

Constructing Rooted Supertrees Using Distances

Stephen J. Willson
Department of Mathematics
Iowa State University
Ames, IA 50011 USA
email: swillson@iastate.edu

May 26, 2004

Abstract: Suppose that a family of rooted phylogenetic trees T_i with different sets X_i of leaves is given. A supertree for the family is a single rooted tree T whose leaf set is the union of all the X_i , such that the branching information in T corresponds to the branching information in all the trees T_i . This paper proposes a polynomial-time method BUILD-WITH-DISTANCES that makes essential use of distance information provided by the trees T_i to construct a rooted tree S_0 . When a supertree also containing the distance information exists, then S_0 is a supertree. The supertree S_0 often shows increased resolution over the trees found by methods that utilize only the topology of the input trees. When no supertree exists because the input trees are incompatible, several variants of the method are described which still produce trees with provable properties.

1 Introduction

The large data bases of DNA for various taxa that are now available have greatly expanded opportunities to build phylogenetic trees. In such trees, leaves correspond to extant taxa, while internal vertices correspond to inferred ancestral species. Typically, such trees are rooted at the common ancestor of all the leaves, so that it is obvious which vertices are ancestral to any given vertex.

Many methods are used to infer phylogenetic trees. See Swofford *et al.* (1996) for an overview. Recently there has been interest in “supertree” methods. See for example the recent book Janowitz *et al.* (2003) and the review paper Bininda-Emonds *et al.* (2002). Such methods take as input a collection \mathcal{F} of rooted trees that have already been constructed with various overlapping sets of taxa. The methods seek to combine these disparate trees into a single tree T called the “supertree” that exhibits the relationships among the entire collection of taxa.

Supertree methods have many theoretical advantages. The initial trees can be based on different kinds of data—for example, DNA of different genes, morphology, behavior, DNA-DNA hybridization. They can be obtained by dif-

ferent methodologies—for example, maximum parsimony, maximum likelihood, neighbor-joining, or some combination. The initial trees often have been selected from competing trees by weighting different factors using professional judgment. In these situations it is likely that no common data set exists for all the species. Even if there were a common data set, methods such as maximum likelihood would often not be computationally tractable because the dataset could be too large.

A number of supertree methods have been proposed. The fast procedure BUILD described in Aho *et al.* (1981) finds a supertree from input rooted trees provided a supertree exists. Fast generalizations when no supertree exists are provided in Semple and Steel (2000), Page (2003), and Bryant *et al.* (2004), and this paper suggests yet another generalization.

A supertree method receiving much recent attention is “matrix representation with parsimony” (MRP). This method was suggested by Baum (1992) and Ragan (1992); for more information see Sanderson *et al.* (1998) and Bininda-Emonds and Sanderson (2001). Suppose we are given a collection \mathcal{F} of rooted trees in which each leaf is labelled by a taxon. The topological information about each tree in \mathcal{F} is encoded into a new character matrix, typically with many entries corresponding to “missing data.” The computationally intensive method of maximum parsimony is then applied to this new character matrix. If a supertree exists, it corresponds to a maximum parsimony tree. Typically, whether or not a supertree exists, there are many different maximum parsimony trees. Some consensus is then presented as a summary of the nature of the maximum parsimony trees. The method is easy to use and there are no alignment difficulties. The software package RADCON (Thorley and Page 2000) is available to perform the reduction, and then packages such as PAUP* (Swofford 2002) may be used to solve the problem by a heuristic search.

The bulk of current supertree methods make use only of the topology of each tree in \mathcal{F} . Yet in fact, many methods of building phylogenetic trees create trees in which each edge has a numerical branch length. Typically the branch length gives an estimate of the rate of DNA mutation along the edge. These branch lengths have often been estimated, even if they have not been published along with the trees. A rooted tree in which each edge has a branch length will be called an “additive tree.”

This paper proposes a supertree method that makes essential use of the branch lengths when \mathcal{F} consists of additive trees. The key observation is that these distances carry a great deal of information about phylogeny which is ignored by purely topological methods that deal only with the branching structure. When purely topological methods such as MRP are utilized, there are typically a great many different topologies compatible with the data. In contrast, when the input trees include lengths of the branches, the number of topologies is often more highly constrained.

More specifically, this paper proposes a polynomial-time algorithm BUILD-WITH-DISTANCES which uses input distance information to construct a supertree when an additive supertree exists. This algorithm generalizes the BUILD algorithm of Aho *et al.* (1981). When an additive supertree exists, BUILD-

WITH-DISTANCES finds a supertree (although not necessarily an additive supertree), here called the “null threshold tree.” Moreover, when the algorithm outputs a tree, the null threshold tree will be a supertree. It is proved in this situation that the null threshold tree contains at least as much resolution as the supertree obtained by BUILD. When the algorithm fails to produce a supertree, a polynomial time modification is also proposed which outputs a tree called the “minimal threshold tree” with interesting properties.

The following example illustrates the increased resolution obtained by using distances in the input trees. Suppose the leaf set is $\{1, 2, 3, 4, 5, 6, 7, 8\}$ and the set \mathcal{F} of input trees is as in Figure 1. Branch lengths are indicated in parentheses, and branch lengths unspecified are equal to 1. Then BUILD-WITH-DISTANCES constructs the additive rooted tree shown in Figure 2, which is in fact the unique additive supertree for \mathcal{F} .

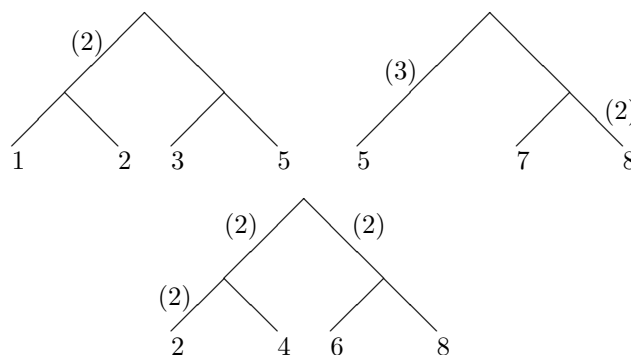


Figure 1: A family \mathcal{F} of input additive rooted trees. Branch lengths are indicated in parentheses. Branch lengths unspecified are 1.

Figure 3 shows a different collection \mathcal{F}' of additive rooted trees, each with the same topology as the corresponding tree in \mathcal{F} , while Figure 4 shows the unique additive rooted supertree constructed from \mathcal{F}' by BUILD-WITH-DISTANCES.

Note the many topological differences between the supertrees in Figures 2 and 4. By contrast, BUILD finds the supertree in which the only nontrivial clusters are $\{1, 2, 3\}$, $\{4, 5\}$, and $\{6, 7, 8\}$, as in Figure 5. The method of MRP using PAUP* finds 142 maximum parsimony trees of which the strict consensus is the star tree, the Adams consensus agrees with Figure 5, and the nontrivial clusters in the 50% majority consensus tree are $\{1, 2, 4\}$, $\{3, 5, 6, 7, 8\}$, and $\{3, 5\}$. None of these contains the high resolution obtained by BUILD-WITH-DISTANCES.

