ELSEVIER

Information Processing Letters 48 (1993) 77-82

Information Processing Letters

Kaikoura tree theorems: Computing the maximum agreement subtree

Mike Steel *,[a], Tandy Warnow **,[b]

[a] Department of Mathematics, Massey University, Palmerston North, New Zealand
[b] Department of Computer and Information Sciences, 200 South 33rd Street, 276 Moore Building, Philadelphia, PA 19104-2398, USA

Communicated by M.J. Atallah
Received 25 January 1993
Revised 2 August 1993

Abstract

The *Maximum Agreement Subtree Problem* was posed by Finden and Gordon in 1985, and is as follows: given a set $S=\{s_1, s_2, \ldots, s_n\}$ and two trees P and Q leaf-labelled by the elements of S, find a maximum cardinality subset S_0 of S such that $P|S_0 = Q|S_0$. This problem arises in evolutionary tree construction, where different methods or data yield (possibly) different trees for the same species on which the trees agree. A superpolynomial time algorithm for finding a maximum agreement subtree of two binary trees was found by Kubicka et al. In this paper, we will present an $O(n^{4.5} \log n + V)$ algorithm to determine the largest agreement subtree of two trees on n leaves, where V is the maximum number of nodes in the trees. For the case of trees of maximum degree k, there are two algorithms presented: one has running time $O(k!n^2 + V)$ while the other has running time $O(k^{2.5}n^2 \log n + V)$. These algorithms also apply when the trees are constrained to be rooted; in this case a maximum agreement subtree is also constrained to be rooted. Each of the algorithms we present can be modified to produce a maximum agreement subtree, rather than computing only the size. Thus, we can solve the search problem in the same running time as above.

Key words: Algorithms; Analysis of algorithms; Combinatorial problems

1. Preliminary definitions

We begin with some definitions. A *tree* $T=(V(T), E(T))$ is a connected acyclic graph, with vertex set $V(T)$ and edge set $E(T)$. Given a finite set $S=\{s_1, s_2, \ldots, s_n\}$, we say that a tree T is

* This work was supported by the New Zealand Lotteries Commission.

** Corresponding author. This work was supported by the U.S. Department of Energy under Contract DE-AC04-76DP00789, and was done while visiting New Zealand supported by NSF.

SSDI 0020-0190(93)E0144-9

leaf-labelled by S if there is a bijection $l_T : S \to \mathscr{L}(T)$ between the elements of S and $\mathscr{L}(T)$, the leaves of T. We will denote by $t \subseteq_h T$ that t is homeomorphic to a (not necessarily proper) subtree of T. For $t \subseteq_h T$, $\mathscr{L}(t)$ is identified naturally with a subset of $\mathscr{L}(T)$, so that $l_T^{-1}(i)$ is defined for $i \in \mathscr{L}(t)$. For $t \subseteq_h T$, we can define the set of labels at the leaves of t to be $L(t) = \{l_T^{-1}(i)\colon i \in \mathscr{L}(t)\}$. Given a subset S_0 of S, $T|S_0$ refers to the minimal homeomorphic subtree of T containing all the leaves labelled by elements of S_0; in this tree, nodes of degree two are suppressed. Given two trees P and Q each leaf-labelled by S and $S_0 \subseteq S$, we say that $P|S_0 = Q|S_0$ if there is a graph isomorphism $\phi : V(P|S_0) \to V(Q|S_0)$ such that $l_Q^{-1}(\phi(l_P(s))) = s$ for all $s \in S_0$. Thus, the mapping ϕ must carry labels to labels. The tree $t = P|S_0 = Q|S_0$ is called an *agreement subtree* for P and Q. Thus, while P and Q may each have nodes of degree two in them, their agreement subtrees will not.

When the tree t is rooted, we let $r(t)$ denote the root of t. We can also then speak of the *least common ancestor* of a set of nodes. Given a set $V_0 \subseteq V(T)$, let $\text{lca}_T(V_0)$ be the node v in the tree T such that

(1) v lies on every path from a node $s \in V_0$ towards $r(T)$, and

(2) if v' satisfies (1), then v' lies on the path from v towards $r(T)$.

We will usually refer to least common ancestors only of pairs of nodes.

