Journal of Classification 12:101-112 (1995)

An Efficient Algorithm for Supertrees

Mariana Constantinescu
Université de Moncton

David Sankoff
Université de Montréeal

Abstract: Given k rooted binary trees $A_1,A_2,..,A_k$, with labeled leaves, we generate C, a unique system of lineage constraints on common ancestors. We then present an algorithm for constructing the set of rooted binary trees B, compatible with all of $A_1,A_2,..,A_k$. The running time to obtain one such supertree is $O(k^2n^2)$, where n is the number of distinct leaves in all of the trees $A_1,A_2,..,A_k$.

Keywords: Tree compatibility; Constraints on trees; Supertrees; Consensus trees.

1. Introduction

The evolutionary history of n species can be described by a rooted tree with labeled leaves in which the internal nodes represent hypothetical ancestral species and the leaves represent the given species. Various methods have been proposed in the literature to reconstruct the evolutionary tree (phylogeny) from data on n given species. Many of these methods require the generation and evaluation of all possible trees on n labeled leaves. This becomes unfeasible for large n because there are $1.3.5...(2n-3) = (2n-2)!/(2^{n-1}(n-1)!)$, Cayley (1881), possible unordered binary trees with n labeled leaves. In some contexts however it is possible to restrict the set of trees to be examined so that the problem becomes tractable for relatively

Authors' Addresses: Mariana Constantinescu, Département d'Informatique, Université de Moncton, Moncton (Nouveau Brunswick) E1A 3E9, and David Sankoff, Centre de Recherches Mathématiques, Université de Montréal, C. P., Succ. A, Montréal (Quebec) H3C 3J7, Canada.

large n. For example, it could be that the n species are already partially classified by means of k rooted binary trees $A_1, A_2, .., A_k$, such that the leaf set of each A_i is a subset of the n species. This situation is the one we study in this paper. The problem then becomes one of generating the rooted binary trees with n leaves *compatible* with all $A_1, A_2, \ldots, A_k$, if such *supertrees* exist. Sankoff, Cedergren and McKay (1982), Gray, Sankoff and Cedergren (1989), Constantinescu and Sankoff (1986) and Gordon (1986) discuss various ways this problem arises. The first, as mentioned above, is simply as a device for getting around the combinatorial explosion in the number of trees. "The analysis of large data sets could proceed by division into overlapping subsets which are classified separately and then recombined to provide a single classification" (Gordon, 1986, pg. 335). The second is a way of ensuring that certain known or well-established evolutionary relationships are incorporated unchanged (i.e., as a constraint) into the phylogeny. A third context would arise in combining phylogenies proposed by different experts or found in studies on somewhat different sets of organisms. Gordon (1986) presented an algorithm for building a "strict consensus supertree" of two rooted binary trees, A_1 with n_1 leaves and A_2 with n_2 leaves. The running time of this algorithm is $O(p^3)$, where $p = \max(n_1, n_2)$.

For a related problem, Aho, Sagiv, Szymanski, and Ullman (1981) have given an $O(n^2 \log n)$ algorithm for constructing (if possible) a single rooted tree from a set of n constraints each constraint involving four leaves. A special case of their problem concerns constraints on three leaves, in which case their algorithm is $O(n^2)$. In the present paper, we make use of the results of Aho et al. to solve the supertree problem. We generate C, a system of constraints (on three leaves), based on the k given rooted binary trees. Those trees are defined on (possibly overlapping) subsets of a set of n leaves. Second, we present an algorithm for constructing the binary tree or trees T, if they exist, on the n leaves, such that each T satisfies the system C. Third, we demonstrate that each such T is *compatible* with all k given rooted binary trees. Our algorithm has running time $O(k^2 n^2)$.

2. Definitions

In this paper "tree" denotes a rooted tree with labeled leaves, where each internal node has at least two children. We say that a tree is "binary" if each internal node has exactly two children. Though we will be dealing with unordered trees, where the only relation among nodes is hierarchical, we temporarily impose an (arbitrary) distinction between the left child and the right child of an internal node only in order to have an unambiguous way, within one of our procedures, to represent a set of constraints. Our results all pertain to unordered trees.

