Polynomial-Time Algorithms for the Ordered Maximum Agreement Subtree Problem

Anders Dessmark, Jesper Jansson, Andrzej Lingas, and Eva-Marta Lundell

Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden
{andersd,jj,andrzej,emj}@cs.lth.se

Abstract. For a set of rooted, unordered, distinctly leaf-labeled trees, the NP-hard maximum agreement subtree problem (MAST) asks for a tree contained (up to isomorphism or homeomorphism) in all of the input trees with as many labeled leaves as possible. We study the ordered variants of MAST where the trees are uniformly or non-uniformly ordered. We provide the first known polynomial-time algorithms for the uniformly and non-uniformly ordered homeomorphic variants as well as the uniformly and non-uniformly ordered isomorphic variants of MAST. Our algorithms run in time $O(kn^3)$, $O(n^3 \min\{nk,\ n + \log^{k-1} n\})$, $O(kn^3)$, and $O((k+n)n^3)$, respectively, where n is the number of leaf labels and k is the number of input trees.

1 Introduction

The basic combinatorial problem of finding a *largest common subsequence* for a set of sequences (LCS) and the well known problem of finding a *maximum agreement subtree* for a set of trees with distinctly labeled leaves (MAST) fall in the general category of problems of finding the *largest common subobject* for an input set of combinatorial objects.

In [7], Fellows *et al.* in particular studied the largest common subobject problem constrained to the so called p-sequences, i.e., sequences where each element occurs at most once. A p-sequence can be seen as an ordered, distinctly leaf-labeled star tree. In this paper, we study a natural generalization of the largest common subobject problem for p-sequences in which the objects are allowed to be arbitrary rooted, ordered trees with distinctly labeled leaves. Since this problem can be also regarded as a restriction of the MAST problem where tree ordering is required, we term it the *ordered maximum agreement subtree problem.*

For an extensive literature and motivations for the LCS and MAST problems, the reader is referred to [3, 13, 17] and [2, 5, 6, 11, 15, 16], respectively. The NP-hardness [2] and approximation NP-hardness [4, 10, 11] of the general MAST problem is one of the motivations for studying its ordered variants in this paper.

1.1 Variants of the Maximum Agreement Subtree Problem

A tree whose leaves are labeled by elements belonging to a finite set S so that no two leaves have the same label is said to be *distinctly leaf-labeled by* S. Throughout this paper, each leaf in such a tree is identified with its corresponding element

S.C. Sahinalp et al. (Eds.): CPM 2004, LNCS 3109, pp. 220–229, 2004.

in S. Let T be a rooted tree distinctly leaf-labeled by a given finite set S. For any subset S' of S, $T|S'$ denotes the tree obtained by first deleting from T all leaves which are not in S' and all internal nodes without any descendants in S' along with their incident edges, and then contracting every edge between a node having just one child and its child. Similarly, $T||S'$ denotes the tree obtained by deleting from T all leaves which are not in S' and all internal nodes without any descendants in S' along with their incident edges.

In *the maximum homeomorphic agreement subtree problem* (MHT), the input is a finite set S and a set $\mathcal{T} = \{T_1, ..., T_k\}$ of rooted, unordered trees, where each $T_i \in \mathcal{T}$ is distinctly leaf-labeled by S and no $T_i \in \mathcal{T}$ has a node of degree 1, and the goal is to find a subset S' of S of maximum cardinality such that $T_1|S' = ... = T_k|S'$. In *the maximum isomorphic agreement subtree problem* (MIT), the input is a finite set S and a set $\mathcal{T} = \{T_1, ..., T_k\}$ of rooted, unordered trees, where each $T_i \in \mathcal{T}$ is distinctly leaf-labeled by S, and the goal is to find a subset S' of S of maximum cardinality such that $T_1||S' = ... = T_k||S'$.

