k)$ for short) is easily seen to be equivalent to the constraint imposed by a full binary tree on the leaves $i,\ j,\ k$ in the local consensus tree problem. For this reason, we shall call the tree inferring problem posed in [1] *the inferred consensus tree problem*.

Unfortunately, it is often impossible to construct an exact consensus tree. This makes a need to deal with an optimization version of the inferred consensus tree problem whose objective is to find a consensus tree for an as large as possible subset of the input set of constraints of the form $\{i,j\} < \{k,l\}$. For brevity, we term this optimization problem *the maximum inferred consensus tree problem* (MICT for short). We also distinguish the restricted case of MICT where all constraints are of the form $(\{i,j\},k)$ and call it *the maximum inferred local consensus tree problem* (MILCT).

In Section 4 we provide an NP-completeness proof for MICT. Section 5 contains a simple $O((n+m)\log n)$-time heuristic for MICT yielding solutions within one third of the optimum and a more involved polynomial-time heuristic for MILCT. Both heuristics work equally well for the weighted versions of MICT and/or MILCT where the objective is to find a consensus tree for a subset of the input constraints of maximum total weight.

2 MHT is hard to approximate

Our main result in this section is as follows.

Theorem 2.1 *For any $0 \le \epsilon < \frac{1}{18}$, MHT (even if restricted to trees of height 2) cannot be approximated within a factor of N^ϵ in polynomial time, unless P=NP.*

Proof: We show that if MHT can be approximated within a factor of N^ϵ in polynomial time then the problem of finding a maximum independent set in a graph with l nodes can be approximated within a factor of $l^{3\epsilon+o(1)}$. In part, our reduction can be seen as a generalization of the reduction of three dimensional perfect matching to MHT restricted to instances with three trees given in [14].

Let $G=(V,E)$ be a graph where $V=\{v_1,\ v_2,...,v_l\}$ and $E=\{e_1,\ e_2,...,e_k\}$. For $1 \le i \le k$ and $1 \le j \le l$, let $e(i,j)=e_i$ if v_j is incident to e_i, otherwise let $e(i,j)=(i,j)$. Furthermore, for $1 \le i \le k$, let $E_i = \bigcup_{1\le j\le l}\{e(i,j)\}$, and for $1 \le j \le l$, let $V_j = \bigcup_{1\le i\le k}\{e(i,j)\}$.

Remark 1 For distinct $1 \le j,\ j' \le l$, $V_j \setminus V_{j'} \ne \emptyset$.

Remark 2 Two nodes $v_j,\ v_{j'}$ are adjacent in G iff $V_j \cap V_{j'} \ne \emptyset$.

Remark 3 For $1 \le i \le k$, $1 \le j,j' \le l$, $j \ne j'$ at most two sets V_j, $V_{j'}$ contain e_i.

Now, generalizing the reduction used in [14], we construct the trees $T_1,\ T_2,...,T_k$ as follows. Each tree T_i has root r_i which is a parent of $l+q$ children. The first l children are in one-to-one correspondence to the elements of E_i. The remaining q are leaves labeled by $z_1,\ z_2,...,z_q$. Next, for each $1 \le j \le l$, we attach a child labeled V_j to the child of r_i corresponding to $e(i,j)$.

Suppose that the roots $r_1,\ r_2,...,r_k$ correspond to the root of a maximum agreement subtree T of $T_1,\ T_2,...,T_k$. Then, no two overlapping sets $V_j,\ V_{j'}$ can simultaneously label leaves in T by the construction of T_1 through T_k and Remark 1. Hence, by Remark 2, the maximum independent set in G has cardinality m iff T has $m + q$ leaves.

In order to force the roots $r_1,\ r_2, ..., r_k$ to correspond to the root of T we set q to 2. To see that this is sufficient, assume that one of the non-leaf children of r_i turned out to be the root, for some i. By Remark 3, each non-leaf child of r_i has at most two leaf children in T_i, so the size of this tree can be no larger than two. But we can always find an agreement tree of size 3 by selecting r_i as root and including $z_1,\ z_2$ in addition to one leaf child.