The lowest common ancestor of two leaves x and y, denoted (x,y), is an ancestor node a of both x and y such that no proper descendant of a is also an ancestor of both x and y. (This terminology implies an "upside down" conception of trees: the root at the top and the leaves at the bottom.) It suffices to treat binary trees having at least three leaves. Aho et al. (1981) write $(i,j) < (k,l)$ to indicate that the lowest common ancestor of leaves i and j is a proper descendant of the lowest common ancestor of leaves k and l. Let x be an internal node of a binary tree. We say that $x^l\,(x^r)$ is *the leftmost (rightmost) leaf of* x, if $x^l\,(x^r)$ is a leaf, x is an ancestor of $x^l\,(x^r)$ and the path between x and $x^l\,(x^r)$, denoted $\{x_1,x_2,..,x_k\}$, has $x_1 = x$, $x_k = x^l\,(x_k = x^r)$ and for each $s = 2, \ldots, k$, x_s is the left (right) child of x_{s-1}. Property 1 follows from tree definitions.

Property 1. *If x is an internal node of a binary tree B, then $x = (x^l,x^r)$.*

3. System of lineage constraints defined by a binary tree

We generate for each edge xy of a binary tree B, where x and y are internal nodes, the constraint $(y^l,y^r) < (x^l,x^r)$, where y is a child of x. Obviously, if y is the left child of x, then $y^l = x^l$, and if y is the right child of x, then $y^r = x^r$. Thus, we generate for B a set of constraints, on three leaves only, of the form $(i,j) < (i,l)$ or $(i,j) < (k,j)$, where the leaves i,j,k, or i,j,l are distinct.[1] We denote this set of constraints by C_B.

Example 1. Suppose we are given the binary tree shown in Figure 1. Its internal nodes are a, b, c, d, e, r. We generate $(2,5) < (1,5)$ for ab, $(4,5) < (2,5)$ for bd, $(2,3) < (2,5)$ for bc, $(1,5) < (1,7)$ for ar, $(6,7) < (1,7)$ for re.

4. The algorithm

The recursive algorithm SUPERB$(F,C,B^1,B^2,..,B^p)$ is based on the algorithm BUILD(S,C) presented by Aho et al. (1981). BUILD(S,C) recursively builds a tree T satisfying a set of constraints C on a non empty leaf set S. It returns the null tree if no such tree exists. The algorithm is of a divisive type: one first determines a partition of a set of leaves S; the classes of this

1. De Soete et al.(1987) express constraints on tree topology in terms of statements about triples of objects, but they incorporate this information into a penalty function.

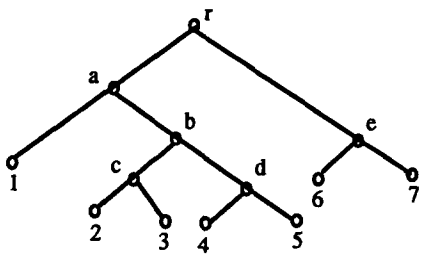

Figure 1. A binary tree *B*.

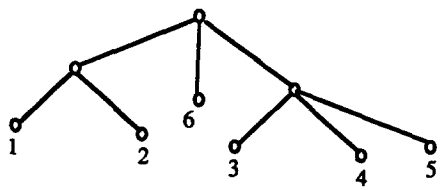

Figure 2. The tree *T*.

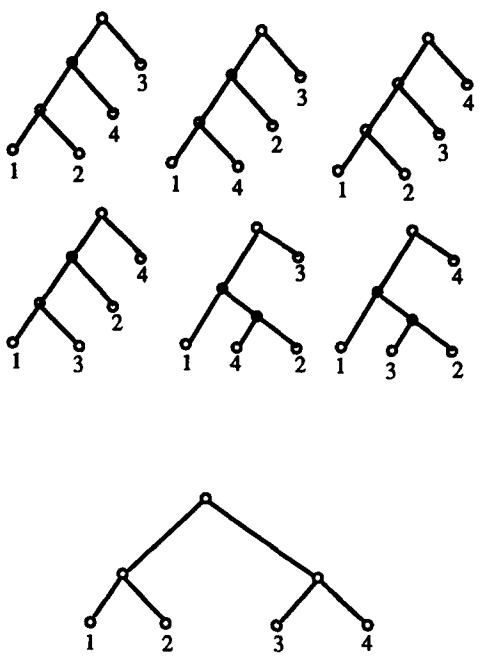

Figure 3. The binary trees considered in example 2.

partition correspond to the children of the root. Then the set of leaves of each of these children is partitioned in the same way, and so on.