An *ordered* tree is a rooted tree in which the left-to-right order of the children of each node is significant. The *leaf ordering* of an ordered, leaf-labeled tree is the sequence of labels obtained by scanning its leaves from left to right. A set $\mathcal{T}$ of ordered trees distinctly leaf-labeled by S is said to be *uniformly ordered* if all trees in $\mathcal{T}$ have the same leaf ordering.

We study the following four ordered variants of MHT and MIT:

- The ordered maximum homeomorphic agreement subtree problem (OMHT)
- The ordered maximum isomorphic agreement subtree problem (OMIT)
- The uniformly ordered maximum homeomorphic agreement subtree problem (UOMHT)
- The uniformly ordered maximum isomorphic agreement subtree problem (UOMIT)

OMHT and OMIT are defined in the same way as MHT and MIT except that $\mathcal{T}$ is required to be a set of *ordered* trees. UOMHT and UOMIT are the special cases of OMHT and OMIT in which $\mathcal{T}$ is required to be uniformly ordered. Note that OMHT and OMIT are the natural generalization of the largest common subobject problem for p-sequences studied by Fellows *et al.* in [7].

From here on, n and k denote the cardinalities of S and $\mathcal{T}$, respectively.

1.2 Motivations

In certain evolutionary tree construction situations, one can determine or accurately estimate the leaf ordering of a planar embedding of the true tree by taking into account other kinds of data such as the geographical distributions of the species or data based on some measurable quantitative characteristics (average life span, size, *etc*). The ordered variants of MHT and MIT might also arise in graphical representation of evolutionary trees where additional restrictions are placed on the leaves (e.g., that they must be ordered alphabetically) for ease of presentation. In the context of the approximation NP-hardness of

MHT and MIT, their ordered restrictions are of theoretical interest in their own rights. Does the leaf ordering restriction make the problems computationally feasible? In [9], Gąsieniec *et al.* showed that an analogous ordering restriction on an NP-hard optimization problem occurring in the construction of evolutionary trees admits a polynomial-time algorithmic solution. Our results on the ordered variants of MHT and MIT will further confirm the power of ordering.

1.3 Related Results

Fellows, Hallet, and Stege studied the LCS problem for p-sequences among many other problems in their interesting article [7], and claimed that they could solve this problem for k p-sequences with n symbols in time $O(kn(k + \log n))$ [1].

Steel and Warnow [15] presented the first exact polynomial-time algorithms to solve MHT and its unrooted counterpart UMHT[2] when $k = 2$ and the degrees are unbounded. Since then, many improvements have been published, the fastest currently known being an algorithm for MHT with $k = 2$, invented by Kao, Lam, Sung, and Ting [12], that runs in $O(\sqrt{D}\, n \log(2n/D))$ time, where D is the maximum degree of the two input trees. Note that this is $O(n \log n)$ for trees with maximum degree bounded by a constant and $O(n^{1.5})$ for trees with unbounded degrees. Finally, for two rooted, *ordered* trees, a maximum agreement subtree can be computed in $O(n \log^2 n)$ time [16].

Amir and Keselman [2] considered the more general case $k \geq 2$. They proved that MHT is NP-hard already for three trees with unbounded degrees, but solvable in polynomial time for three or more trees if the degree of at least one of the trees is bounded by a constant. For the latter case, Farach, Przytycka, and Thorup [6] gave an algorithm with improved efficiency running in $O(kn^3 + n^d)$ time, where d is an upper bound on at least one of the input trees degrees; Bryant [5] proposed a conceptually different algorithm with the same running time.