The total size N of the trees $T_1,\ T_2, ...,\ T_k$ is $k \cdot O(l) = O(l^3) = l^{3+o(1)}$. Clearly, they can be constructed in polynomial time from G. Also, note that they are of height 2. Below we will only consider approximations that can be carried out in polynomial time. If MHT could be approximated within a factor of N^ϵ, then $\frac{OPT+2}{s+2} \leq N^\epsilon$, where $OPT + 2$ refers to the number of leaves in an optimal solution for a given instance of MHT and $s + 2$ is the number of leaves in its corresponding, approximative solution. For $s \geq 1$, it follows that $\frac{OPT}{s} \leq 3 \cdot \frac{OPT+2}{s+2} \leq 3N^\epsilon = l^{3\epsilon+o(1)}$, which would imply that the problem of finding a maximum independent set in a graph could be approximated within a factor of $l^{3\epsilon+o(1)}$. However, Bellare and Sudan proved in [2] that this problem isn't approximable within $l^{1/6-\delta}$ for any $\delta > 0$, unless P=NP. Hence, if P$\neq$NP, MHT cannot be approximated within a factor of N^ϵ for any $0 \leq \epsilon \leq \frac{1}{18} - o(1)$. Finally, since $\frac{1}{18} - o(1)$ can be made arbitrarily close to $\frac{1}{18}$ by choosing N large enough, there exist instances of MHT which cannot be approximated within a factor of N^ϵ for any constant $0 \leq \epsilon < \frac{1}{18}$ in polynomial time (unless P=NP).

$\square$

3 Approximations of MHT with $O(1)$ trees of $O(1)$ height

We know that MHT is NP-complete already for instances with three trees [16] and hard to approximate for instances with an arbitrary number of trees of height 2 by Theorem 2.1. The natural question arises whether or not MHT for instances with bounded number of trees can be tightly approximated in polynomial time. Here we partially settle this question by proving the approximability of MHT for instances with $O(1)$ trees of height $O(1)$. This result together with Theorem 2.1 yield a complete characterization of approximability of MHT restricted to instances with trees of $O(1)$ height.

Theorem 3.1 *MHT restricted to instances with k trees of height not exceeding h can be approximated within a factor of $(\frac{1}{k})^h$ in time $O(n \log n)$.*

To begin the proof of Theorem 3.1, we need to introduce the following notation. For a tree T, $V(T)$ stands for the set of nodes of T. Let v be a node of a rooted tree T. The minimal subtree of T rooted at v, including v and all its descendants is denoted by T_v. $L(T_v)$ stands for the set of labels of the leaves in T_v. The set of children of v in T is denoted by $C(v)$. Furthermore, by a *k-partite hypergraph* we shall mean a pair $(V_1 \cup ... \cup V_k,\ E)$ where V_1

through V_k are pairwise disjoint sets and E is a subset of $V_1 \times V_k \times ... \times V_k$. The elements of $V_1 \cup ... \cup V_k$ are called the *nodes* of H whereas the elements of E are called the *edges* of H. A *matching* of H is a subset of E in which no pair of edges includes a common node.

Let T_1, ..., T_k be the input trees. For $(v_1, v_2, ..., v_k)$, where $v_i \in V(T_i)$ for $i = 1, ..., k$, let $Mht(v_1, v_2, ..., v_k)$ denote the maximum size of an agreement subtree of the trees $T_1, ..., T_k$ restricted to $B = L((T_1)_{v_1}) \cap L((T_2)_{v_2}) \cap ... \cap L((T_k)_{v_k})$. We can view $Mht(v_1, v_2, ..., v_k)$ as the solution of MHT for $(T_1)_{v_1}, ..., (T_k)_{v_k}$. Next, let $H(v_1, ..., v_k)$ denote the k-partite hypergraph $(C(v_1) \cup .... \cup C(v_k), C(v_1) \times C(v_2) \times ... \times C(v_k))$ whose edges $(w_1, ..., w_k)$ have weight $Mht(w_1, ..., w_k)$. Finally, let $Match(v_1, ..., v_k)$ be the maximum weight of a matching in $H(v_1, ..., v_k)$ and $Diag(v_1, ..., v_k) = \max\{Mht(w_1, ..., w_k) | (w_1, ..., w_k) \in (\{v_1\} \cup C(v_1)) \times ... \times (\{v_k\} \cup C(v_k)) - \{(v_1, ..., v_k)\}$.

Intuitively, in the final agreement subtree of $(T_1)_{v_1}, ..., (T_k)_{v_k}$ either the roots of the trees, i.e., v_1 through v_k, are matched together which forces their children to be optimally matched together (*Match*), or only some of the roots are matched together with some children of the remaining roots (*Diag*). This yields the following lemma which is a straightforward generalization of the basic lemma in the dynamic programming approach to MAST in [18] (see also [4]).

Lemma 3.2 *For any $(v_1, ..., v_k)$, where $v_i \in V(T_i)$ for $i = 1, ..., k$, if at least one of the v_i's is a leaf then:*
– $Mht(v_1, ..., v_k) = |L((T_1)_{v_1}) \cap ... \cap L((T_k)_{v_k})|$ *else*
– $Mht(v_1, ..., v_k) = \max\{Match(v_1, ..., v_k), Diag(v_1, ..., v_k)\}$.