Let x be an internal node, X the set of leaves that are descendants of x, CX a set of constraints on X of form $(i,j) < (k,l)$, $X \ni i,j,k,l$, and $\pi_x = \{S_1, \ldots, S_r\}$, a partition of X where each S_i contains the descendants of one of the children of x. We call each S_m of π_x a block. We say that π_x satisfies the constraint $(i,j) < (k,l)$ if i,j belong to the same block, and k,l belong to two distinct blocks. π_x satisfies the set of constraints CX if it satisfies at least one constraint of CX and for the others all four leaves are in the same block.

To calculate π_x from X and CX the following procedure (which we call compute π_x) starts with $\pi_x^0 = \{S_1, \ldots, S_u\}$, where each S_p, $1 \le p \le u$, consists of a single leaf of X. Then for some constraint of CX, $(i,j) < (k,l)$, the two blocks S_i and S_j are replaced by the block $S_i \cup S_j$ in forming π_x^1. In this way, the partition $\pi_x = \pi_x^t = \{S_1, \ldots, S_r\}$, $t < u$, is obtained by successive fusions of the blocks containing i and the blocks containing j, for all constraints $(i,j) < (k,l)$. If $r = 1$, then CX is not consistent with a tree and the algorithm stops (it returns the null tree). Otherwise, for each S_i of π_x the algorithm determines CS_i, the corresponding set of constraints, $CS_i = \{(i,j) < (k,l) \mid CX \ni (i,j) < (k,l)$ and $S_i \ni i,j,k,l\}$. S_i represents the leaves descending from a child of x and CS_i the subset of CX pertinent to these leaves. We apply the above procedure to the root, to each of its children, represented by S_i and CS_i, and so on until we are left with empty constraint sets only.

Example 2. Suppose $C = \{\{(1,2) < (1,4),(3,4) < (1,4)\}, (3,5) < (1,4)\}$ and $S = \{1,2,3,4,5,6\}$. BUILD(S,C) constructs $\pi_r = \{\{1,2\}, \{3,4,5\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$. The corresponding tree T is presented in Figure 2.

Though such trees are of undoubted interest in some contexts, we recall our initial motivation for building binary trees. There are 6 binary trees (presented in Figure 3a) on the leaf set S and satisfying C. There is no direct way of proceeding from the tree in Figure 2 to obtain the six trees in Figure 3a, and that excludes the one in Figure 3b not satisfying C, without reconsidering C itself.

The algorithm SUPERB$(S,C,B^1,B^2,..,B^p)$, presented in the Appendix, is designed to output the binary trees $B^1,B^2,..,B^p$, where $p \ge 1$, directly. All these binary trees have the same leaf set S and each satisfies C. The main idea is to determine, for each internal node x, from $\pi_x = \{S_1, \ldots, S_r\}$ all distinct pairs of sets $\{F_1,F_2\}$, denoted π^x, such that: $F_1 = \cup S_i, I \ni i$, $F_2 = \cup S_j, J \ni j$, $I \cup J = \{1,2,..,r\}$, $I \cap J = \varnothing$, for each π^x there is at least a constraint $(i,j) < (k,l)$ of CX having i,j in a set and k,l in distinct sets, and for

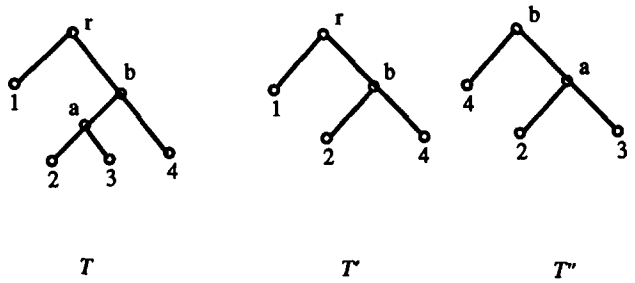

Figure 4. *T´* is obtained from *T* by the contraction of the edge *a3Fr, and T´´ is obtained from T* by the contraction of the edge *1r*.

each constraint $(i,j) < (k,l)$ of CX either i,j,k,l are in the same set of π^x or k,l belong to distinct sets of π^x. Each $\{F_1,F_2\}$ is a bipartition, or partition into two subsets, of X. The following property is clear.