Hein, Jiang, Wang, and Zhang [11] proved that MHT with three trees with unbounded degrees cannot be approximated within a factor of $2^{\log^\delta n}$ in polynomial time for any constant $\delta < 1$, unless NP $\subseteq$ DTIME$[2^{\text{polylog}\, n}]$. This inapproximability result also holds for UMHT [11]. Bonizzoni, Della Vedova, and Mauri [4] showed that it can be carried over to MIT restricted to three trees with unbounded degrees as well, and that even stronger bounds can be proved for MIT in the general case. Gąsieniec, Jansson, Lingas, and Östlin [10] proved that MHT is hard to approximate in polynomial time even for instances containing only trees of height 2, and showed that if the number of trees is bounded by a constant and all of the input trees' heights are bounded by a constant, then MHT can be approximated within a constant factor in $O(n \log n)$ time.

[1] Their algorithm, allowing different interpretations, seems to fail already for $x_1 = 1234$ and $x_2 = 1342$, producing either 12 or 34 instead of 134. Therefore, we were forced to use a weaker upper time-bound for this problem, given in Lemma 5 in this paper, in order to derive one of our main results.

[2] UMHT is defined like MHT except that all trees are unrooted and $T|S'$ now denotes the tree obtained by first deleting from T all nodes (and their incident edges) not on any path between two leaves in S', and then contracting every node with degree 2.

1.4 Our Results and Organization of the Paper

We present the first known polynomial-time algorithms for the uniformly and non-uniformly ordered homeomorphic variants (UOMHT and OMHT) as well as the uniformly and non-uniformly ordered isomorphic variants (UOMIT and OMIT) of the maximum agreement subtree problem. They run in time $O(kn^3)$, $O(n^3 \min\{nk,\ n+\log^{k-1} n\})$, $O(kn^3)$, and $O((k+n)n^3)$, respectively. Our results have been obtained by utilization of deep structural properties of ordered agreement subtrees which allowed for significant pruning of the otherwise unfeasible number of combinations of subproblems needed for the exact solution.

In Section 2, we introduce some common notation for our algorithms. In Section 3, we present the algorithm for UOMHT. Section 4 is devoted to the algorithm for OMHT. Section 5 describes the algorithms for UOMIT and OMIT.

2 Notation

We use the following notation. For any $a, b \in S$, denote by $OMHT_{a,b}$ the problem OMHT under the additional constraint that for any valid solution S', the leaf ordering of $T_1|S'$ must begin with a and end with b. Define $OMIT_{a,b}$, $UOMHT_{a,b}$, and $UOMIT_{a,b}$ analogously. Furthermore, let $UOMHT_{(a),(b)}$ be UOMHT restricted to the subset of S consisting of all elements in the interval $[a, b]$ in the leaf ordering. Note that while a and b are required to belong to any solution to $UOMHT_{a,b}$, they are not necessarily included in a solution to $UOMHT_{(a),(b)}$.

We also find it convenient to write $S = \{a_1, ..., a_n\}$ and in the uniformly ordered case let a_l precede a_r in the leaf ordering if $l < r$.

3 A Polynomial-Time Algorithm for UOMHT

In this section, we present an algorithm called *All-Pairs* that solves the uniformly ordered maximum homeomorphic agreement subtree problem. Our algorithm employs dynamic programming to build a table of solutions for $UOMHT_{(a_l),(a_r)}$ for all pairs of leaves a_l and a_r with $l \leq r$, using a procedure called *One-Pair*. *All-Pairs* and *One-Pair* are listed in Fig. 1 and Fig. 2, respectively.

Given indices l and r, *One-Pair* first computes $UOMHT_{a_l,a_r}$ and then finds the solution to $UOMHT_{(a_l),(a_r)}$ by taking the largest of $UOMHT_{a_l,a_r}$ and the two previously computed optimal solutions to subproblems $UOMHT_{(a_l),(a_{r-1})}$ and $UOMHT_{(a_{l+1}),(a_r)}$, both stored in the dynamic programming table.

When computing $UOMHT_{a_l,a_r}$, the algorithm considers the path P between a_l and a_r and the positions along P where the intervening leaves have their *lowest ancestors*. By *the lowest ancestor of leaf* a_j, we mean the node which is the lowest common ancestor of a_j and either a_l or a_r, i.e., the position on P where the path from a_j to the root joins P. By *the lowest ancestor edge of leaf* a_j, we mean the first edge on the path leading from the lowest ancestor of a_j to a_j.