The *Agreement Subtree Problem* is then as follows:

Problem: The Agreement Subtree Problem

Instance: A set $S = \{s_1, s_2, \ldots, s_n\}$, two trees P and Q which are leaf-labelled by S, and an integer k.

Question: Does there exist a subset S_0 of cardinality at least k such that $P|S_0 = Q|S_0$?

This problem arises naturally in the application to phylogenetic tree construction, where trees for the same species set may be constructed in a variety of ways (either the optimality criteria may differ, or the trees may be based upon different data sets).

The agreement subtree problem was first posed by Finden and Gordon in 1985 and a method for finding a subtree on which two binary trees agreed was presented [1]. Unfortunately, the heuristic did not guarantee that the subtree would be of maximum cardinality. In [3], Kubicka et al. presented an algorithm for the agreement subtree problem on binary trees, which had running time $O(n^{(1/2+\varepsilon)\log_2 n})$. Lower bounds on the minimum size of the agreement subtree of two n-leaf binary trees were found by Kubicka et al. in [4]. In this paper we present the first polynomial-time algorithm for the problem of computing the maximum agreement subtree of two trees. The algorithm we present has running time $O(n^{4.5} \log n + V)$. For trees of maximum degree k we present two algorithms: one has running time $O(k!n^2 + V)$ and the other has running time $O(k^{2.5}n^2 \log n + V)$.

When the trees P and Q are constrained to be rooted, then the definition of an agreement subtree t carries the additional requirement that t is homomorphic to a rooted subtree of P and Q. Thus, relationships between least common ancestors of leaves in t must hold in P and Q as well. Let $\phi_P : V(t) \to V(P)$ denote the function mapping t homeomorphically onto its image in P. Then the additional requirement for rooted trees is that

$$\text{lca}_P(\phi_P(a), \phi_P(b)) = \phi_P(\text{lca}_T(a, b)).$$

Of course, the same condition must be true for the function ϕ_Q.

The algorithms we give for computing the size of the maximum agreement subtrees of unrooted trees can be modified for the rooted case, or to construct the maximum agreement subtree itself, with the same running times.

2. Finding the maximum agreement subtree of two trees

We now present a dynamic programming algorithm for finding a maximum agreement subtree of two trees on n labelled leaves. For the general case, the algorithm has running time $O(n^{4.5} \log n + V)$, where V is the number of nodes between the two trees. For trees of maximum degree k, the running time is bounded by $O(k^{2.5}n^2 \log n + V)$. We will also give a variant on the algorithm for bounded degree trees which has running time $O(k!n^2 + V)$.

The first step of the algorithm is to replace P and Q, respectively, by $P|S$ and $Q|S$. This can be accomplished in $O(V)$ time, where V is the number of nodes in P and Q, by repeatedly contracting an edge incident to a degree-two node until there are no degree-two nodes left. Thus, the tree t is an agreement subtree for P and Q if and only if it is an agreement subtree for $P|S$ and $Q|S$. Therefore, replacing P and Q by $P|S$ and $Q|S$, respectively, will not cause any problems, we will henceforth assume that there are no internal nodes of degree of two in either P or Q, so that the number of nodes in P and Q is then linear in the number of leaves, n.

The problem we solve in this paper is to compute the maximum k such that P and Q have an agreement subtree on k leaves. We will let $MAST(P, Q)$ denote that maximum value. If $|S| \leqslant 3$ so that $P|S = Q|S$, then $MAST(P, Q) = |S|$. The algorithm we will describe in the following section is used only when $|S| > 3$, in which case more care is needed to compute the size of the maximum agreement subtree.

2.1. Definitions

Let T be an unrooted tree leaf-labelled by $S = \{s_1, s_2, \dots, s_n\}$. We will use the term *e-subtree* to indicate a subgraph of T which is component of $T - \{e\}$ for some edge $e \in E(T)$. Each e-subtree is naturally rooted at the vertex incident to the deleted edge. The deletion of the root of an e-subtree x will create subgraphs $x^1, x^2, \dots, x^m$. These subgraphs are also e-subtrees by our definition.