Property 2. *There are* $2^{r-1} - 1$ *distinct bipartitions* π^x *of* $\pi_x = \{S_1, \ldots, S_r\}$.

■

Two rooted binary trees A and B are *equivalent*, denoted $A \approx B$, if they have the same leaf set and there is a bijection j from the nodes of A to the nodes of B such that for each leaf z of A, $\phi(z)$ is a leaf of B having the same label as z in A, the image of the root of A by ϕ is the root of B and for each xy edge of A, $\phi(x)\phi(y)$ is an edge of B. A binary tree B is *compatible* with a binary tree A if B has a binary subtree B' (possibly B itself), such that either $B' \approx A$, or B' can be transformed to a tree equivalent to A through a series of edge contractions. Let xy, xz, xt be three edges of a binary tree T, where y is a leaf of T and x an internal node distinct from the root. The *contraction* of the edge xy transforms T in a binary tree T' by replacing edges xy, xt, xz by the edge zt and by removing nodes x and y. If x is the root and y,z are its children, the contraction of the edge xy transforms T into a binary tree T'' which has the root z, by removing the edges xy, xz and the nodes y, x.

Example 3.

Property 3. *The binary tree B is compatible with the binary tree A iff B satisfies* C_A.

Proof. Suppose B is compatible with A. Then there is a subtree B' of B, such that either $B' \approx A$, or B' can be transformed to a tree B'' equivalent to A through a series of edge contractions. Suppose $B' \approx A$. Let ϕ be the bijection between A and B'. Consider a constraint of C_A, say $(i,j) < (i,l)$. Then, $(i,j),(i,l)$ are two internal nodes of A and the first is the child of the second. Consequently, $(\phi(i), \phi(j)) = (i,j)$, $(\phi(i), \phi(l)) = (i,l)$, $(i,j),(i,l)$ are internal

nodes of B', and $(i,j)(i,l)$ is an edge of B' with (i,j) the child of (i,l). Thus, B' satisfies the constraint $(i,j) < (i,l)$. If there is a B'', $B'' \approx A$, obtained from B' through a series of edge contractions, then B'' satisfies C_A and thus B' satisfies C_A.

Suppose B satisfies C_A. We denote by L_B the leaf set of B. Let B' be the minimal (having the minimal leaf set) subtree of B which satisfies C_A. Let B^* be the binary tree obtained from B' by the contraction of all edges xy which have one endpoint in $L_{B'} - L_A$. Clearly, B^* satisfies C_A. We denote by V_A, V_{B^*} the node sets of A, B^* and by r_A, r_{B^*} their roots. The function $\phi : V_A \rightarrow V_{B^*}$ where $\phi(x) = x$ if x is a leaf and $\phi(x) = (\phi(i),\phi(j))$ if $x = (i,j)$, is a bijection from A to B and obviously $\phi(r_A) = r_{B^*}$. Consequently, B is compatible with A. ■

Property 4. *If there exists a binary tree on the leaf set S which satisfies the constraint set C, then SUPERB$(S,C,B^1,B^2,..,B^p)$ returns at least one such tree.*

Proof. Suppose there is a binary tree on the leaf set S which satisfies the constraint set C. Aho et al. (1981) demonstrate that BUILD(S,C) returns one nonnull tree. If there is such a tree, according to Property 2, at least one binary tree is returned by SUPERB. ■

Property 5. *If SUPERB$(S,C,B^1,B^2,..,B^p)$ returns a binary tree B, then B satisfies the constraint set C.*

Proof. Let $(i,j) < (i,l)$ be a constraint of C. Suppose that SUPERB$(S,C,B^1,B^2,..,B^p)$ returns a binary tree B. Let $\pi^x = \{F_1,F_2\}$ be the bipartition calculated for the root of B. If i,j belong to one set of π_x and l belongs to the other, then B respects the constraint. If i,j,l belong to the same set, say F_1, then there is a recursive call of SUPERB$(F_1,C_1,B'^1,B'^2,..,B'^q)$ with $(i,j) < (i,l)$ belonging to C_1. At each recursive call at least one constraint is satisfied. Consequently, there is a recursive call of SUPERB, say SUPERB $(F',C',B^1,B^2,..,B^t)$, such that i, j, l are no longer all in the same set of the corresponding bipartition of F', but i, j are (Property 2). Thus, B respects the constraint $(i,j) < (i,l)$. ■

5. Binary tree B compatible with the binary trees $A_1,A_2,\ldots,A_k$.

Suppose there are the binary trees $A_1,A_2,\ldots,A_k$.