This paper is organized as follows: Section 2 introduces the basic notions of additive rooted trees. Each branch in these trees has a branch length. The

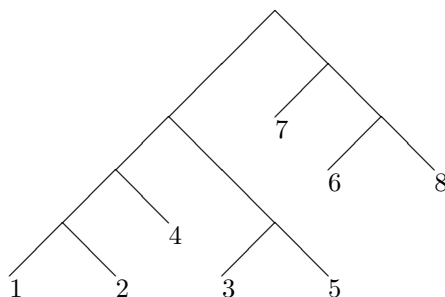


Figure 2: The additive rooted supertree constructed from Figure 1 by BUILD-WITH-DISTANCES. All edges have unit length.

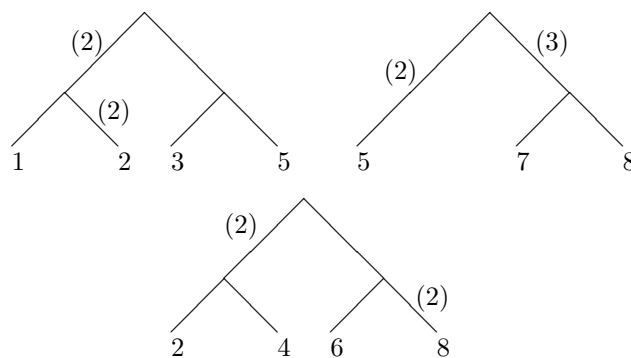


Figure 3: A family \mathcal{F}' of additive rooted trees with the same topology as in Figure 1 but different branch lengths. Unspecified branch lengths are 1.

basic construct is that of the distance function $\lambda(x, y)$ which tells the length in the tree of the path from x to the most recent common ancestor of x and y . This construct is useful since it includes the root information; moreover, when x and y are leaves, $\lambda(x, y)$ is unchanged between a tree and a supertree. While the ordinary distance $d(x, y)$ telling the length of the path between vertices x and y would also be invariant when x and y are leaves, $d(x, y)$ does not contain information about the rooting of trees.

Section 3 gives existence and uniqueness theorems which tell when a proposed distance function λ actually determines a tree uniquely. Section 4 defines

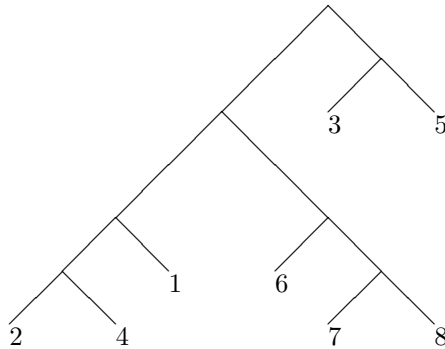


Figure 4: The additive rooted supertree constructed from Figure 3 by BUILD-WITH-DISTANCES. All branches have unit length.

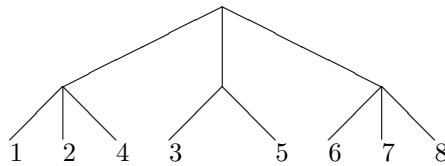


Figure 5: The supertree constructed by BUILD from the input trees in Figures 1 or 3.

the notions of a “supertree” and an “additive supertree” and describes the algorithm BUILD-WITH-DISTANCES to find a supertree, called the “null threshold tree.”

In most practical situations, no supertree in a strict sense will exist. Section 5 gives a modification of BUILD-WITH-DISTANCES which yields a “minimal threshold tree,” given a measure s of the “support.” When no supertree exists, the minimal threshold tree arguably contains the most reliable phylogenetic information in the input trees. The idea is that groupings with larger numerical support from the distance information should take precedence over groupings with smaller support. Section 6 contains a comparison of the minimal threshold trees with different support functions. There is concluding discussion in Section 7.

Other methods exist for building supertrees by making use of branch length information. Consider the “average consensus” method of Lapointe and Cucumel (1997) or Lapointe *et al.* (2003). It seeks the additive tree that is closest to the input trees using a least-squares criterion. It is not, however, a polynomial

time algorithm since it uses a Fitch and Margoliash (1967) method to find the tree which best fits the matrix in the sense of least-squares. Implementations such as FITCH in PHYLIP (Felsenstein 1993) make use of heuristic search, and the computer requirements can grow exponentially with the number of taxa.

The use of trees with distances rather than topological trees is not without cost. Certainly there is additional analysis required to find good distances on the input trees. Many published trees include only topology and omit distances. Some standard banks of trees such as Tree of Life (<http://tolweb.org/tree/phylogeny.html>) include topology but not distances.

Moreover, Lapointe and Levasseur (2004), when studying simulations using the average consensus method, found that when branch lengths in the true tree were very heterogeneous, then the use of branch lengths yielded reduced accuracy compared with the use only of topological information. Theorem 4.9 shows, however, that when an additive supertree exists, the tree found by BUILD-WITH-DISTANCES has at least as much resolution as the tree constructed by BUILD using only topology. Studies need to be performed to tell whether analogous results are true in the case of incompatible data.

For the current methods to work, when several different input trees have distances, the distances should have comparable meanings. For example, one expects that if both taxa x and y are in two different trees, their distances in the two trees should be the same. This constraint poses difficulties in combining trees obtained by substantially different kinds of data. Some uniform notions of distance, such as substitution rate per base pair, might differ on different genes from the same taxa and would need reconciliation. Even distances obtained by different methods from the same DNA might need to be reconciled before the methods in this paper are utilized. If the current methods are to be practical, careful study must be made of ways to reconcile different distances before applying BUILD-WITH-DISTANCES.

While the use of additive rooted trees may increase resolution over purely topological methods, it also may increase the likelihood that the data are incompatible, so that no additive supertree exists. When a supertree exists but there is no additive supertree, the algorithm may still yield a supertree with more resolution than the supertree constructed by BUILD. For example, if the branch lengths in Figure 1 or 3 are slightly perturbed, it is likely that there is no longer an additive supertree; yet the trees in Figures 2 or 4 (without the branch lengths) would still be constructed by BUILD-WITH-DISTANCES.

A common assumption for rooted trees with distances is that the distances form an ultrametric. See Johnson (1967), Hartigan (1967), Hubert (1972, 1973), Colonius and Schulze (1981), or Semple and Steel (2003). For an ultrametric, all paths from the root to a leaf have the same length, and $\lambda(x, y) = \lambda(y, x)$ for all x and y . This paper does not make the ultrametric assumption, and no assumption is made about the relation between $\lambda(x, y)$ and $\lambda(y, x)$. There are advantages to avoiding this assumption. If T is an unrooted tree in which each edge has a distance and subsequently the tree is rooted at r to become the rooted tree T^r , then T^r becomes an additive rooted tree in the sense of this paper but need not become an ultrametric tree. Biologically, an ultrametric

corresponds to the existence of a uniform scale for evolution, so that perhaps $\lambda(x, y)$ is the number of years that have elapsed since the most recent common ancestor z of x and y . This computation of the number of years is problematic since it often requires the assumption of a molecular clock. In this paper, $\lambda(x, y)$ should be considered rather the expected number of substitutions in the DNA between z and x , while $\lambda(y, x)$ would be the expected number of substitutions in the DNA between z and y . Since it is more likely that mutation rates are observed than years, this interpretation of $\lambda(x, y)$ is closer to the data.