Now let p and q be e-subtrees of P and Q respectively. Note that here p and q need not have the same label set. When two e-subtrees p and p' arise by deleting the same edge from P, we will say that p and p' are *complementary*. Since $P|S = P$, there are $O(n)$ e-subtrees of P, and similarly there are $O(n)$ e-subtrees of Q. We order the e-subtrees of P by inclusion, and compute a linear extension $\mathscr{O}(P)$ of this partial ordering. In the same way we construct the linear ordering $\mathscr{O}(Q)$ and compute a linear ordering $\mathscr{O}$ on $\mathscr{O}(P) \times \mathscr{O}(Q)$. There are $O(n^2)$ pairs (p, q) in this order $\mathscr{O}$.

2.2. The algorithm

The algorithm is a dynamic programming algorithm, in that it computes $MAST(p, q)$ for each $(p, q) \in \mathscr{O}$ only after we have computed $MAST(p', q')$ for all e-subtrees p', q' such that (p', q') precedes (p, q) in the order $\mathscr{O}$. The minimal elements (p, q) of the order $\mathscr{O}$ are where p or q is a single leaf. For these cases, $MAST(p, q) = |L(p) \cap L(q)|$. Thus, we need to compute the set of labels $L(x)$ which appear in each e-subtree x. These sets are given by characteristic vectors, and can be computed using depth-first-search. The value of $MAST(p, q)$ for the remaining pairs (p, q) can be determined in a somewhat complicated manner, which involves examining certain predecessors in the containment order, and computing a maximum matching in a weighted bipartite graph. This is now described.

Let $G(p, q)$ be the weighted version of $K_{r,s}$ with $w(i, j) = MAST(p^i, q^j)$, and let $\mathscr{W}(p, q)$ be the weight of a maximum matching in this weighted bipartite graph.

The value for $MAST(p, q)$ is then given as follows:

$MAST(p, q) = |L(p) \cap L(q)|$ if either p or q is a singleton, else

$$\max\{\mathscr{W}(p, q),\ MAST(p, q^1),\ MAST(p, q^2), \dots, MAST(p, q^s),\ MAST(q, p^1),\ MAST(q, p^2), \dots, MAST(q, p^r)\}$$

Note that $MAST$ takes as its arguments two rooted trees. Since the input to the Maximum Agreement Subtree may be unrooted, we will define the function $MAX(P, Q)$ to be the number of leaves in a maximum agreement subtree of

P and Q. The computation of $MAX(P, Q)$ is then set to

$$\max_{p_1,p_2,q_1,q_2} \left\{ \begin{matrix} MAST(p_1, q_1) + MAST(p_2, q_2) \\ MAST(p_1, q_2) + MAST(p_2, q_1) \end{matrix} \right\},$$

where (p_1, p_2) and (q_1, q_2) are both pairs of complementary e-subtrees.

The entire algorithm thus has the following structure:

```
Algorithm:
If $|S| \leqslant 3$, then $MAX(P, Q) = |S|$ and EXIT. Else
    Compute the lists $\mathscr{O}$ of pairs $(p, q)$ of $e$-subtrees as described above, where $p \subseteq P$ and $q \subset Q$.
    For each $(p, q) \in \mathscr{O}$ Compute $MAST(p, q)$
    Compute $MAX(P, Q)$
end of algorithm
```

2.3. *Proof of correctness*

We now show that this algorithm finds the correct value for the maximum agreement subtree of two rooted trees p and q.

Lemma 1. *Let p and q be two e-subtrees of P and Q respectively, and let t be a maximum agreement subtree of p and q. Then $|\mathscr{L}(t)| = MAST(p, q)$.*

Proof. The proof is by strong induction on $l = |\mathscr{L}(p)| + |\mathscr{L}(q)|$.

The trees p, q, and t are each rooted and thus we have subtrees $p^1, p^2, \ldots, p^r$, $q^1, q^2, \ldots, q^s$, and $t^1, t^2, \ldots, t^h$. The base case is where $l = 2$, so that p and q are each leaves, and $MAST(p, q) = |L(p) \cap L(q)|$, which is obviously correct. So assume true for all pairs of e-subtrees with at most l leaves between them, let p, q be given with $l+1$ leaves between them, and let t be a maximum agreement subtree of p and q. We will first show that $MAST(p, q) \leqslant |\mathscr{L}(t)|$.