Step 1. Generate for each binary tree A_s, $s = 1,\ldots,k$, a set C_s, of lineage constraints of the form $(i,j) < (i,l)$ or $(i,j) < (k,j)$ on L_{As}. Note that the left-right convention on the children of a node is used only within this step. It is

neither input nor output of the step.
Step 2. Compute $C = \cup C_i$, $S = \cup L_{Ai}$ where L_{Ai} is the leaf set of A_i, $i = 1,2,\ldots,k$.
Step 3. Execute SUPERB($S,C,B^1,..,B^p$) to construct p binary trees $B^1,B^2,..,B^p$.
Each B^t, $t = 1,2,\ldots,p$ is compatible with $A_1,A_2,\ldots,A_k$ (Properties 5 and 3). If $n = |F|$, then $|C|$ is $O(kn)$.
If $k = 2$, the binary trees $B^1,B^2,..,B^p$ are the same as the supertrees obtained by Gordon (1986).

Example 4.
Given the binary trees A_1, A_2, A_3, presented in Figure 5. We generate the constraints (1,2) < (1,4), (3,4) < (1,4) for A_1, (5,6) < (5,7), (4,7) < (5,7) for A_2, (3,4) < (3,7), (3,7) < (3,8) for A_3. We execute SUPERB ($S,C,B^1,B^2,\ldots,B^p$), where $S = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $C = \{(1,2) < (1,4), (3,4) < (1,4), (5,6) < (5,7), (4,7) < (5,7), (3,4) < (3,7), (3,7) < (3,8)\}$. The procedure determines the sets of leaves descending from the children of the root r (Figure 6). $S_1 = \{1,2\}$, $S_2 = \{3,4,7\}$, $S_3 = \{5,6\}$, $S_4 = \{8\}$. Consequently, $r = 4$ and $2^{r-1} - 1 = 7$. There are seven possible bipartitions, represented in the first two rows of Figure 7, each of them being a possible bifurcation for the root.

Continuing the execution for the last bipartition: $F_1 = \{8\}$, $F_2 = \{1,2,3,4,5,6,7,8\}$, the corresponding systems of constraints is $C_1 = \varnothing$, $C_2 = \{(1,2) < (1,4), (3,4) < (1,4), (5,6) < (5,7), (4,7) < (5,7), (3,4) < (3,7)\}$. Let $x1$, $x2$ be the children of the root. Then one of these children, say $x1$, is the leaf 8. We obtain: $S_{1'} = \{1,2\}$, $S_{2'} = \{3,4,7\}$, $S_{3'} = \{5,6\}$, presented in Figure 8. Then $r = 3$, $2^{r-1} - 1 = 3$. The three possible bipartitions π^{x2} are presented in Figure 9.

Continuing the execution of the procedure for $F_{1'} = \{1,2,5,6\}$, $F_{2'} = \{3,4,7\}$, the corresponding systems of constraints is $C_{1'} = \varnothing$, $C_{2'} = \{(3,4) < (3,7)\}$. We denote by $x3$, $x4$ the children of $x2$. Let $x3$ be the root of the binary subtree obtained for the leaf set {1, 2, 5, 6}. There are 15 distinct binary trees on leaves 1, 2, 5, 6.

We obtain: $S''_1 = \{3,4\}$, $S''_2 = \{7\}$, presented in Figure 10. Consequently, $F''_1 = S''_1$, $F''_2 = S''_2$. We denote by $x5$, $x6$ the children of $x4$. Let $x5$ be the leaf 7 and $x6$ be the root of the binary subtree with leaves 3 and 4. Figure 11 presents two of the twenty one binary trees obtained for the last bipartition in Figure 7.

In the last row of the table presented in Figure 7, for each bipartition π_x we have written the number of binary trees obtained with SUPERB. It produces 173 distinct binary trees compatible with A_1, A_2, A_3.

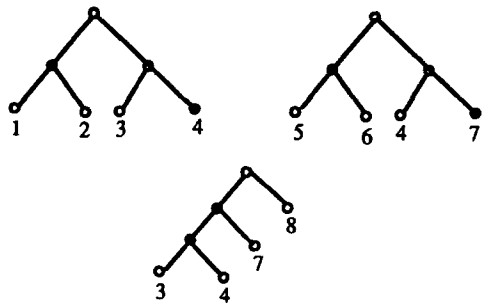

Figure 5. The binary trees A_1, A_2, A_3.