Another related method is that of Bryant *et al.* (2004) in which BUILD is generalized by the use of ranked divergence dates. A polynomial time algorithm is described in which the input includes not only rooted trees but also statements about whether certain divergence events preceded or followed other such events and time estimates about such divergence events. Since dates are utilized, the input is similar to the input of an ultrametric. Yet another related method, described in Dress and Erdős (2003), characterizes when edge-weighted quartet trees have an edge-weighted supertree.

Software for BUILD-WITH-DISTANCES is available from the author's web-site (<http://www.public.iastate.edu/~swillson/software.html>).

I wish to thank the Institute for Mathematics and Its Applications for its support while I wrote part of this paper. I also thank the referees for helpful suggestions to improve the paper.

2 Additive rooted trees

Let X be a finite nonempty set. A *rooted tree* (T, X) with leaf set X is a collection T of subsets U of X such that

- (1) $X \in T$.
- (2) (nesting): If U and V are in T , either $U \subseteq V$ or $V \subseteq U$ or $U \cap V = \emptyset$.
- (3) For each $x \in X$, the singleton $\{x\}$ lies in T .
- (4) The empty set is not a member of T .

Each member of T will be called a *cluster* in T . The clusters X and the singleton clusters $\{x\}$ are called *trivial*; all other clusters are *nontrivial*. The *star* tree contains all the trivial clusters but no other clusters.

If T is a rooted tree, $A \in T$, $B \in T$, $A \subset B$, $A \neq B$, and there is no $C \in T$ with $A \subset C \subset B$, $A \neq C$, $B \neq C$, then we say that A is a *child* of B and B is a *parent* of A . If A_1, A_2, \dots, A_k is a sequence of members of T such that each A_{i+1} is a child of A_i , then A_k is a *descendent* of A_1 and A_1 is an *ancestor* of A_k . By using the sequence with only one term, we see that A_1 is both a descendent and an ancestor of itself. We say A_k is a *proper descendent* of A_1 and A_1 is a *proper ancestor* of A_k if A_k is a descendent of A_1 and $A_1 \neq A_k$. We call X the *root*. Each member of T other than the root has a unique parent. Each singleton set $\{x\}$ is a *leaf*. Each member of T that is not a leaf has at least two children. A rooted tree T is *binary* if each cluster that is not a leaf has exactly two children.

The *graph* of T has vertex set the members of T ; there is an edge $\{A, B\}$

iff B is a child of A or A is a child of B . A *tree* is a connected graph with no cycles. It is easy to see that the graph of a rooted tree is always a tree. If X has n members, then any rooted tree with leaf set X will have at most $2n - 1$ clusters, and a binary rooted tree will have exactly $2n - 1$ clusters.

Let T be a rooted tree with leafset X . If U is a nonempty subset of X , then $\text{lca}(U)$ or $\text{lca}(U; T)$ denotes the *lowest common ancestor* or *most recent common ancestor* of all members of U ; more precisely it is the smallest member of T that contains U . If x and y are members of X write $\text{lca}(x, y)$ for $\text{lca}(\{x, y\})$. If V is a member of T that is not a leaf, let A and B be two of its children. Then for any $x \in A$ and any $y \in B$ it follows that $V = \text{lca}(x, y)$.

Since we will deal with complicated sums, I introduce the following notation: Suppose S is a set and $f : S \rightarrow \mathfrak{R}$ is a function. Write $[f(s) : s \in S]$ for the multiset which has one value of the number $f(s)$ for each $s \in S$. Thus, $[f(s) : s \in S]$ consists of $|S|$ values, even if some of the values are the same. Write $\sum[f(s) : s \in S]$ for the sum of the members of the multiset.

Suppose that whenever C is a child of U in a rooted tree T , then there is associated with the edge $\{C, U\}$ of its graph a nonnegative *branch length* $w(C, U)$. Define the *lca distance function* $\lambda : X \times X \rightarrow R$ by

$$\lambda(x, y) = \sum[w(U, V) : U \text{ is a child of } V, x \in U, y \notin U].$$

Note that $\lambda(x, y)$ is the length of the path in the graph of T from $\{x\}$ to $\text{lca}(x, y)$, and $\lambda(x, x) = 0$. When desired, the value $\lambda(x, y)$ may also be denoted $\lambda(x, y; T)$. An *additive rooted tree* is a tuple (T, X, w, λ) where T is a rooted tree with leaf set X , w is a branch length function, and λ is the corresponding lca distance function. The additive rooted tree is *positive* if for each C and parent U , $w(C, U) > 0$. The function λ can be extended to a function $\lambda : T \times T \rightarrow R$ by defining

$$\lambda(A, B) = \sum[w(U, V) : U \text{ is a child of } V, A \text{ is a descendent of } U, \text{lca}(A, B) \text{ is an ancestor of } V].$$

Note $\lambda(A, B)$ is the distance in the graph of T from A to $\text{lca}(A, B)$, and $\lambda(A, A) = 0$. In general, if A and B are in T , the function $d(A, B) = \lambda(A, B) + \lambda(B, A)$ is an additive distance function on the graph of T .

Theorem 2.1. *Let (T, X, w, λ) be a positive additive rooted tree. Then*

- (i) $\lambda(x, x) = 0$ for all $x \in X$.
- (ii) $\lambda(x, y) > 0$ for all $x \in X$ and $y \in X$ if $x \neq y$.
- (iii) $\lambda(x, y) < \lambda(x, z)$ iff $\text{lca}(x, y)$ is a proper descendent of $\text{lca}(x, z)$.
- (iv) If $\lambda(x, y) = \lambda(x, z)$ then $\text{lca}(x, y) = \text{lca}(x, z)$.
- (v) If $\lambda(w, x) < \lambda(w, z)$ and $\lambda(w, y) < \lambda(w, z)$ then $\lambda(x, y) < \lambda(x, z)$.
- (vi) If $\lambda(x, y) < \lambda(x, z)$ then $\lambda(y, x) < \lambda(y, z)$ and $\lambda(x, z) - \lambda(x, y) = \lambda(y, z) - \lambda(y, x)$.
- (vii) Suppose $U \in T$, C is a child of U in T , and E is a child of C in T . Let $e \in E$, $c \in C - E$, $u \in U - C$. Then $w(C, U) = \lambda(e, u) - \lambda(e, c)$.

(viii) Suppose $U \in T$ and C is a child of U in T which is a singleton, $C = \{c\}$. Let $u \in U - C$. Then $w(C, U) = \lambda(c, u)$.