Assume, without loss of generality, the value of $MAST(p, q)$ is either $\mathscr{W}(p, q)$ or $MAST(p, q^y)$, for some y. If $MAST(p, q) = \mathscr{W}(p, q)$, then there is a matching in the weighted bipartite graph $G(p, q)$ of weight $\mathscr{W}(p, q)$. Inductively, the weight of an edge (p^i, q^j) in that matching corresponds to the number of leaves in an agreement subtree $t(i, j)$ of p^i and q^j. Construct the rooted tree $T(p, q)$ by adding a node r and making the root of each $t(i, j)$ the child of r. $T(p, q)$ is an agreement subtree of p and q, so that $|\mathscr{L}(T(p, q))| \leqslant |\mathscr{L}(t)|$. Thus, $MAST(p, q) = \mathscr{W}(p, q) = |\mathscr{L}(T(p, q))| \leqslant |\mathscr{L}(t)|$. The other case is where $MAST(p, q) = MAST(p, q^y)$ for some y. In this case, inductively, $MAST(p, q^y) = |\mathscr{L}(t')|$, where t' is a maximum agreement subtree of p and q^y. Clearly t' is also an agreement subtree of p and q, so that $MAST(p, q) = |\mathscr{L}(t')| \leqslant |\mathscr{L}(t)|$. Thus, in each case, $MAST(p, q) \leqslant |\mathscr{L}(t)|$. To complete the proof we only need to show that $|\mathscr{L}(t)| \leqslant MAST(p, q)$, for all p, q.

Let $v_p = \phi_p(r(t))$ and $v_q = \phi_q(r(t))$. If $v_p \neq r(p)$, then $v_p \in V(p^i)$ for some $i = 1, 2, \ldots, r$. In this case, $t \subseteq_{\text{h}} p^i$ and so $|\mathscr{L}(t)| \leqslant MAST(p^i, q) \leqslant MAST(p, q)$. Similarly, if $v_q \neq r(q)$, then $t \subseteq_{\text{h}} q^j$ for some $j = 1, 2, \ldots, s$, and so $|\mathscr{L}(t)| \leqslant MAST(q^j, p) \leqslant MAST(p, q)$. The remaining case is where $v_p = r(p)$ and $v_q = r(q)$. We will show that in this case $|\mathscr{L}(t)| \leqslant \mathscr{W}(p, q)$, and since $\mathscr{W}(p, q) \leqslant MAST(p, q)$, the assertion that $|\mathscr{L}(t)| \leqslant MAST(p, q)$ follows.

To complete our proof, we will show that

(1) for each i, there exist a and b such that $\phi_p(t^i) \subseteq_{\text{h}} p^a$ and $\phi_q(t^i) \subseteq_{\text{h}} q^b$,

(2) if $\phi_p(t^i) \subseteq_{\text{h}} p^a$ and $\phi_p(t^j) \subseteq_{\text{h}} p^a$, then $i = j$, and

(3) if $\phi_q(t^i) \subseteq_{\text{h}} q^b$ and $\phi_q(t^j) \subseteq_{\text{h}} q^a$, then $i = j$.

We will show that these assertions simply that t defines a matching in $G(p, q)$ of weight at least $|\mathscr{L}(t)|$. Therefore, $|\mathscr{L}(t)| \leqslant \mathscr{W}(p, q)$, as required.

To prove these assertions, we will show that if these conditions do not hold, then without loss of generality there exist nodes $u, v \in V(t)$ such that $\phi_p(\text{lca}_t(u, v)) \neq \text{lca}_p(\phi_p(u), \phi_p(v))$, contradicting that ϕ_p maps t homeomorphically to a subtree of p.

