Kaikoura Tree Theorems: Computing The Maximum Agreement Subtree

Mike Steel* Tandy Warnow†

February 11, 2003

Abstract

The *Maximum Agreement Subtree Problem* was posed by Finden and Gordon in 1985 [2], 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 set, and the problem is to determine the largest set of species on which the trees agree. An exponential time algorithm for finding the maximum agreement subtree of two binary trees was found by Kubicka et. al. [4]. In this paper, we will present an $O(n^{4.5}\alpha(n^2))$ algorithm to determine the largest agreement subtree of two trees. For the case of trees of maximum degree k, the algorithm has running time $O(n^2\alpha(n^2))$.

1 Preliminary Definitions

We begin with some definitions. A *tree* T is a connected acyclic graph. Given a finite set $S = \{s_1, s_2, \ldots, s_n\}$, we say that a tree T is *leaf-labelled* by S if there is a bijection between the leaves of T and the elements of S. The leaf labelled by the element $s \in S$ is indicated by $L(s)$, and the label at leaf v is given by $L^{-1}(v)$. 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 supressed. Given two trees P and Q each leaf-labelled by S, we say that $P|S_0 = Q|S_0$ if there is a graph isomorphism (i.e. edge-preserving) $\phi : V(P|S_0) \to V(Q|S_0)$ such that $L^{-1}(\phi(L(s))) = s$ for all $s \in S$. Thus, the mapping ϕ must carry labels to labels.

*Department of Mathematics, Massey University, Palmerston North, New Zealand; this work was supported by the New Zealand Lotteries Commission

†Division 1423, Sandia National Laboratories, Albuquerque, NM 87185-5800; 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

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 binary 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[2], and a method for finding a subtree on which two binary trees agreed was presented. Unfortunately, the heuristic did not guarantee that the subtree would be of maximum cardinality. In [4], Kubicka et. al. presented an algorithm for the agreement subtree problem on binary trees, which had running time $O(n^{(\frac{1}{2}+\epsilon)\log_2 n})$. Lower bounds on the minimum size of the agreement subtree of two n-leaf binary trees were found by Kubicka et al in [5]. 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 $O(n^{4.5}\alpha(n^2))$, where $\alpha(n)$ is the inverse Ackerman function. For trees of maximum degree k, the algorithm has running time $O(n^2\alpha(n^2))$.

The organization of the remainder of the paper is as follows. In section 2 we describe an $O(n^2\alpha(n^2))$ algorithm for finding the maximum agreement subtree of two binary trees. In section 3 we show how to extend the techniques of section 2 for the general case, where the maximum degree of the trees are not constrained. We then discuss open problems in section 4.

2 Finding the Maximum Agreement Subtree of Binary Trees

We now present an $O(n^2\alpha(n^2))$ dynamic programming algorithm for finding the maximum agreement subtree of two binary trees on n labelled leaves.

2.1 Definitions

Define a *subtree* of a tree T to be a subgraph of T which is a component of $T - \{e\}$ for some edge $e \in E(T)$. For subtrees p, q of P, Q respectively, we will compute the size of the maximum agreement subtree $t = MAST(p, q)$ on which they agree. Note that here p and q may not have the same label set. When two subtrees p and q arise by deleting a single edge, we will say that p and q are *complementary.* We specifically need to keep track of all pairs of complementary subtrees.

We order the subtrees of P by inclusion, and compute a linear extension $L(P)$ of this partial ordering. In the same way we construct the linear ordering $L(Q)$, and similarly we can compute a linear ordering $\mathcal{L}$ on $L(P)$ x $L(Q)$. We then compute $MAST(p,q)$ for each $p \in L(P)$ and $q \in L(Q)$, ordered by $\mathcal{L}$, where *MAST(t,t')* will be the number of leaves in the maximum agreement subtree of t and t',

Note that we do not compute $MAST(p,q)$ until we have computed $MAST(t,t')$ for all subtrees $t \subset p$ and $t' \subset q$. There are $O(n)$ of these subtrees formed by these edge deletions, and each such subtree t is naturally rooted at the vertex y incident to the edge e deleted in order to form t (recall t is a component of $T - \{e\}$, for $T \in \{P,Q\}$). Furthermore, each subtree t has two children subgraphs t^1 and t^2, since the trees P and Q are binary; thus, the removal of the node y from t will create two subgraphs t^1 and t^2. These subgraphs are also subtrees by our definition.