Proof. (i) is immediate, and (ii) follows because each branch length is positive. For (iii), note that x lies in both $\text{lca}(x, y)$ and $\text{lca}(x, z)$, whence by nesting one must be a descendent of the other. If $\text{lca}(x, y)$ is a proper descendent of $\text{lca}(x, z)$ then the path from $\{x\}$ to $\text{lca}(x, z)$ must pass through $\text{lca}(x, y)$; since the branch lengths are positive, $\lambda(x, y) < \lambda(x, z)$. For (iv), note that since x is in both $\text{lca}(x, y)$ and $\text{lca}(x, z)$ one must be a descendent of the other; by (iii) if one were a proper descendent, we could not have $\lambda(x, y) = \lambda(x, z)$.

Under the hypotheses of (v), $\text{lca}(w, x)$ and $\text{lca}(w, y)$ are proper descendents of $\text{lca}(w, z)$ by (iii). By nesting, one of $\text{lca}(w, x)$ and $\text{lca}(w, y)$ must contain the other since both contain w ; hence one of them is $\text{lca}(w, x, y)$. Since $\text{lca}(x, y) \subseteq \text{lca}(w, x, y)$, $\text{lca}(x, y)$ is a descendent of either $\text{lca}(w, x)$ or $\text{lca}(w, y)$. Now $\text{lca}(x, z)$ cannot be a descendent of $\text{lca}(x, y)$ since then z would lie in $\text{lca}(x, y)$ hence in $\text{lca}(w, x, y)$ hence in one of $\text{lca}(w, x)$ or $\text{lca}(w, y)$, contradicting that they are proper descendents of $\text{lca}(w, z)$. Hence $\text{lca}(x, y)$ must be a proper descendent of $\text{lca}(x, z)$ since by nesting one must contain the other. From (iii), $\lambda(x, y) < \lambda(x, z)$, proving (v).

For (vi) note by (iii) that $\text{lca}(x, z)$ is a proper ancestor of $\text{lca}(x, y)$, and the path from $\text{lca}(x, y)$ to $\text{lca}(x, z)$ has positive length $\lambda(x, z) - \lambda(x, y)$. By nesting, $\text{lca}(x, y) \subset \text{lca}(x, z)$, so $\text{lca}(x, y, z) = \text{lca}(x, z)$ and then $\text{lca}(y, z) = \text{lca}(x, y, z)$. Since $\text{lca}(y, x) \subset \text{lca}(x, z) = \text{lca}(y, z)$, it follows that $\text{lca}(y, x)$ is a proper descendent of $\text{lca}(y, z)$ so $\lambda(y, x) < \lambda(y, z)$ by (iii). But $\lambda(y, z) - \lambda(y, x)$ is the length of the path from $\text{lca}(y, x)$ to $\text{lca}(y, z)$ which is the same as the path from $\text{lca}(y, x)$ to $\text{lca}(x, z)$ since $\text{lca}(x, z) = \text{lca}(y, z)$. Hence $\lambda(y, z) - \lambda(y, x) = \lambda(x, z) - \lambda(x, y)$.

For (vii), $C = \text{lca}(e, c)$ and $U = \text{lca}(e, u)$. The path from e to U has length $\lambda(e, u)$ while the path from e to C has length $\lambda(e, c)$. Since the paths differ only on the edge between C and U , it follows that $w(C, U) = \lambda(e, u) - \lambda(e, c)$.

For (viii), note $U = \text{lca}(c, u)$ and the path from c to U has length $\lambda(c, u)$ but consists only of the single edge between C and U . \square

Note that (vii) and (viii) imply that given T and the numbers $\lambda(x, y)$ for all x and y in X , the branch lengths of T are determined. Property (v) is related to Theorem 1 of Colonius and Schulze (1981).

3 Existence and uniqueness theorems

In this section I show that given a function λ , there exists at most one additive rooted tree T for which λ is the lca distance function. We also see sufficient conditions on λ such that there exists an additive rooted tree for which λ is the lca distance function.

Given an additive rooted tree (T, X, w, λ) , for any leaf a and for any real number $r \geq 0$, let $C(a, r) = \{x \in X : \lambda(a, x) \leq r\}$. Call $C(a, r)$ the *cluster of points with lca distance from a at most r* .

Lemma 3.1. *Let (T, X, w, λ) be an additive rooted tree, $a \in X$, $r \geq 0$. Then $C(a, r)$ is a member of T .*

Proof. a lies in $C(a, r)$ since $\lambda(a, a) = 0 \leq r$, so $C(a, r)$ is nonempty and is clearly a subset of X . I show that $C(a, r) = \text{lca}(C(a, r))$, whence the result follows since the latter is in T . It suffices to show that, if $x \in C(a, r)$, $y \in C(a, r)$, and $z \in \text{lca}(x, y)$, then $z \in C(a, r)$ as well. So assume $\lambda(a, x) \leq r$, $\lambda(a, y) \leq r$, and $z \in \text{lca}(x, y)$; we show $\lambda(a, z) \leq r$.

By nesting, either $\text{lca}(a, x) \subseteq \text{lca}(a, y)$ or $\text{lca}(a, x) \supseteq \text{lca}(a, y)$ since both contain a . Without loss of generality, assume $\text{lca}(a, x) \subseteq \text{lca}(a, y)$. Hence $\text{lca}(a, x, y) = \text{lca}(a, y)$ and $\text{lca}(a, z, x, y) = \text{lca}(a, y)$. Now it follows that $\lambda(a, z) \leq \lambda(a, \text{lca}(a, z, x, y))$ [since the path from $\{a\}$ to $\text{lca}(a, z)$ is a portion of the path from $\{a\}$ to $\text{lca}(a, z, x, y)$] $= \lambda(a, \text{lca}(a, y)) = \lambda(a, y) \leq r$. \square

Lemma 3.2. $\lambda(a, C(a, r)) \leq r$.

Proof. If $C(a, r) = \{a\}$ then the result is immediate. Otherwise, let A and B be distinct children of $C(a, r)$ with $a \in A$, and let $b \in B$. Then $C(a, r) = \text{lca}(a, b)$. Now $\lambda(a, C(a, r)) = \lambda(a, b) \leq r$ since $b \in C(a, r)$. \square

Lemma 3.3. *Assume all branch lengths of T are positive, U is a cluster of T , and $a \in U$. Then there exists $r \geq 0$ such that $U = C(a, r)$.*

Proof. Let $r = \max\{\lambda(a, u) : u \in U\}$. Then each member of U lies in $C(a, r)$. If $U \neq C(a, r)$, then $C(a, r)$ is in T (by Lemma 3.1) and U is contained in a child W of $C(a, r)$. Since $w(W, C(a, r)) > 0$ it follows that $\lambda(a, W) < \lambda(a, C(a, r)) \leq r$. Now since U is contained in W , for each $u \in U$ we have $\lambda(a, u) \leq \lambda(a, W) < r$, which contradicts the definition of r . Hence $U = C(a, r)$. \square

Theorem 3.4. (*Uniqueness*). *Suppose that X is a finite set and numbers $\lambda(x, y) \geq 0$ are given for all a and b in X . Then there exists at most one positive additive rooted tree for which λ is the lca distance function.*

Proof. If there exists such a tree, then by Lemma 3.3 each member has the form $C(a, r)$ for $a \in X$ and $r \geq 0$, while by Lemma 3.1 each set $C(a, r)$ is in T . Hence the clusters of T are determined by the given numbers. \square

Suppose that we are given the distances $\lambda(x, y)$ for all x and y , and suppose they come from a positive additive rooted tree T with leaf set X . It follows that we can easily reconstruct the rooted tree T as follows: Each set $C(a, r) = \{x \in X : \lambda(a, x) \leq r\}$ is a cluster in T ; and every cluster in T has this form. For each $a \in X$ (with n choices) we sort the n values $r = \lambda(a, x)$ to form all the clusters $C(a, r)$. Clearly this method requires only polynomial time.