For each input tree T_i, let v_i be the lowest common ancestor of a_l and a_r. In Step 1, the intervening leaves are divided into three classes depending on whether

Algorithm *All-Pairs*
Input: An instance of UOMHT.
Output: The subset of leaves in a maximum agreement subtree of $\mathcal{T}$.

for $length = 1$ to n **do**
  for $left = 1$ to $n - length + 1$ **do**
    Compute $UOMHT_{(a_{left}),(a_{left+length-1})}$ by a subroutine call to *One-Pair*$(left, left + length - 1)$ and enter the result in the table.
  endfor
endfor
return $UOMHT_{(a_1),(a_n)}$
End *All-Pairs*

Fig. 1. A dynamic programming algorithm for UOMHT.

their lowest ancestor is inside the path from v_i to a_l, equal to v_i itself, or inside the path from v_i to a_r. Any leaf that does not belong to the same class in all input trees can be discarded, since it can not exist in a solution together with a_l and a_r (Step 2). Next, in Step 3, the remaining intervening leaves are further processed and divided into sets according to their lowest ancestors so that two leaves having the same lowest ancestor belong to the same set. The divisions created by the different trees are merged into sets $S_1, ..., S_m$ so that two leaves belong to the same set S_p if and only if they have the same lowest ancestor in every tree in $\mathcal{T}$. The leaves in any such set S_p clearly form a consecutive sequence.

Each set S_p is further divided into subsets in Step 4. As in the previous division, every input tree is traversed to find the lowest ancestor edge for each leaf which connects it to the path between a_l and a_r. The resulting divisions are then merged to create the subsets $S_{p,1}, S_{p,2}, ...$ with the property that two leaves are in the same subset if and only if they have the same lowest ancestor edge in every tree in $\mathcal{T}$ (see Fig. 3). As above, the leaves in such a subset form a consecutive sequence. It also holds for any subset $A \subseteq S_{p,q}$ that if A induces a homeomorphic agreement subtree then the set $A \cup \{a_l, a_r\}$ also induces a homeomorphic agreement subtree. Such a maximum subset A can be looked up in the table (Step 5). However, two members of two different $S_{p,q}$ subsets may have the same lowest ancestor edge in some tree even though no such pair of leaves can belong to the same solution. Since members of a subset share a lowest ancestor edge in all trees, a conflict in one tree must hold for all pairs of leaves induced by the two subsets concerned. Thus, we can represent such a conflict as an edge in a special graph having the $S_{p,q}$ sets as vertices and the sizes of UOMHT solutions looked up in Step 5 as vertex weights (Step 6), and then compute a maximum weighted independent set in this graph (Step 7).

The maximum subset of leaves, connected to the same lowest ancestor in all trees, that can be included together with a_l and a_r in a homeomorphic agreement subtree are now known. Next, we choose the sets to be included in $UOMHT_{a_l,a_r}$. This is done in Steps 8 and 9 similarly as for the subsets of a single set. We build a

Algorithm *One-Pair*

Input: An instance of UOMHT and the index of two leaves a_l and a_r.

Output: The subset of leaves in a maximum agreement subtree of $\mathcal{T}$ restricted to the set of leaves in the range from a_l to a_r (i.e., $UOMHT_{(a_l),(a_r)}$).

1 For each $T_i \in \mathcal{T}$, divide the leaves $\{a_{l+1}, ..., a_{r-1}\}$ into sets according to the location of their lowest ancestor on the path between a_l and a_r.

2 Remove every leaf a_j, such that all subtrees induced by $\{a_l, a_j, a_r\}$ are not homeomorphic, from further consideration.