For each $i = 1, 2, \ldots, m$ there is an a, $1 \leqslant a \leqslant r$ such that $\phi_p(r(t^i)) = p_i \in V(p^a)$. Thus, we can define the function P by $P(i) = a$. Similarly, for each $i = 1, 2, \ldots, m$ there is a b, $1 \leqslant b \leqslant s$ such that $\phi_q(r(t^i)) = q_i \in V(q^b)$, and we can define the function Q by $Q(i) = b$. Therefore, $t^i \subseteq_{\text{h}} p^{P(i)}$

and $t^i \subseteq_h q^{Q(i)}$. Thus assertion (1) is proved. To prove assertion (2), suppose that $P(i) = P(j) = a$, for $i \neq j$. Then

$$\mathrm{lca}_p(\phi(r(t^i)), \phi_p(r(t^j))) \leqslant r(p^a) < r(p).$$

On the other hand,

$$\phi_p(\mathrm{lca}_t(r(t^i)), r(t^j)) = \phi_p(r(t)) = r(p).$$

Thus, it is not the case that $\mathrm{lca}_p(\phi_p(u), \phi_p(v)) = \phi_p(\mathrm{lca}_t(u, v))$ for all $u, v \in t$, contradicting our assumption. Thus assertion (2) is proved. The proof of (3) follows exactly along the same lines.

To construct a matching in $G(p, q)$ having weight at least $|\mathscr{L}(t)|$, select edge (p^a, q^b) if and only if for some i, $P(i) = a$ and $Q(i) = b$. These edges form a matching in $G(p, q)$ by conditions (1), (2), (3) above, and t^i is an agreement subtree of p^a and q^b of at most $w(a, b)$ leaves. Thus,

$$|\mathscr{L}(t)| = \sum_{i=1}^{h} |\mathscr{L}(t^i)| \leqslant \sum_{i=1}^{h} w(P(i), Q(i)) \leqslant \mathscr{W}(p, q).$$

Therefore, $|\mathscr{L}(t)| \leqslant MAST(p, q)$ for all p, q.

Since $|\mathscr{L}(t)| \leqslant MAST(p, q) \leqslant |\mathscr{L}(t)|$ for all p, q, it follows that $MAST(p, q) = |\mathscr{L}(t)|$ for all p, q. □

We complete the proof of correctness by showing that the algorithm correctly determines the number of leaves in a maximum agreement subtree of two *unrooted* trees.

Theorem 2. *The algorithm correctly determines the size of a maximum agreement subtree for two trees on n labelled leaves.*

Proof. Let P and Q be two trees on n leaves each, and let T be a maximum agreement subtree of P and Q. Note that if $|S| \leqslant 3$ then $P|S$ and $Q|S$ are identical (since P and Q are unrooted trees), and the algorithm will correctly determine this by setting $MAX(P, Q) = |S|$. Now suppose $|S| \geqslant 4$.

We will first show that $MAX(P, Q) \leqslant |\mathscr{L}(T)|$. Since $|S| \geqslant 4$,

$$MAX(P, Q) = \max_{p_1, p_2, q_1, q_2} \{MAST(p_1, q_1) + MAST(p_2, q_2)\},$$

where p_1, p_2 and q_1, q_2 are pairs of complementary e-subtrees of P and Q respectively. Suppose that $MAX(P, Q) = MAST(p', q') + MAST(p'', q'')$, for (p', p'') and (q', q'') pairs of complementary e-subtrees. Let t' be a maximum agreement subtree of p' and q' and let t'' be a maximum agreement subtree of p'' and q''. By the previous lemma, $MAST(p', q') = |\mathscr{L}(t')|$ and $MAST(p'', q'') = |\mathscr{L}(t'')|$. Let t be the unrooted tree formed by adding an edge between the roots of t' and t''. Then

$$\begin{aligned} |\mathscr{L}(t)| &= |\mathscr{L}(t')| + |\mathscr{L}(t'')| \\ &= MAST(p', q') + MAST(p'', q'') \\ &= MAX(P, Q). \end{aligned}$$

Since (p', p'') and (q', q'') are both pairs of complementary e-subtrees, t is an agreement subtree of P and Q, so that $|\mathscr{L}(t)| \leqslant |\mathscr{L}(T)|$. It follows that $MAX(P, Q) \leqslant |\mathscr{L}(T)|$. To complete the proof we show that $|\mathscr{L}(T)| \leqslant MAX(P, Q)$.