We now turn to questions of existence.

Theorem 3.5. *Let X be a finite set, and let $\lambda : X \times X \rightarrow \mathfrak{R}$. There exists a rooted tree T and a positive branch length function w such that (T, X, w, λ) is a positive additive rooted tree if and only if the following conditions all hold:*

- (i) $\lambda(x, x) = 0$ for all $x \in X$.
- (ii) $\lambda(x, y) > 0$ for all x and y in X if $x \neq y$.
- (iii) If $\lambda(w, x) < \lambda(w, z)$ and $\lambda(w, y) < \lambda(w, z)$ then $\lambda(x, y) < \lambda(x, z)$.
- (iv) If $\lambda(x, y) < \lambda(x, z)$ then $\lambda(y, x) < \lambda(y, z)$ and $\lambda(x, z) - \lambda(x, y) = \lambda(y, z) - \lambda(y, x)$.

The necessity of the conditions follows from Theorem 2.1. The proof of their sufficiency will consist of Lemmas 3.6 through 3.9.

For each $a \in X$ and each $r \geq 0$ let $C(a, r) = \{x \in X : \lambda(a, x) \leq r\}$. Let T be the collection of all $C(a, r)$ with $a \in X$ and $r \geq 0$.

Note that if $U = C(a, r)$ and $r_0 = \max\{\lambda(a, x) : \lambda(a, x) \leq r\}$, then $U = C(a, r_0)$ and r_0 is the minimal value of t such that $C(a, r) = C(a, t)$; moreover, there exists $x \in U$ such that $\lambda(a, x) = r_0$. We call r_0 the *minimal radius* of U with respect to a .

Lemma 3.6. *T is a rooted tree with leaf set X .*

Proof. The empty set is not a member of T since $C(a, r)$ contains a by (i). For each $a \in X$, $C(a, 0) = \{a\}$ by (i) and (ii). If $r > \lambda(x, y)$ for all x and y in X , then $C(a, r) = X$ for all $a \in X$. There remains to show the nesting property. Suppose $y \in C(a, r) \cap C(b, s)$. In order to show that either $C(a, r) \subseteq C(b, s)$ or $C(b, s) \subseteq C(a, r)$, we assume that $z \in C(a, r)$ but $z \notin C(b, s)$, and we prove that $C(b, s) \subseteq C(a, r)$. So assume $x \in C(b, s)$; we must show $x \in C(a, r)$.

Now $\lambda(a, y) \leq r$, $\lambda(b, y) \leq s$, $\lambda(a, z) \leq r$, $\lambda(b, z) > s$, $\lambda(b, x) \leq s$, and we wish to show $\lambda(a, x) \leq r$. If not, we may assume $\lambda(a, x) > r$.

Now $\lambda(a, y) \leq r < \lambda(a, x)$ and $\lambda(a, z) \leq r < \lambda(a, x)$. By (iii), $\lambda(y, z) < \lambda(y, x)$. Similarly, $\lambda(b, y) \leq s < \lambda(b, z)$ and $\lambda(b, x) \leq s < \lambda(b, z)$, whence by (iii) $\lambda(y, x) < \lambda(y, z)$. This contradiction proves the nesting property and hence that T is a rooted tree. \square

Lemma 3.7. *Each cluster in T containing a has the form $C(a, t)$ for some $t \geq 0$.*

Proof. Let U be a cluster in T which contains a . There exists $b \in X$ and $s \geq 0$ such that $U = C(b, s)$. If $b = a$ then we are done. Assume $b \neq a$ and let r be the minimal radius of U with respect to b . Then $U = C(b, r)$ and moreover, there exists $u \in U$ such that $\lambda(b, u) = r$. Since $a \in U$, it follows $\lambda(b, a) \leq r = \lambda(b, u)$.

Let $t = \max\{\lambda(a, x) : x \in U\}$. Then $C(a, t) \in T$ and $C(a, t)$ contains U . I claim that $C(a, t) = U$. If not, there exists $z \in C(a, t)$ but $z \notin U$. Hence we know $\lambda(a, z) \leq t$, $\lambda(b, x) > r$. By the definition of t choose $v \in U$ such that $\lambda(a, v) = t$; since $v \in U$, $\lambda(b, v) \leq r$.

Now $\lambda(b, v) \leq r < \lambda(b, z)$ and $\lambda(b, a) \leq r < \lambda(b, z)$, whence by (iii) it follows that $t = \lambda(a, v) < \lambda(a, z)$. This contradicts that $\lambda(a, z) \leq t$, proving the lemma. \square

Suppose $U = C(a, r) \in T$. We may assume r is the minimal radius of U with respect to a , so there exists z such that $\lambda(a, z) = r$. If $r = 0$, then $C(a, 0) = \{a\}$ by (i) and (ii), so U is a leaf. Otherwise U has at least two children E and F . We show how to define the branch length $w(E, U)$. There are two cases:

- (1) If E is a leaf $\{e\}$, select $f \in F$. Define $w(E, U) := \lambda(e, f)$.
- (2) If E is not a leaf, it has children E_1, E_2, \dots, E_k with $k \geq 2$. Choose e and e' in distinct children of E , and choose $f \in F$. Define $w(E, U) := \lambda(e, f) - \lambda(e, e')$.

Lemma 3.8. *These definitions are independent of the choices, yielding a well-defined value for $w(E, U)$. Moreover, $w(E, U) > 0$.*

Proof. In case (1), choose a minimal value $s > 0$ such that $C(e, s) \neq \{e\}$. Note that $C(e, s) \in T$. Then $C(a, r) = C(e, s)$ since $C(a, r)$ is the parent of $\{e\}$. Hence $\lambda(e, x) = s$ for all $x \in C(e, s) = C(a, r)$, $x \neq e$, by minimality. In particular, for every child F of $C(a, r)$ other than E , for all $f \in F$, $\lambda(e, f) = s$. Hence in case (1), $w(E, U)$ is well-defined and is positive since $s > 0$.