3 Construct the sets $S_1, ..., S_m$ of leaves such that two leaves are in the same set if and only if they are in the same set constructed in Step 1 for all trees in $\mathcal{T}$, and furthermore, for any two leaves $a_i \in S_p$ and $a_j \in S_q$, if $p < q$ then $i < j$.

for $p = 1$ to m **do**

4 Construct the sets $S_{p,1}, ..., S_{p,m_p}$ of leaves such that two leaves are in the same set if and only if they are connected by the same edge to the path between a_l and a_r for all trees in $\mathcal{T}$, and furthermore, for any two leaves $a_i \in S_{p,s}$ and $a_j \in S_{p,t}$, if $s < t$ then $i < j$.

5 **for** $q = 1$ to m_p **do**
$S_{p,q} = UOMHT_{(a_i),(a_j)}$ where i and j are the smallest and largest index, respectively, such that $a_i \in S_{p,q}$ and $a_j \in S_{p,q}$.
endfor

6 Construct a weighted graph $G_p = (V_p, E_p)$, where $V_p = \{S_{p,1}, ..., S_{p,m_p}\}$ with weights according to their sizes, and E_p contains every pair of subsets of leaves such that they share a lowest ancestor edge in some $T_i \in \mathcal{T}$.

7 Compute the maximum weighted independent set of G_p and let S_p be the union of the sets in the solution.

endfor

8 Construct a weighted graph $G = (V, E)$ where V is the set $\{S_1, ..., S_m\}$ with weights according to their sizes and E contains every pair of sets of leaves such that they are in the same set for some $T_i \in \mathcal{T}$.

9 Compute the maximum weighted independent set of G and let W be the union of the sets in the solution.

10 return The largest of W, $UOMHT_{(a_l),(a_{r-1})}$ and $UOMHT_{(a_{l+1}),(a_r)}$.

End *One-Pair*

Fig. 2. The subroutine for computing one entry of the dynamic programming table.

graph with the S_p sets as the vertices, with weights according to the solutions to the maximum weighted independent set problems for the subsets of the sets. The conflict edges now represent two separate sets sometimes connected to the same lowest ancestor (regardless of lowest ancestor edge). The maximum weighted independent set for this graph provides us with the solution to $UOMHT_{a_l,a_r}$.

Lemma 1. *One-Pair solves* $UOMHT_{(a_l),(a_r)}$ *in* $O(km)$ *time, with* $m = r - l + 1$.

Proof. The correctness follows from the discussion above. As for the time complexity, Steps 1 to 3 can be done in one traversal of each tree in $\mathcal{T}$ (this takes $O(km)$ time). Whenever the ancestor changes, this is recorded in the present

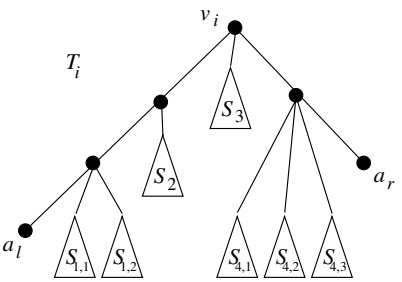

Fig. 3. An input tree T_i where the leaves $\{a_{l+1}, .., a_{r-1}\}$ are divided into sets according to lowest ancestor and lowest ancestor edge.

leaf. After all trees have been traversed, the union of these recorded changes will divide the leaves into the sought sequence of sets. Another traversal of the trees in $\mathcal{T}$ is sufficient to make the further division in Step 4.