2.2 Algorithmic Details

We need to compute the labels $L(t_i)$ which appear in each subtree t_i. To compute $L(t_i)$, we first compute $L(t'_i)$ for each $t'_i \subset t_i$; then, $L(t_i)$ is just the union of two label sets, since each T_i is binary. Thus, these computations can be completed using $O(n)$ *find* operations, for a total cost of $O(n\alpha(n))$, where $\alpha(n)$ is the inverse ackerman function[8].

Now assume p and q are subtrees of P and Q respectively, and that we have computed $MAST(p',q')$ for all subtrees $p' \subset p$ and $q' \subset q$. In particular, we will have computed $MAST(p^i,q^j)$, for $i = 1,2$ and $j = 1,2$.

The computation of $MAST(p,q)$ then depends upon whether p or q are both subtrees containing more than one leaf. If p contains only one leaf x, then $MAST(p,q) = 1$ if $L(x) \in L(q)$, and otherwise $MAST(p,q) = 0$. The case where q contains only one leaf is handled similarly. When both subtrees p and q contain at least two leaves, then the value of $MAST(p,q)$ is obtained by maximizing the score obtained from the different combinations of their constituent subtrees, p^1, p^2, q^1, and q^2. To summarize:

$$\begin{aligned}
&MAST(p,q) = \\
&\quad |L(p) \cap L(q)|, \text{ if either } p \text{ or } q \text{ is a singleton, or} \\
&\quad max\{MAST(p^1,q^1) + MAST(p^2,q^2), \\
&\qquad MAST(p^1,q^2) + MAST(p^2,q^1)\}, \text{ otherwise.}
\end{aligned}$$

The computation of $MAST(P,Q)$ is then set to

$$max_{(p_1,p_2)\ (q_1,q_2)}\{MAST(p_1,q_1)+MAST(p_1,q_2), MAST(p_1,q_2)+MAST(p_2,q_1)\},$$

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

2.3 The Algorithm

Algorithm:
Compute the list $L(P)$ of subtrees $t \subset P$, so that
if $t \subset t'$ then t appears before t' in the list $L(P)$.
Compute list $L(Q)$ in the same way.
Compute a total order $\mathcal{L}$ on $L(P)$ x $L(Q)$, the
pairs of subtree p, q, where $p \subset P, q \subset Q$.
For each $(p, q) \in \mathcal{L}$ **DO**
Compute $MAST(p, q)$
end-do
Compute $MAST(P, Q)$
end-do
end of algorithm

2.4 Running time analysis

The initialization (computing all subtrees t and $L(t)$ for all subtrees) involves $O(n)$ unions, and hence costs us $O(n\alpha(n))$. Computing $MAST(p, q)$ costs us a single find if one (or both) of p or q is a singleton; else it costs us two additions and one comparison. Since there are $O(n^2)$ pairs (p, q) of subtrees, this costs us $O(n^2)$ finds, additions, and comparisons, for a total of $O(n^2\alpha(n^2))$. The final computation of selecting the max among $O(n^2)$ values involves $O(n^2)$ additions, and $O(n^2)$ comparisons. All in all, a total cost of $O(n^2\alpha(n^2))$.

2.5 Proof of Correctness

Theorem 1 *The algorithm correctly determines the size of the maximum agreement subtree for two binary trees on n labelled leaves.*

Proof: The proof is by induction on n. If $n = 1$, the proof is trivial: either the trees are identical, or they are disjoint, and the algorithm handles each case correctly. So assume true for all trees on fewer than n leaves.