It is easy to see that the recursive computation of $Mht(v_1, ..., v_k)$ for $(v_1, ..., v_k) \in V(T_1) \times ... \times V(T_k)$ suggested by Lemma 3.2 can be bottom-up ordered by $Hs(v_1, ..., v_k) = \sum_{i=1}^{k} height(v_i)$. Hence, we have the following algorithm for MHT.

Algorithm 1

1. <u>input</u> $T_1,\ T_2, ..., T_k$
2. <u>for</u> each $(v_1, ..., v_k) \in V(T_1) \times ... \times V(T_k)$ in the increasing order of $Hs(v_1, ..., v_k)$ <u>do</u> compute $Mht(v_1, ..., v_k)$ by using the expression of Lemma 3.2.
3. <u>output</u> $Mht(r_1, ..., r_k)$ where r_i is the root of T_i for $i = 1, ..., k$.

It is hard to compute the exact value of $Match(v_1, ..., v_k)$ in the expression of Lemma 3.2 since the problem of computing maximum matching in a 3-partite hypergraph is NP-complete [16]. For this reason, we shall rely on a greedy method for approximating $Match(v_1, ..., v_k)$ yielding an approximation of $Mht(v_1, ..., v_k)$. The greedy method consists in repeatedly picking the heaviest edge e and removing all edges overlapping e. It can be easily implemented using a priority queue. Since e can overlap with at most k edges in an optimum solution having total weight $\leq k \cdot weight(e)$, we obtain:

Lemma 3.3 *Let $H = (V, E)$ be a k-partite hypergraph on m edges with positive integer weights. A matching in H of total weight within $\frac{1}{k}$ of the maximum can be constructed in a greedy fashion in time $O(k|E| + |V| + m \log m)$.*[*]

By combining the scheme of Algorithm 1 with the greedy method for approximating $Match(v_1, ..., v_k)$, we obtain the following lemma yielding Theorem 3.1.

Lemma 3.4 *For all $(v_1, ..., v_k) \in V(T_1) \times ... \times V(T_k)$, we can approximate $Mht(v_1, ..., v_k)$ within a factor of $(\frac{1}{k})^h$ where $h = \max\{height(v_i) | 1 \le i \le k\}$ in time $O(n \log n)$.*

Proof: For $(v_1, ..., v_k) \in V(T_1) \times .... \times V(T_k)$, let $s(v_1, ..., v_k)$ denote the size of the intersections $L((T_1)_{v_1}) \cap ... \cap L((T_k)_{v_k})$. Clearly, we have $Mht(v_1, ..., v_k) \le s(v_1, ..., v_k)$, and in particular if one of the v_i's is a leaf then $Mht(v_1, ..., v_k) = s(v_1, ..., v_k)$. For a leaf label j, we can determine all k-tuples $(v_1, ..., v_k)$ for which $j \in L((T_1)_{v_1}) \cap ... \cap L((T_k)_{v_k})$ by finding, in each T_i, $i = 1, ..., k$, the nodes on the path of length $\le h$ from the leaf labeled by j to the root. It follows that the number of these tuples is $(h+1)^k$. Consequently, the set L of all k-tuples for which $s(v_1, ..., v_k) > 0$ has size not exceeding $n(h+1)^k$. To list L efficiently, we sort the pointers to leaves in the trees T_1 through T_k by the leaf labels. Such a sorted list of pointers can be produced in time $O(|V(T_1)| + ... + |V(T_k)|)$. Using it, we can easily generate L by finding appropriate tree paths in time $O(|V(T_1)| + ... + |V(T_k)| + (h+1)^k)$.

For the k-tuples $(v_1, ..., v_k)$ in L including at least one leaf we have clearly $s(v_1, ..., v_k) = 1$ and $Mht(v_1, ..., v_k) = 1$. To compute approximations of $Mht(v_1, ..., v_k)$ for the remaining k-tuples in L, we build a balanced search tree S_L for L, with respect to the lexicographic order of k-tuples in $V(T_1) \times .... \times V(T_k)$, in time $O(|L| \log |L|)$. Next, we follow the scheme of Algorithm 1 using the greedy method to approximate $Match(v_1, ..., v_k)$ in the hypergraph $H_L(v_1, ..., v_k)$ which is the hypergraph $H(v_1, ..., v_k)$ defined in Lemma 3.2 restricted to edges in L.