Computing the weights for the subsets is done in $O(m)$ time, as there are $O(m)$ subsets. The edge sets can also be computed in the course of the traversal, by noting the lowest index of leaf that so far has shared a lowest ancestor edge with the current leaf. Since the leaves are uniformly ordered, a conflict with one leaf must also result in a conflict with all the intervening leaves. Solving the maximum weighted independent set problems in Step 7 for the constructed graphs can be done in linear time by the following dynamic programming procedure:

Process the vertices in the order of leaves contained. Keep a table of the maximum weight subset of vertices computed thus far. For every new entry (corresponding to the set $S_{p,q}$), compare the entry for set $S_{p,q-1}$ with the weight of the current vertex ($|S_{p,q}|$) added to the entry corresponding to $S_{p,s}$ where s is the largest index $s < q$ such that there is no conflict between $S_{p,s}$ and $S_{p,q}$, and choose the larger of the two values.

The edge set in Step 8 can be computed during the initial traversal. The final maximum weighted independent set problem in Step 9 can be solved in linear time with a dynamic programming procedure analogous to that of Step 7. □

Theorem 1. *Algorithm All-Pairs solves UOMHT in $O(kn^3)$ time.*

Proof. There are $O(n^2)$ pairs; each pair requires $O(kn)$ time by Lemma 1. □

4 An Algorithm for OMHT

In this section we present an algorithm for the ordered maximum homeomorphic agreement subtree problem running in time $O(n^3 \min\{nk,\ n + \log^{k-1} n\})$.

For each $a_l, a_r \in S$ where a_l precedes a_r in the leaf ordering of all trees in $\mathcal{T}$, or $a_l = a_r$, we consider the subproblem $OMHT_{a_l,a_r}$ (if $a_l = a_r$, the solution to $OMHT_{a_l,a_r}$ is trivially the singleton $\{a_l\}$). Our algorithm for $OMHT_{a_l,a_r}$ is similar to that for $UOMHT_{a_l,a_r}$, again focusing on the path between a_l and a_r. Let v_i be the lowest common ancestor of a_l and a_r in T_i. As before, the intervening leaves are divided into three classes depending on whether their lowest

ancestor is inside the path from v_i to a_l, equal to v_i, or inside the path from v_i to a_r. Any leaf that does not belong to the same class in all trees is immediately discarded as it cannot exist in a solution together with a_l and a_r.

With each of the remaining intervening leaves b, we associate the point $p(b) = (p(b)_1, ..., p(b)_k)$ in $\mathbb{N}^k$, where for $i = 1, ..., k$, $p(b)_i$ is defined as the distance between a_l and the lowest ancestor of b on the path from a_l to a_r in T_i. With each of the points p, we associate the set S_p of remaining leaves b for which $p = p(b)$. In turn, with each leaf $b \in S_p$, we associate another point $q(b) = (q(b)_1, ..., q(b)_k)$ in $\mathbb{N}^k$, where for $i = 1, ..., k$, $q(b)_i$ is the rank of the lowest ancestor edge of b on the path from a_l to a_r in T_i. We say that a point x in $\mathbb{N}^k$ *strictly dominates* a point y in $\mathbb{N}^k$ if each coordinate of x is greater than the corresponding one of y.

Lemma 2. *A pair of remaining leaves b, d, where $p(b) \neq p(d)$, can jointly with a_l and a_r be a subset of a leaf set inducing a homeomorphic agreement subtree for $\mathcal{T}$ if and only if $p(b)$ strictly dominates $p(d)$ or* vice versa.

Lemma 3. *A pair of remaining leaves b, $d \in S_p$, where $q(b) \neq q(d)$, can jointly with a_l and a_r be a subset of a leaf set inducing a homeomorphic agreement subtree for $\mathcal{T}$ if and only if $q(b)$ strictly dominates $q(d)$ or* vice versa.

Following the idea of double set subdivision from the previous section, we associate with each of the points q a set $S_{p,q}$ of remaining leaves $b \in S_p$ for which $q = q(b)$. We let the weight of each such point q be the weight the maximum cardinality of a solution to $OMHT_{c,d}$, where c, $d \in S_{p,q}$ and c and d occur in the same order in the leaf ordering of all trees in $\mathcal{T}$.