In case (2), choose a child E_1 of E and choose $e \in E_1$. By Lemma 3.7, $U = C(e, s)$ where we may choose s to be the minimal radius of U with respect to e . Similarly choose t the minimal radius of E with respect to e and v the minimal radius of E_1 with respect to e . Thus $C(a, r) = C(e, s)$, $E = C(e, t)$, and $E_1 = C(e, v)$; clearly $v < t < s$ since E_1 is contained properly in E and E is contained properly in U . Then for each $x \in C(e, s) - E$ we have $\lambda(e, x) = s$ (otherwise $C(e, s)$ would not be a parent of E). In particular, $\lambda(e, f) = s$ for every choice of $f \in F$. Similarly, for each $x \in C(e, t) - E_1$ we have $\lambda(e, x) = t$, whence $\lambda(e, e') = t$ for every choice of $e' \in E - E_1$. Hence given $e \in E_1$, we see that $w(E, U) = s - t$ independent of the choices for e' and f . Moreover $w(E, U) > 0$ since $t < s$.

But by (iv), $\lambda(e, f) - \lambda(e, e') = \lambda(e', f) - \lambda(e', e)$. Hence for any e' , we have $\lambda(e', f) - \lambda(e', e)$ equals the same value obtained for $w(E, U)$. As above, the value of $w(E, U)$ is then independent of e and f , given e' . Hence $w(E, U)$ is independent of the choices of e, e' , and f . \square

Lemma 3.9. *The sum of the branch lengths in the path in T between $\{x\}$ and $\text{lca}(x, y)$ is $\lambda(x, y)$.*

Proof. If $x = y$, then $\text{lca}\{x, x\} = \{x\} = C(x, 0)$, so the length of the path is $0 = \lambda(x, x)$. If $x \neq y$, let $U_0 = \{x\}$ and $U = \text{lca}(x, y)$. Since U_0 is contained in U , there is in T a path $U_0, U_1, U_2, \dots, U_k = U$ where U_i is a child of U_{i+1} in T for $0 \leq i < k$. For each $i > 0$ select $u_i \in U_i - U_{i-1}$, with $u_0 = x$. Since $U_k = \text{lca}(x, y)$ and $x \in U_{k-1}$, we may take $u_k = y$. Then $w(U_0, U_1) = \lambda(u_0, u_1)$, $w(U_1, U_2) = \lambda(u_0, u_2) - \lambda(u_0, u_1)$, $w(U_2, U_3) = \lambda(u_0, u_3) - \lambda(u_0, u_2)$, and in general,

$$w(U_i, U_{i+1}) = \lambda(u_0, u_{i+1}) - \lambda(u_0, u_i).$$

Hence the length of the path in T is $\sum[w(U_i, U_{i+1}) : i = 0, \dots, k-1]$ which telescopes into $\lambda(u_0, u_k) = \lambda(x, y)$. \square

These lemmas complete the proof of Theorem 3.5.

Lemma 3.10. For any a and b in X , $\text{lca}(a, b; T) = C(a, \lambda(a, b))$.

Proof. Since $C(a, r) = \{x : \lambda(a, x) \leq r\}$ it follows that $b \in C(a, \lambda(a, b))$ while $b \notin C(a, t)$ for $t < \lambda(a, b)$. \square

Lemma 3.11. Let E be a child of U in T . Then

$$w(E, U) = \min\{\lambda(e, u) - \lambda(e, e') : e \in E, e' \in E, u \in U - E\}.$$

Proof. This is immediate if E is a leaf $\{e\}$. Otherwise assume E has a child E_1 and $e \in E_1$. Let $u \in U - E$ and $e' \in E - E_1$. From Lemma 3.8 we know that $w(E, U) = \lambda(e, u) - \lambda(e, e')$. It suffices to show that when $e'' \in E_1$, then $\lambda(e, u) - \lambda(e, e') < \lambda(e, u) - \lambda(e, e'')$. By algebra it suffices to show $\lambda(e, e'') < \lambda(e, e')$. But $\text{lca}(e, e') = E$ while $\text{lca}(e, e'')$ is contained in E_1 . By Lemma 3.9, $\lambda(e, e'')$ is the sum of the branch lengths in the path between $\{e\}$ and $\text{lca}(e, e'')$, and each of these edges lies in the path between $\{e\}$ and $\text{lca}(e, e')$. Since branch lengths are positive, we obtain $\lambda(e, e'') < \lambda(e, e')$. \square

4 Additive rooted supertrees

If T is a rooted tree with leaf set X and X' is a nonempty subset of X , then the *restriction* of T to X' , denoted $T|X'$, consists of the collection of sets $U \cap X'$ such that

- (i) $U \in T$, and
- (ii) $U \cap X'$ is nonempty.

Lemma 4.1. $T|X'$ is a rooted tree with leaf set X' .

Proof. $X \cap X' = X'$, so $X' \in T|X'$. If $x \in X'$, then $\{x\} \in T$, so $\{x\} \cap X' = \{x\} \in T|X'$. For the nesting property suppose U and V in T , $U \cap X'$ and $V \cap X'$ are nonempty, that $U \cap X'$ meets $V \cap X'$ (say in a point w), and $U \cap X'$ contains an element y not in $V \cap X'$. We must see that $U \cap X' \supseteq V \cap X'$. But $w \in U \cap V$ and $y \in U - V$, whence since T is a tree we have that $U \supseteq V$. Hence $U \cap X' \supseteq V \cap X'$. \square

If (T, X, w, λ) is an additive rooted tree, make $T|X'$ an additive rooted tree as follows: If A is a child of B in $T|X'$, let the branch length between A and B be $w(A, B; T|X') = \lambda(\text{lca}(A; T), \text{lca}(B; T); T)$. Thus $w(A, B; T|X')$ is the length of the path in T from $\text{lca}(A; T)$ to $\text{lca}(B; T)$. The next result shows that the values $\lambda(x, y)$ are unchanged under restriction. This fact underlies their importance in the reconstruction process. (While it might be tempting to utilize the collection of distances $d(x, y; T)$ where $d(x, y; T)$ is the length of the path in T from x to y , this notion is inadequate for rooted trees. For example, if r is the root of T and r' is the root of $T|X'$, then it need not be true that $d(x, r'; T|X') = d(x, r; T)$.)

Lemma 4.2. If x and y are in X' , then $\lambda(x, y; T|X') = \lambda(x, y; T)$.

Proof. $\lambda(x, y; T|X') = \sum[w(C, U; T|X') : C \text{ is a child of } U \text{ in } T|X', x \in C, y \notin C] = \sum[\lambda(\text{lca}(C; T), \text{lca}(U; T); T) : C \text{ is a child of } U \text{ in } T|X', x \in C, y \notin C]$. The relevant sets form a nested chain in T with smallest set $\{x\}$ and largest set $\text{lca}(x, y; T)$. Hence the sum is the length of the path in T from $\{x\}$ to $\text{lca}(x, y; T)$, which equals $\lambda(x, y; T)$. \square

A *rooted tree family* \mathcal{F} is a collection of rooted trees T_i for $i = 1, \dots, k$, where T_i is a rooted tree on the leaf set X_i . Let X contain all X_i (typically, X is the union of the X_i). A rooted tree T with leaf set X is a *supertree* for \mathcal{F} provided for $i = 1, \dots, k$ it is true that $T|X_i$ contains T_i . Note that it is possible that $T|X_i$ is more highly resolved (contains more members) than T_i . Equality is not required because in typical biological applications a lack of resolution is interpreted merely as inadequate information about the true resolution.