Each k-tuple $(w_1, ..., w_k) \in L$ occurs at most once as an edge in every hypergraph $H_L(v_1, ..., v_k)$ for $(v_1, ..., v_k) \in L$ (only when $w_i \in C(v_i)$ for $i = 1, ..., k$). Hence, the hypergraphs $H_L(v_1, ..., v_k)$ for $(w_1, ..., w_k) \in L$, have no more than $|L|$ edges totally and can be constructed (without weights) by scanning L and using S_L in total time $O(|L| \log |L|)$. Clearly, each $H_L(v_1, ..., v_k)$ has at most $s(v_1, ..., v_k)$ edges with positive weights. For each of its edges $(w_1, ..., w_k)$, we have $Hs(w_1, ..., w_k) = Hs(v_1, ..., v_k) - k$ and $\max\{height(w_i) | 1 \le i \le k\} = h - 1$. Hence, we may inductively assume that we have already $(\frac{1}{k})^{h-1}$ approximations of $Mht(w_1, ..., w_k)$, i.e., of the weights of $(w_1, ..., w_k)$ in the hypergraph. Consequently, we obtain an approximation of $Match(v_1, ..., v_k)$ within a factor of $(\frac{1}{k})^h$ by applying the greedy method. By Lemma 3.3, the total cost of the greedy method is $O(k|L| + (\sum_{(v_1,...,v_k) \in L} s(v_1, ..., v_k)) \log n)$ time. By induction on $Hs(v_1, ..., v_k)$, we also obtain an approximation of $Diag(v_1, ..., v_k)$ within a factor of $(\frac{1}{k})^h$ by considering solely k-tuples $(w_1, ..., w_k)$ in $L \cap ((\{v_1\} \cup C(v_1)) \times$

[*]Interestingly, in the unweighted case there are known (much slower, but still) polynomial-time heuristics yielding solutions within almost $\frac{2}{k}$ of the optimum [10].

$\ldots \times (\{v_k\} \cup C(v_k)) - \{(v_1, \ldots, v_k)\})$. Each $(w_1, \ldots, w_k) \in L$ can contribute to the value of $Diag(v_1, \ldots, v_k)$ for at most $2^k - 1$ k-tuples $(v_1, \ldots, v_k) \in L$. Hence, the total size of the subsets of L contributing to $Diag(v_1, \ldots, v_k)$ over all $(v_1, \ldots, v_k) \in L$, and consequently the total cost of finding maxima of Mht-approximations over these subsets, is $O(2^k|L|)$. We can create these subsets, again by scanning L and using S_L, in total time $O(2^k|L| \log |L|)$.

Each of the trees T_1 through T_k has size not exceeding $2n$ by its binarity. Hence, we obtain the $O(n \log n)$ bound by $|L| \leq (h+1)^k n$, $\sum_{(v_1,\ldots,v_k)\in L} s(v_1, \ldots, v_k) \leq (h+1)^k n$ $h = O(1)$, $k = O(1)$, and straightforward calculations. □

4 MICT is NP-complete

The problem of deciding whether or not a 3-partite hypergraph (V, E) has a perfect matching (3PM), i.e., if V is covered by a subset of pairwise disjoint edges in E, is known to be NP-complete [16]. To show the NP-completeness of MICT, we provide a reduction of 3PM to MICT.

Let $H = (V, E)$ be a 3-partite hypergraph and k a parameter that will be specified later on. Let C be the minimal set of constraints satisfying:

1. for each $e,\ f \in E$ with $e \neq f$, $(\{e_i, e_l\}, f_j) \in C$, where $i = 0, 1$, $\quad l = 2, \ldots, k+1$, $j = 2, \ldots, k+1$.
2. for each $e = (a, b, c) \in E$, the three constraints $\{a, b\} < \{e_0, e_1\}$, $\{a, c\} < \{e_0, e_1\}$, and $\{b, c\} < \{e_0, e_1\} \in C$.

Thus, C consists of $2k^2(|E|^2 - |E|)$ constraints of the first type and $3|E|$ constraints of the second type.

To characterize consensus trees for large subsets of C, we need the following definitions.

Definition 4.1 *In a rooted tree T, the lowest common ancestor of a sequence of nodes $v_1, \ldots, v_m$ will be denoted by $lca(v_1, \ldots, v_m)$. Furthermore, the path from a node v to the root of T will be denoted by $R(v)$. The subtree of T induced by a sequence of nodes $v_1, \ldots, v_m$ is the smallest subtree of T including the paths $R(v_i)$, $i = 1, \ldots, m$.*

Definition 4.2 *The full binary tree on four leaves a, b, c, d, where $lca(a, b)$ and $lca(c, d)$ form the intermediate level, will be denoted by $B_4(a, b, c, d)$.*

Lemma 4.3 *If T is a consensus tree for at least $|C| - k^2 + 1$ constraints in C, then for each $e,\ f \in E$ with $e \neq f$, the subtree induced of T by $\{e_0, e_1, f_0, f_1\}$ is homeomorphic to $B_4(e_0, e_1, f_0, f_1)$.*

Proof: By the assumption on the number of constraints satisfied by T, for each $e,\ f \in E$ with $e \neq f$, there are indices $l, j \in \{2, ..., k+1\}$ such that for $i = 0, 1$, the constraints $(\{e_i, e_l\}, f_j)$, $(\{f_i, f_j\}, e_l)$ are satisfied by T.