The points q can be interpreted as degenerate k-trapezoids [8]. By Lemma 3, it suffices to find a maximum weighted independent set in the k-trapezoid graph induced by the points q, i.e., in the intersection graph of the k-trapezoids, in order to find a largest subset of S_p that can jointly with a_l and a_r be a subset of a leaf set inducing a homeomorphic agreement subtree for $\mathcal{T}$.

Lemma 4. *[8] A maximum weighted independent set in a k-trapezoid graph on n vertices given with its box representation can be found in $O(n \log^{k-1} n)$ time.*

Consequently, we assign as the weight of p the cardinality of such a largest subset. Analogously, the points p can be interpreted as degenerate k-trapezoids. By Lemma 2, it is sufficient to find a maximum weighted independent set in the k-trapezoid graph induced by the points p to solve $OMHT_{a_l,a_r}$.

When k is large, the above method for finding the maximum weighted independent set in a k-trapezoid graph is super-polynomial. Therefore, for large values of k, we reduce the latter problem to a restricted version of LCS. Given k sequences, each consisting of a permutation of the elements in a set Σ, the problem is to find the longest sequence that is a subsequence of all input sequences.

Lemma 5. *Restricted LCS can be solved in $O(kn^2)$ time, where $n = |\Sigma|$.*

Proof. We first compute and store the ranks of all elements in all sequences in an $n \times k$ table, which takes $O(kn)$ time. For any $a, b \in \Sigma$, we can then determine if

a occurs before b in all sequences, if b occurs before a in all sequences, or neither. Build a directed graph $G = (\Sigma, E)$ where $(a, b) \in E$ if and only if a occurs before b in all sequences. G is clearly acyclic and can be constructed in $O(kn^2)$ time. A topological sort on G yields the longest directed path and this corresponds to a longest common subsequence. This can be done within the given time bound. □

In our reduction, each point corresponds to an element in Σ and each input tree to a sequence of elements. The reduction must take into account two difficulties: our points are weighted and two points may have the same coordinate for some input tree. The latter is handled by representing each input tree by *two* sequences. If a set of points share a coordinate for a tree, we put them in arbitrary order with respect to each other in the first sequence and reverse this order in the second sequence to ensure that no pair of such points can occur together in the solution. The weights of the points are accounted for by replacing an element a of weight w in all the $2k$ sequences by w consecutive elements $a_1, ..., a_w$, uniformly ordered throughout the sequences. Since the weights correspond to disjoint sets of leaves, the length of the sequences will be at most n.

The construction of the set of the remaining leaves as well as the construction of the set of points $p(b)$ and $q(b)$ can be carried out in total time $O(nk)$, the latter by using lexicographic sort (see for instance [1]). The weights of the points q can be determined in total time $O(n^2)$. The two level application of Lemma 4 or Lemma 5 in order to determine maximum weight independent sets of points q and p takes $O(n \min\{kn, \log^{k-1} n\})$ time. Hence, we obtain the following lemma.

Lemma 6. *If, for each pair a_m, a_q which follows a_l and precedes a_r in the leaf ordering of all trees in $\mathcal{T}$, the cardinalities of solutions to $OMHT_{a_m,a_q}$ are already stored then $OMHT_{a_l,a_r}$ can be solved in $O(n \min\{kn, n + \log^{k-1} n\})$ time.*

There are $O(n^2)$ subproblems $OMHT_{a_l,a_r}$. For any a_m, a_q in Lemma 6, the quadruple a_l, a_m, a_q, a_r is in particular a subsequence of the leaf ordering in the first tree. Hence, it suffices to solve the subproblems in order of non-decreasing distance between a_l and a_r in the leaf ordering of the first tree to solve them with dynamic programming by Lemma 6. We thus obtain our next main result.