Let P and Q be two trees on n leaves each, and let T be the maximum agreement subtree of P and Q. T must contain at least three leaves, trivially, no matter how small n is. Let e_T be an edge in T, creating a bipartition on the label set $S|L(T)$ into two parts, S_1 and S_2, with subtrees T^1 and T^2. Since P and Q both agree on T, each of P and Q must contain edges creating the same bipartition on $S|L(T)$; 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. Since P and Q agree with T, we can say (without loss of generality) that $S_i \subset L(p^i) \cap L(q^i)$, $i = 1, 2$. The algorithm will compute the maximum agreement subtrees of p^i and q^j for each i, j, and thus be able to determine that the maximum is obtained at p^1, q^1 and p^2, q^2, so that the maximum agreement subtree is obtained by using the leaves in T.

One point to note is that p^i and q^j are not truly binary trees, since they have each a single node of degree two; however, supressing these degree two nodes creates binary trees which we will call P^1, P^2, Q^1, and Q^2. It is then clear to see that the algorithm would correctly handle the determination of the maximum agreement subtree of the various combinations of P^i and Q^j, and a quick check of the algorithm reveals that this translates into the correct handling of these rooted binary trees. Thus, the algorithm will correctly determine the maximum agreement subtrees of the various combinations of the p^i and q^j, and thus discover the way that T was constructed. ∎

3 Finding the Maximum Agreement Subtree of Arbitrary Trees

The only modification we need to make here is that each subtree (as defined above) may consist of more than two subtrees, so that the computation of $MAST(p, q)$ for p and q subtrees of P and Q respectively, will involve solving a maximum matching problem on a bipartite graph. That is, if p is a subtree of P, and q a subtree of Q, and the subtrees of p and q are $p^1, p^2, \ldots, p^k$ and $q^1, q^2, \ldots, q^r$, respectively, then we have computed $MAST(p^i, q^j)$ for each $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, r$. We can therefore weight the complete bipartite graph $K_{k,r}$ by $w(i, j) = MAST(p^i, q^j)$, and compute the maximum matching in this bipartite graph. This costs us $O((|p| + |q|)^{2.5})$ [3], where $|p|$ equals the degree of the root of p, which in turn equals the number of subtrees of p involved in the computation.

Since there are $O(n)$ edges, there are $O(n)$ subtrees, and hence $O(n^2)$ $MAST$ computations to perform. The worst case occurs when each of the trees are stars, and thus each subtree (other than the single node subtrees) has $n-1$ sub-subtrees. For this case, the $MAST$ computations incur a cost of $O(n^{4.5}\alpha(n^2))$.

However, for trees P and Q where the maximum degree of nodes are bounded by k, this algorithm has running time bounded by $O(n^2\alpha(n^2))$, as in the case for binary trees.

4 Open Problems

Can a better running time be achieved for this problem? Note that improving the running time on this algorithm probably is as hard as improving the bipartite matching problem.

References

[1] G. Chartrand, F. Saba, H. Zou, *Greatest common subgraphs of graphs*, Casopis Pest. Math, 10 (1985) 87-91.

[2] C.R. Finden and A.D. Gordon, *Obtaining Common Pruned Trees*, Journal of Classification, 2 (1985) 255-176.

[3] Hopcroft, J. E. and Karp, R.M. , *An $O(n^{5/2})$ algorithm for maximum matching in bipartite graphs*, SIAM J. of Comput. 2 (1973) pp. 225-231

[4] E. Kubicka, G. Kubicki, and F.R. McMorris, *An algorithm to find agreement subtrees*, to appear, Journal of Classification, 1992.

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

[6] G. Kubicki, *Greatest common subgraph index of graphs*, Congressus Numerantium 76 (1990) 101-113.

[7] D. Swofford, *When are phylogeny estimates from molecular and morphological data incongruent?*, in **Phylogenetic Analysis of DNA Sequences**, eds. M.M. Miyamoto and J. Cracraft, New York (1991), Oxford University Press.

[8] Robert E. Tarjan, *Efficiency of a good but not linear set union algorithm*, JACM 22(2):215-225, 1975.

[9] E.O.Wilson, *A Consistency Test for Phylogenies Based upon Contemporaneous Species*, Systematic Zoology, 14, pp. 214-220.