By $(\{e_0, e_l\}, f_j)$ and $(\{e_1, e_l\}, f_j)$, the path $R(lca(e_0, e_1, e_l))$ cannot be included in the path $R(f_j)$. Thus, $R(lca(e_0, e_1, e_l)) \not\subseteq R(lca(f_0, f_1, f_j))$. Similarly, by $(\{f_0, f_j\}, e_l)$ and $(\{f_1, f_j\}, e_l)$, we have $R(lca(f_0, f_1, f_j)) \not\subseteq R(lca(e_0, e_1, e_l))$. This means that the paths from $lca(e_0, e_1, e_l)$ and $lca(f_0, f_1, f_j)$ to $lca(e_0, e_1, e_l, f_0, f_1, f_j)$, respectively, must be edge-disjoint. $\square$

Corollary 4.4 *Let T be a consensus tree for at least $|C| - k^2 + 1$ constraints in C. For each node $a \in V$ and two different edges $e,\ f$ in E, if T satisfies a constraint of the form $\{a,\ \} < \{e_0, e_1\}$ then T cannot satisfy any constraints of the form $\{a,\ \} < \{f_0, f_1\}$.*

Lemma 4.5 *Let $k > \sqrt{3|E| - |V|}$. The hypergraph H has a perfect matching iff there is a consensus tree for a subset of $2k^2(|E|^2 - |E|) + |V|$ constraints in C.*

Proof: Suppose first that H has a perfect matching M. We can easily construct a consensus tree T satisfying at least $2k^2(|E|^2 - |E|) + |V|$ of the constraints in C. The root of T has $|E|$ children which are in one-to-one correspondence with the edges in E. For every $e \in E$, a subtree rooted in the corresponding child has as children the leaves $e_0, e_1, ..., e_{k+1}$. Furthermore, if $e = \{a, b, c\}$ is in M, then the subtree has another child which in turn is the parent of the leaves labeled $a,\ b,\ c$.

Suppose in turn that there is a consensus tree T satisfying $2k^2(|E|^2 - |E|) + |V|$ constraints in C. The total number of constraints in C is $2k^2(|E|^2 - |E|) + 3|E|$. It follows by $k > \sqrt{3|E| - |V|}$ that T satisfies at least $|C| - k^2 + 1$ constraints. Thus, by Corollary 4.4, for each node $a \in V$, there is at most one edge $e \in E$ such that some constraint of the form $\{a,\ \} < \{e_0, e_1\}$ is satisfied by T. On the other hand, for a given node a and a given edge e, at most two constraints of the form $\{a,\ \} < \{e_0, e_1\}$ can be satisfied by T by the construction of C. Consequently, V can be partitioned into three disjoint subsets V_r, $r = 0, 1, 2$, respectively consisting of nodes $a \in V$ for which r constraints of the form $\{a,\ \} < \{\ ,\ \}$ are satisfied by T. It is easily seen that T satisfies at most $|V_2| + \frac{|V_1|}{2}$ constraints of the form $\{\ ,\ \} < \{\ ,\ \}$. Since there are $2k^2(|E|^2 - |E|)$ constraints of the form $(\{\ ,\ \},\ )$, we conclude that V has to be as large as possible, i.e., $V_2 = V$. It follows that for each edge $e \in E$, if a constraint of the form $\{\ ,\ \} < \{e_0, e_1\}$ is satisfied by T, then all the three constraints of this form are satisfied by T. Hence, H has a perfect matching. $\square$

The construction of C for k equal to $|E|$ can be done in polynomial time. Hence, MICT is NP-hard by the NP-completeness of 3PM and Lemma 4.5. The membership of MICT in NP is obvious.

Theorem 4.6 *MICT is NP-complete.*

5 Approximation heuristics for MICT

Our heuristics in fact work for the generalization of MICT where with each input constraint c a positive weight $w(c)$ is associated, and the objective is to construct a consensus tree for a subset of constraints of maximum total weight.

5.1 Heuristic 1

For a constraint $\{i,j\} < \{k,l\}$, where all the leaves are different, k and l are said to have an *upper occurrence* in the constraint, and i and j are said to have a *lower occurrence* in the constraint. For a constraint $\{i,j\} < \{i,k\}$, where i, j, k are different, i and j are said to have a lower occurrence in the constraint and k is said to have an upper occurrence in the constraint. The weight of an upper or lower occurrence in a constraint equals the weight of the constraint.