Let e_T be an edge in T, and let $T - \{e\}$ have e-subtrees T_1 and T_2 with label sets $L(T_i) = S_i$ for $i = 1, 2$. Thus, e defines a partition of the label set $L(T)$ into two parts, S_1 and S_2. Since P and Q both contain homeomorphic images of T, each of P and Q must contain edges creating the same bipartition on $L(T)$; let these edges be e_P and e_Q.

The removal of e_P from P creates trees P_1 and P_2, and similarly we have trees Q_1 and Q_2 created by removing e_Q from Q. Then (P_1, P_2) and (Q_1, Q_2) are both pairs of complementary e-subtrees. Without loss of generality we can claim that $S_i \subseteq L(P_i) \cap L(Q_i)$, for $i = 1, 2$. Thus P_i and Q_i each contain a rooted subtree homeomorphic to T_i, for $i = 1, 2$. Therefore, T_i is an agreement subtree of P_i and Q_i, for $i = 1, 2$, and $MAST(P_i, Q_i) \geqslant |\mathscr{L}(T_i)|$, by the previous lemma. Then

$$\begin{aligned} MAX(P, Q) &\geqslant MAST(P_1, Q_1) + MAST(P_2, Q_2) \\ &\geqslant |\mathscr{L}(T_1)| + |\mathscr{L}(T_2)| = |\mathscr{L}(T)|. \end{aligned}$$

Since $|\mathscr{S}(T)| \geqslant MAX(P, Q) \geqslant |\mathscr{S}(T)|$, it follows that $MAX(P, Q) = |\mathscr{S}(T)|$, as stated. □

2.4. Running time analysis

The initialization involves computing all e-subtrees t of P or Q, computing the label set $L(t)$ for each e-subtree t, and computing the linear order $\mathscr{O}$. Using depth-first-search, we can compute $L(t)$ in $O(n)$ time per e-subtree t, and thus accomplish the initialization in $O(n^2)$ time.

When neither p nor q is a single leaf, the cost of the $MAST(p, q)$ computation includes the cost of computing a maximum matching on the bipartite weighted graph $G(p, q)$. One way to do this is to use the Gabow and Tarjan algorithm of [2] which has running time $O(m\sqrt{n}\ \log(Nn))$, where m is the number of edges, n the number of nodes, and N the maximum weight of an edge. This results in a cost of $O(n^{2.5} \log n)$ for the general case, and $O(k^{2.5} \log n)$ for the case where each node is constrained to have degree at most k. As an alternative, if k is small enough, the $(k-1)!$ possible matchings can each be evaluated, at a cost of $O(k)$ additions per matching, for a total cost of $O(k!)$ to find the maximum matching. The weight $\mathscr{W}(p, q)$ of this maximum matching is then compared to $r + s \leqslant n$ other values, where p has r subtrees and q has s subtrees, for an additional cost of $O(n)$ at worst for the general case, and $O(k)$ for the case where the degrees are constrained to be at most k.

Since there are $O(n^2)$ pairs (p, q) of e-subtrees, this costs at worst $O(n^{4.5} \log n)$ for the general case, and $O(k!n^2)$ or $O(k^{2.5}n^2 \log n)$ for the case where the degrees in P and Q are constrained to be at most k.

The final computation of selecting the maximum among $O(n^2)$ values involves $O(n^2)$ additions, and $O(n^2)$ comparisons. Thus, the overall cost of the algorithm is $O(n^{4.5} \log n)$ for arbitrary trees P, Q, and either $O(k!n^2)$ or $O(k^{2.5}n^2 \log n)$ for trees in which all nodes have degree at most k.

Acknowledgements

The authors wish to thank NSF for providing funds for the second author to travel to New Zealand and work with the first author. The referee is also thanked for his/her very careful reading of the text and thoughtful comments.

References

[1] C.R. Finden and A.D. Gordon, Obtaining common pruned trees, *J. Classification* **2** (1985) 255–176.

[2] H.N. Gabow and R.E. Tarjan, Faster scaling algorithms for network problems, *SIAM J. Comput.* **18** (5) (1989) 1013–1036.

[3] E. Kubicka, G. Kubicki, and F.R. McMorris, An algorithm to find agreement subtrees, *J. Classification* (1992), to appear.

[4] E. Kubicka, G. Kubicki, and F.R. McMorris, On agreement subtrees of two binary trees, Manuscript, 1992.