	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}
$(1,2) < (1,4)$,	{1, 2}		{3}	{4}	{5}	{6}	{7}	{8}
$(3,4) < (1,4)$,	{1, 2}		{3, 4}		{5}	{6}	{7}	{8}
$(5,6) < (5,7)$,	{1, 2}		{3, 4}		{5, 6}		{7}	{8}
$(4,7) < (5,7)$,	{1, 2}		{3, 4, 7}		{5, 6}		{8}	
$(3,4) < (3,7)$,	{1, 2}		{3, 4, 7}		{5, 6}		{8}	
$(3,7) < (3,8)$	{1, 2}		{3, 4, 7}		{5, 6}		{8}	

Figure 6. Calculation of π_r.

F_1	1, 2, 3, 4, 7	1, 2, 5, 6	1, 2, 8	1, 2	3, 4, 7	5, 6	8
F_2	5, 6, 8	3, 4, 7, 8	3, 4, 5, 6, 7	3, 4, 5, 6, 7, 8	1, 2, 5, 6, 8	1, 2, 3, 4, 7, 8	1, 2, 3, 4, 5, 6, 7
Nbr. of binary trees	15	15	3	5	105	9	21

Figure 7. Possible bipartitions for the root of the binary trees compatible with A_1, A_2, A_3.

	{1}	{2}	{3}	{4}	{5}	{6}	{7}
$(1,2) < (1,4)$,	{1, 2}		{3}	{4}	{5}	{6}	{7}
$(3,4) < (1,4)$,	{1, 2}		{3, 4}		{5}	{6}	{7}
$(5,6) < (5,7)$,	{1, 2}		{3, 4}		{5, 6}		{7}
$(4,7) < (5,7)$,	{1, 2}		{3, 4, 7}		{5, 6}		
$(3,4) < (3,7)$,	{1, 2}		{3, 4, 7}		{5, 6}		

Figure 8. Calculation of π_{x2}.

F_1'	1, 2, 3, 4, 7	1, 2, 5, 6	1, 2
F_2'	5, 6	3, 4, 7	3, 4, 5, 6, 7

Figure 9. Calculation of π^{x2}.

		{3}	{4} {7}
$(3,4) < (3,7)$,		{3, 4}	{7}

Figure 10. Calculation of π_{x4}.

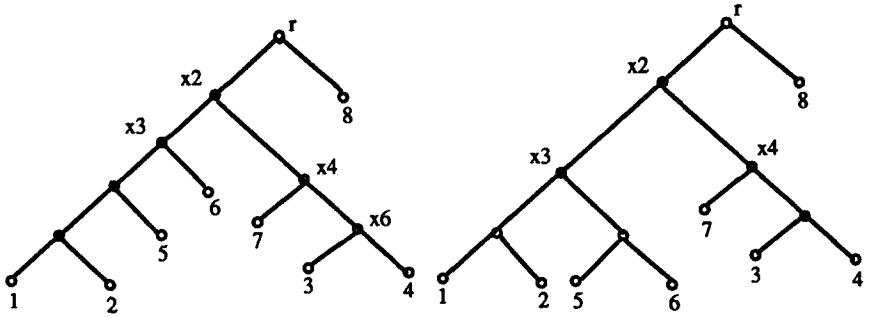

Figure 11. Binary trees compatible with A_1, A_2, A_3.

6. Time complexity for the construction of one binary tree using SUPERB.

Given a leaf set S, $|S| > 3$, and a set of constraints C, the running time to obtain the null tree or to construct one binary tree, if such a tree exists, using SUPERB(S,C,B^1,B^2, . . . ,B^p) is no more that the running time needed by BUILD(S,C) to construct the tree T. Aho et al (1981) demonstrate that if all constraints are of the form $(i,j) < (i,l)$, $(i,j) < (k,j)$, then the running time of BUILD is $O(m^2)$, where m is the number of constraints. Consequently, because there is a constraint for each edge of $A_1, A_2, .., A_k$, $k \geq 1$, and there are at most $k(2n-2)$ edges in all these binary trees, the running time needed by SUPERB, to construct one binary tree B, compatible with k given trees, is at most $O(k^2n^2)$, where $n = |S|$.