An *additive rooted tree family* \mathcal{F} is a collection of additive rooted trees $(T_i, X_i, w_i, \lambda_i)$ for $i = 1, \dots, k$. An additive rooted tree (T, X, w, λ) is an *additive rooted supertree* for \mathcal{F} provided T is a supertree for \mathcal{F} and moreover, whenever x and y are in X_i , then $\lambda(x, y) = \lambda_i(x, y)$. We will often merely write that T is an additive rooted supertree for \mathcal{F} . If in addition each branch length of T is positive, then (T, X, w, λ) is a *positive additive rooted supertree* for T .

Theorem 4.3. (*Uniqueness*) *Suppose that $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ is an additive rooted tree family. Let X be the union of the X_i . Assume that for each x and y in X there exists at least one i such that x and y are both in X_i . Then there exists at most one positive additive rooted supertree T for \mathcal{F} .*

Proof. We may assume that whenever x and y are in both X_i and X_j , then $\lambda_i(x, y) = \lambda_j(x, y)$. (If the condition fails, then clearly there does not exist an additive rooted supertree.) In this case, define $\lambda(x, y) = \lambda_i(x, y)$ when x and y are in X_i ; note that λ is well-defined. The result follows from Theorem 3.4. \square

Theorem 4.3 contrasts sharply with the situation when the trees in \mathcal{F} do not have branch lengths. In that situation, even if each pair x and y from X lies in X_i for some i , there may be many distinct supertrees.

The polynomial-time algorithm BUILD of Aho *et al.* (1981) has as input a collection \mathcal{F} of rooted trees. Its output is a particular rooted supertree if one exists, and it terminates without finding a tree if no supertree exists. Some interesting variants such as Semple and Steel (2000) or Page (2003) modify the procedure so as always to output a tree even if no supertree exists in a strict sense. This paper will describe another variant.

An *lca-domain* is a triple (X, D, λ) where X is a finite set, D is a subset of $X \times X$ containing (x, x) for all $x \in X$, and $\lambda : D \rightarrow \mathfrak{R}_{\geq}$ satisfies that $\lambda(x, x) = 0$ for all $x \in X$. Here \mathfrak{R}_{\geq} denotes the nonnegative real numbers.

As a paradigm example, let $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i) : i = 1, \dots, r\}$ be an additive rooted tree family. Let $X = \cup X_i$ and $D = \{(x, y) \in X \times X : \text{there exists } i \text{ with } x \text{ and } y \text{ in } X_i\}$. Define $\lambda : D \rightarrow \mathfrak{R}_{\geq}$ by setting $\lambda(x, y)$ to be the average of the values $\lambda_i(x, y)$ such that x and y are in X_i . Then (X, D, λ) is an lca-domain, called the *lca-domain from \mathcal{F}* . An especially interesting case occurs

if, whenever x and y are in both X_i and X_j , then $\lambda_i(x, y) = \lambda_j(x, y)$. In that case, $\lambda(x, y) = \lambda_i(x, y)$ when x and y are in X_i , and we say that the lca-domain from \mathcal{F} is *exact*. If \mathcal{F} is positive, then for $(x, y) \in D$, $x \neq y$, it follows that $\lambda(x, y) > 0$.

Suppose (X, D, λ) is an lca-domain and U is a nonempty subset of X . Define the *generalized Aho graph* $A(U)$ as follows: The vertex set consists of the members of U . If x and y are distinct members of U , there is an edge $\{x, y\}$ iff there exists $z \in U$ such that either

- (1) $(x, y) \in D$, $(x, z) \in D$, and $\lambda(x, y) < \lambda(x, z)$; or
- (2) $(y, x) \in D$, $(y, z) \in D$, and $\lambda(y, x) < \lambda(y, z)$.

Let C_1, C_2, \dots, C_k be the components of $A(U)$. If $k = 1$, U has no child. Otherwise, the k *children* of U will be the members of U (i.e., the vertices) in the components C_1, C_2, \dots, C_k and will also be denoted C_1, C_2, \dots, C_k respectively.

The graph $A(U)$ generalizes the graph in the BUILD algorithm of Aho *et al.* 1981. In BUILD there is an edge $\{x, y\}$ in the graph if for some $T_i \in \mathcal{F}$ there exists $z \in U \cap X_i$ such that T_i includes a cluster containing both x and y but not z . In this situation, $\lambda_i(x, y) < \lambda_i(x, z)$ in T_i ; if the lca-domain is exact then $A(U)$ contains an edge $\{x, y\}$ as well. On the other hand, the current construction is more general in that other edges may exist: Maybe x and y are in one input tree while x and z are in another and no input tree has all three; yet if $\lambda(x, y) < \lambda(x, z)$ then $A(U)$ still contains an edge $\{x, y\}$.

Algorithm 4.4. (*BUILD-WITH-DISTANCES*):

Input: An lca-domain (X, D, λ) such that $\lambda(x, y) > 0$ for $(x, y) \in D$, $x \neq y$.

(Typically, (X, D, λ) is the lca-domain from a positive additive tree family \mathcal{F} .)

Procedure: Form a set L of subsets of X as follows:

- (1) Initially, L contains only the cluster X which has not been checked off.
- (2) Repeat the following (a) and (b) as long as some cluster in L has not been checked off:
 - (a) If U is a singleton cluster in L that has not been checked off, then check U off.
 - (b) If U is a cluster in L that is not a singleton and has not been checked off, form the generalized Aho graph $A(U)$. For each component C of $A(U)$, if $C \neq U$, the set of vertices of C is placed into L as a new cluster which has not been checked off; and then U is checked off. (If $C = U$, then U is merely checked off.)
- (3) When all clusters in L have been checked off, output L .

Theorem 4.5. *The procedure BUILD-WITH-DISTANCES outputs in polynomial time a collection $S_0 = S_0(X, D, \lambda)$ of subsets of X satisfying:*

- (1) $X \in S_0$.
- (2) The empty set is not in S_0 .
- (3) (Nestedness) If U and V are in S_0 and $U \cap V$ is nonempty, then either U is contained in V or V is contained in U .
- (4) If $U \in S_0$, then each component of $A(U)$ is in S_0 .

Proof. For $U \in L$, the components of $A(U)$ can be found in polynomial time. Either $A(U)$ is connected and U has no children, or else the children of U form a partition of U . Hence it is clear that L always satisfies the analogues of (1), (2), and (3). If X contains n points, then L contains at most $2n - 1$ subsets. Hence the algorithm terminates in polynomial time. Let S_0 be the collection of members of L when the algorithm terminates. Clearly S_0 satisfies (4). \square

Corollary 4.6. *If S_0 contains all the singleton sets from X , then S_0 is a rooted tree.*

In general, call $S_0(X, D, \lambda)$ the *null threshold graph* for (X, D, λ) . If it contains all the singleton sets, call it the *null threshold tree*. Its definition will be generalized in Section 5 using the notion of “threshold.”