Lemma 5.1 *For any instance I of MICT, the total weight of upper occurrences is at least one third (half if all constraints contain four leaves) of the total weight of all occurrences in the constraints in I.*

Heuristic 1

input: a set C of m weighted constraints on leaves 1 through n;
output: a consensus tree T for a subset of C whose weight is at least one third (half if all constraints contain four leaves) of the total weight of the constraints in C;

1. $LEFT \leftarrow C$; $LEAVES \leftarrow \{1, ..., n\}$;
   $T \leftarrow \{v\}$;
2. if $LEFT = \emptyset$ then extend T by adding $|LEAVES|$ children to v, label them uniquely with elements in $LEAVES$, and return T;
3. pick a leaf y in $LEAVES$ which achieves the maximum ratio between the total weight of its upper occurrences and the total weight of its lower occurrences in the constraints in $LEFT$;
4. set Y to the set of constraints in $LEFT$ which contain y;
5. $LEFT \leftarrow LEFT \setminus Y$;
6. $LEAVES \leftarrow LEAVES \setminus \{y\}$;
7. extend T by adding two children to v; label the first child by y; set v to the second child;
8. go to 2

Theorem 5.2 *Heuristic 1 constructs a consensus tree for a subset of the input set of constraints C, whose total weight is at least one third (half if all constraints contain four leaves) of the total weight of C, in time $O((m+n)\log n)$.*

Proof: By Lemma 5.1 and the choice of y, the ratio between the total weight of upper occurrences and lower occurrences of y in the constraints in $LEFT$ is at least one third. All the constraints in Y in which y has an upper occurrence are satisfied by T by the construction of T.

To implement Steps 3, 6 efficiently, we arrange $LEAVES$ in a priority queue partially ordered by the ratio between the total weight of their upper and lower occurrences in constraints in $LEFT$. All the priority queue operations, i.e., creating the priority queue, picking the y's, updating the priority queue after Step 5, totally take $O((n+m)\log n)$ time.

To implement Steps 4, 5, we sort lexicographically C four times according to four cyclic permutations of the four leaves in each constraint. For $i = 1, ..., 4$, the i-th permutation puts the i-th leaf as the first, the $i+1$-st (in the cyclic order) as the second *etc.* Next, four search trees are built on the basis of the sorted lists. Using the search trees, we can find Y in $LEFT$ and remove it from $LEFT$ in time $O(|Y|\log n)$. We conclude that Steps 4, 5 totally take time $O((m+n)\log n)$ (inclusive the preprocessing). □

The absolute factors of one third and half respectively provided by Heuristic 1 are in fact worst-case optimal. For instance, in a sequence $(\{a_i, b_i\}, c_i)$, $(\{b_i, c_i\}, a_i)$, $(\{c_i, a_i\}, b_i)$, $i = 1, ..., k$, any consensus tree can satisfy at most one constraint from each consecutive triple. In case all constraints contain four leaves, the following sequence $\{a_i, b_i\} < \{c_i, d_i\}$, $\{c_i, d_i\} < \{a_i, b_i\}$, $i = 1, ..., k$, yields the lower bound $\frac{1}{2}$.

The consensus tree produced by Heuristic 1 has the form of a linear chain with singular leaves pending where only the last chain node can have larger degree. It is easy to slightly modify Heuristic 1 to output the set of the input constraints (*a priori*) satisfied by the tree. By running the algorithm of Aho *et al.* for the inferred consensus tree problem [1] on this set we can obtain a minimum height consensus tree for at least one third of the input constraints in time $O(mn\log n)$.

In case the minimum number of constraints necessary to delete in order to build a consensus tree for the remaining part is very small, and the number m of constraints relative to the number of leaves is high (it is always $O(n^4)$), an approach different from that of Heuristic 1 might be more useful.

5.2 Heuristic 2

Heuristic 2 for MILCT simply mimics the algorithm of Aho *et al.* for the inferred consensus tree problem restricted to constraints of the form $(\{i, j\}, k)$ given in [1]. The basic idea of the latter algorithm is simple. The input set of the leaves $1, 2, ..., n$ is partitioned into a minimal set of blocks satisfying the following requirement:

(*) If $(\{i,j\},k)$ is a constraint then i and j are in the same block.

Now, if the number of blocks in the minimal set is at least two then the algorithm of Aho *et al.* creates the consensus tree by connecting the roots of the consensus trees recursively computed for the respective blocks with a common parent root node. Otherwise, the number is one and it returns a null consensus tree.

For a subset S of leaves, let $G(S)$ denote the auxiliary graph on S where the edges are induced by the requirement (*), and their weights are equal to the total weight of the constraints inducing them.