7. Discussion.

In this paper we have presented an algorithm that will construct, whenever possible, a rooted binary tree to satisfy m given constraints on common ancestors of a leaf set. We have also shown how to generate a set of constraints for k given binary trees. Thus the above algorithm can be used to find rooted binary trees compatible with these k rooted binary trees. This method

is more general than that of Gordon (1986), since ours applies for $k \geq 2$, and explicitly returns all binary supertrees, not just one. At the same time it represents a dramatic speed-up in the construction of a single supertree, from a cubic to a quadratic algorithm.

If the supertree is not required to be binary, less resolved supertrees may be found by taking the strict consensus of several or all of the supertrees generated by our algorithm. Or we may directly obtain the most highly reduced supertree possible by using the constraints we have derived for $A_1,A_2,..,A_k$, on three leaves only, as input to BUILD(S,C). For the case where $A_1,A_2,..,A_k$ are not binary trees, we are currently extending the constraint generation in Section 3 above for trees with nodes of higher degree. If $A_1,A_2,..,A_k$ are binary unrooted trees, we are developing a method to build the unrooted supertree compatible with them.

Appendix

```
procedure SUPERB (S,C,B^1,B^2,..,B^p)
begin
  if C = ∅ then return B^1,B^2,..,B^p, p = 1.3.5...(2n − 3),n ≥ 3,
distinct binary rooted trees having the leaf set S, where | S | = n
  else
    COMPUTE π_x = S_1,S_2, . . . ,S_r, r ≥ 1,
  if r > 1 then
     p = 0
     for s = 1 to 2^(r−1) − 1 do
       compute a distinct π_s^x = {F_1,F_2}
       if | F_1 | ≤ 2 then
         q = 1
         B_s'^1 is the tree on the leaf set F_1
       else
         C_1 = {(i,j) < (k,l) in C | i,j,k,l are in F_1roman}
         SUPERB(F_1,C_1,B_s'^1,B_s'^2,..,B_s'^q)
       end (if)
       if | F_2 | ≤ 2 then
         v = 1
         B_s''^1 is the tree on the leaf set F_2
       else
         C_2 = {(i,j) < (k,l) in C | i,j,k,l are in F_2}
         SUPERB(F_2,C_2,B_s''^1,B_s''^2,..,B_s''^v)
       end (if)
       for t = 1 to q do
         if B_s'^t is not the null tree then
           for u = 1 to v do
             if B_s''^u is not the null tree then
               p = p + 1
               B^p is the binary rooted tree having the root r^p and
               the subtrees B_s'^t and B_s''^u
```

```
            which roots are the children of r^p.
          end (if)
        end (for)
      end (if)
    end (for)
  end (for)
else
  p = 1
  B^1 = the null tree
end (if)
end (if)
end.
```

References

AHO, A., SAGI, Y., SZYMANSKI, T. G., and ULLMAN, J. D. (1981), "Inferring a Tree from Lowest Common Ancestors with an Application to the Optimization of Relational Expressions," *Journal of Computing, 3*, 405-421.

CAYLEY, A. (1881), "On the Analytical Forms Called Trees," American *Mathematical Journal, 4*, 266-268.

CONSTANTINESCU, M., and SANKOFF, D. (1986), "Tree Enumeration Modulo a Consensus," *Journal of Classification, 3*, 349-356.

DE SOETE, G., CARROLL, J. D., and DESARBO, W. S. (1987), "Least Squares Algorithms for Constructing Constrained Ultrametric and Additive Tree Representation of Symmetric Proximity Data," *Journal of Classification, 4*, 155-173.

GORDON, A. D. (1986), "Consensus Supertrees: The Synthesis of Rooted Trees Containing Overlapping Sets of Labeled Leaves," *Journal of Classification, 3*, 335-348.

GRAY, M. W., SANKOFF, D., and CEDERGREN, R. J. (1984), "On the Evolutionary Descent of Organisma and Organelles: A Global Phylogeny Based on a Highly Conserved Structural Core in Small Subunit Ribosomal RNA," *Nucleic Acids Research, 12*, 5837-5852.

SANKOFF, D., CEDERGREN, R. J., and MCKAY, W. (1982), "A Strategy for Sequence Phylogeny Research," *Nucleic Acid Research, 10*, 421-431.