Theorem 2. *OMHT can be solved in $O(n^3 \min\{nk,\ n + \log^{k-1} n\})$ time.*

5 Polynomial-Time Solutions for UOMIT and OMIT

Our polynomial-time algorithms for UOMHT and OMHT can easily be adapted to UOMIT and OMIT, respectively. In order to solve the corresponding subproblems $UOMIT_{a_l,a_r}$ and $OMIT_{a_l,a_r}$, we rely on the following lemma.

Lemma 7. *(Smolenskii [14]) Two labeled trees are isomorphic if and only if the distance between any two leaves with corresponding labels is the same.*

By Lemma 7, we can require the subtrees induced by the leaves a_l and a_r to be pairwise isomorphic. In particular, the paths connecting a_l with a_r must be of

equal length. Also, any relevant remaining leaf must have the same distances to a_l and a_r in all input trees. This immediately implies that such a leaf has the same lowest ancestor on the path connecting a_l with a_r in all the input trees. Hence, any pair of such leaves with different lowest ancestors on the aforementioned path can occur in a feasible solution to $UOMIT_{a_l,a_r}$ or $OMIT_{a_l,a_r}$, respectively. Consequently, the corresponding weighted graphs have no edges and there is no need to use special algorithms for computing maximum weighted independent sets. The time analysis simplifies and we obtain the following theorems.

Theorem 3. *$UOMIT$ can be solved in $O(kn^3)$ time.*

Theorem 4. *$OMIT$ can be solved in $O((k+n)n^3)$ time.*

References

1. A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
2. A. Amir and D. Keselman. Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. *SIAM J. Computing*, 26(6):1656–1669, 1997.
3. H. Bodlaender, R. Downey, M. Fellows, and T. Wareham. The parameterized complexity of sequence alignment and consensus. *Theor. Comput. Sci.*, 147:31–54, 1995.
4. P. Bonizzoni, G. Della Vedova, and G. Mauri. Approximating the maximum isomorphic agreement subtree is hard. *International Journal of Foundations of Computer Science*, 11(4):579–590, 2000.
5. D. Bryant. *Building Trees, Hunting for Trees, and Comparing Trees: Theory and Methods in Phylogenetic Analysis*. PhD thesis, University of Canterbury, 1997.
6. M. Farach, T. Przytycka, and M. Thorup. On the agreement of many trees. *Information Processing Letters*, 55:297–301, 1995.
7. M. Fellows, M. Hallett, and U. Stege. Analogs & duals of the MAST problem for sequences & trees. *Journal of Algorithms*, 49(1):192–216, 2003.
8. S. Felsner, R. Müller, and L. Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Applied Mathematics*, 74:13–32, 1997.
9. L. Gąsieniec, J. Jansson, A. Lingas, and A. Östlin. Inferring ordered trees from local constraints. In proceedings of CATS'98, vol. 20(3) of *Australian Computer Science Communications*, pages 67–76. Springer-Verlag, 1998.
10. L. Gąsieniec, J. Jansson, A. Lingas, and A. Östlin. On the complexity of constructing evolutionary trees. *Journal of Combinatorial Optimization*, 3:183–197, 1999.
11. J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71:153–169, 1996.
12. M.-Y. Kao, T.-W. Lam, W.-K. Sung, and H.-F. Ting. An even faster and more unifying algorithm for comparing trees via unbalanced bipartite matchings. *Journal of Algorithms*, 40(2):212–233, 2001.
13. D. Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, 25(2):322–336, 1978.
14. E. A. Smolenskii. Jurnal Vicisl. Mat. i Matem. Fiz, 2:371–372, 1962.
15. M. Steel and T. Warnow. Kaikoura tree theorems: Computing the maximum agreement subtree. *Information Processing Letters*, 48:77–82, 1993.
16. W.-K. Sung. *Fast Labeled Tree Comparison via Better Matching Algorithms*. PhD thesis, University of Hong Kong, 1998.
17. V. G. Timkovsky. Complexity of common subsequence and supersequence problems and related problems. *Cybernetics*, 25:1–13, 1990.