Whenever the algorithm of Aho *et al.* is stuck at a non-divisible subset S of the set of leaves and has to return a null tree, Heuristic 2 simply finds a minimum weight edge cut of the auxiliary graph $G(S)$ (with respect to the current set of constraints). Next, the edges of the min-cut are deleted from $G(S)$ and the connected components of $G(S)$ are computed. Consequently, the constraints corresponding to the edges of the min-cut are also deleted. Finally, the approximation consensus trees for the connected components are recursively computed and connected by a common parent node.

In the appendix, we present an implementation of Heuristic 2 based on the recent, efficient implementation of the algorithm of Aho *et al.* restricted to constraints of the form $(\{i,j\},k)$ due to Henzinger *et al.* [8]. As a result, we obtain the following lemma.

Lemma 5.3 *Heuristic 2 can be implemented in expected time $O(n^3 \log n + m \log^3 n)$.*

Proof: A minimum weight edge cut can be computed with high probability in time $O(n^2 \log n)$ [13], and in the worst case it has to be done n times. Hence, all calls for minimum weight edge cut take $O(n^3 \log n)$ expected time. All other operations can be performed in expected time $O(m \log^3 n)$ like in the algorithm of Henzinger *et al.* (cf. [8]). Thus, the total expected time is $O(n^3 \log n + m \log^3 n)$. □

Lemma 5.4 *Let I be an instance of MICT, and let T be the tree produced by Heuristic 2 for I. The total weight of constraints in I not satisfied by T is within $height(T)$ of the minimum.*

Proof: Let J be a subset of I of minimum total weight such that $I \setminus J$ has a consensus tree. Next, let D be the set of connected components in the auxiliary graph where the edges corresponding to the constraints in J are deleted. Suppose that Heuristic 2 at some stage finds a min-cut in a currently connected fragment C. Clearly, C cannot be a subset of a simple component in D since then there wouldn't exist a consensus tree for $I \setminus J$. Hence, there is a subset J_C of J such that the set of edges corresponding to the constraints in J_C disconnects $G(C)$ into disjoint components. Clearly, the total weight of J_C is not smaller than the weight of minimum cut of $G(C)$. Now, it is sufficient to observe that the subsets J_C for distinct $C's$ on the same recursion level of Heuristic 2 are pairwise disjoint. □

Theorem 5.5 *Let n, w, t be respectively the number of leaves, the total weight of constraints, and the minimum total weight of the constraints to remove in an instance I of MILCT. Heuristic 2 constructs a consensus tree for a subset of the constraints in I whose total weight is not smaller than $w - nt$.*

Note that the number of constraints in I might be even cubic in n and that Heuristic 2 yields a better approximation factor than Heuristic 1 for MILCT whenever $t < \frac{2w}{3n}$.

6 Open problems

The approximability of MHT for instances with a constant number of trees of unbounded height is an open problem. Neither do we know whether or not it is possible to find a polynomial-time approximation scheme for instances of MHT with $O(1)$ trees of height $O(1)$.

It follows from Theorem 4.6 and the definition of MICT that this problem is strongly NP-complete. Hence, it cannot admit a fully polynomial-time approximation scheme [16]. However, it is an open question whether MICT admits a polynomial-time approximation scheme or at least a polynomial-time heuristic with a smaller approximation factor.

The complexity status of MILCT is also an interesting open question. If MILCT is NP-complete, does it admit a polynomial-time approximation scheme?

On a high level, the definitions of MICT and MILCT resemble those of MAX SAT and MAX k-SAT (see [9, 11]). In the design of Heuristic 1 we have utilized this similarity taking inspiration from the early heuristic for MAX k-SAT due to Johnson [11]. Recently, substantial progress in approximating MAX SAT and MAX k-SAT has been made by using linear programming, semidefinite programming, and randomized rounding [7, 9]. One of the main obstacles in applying these techniques to MICT is the complexity of "arithmetization" of the proper-descendant lowest-common-ancestor relation (the case of MILCT seems more promising).

References

[1] A.V. Aho, Y. Sagiv, T.G. Szymanski, and J.D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM Journal of Computing, Vol. 10, No. 3, 1981, pp. 405-421.

[2] M. Bellare and M. Sudan. Improved non-approximability results. Proc. of the 26th Annual ACM Symposium on the Theory of Computing (STOC), 1994, pp. 184-193.

[3] M. Farach, T. Przytycka, and M. Thorup. Computing the agreement of trees with bounded degrees. Proc. of the 3rd Annual European Symposium on Algorithms (ESA), 1995, pp. 381-393.

[4] M. Farach and M. Thorup. Fast Comparison of Evolutionary Trees. Proc. of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1994, pp. 481-488.

[5] M. Farach and M. Thorup. Optimal evolutionary tree comparison by sparse dynamic programming. Proc. of the 35th Annual Symposium on the Foundations of Computer Science (FOCS), 1994, pp. 770-779.

[6] C.R. Finden and A.D. Gordon. Obtaining common pruned trees. Journal of Classification 2, 1985, pp. 255-276.

[7] M.X. Goemans and D.P. Williamson. New $\frac{3}{4}$-approximation algorithms for MAX SAT. SIAM Journal of Discrete Mathematics, 7, pp. 656-666, 1994.

[8] M.R. Henzinger, V. King, and T. Warnow. Constructing a Tree from Homeomorphic Subtrees, with Applications to Computational Biology. Proc. of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1996, pp. 333-340.

[9] D.S. Hochbaum (editor). Approximation Algorithms for NP-hard Problems. PWS Publishing Company, Boston, 1995.

[10] C.A.J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. SIAM Journal of Discrete Mathematics, Vol. 2, No. 1, 1989, pp. 68-72.

[11] D.S. Johnson. Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences, 9, pp. 256-278, 1974.

[12] S. Kannan, T. Warnow, and S. Yooseph. Computing the Local Consensus of Trees. Proc. of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1995, pp. 68-77.

[13] D.R. Karger. Minimum Cuts in Near-Linear Time. Proc. of the 28th Annual ACM Symposium on Theory of Computing (STOC), 1996, pp. 56-63.

[14] D. Keselman and A. Amir. Maximum agreement subtree in a set of evolutionary trees - Metrics and efficient algorithms. Proc. of the 35th Annual Symposium on the Foundations of Computer Science (FOCS), 1994, pp. 758-769.

[15] T.W. Lam, W.K. Sung, and H.F. Ting. Computing the Unrooted Maximum Agreement Subtree in Sub-quadratic Time. Proc. of the 5th Scandinavian Workshop on Algorithm Theory (SWAT), 1996, pp. 124-135.

[16] C.H. Papadimitriou. Computational Complexity, Addison-Wesley, Reading, 1994.

[17] C. Phillips and T.J. Warnow. The Asymmetric Median Tree - A New Model for Building Consensus Trees. Proc. Combinatorial Pattern Matching, LNCS 1075, 1996, pp. 234-252.

[18] M. Steel and T. Warnow. Kaikoura tree theorems: Computing the maximum agreement subtree. Information Processing Letters 48, 1993, pp. 77-82.

Appendix: the implementation of Heuristic 2

Using recent dynamic data structures for graph connectivity, Henzinger *et al.* gave efficient implementations of the algorithm of Aho *et al.* restricted to constraints of the form $(\{i, j\}, k)$ [8]. Their randomized implementation takes $O(m \log^3 n)$ expected time. They use the undirected graph U and the directed graph D defined as follows.

- $U = (V, E)$ with V equal to the set $\{1, 2, ..., n\}$ of leaves and where for each constrain $(\{a, b\}, c)$ in C edges $\{a, b\}$ and $\{b, c\}$ are in E.
- $D = (V', A)$ where for each constrain $(\{a, b\}, c)$ in C nodes $\{a, b\}$ and $\{b, c\}$ are in V' and $\{a, b\} \to \{b, c\}$ is in A.

At the beginning the graph U is colored yellow. The graph U is used for finding yellow components. The consensus tree returned is found by combining the trees constructed for the yellow components. The graph D is used for finding edges in U that can be colored red, these edges correspond to the so-called maximal nodes in D.

A *maximal node* in D is a node with no outgoing edges, and a red edge whose endpoints are in different yellow components is called a *separable red edge*.

By slightly modifying the algorithm of Henzinger *et al.* and combining it with an algorithm for minimum weight edge cut [13], we can implement Heuristic 2 as follows.

Heuristic 2

1. Construct U and D. Add weights to the edges in U. The weight of an edge $\{a, b\}$ in U is equal to the sum of the weights of constraints of the form $(\{a, b\},\ )$. Color all nodes in D and edges in U yellow.
2. Identify maximal nodes in D. Recolor these nodes and the corresponding edges of U red.
3. If U has no edges, then return the consensus tree T with a root and all nodes in U children of the root. Otherwise, compute yellow components of U. If there is only one yellow component then find a minimum weight edge cut, delete the edges in the cut from U and the corresponding nodes from D, and recompute the yellow components. Let $\mathcal{C}_\infty, \mathcal{C}_\in, ..., \mathcal{C}_{||}$ be the current yellow components. Form a tree T by creating the root of T and connecting the root of the consensus trees recursively computed for the components $\mathcal{C}_\infty, \mathcal{C}_\in, ..., \mathcal{C}_{||}$ to it. For each current yellow component, identify the set E_{sep} of separable red edges incident to that new component. Delete these edges from U and the corresponding nodes from D. Go to Step 2.