Suppose X contains n taxa. Then for each subset U , the edges of $A(U)$ can be found in time $O(n^3)$ and the components in time $O(n^2)$, so the children of U can be found in time $O(n^3)$. Since there are at most $n - 1$ clusters U that are not singleton sets, it follows that a naive implementation of the algorithm takes time $O(n^4)$.

Theorem 4.7. *Let $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ be a positive additive rooted tree family and (X, D, λ) be its lca-domain. Suppose there exists a positive additive rooted supertree T for \mathcal{F} . Then S_0 contains all the singleton sets, and S_0 is a supertree for \mathcal{F} .*

Proof. While a direct proof can be given at this point, the theorem follows as a special case of Theorem 5.3, to be proved later. \square

Theorem 4.8. *Let $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ be a positive additive tree family with exact lca-domain (X, D, λ) . Suppose that S_0 contains all the singleton sets. Then S_0 is a supertree for \mathcal{F} .*

Proof. Since S_0 contains all the singleton sets, it is a rooted tree by Corollary 4.6. For each i , $S_0|X_i$ is a rooted tree by Lemma 4.1. We show that $S_0|X_i$ contains T_i . Observe that $X \in S_0$, whence $X \cap X_i = X_i \in S_0|X_i$. It therefore suffices to show that, whenever $U \in T_i$ lies in $S_0|X_i$ and C is a child of U in T_i , then $C \in S_0|X_i$. If C is a singleton set, the result is immediate, so we may assume C is not a singleton. Since $U \in T_i$ lies in $S_0|X_i$, there exists $U' \in S_0$ satisfying $U = U' \cap X_i$; we may assume that U' is the minimal such cluster in S_0 . If U' were a singleton set, then U would also be a singleton set and would have no children. Hence U' is not a singleton and $A(U')$ is disconnected since S_0 contains all the singleton sets. If c and c' are distinct members of C and $u \in U - C$, then $(c, u) \in D$, $(c, c') \in D$, and $\lambda_i(c, c') < \lambda_i(c, u)$ by Theorem 2.1 since T_i is a positive additive rooted tree. By exactness, $\lambda(c, c') < \lambda(c, u)$. Hence in $A(U')$ there is an edge between c and c' . It follows that C is contained in a child W of U' in S_0 .

Since U' is minimal, it follows that U is not contained in W . Let Y denote the collection of members of S_0 that contain C but do not contain U ; it follows that $W \in Y$. Since Y is nonempty we may find a minimal member V of Y .

I claim that $V \cap X_i = C$. Note $V \cap X_i \supseteq C$ since $V \supseteq C$ and $C \subseteq X_i$. If there exists $v \in V \cap X_i - C$ then whenever c and c' are in C , it follows that $(c, v) \in D$, $(c, c') \in D$, and $\lambda_i(c, c') < \lambda_i(c, v)$ by Theorem 2.1. By exactness, $\lambda(c, c') < \lambda(c, v)$. Hence C is contained in a child of V in S_0 , contradicting that V is the minimal member of Y . It follows that $V \cap X_i = C$. \square

Given two rooted trees S and T with the same leaf set X , say S is *monotone* for T provided whenever $U \in S$ has a child $C \in S$ and $V \in T$ contains U , then there exists a child C' of V in T that contains C . The following result relates S_0 with the supertree W constructed by BUILD of Aho *et al.* (1981), and a general positive additive supertree T :

Theorem 4.9. *Let $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ be a positive additive tree family with lca-domain (X, D, λ) . Suppose there exists a positive additive rooted supertree (T, X, w, λ) for \mathcal{F} . Let W be the supertree constructed from $\{(T_i, X_i)\}$ by BUILD. Then*

- (a) S_0 is monotone for T .
- (b) W is monotone for S_0 .

Proof. (a) By Theorem 4.7, S_0 is a rooted tree. Suppose $U \in S_0$ has child $C \in S_0$, while the cluster $V \in T$ satisfies $V \supseteq U$. If $\{x, y\}$ is an edge of $A(U)$, then we may assume there exists $z \in U$ such that $\lambda(x, z) - \lambda(x, y) > 0$. By Lemma 3.1, T contains the cluster $Y = C(a, \lambda(x, y); T)$ containing x and y but not z . By nesting, Y is a proper descendent of V , whence Y is contained in a child C' of V . In particular, for each edge $\{x, y\}$ of $A(U)$, there is a child of V that contains both x and y . Since C is a component of $A(U)$ and the children of V are disjoint, it follows that there is a child of V that contains C .

(b) Suppose $U \in W$ has child $C \in W$, while the cluster $V \in S_0$ satisfies $V \supseteq U$. The child C is a component of Aho's graph A in which the vertices are the members of U and there is an edge $\{x, y\}$ iff there exists $z \in U$ and i such that X_i contains $\{x, y, z\}$ and T_i contains a cluster Y including x and y but not z . Suppose $\{x, y\}$ is an edge of A , with corresponding z and i . Since T_i is a positive additive rooted tree, $\lambda_i(x, z) > \lambda_i(x, y)$ by Theorem 2.1(iii). Since there exists an additive rooted supertree (T, X, w, λ) it follows that $\lambda(x, z) > \lambda(x, y)$. Hence $A(U)$ also has an edge $\{x, y\}$. It follows that each edge of A is an edge of $A(U)$, whence the component C of A must be contained in a component of $A(U)$. \square

If S_0 is the star tree, then it is monotone for any tree and little information has been gained. On the other hand, if S_0 is binary, then there is at most one positive additive rooted supertree, as shown in Corollary 4.10:

Corollary 4.10. *Let $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ be a positive additive tree family with lca-domain (X, D, λ) . Suppose there exists a positive additive rooted supertree (T, X, w, λ) for \mathcal{F} . If S_0 is binary, then $S_0 = T$.*

Proof. Note that S_0 is a rooted tree by Theorem 4.7. I first show that $S_0 \subseteq T$. Note that X is in both S_0 and T . Suppose U is in both S_0 and T ; I claim that each child of U in S_0 will also be in T . Since S_0 is binary, there are exactly two children C_1 and C_2 of U in S_0 . Since $U \in T$ and S_0 is monotone for T , there exists a child C'_1 of U in T that contains C_1 and a child C'_2 of U in T that contains C_2 . Since C_1 and C_2 form a partition of U , U can have no other children in T . If C'_1 contained C_1 properly, then C'_1 would contain an element of C_2 and hence would meet C'_2 , contradicting the nesting property of T . Hence $C'_1 = C_1$ and similarly $C'_2 = C_2$. Hence each child of U is in T . It follows that $S_0 \subseteq T$.

Since S_0 is binary, if X contains n members, then S_0 contains $2n - 1$ clusters, which is the maximal number of vertices in a rooted tree with n leaves. Since T is also a rooted tree with n leaves, T cannot have any more clusters. Hence $S_0 = T$. \square

If S is a rooted tree with leaf set X and $i \in X$, define the *height* $h(i; S)$ of i in S to be the number of edges on the path in S from the root X to the leaf i . The *total height* of S is $h(S) = \sum h(i; S)$. Clearly $h(S)$ is a measure of the amount of resolution in the tree S .

Theorem 4.11. *Let S and T be rooted trees with leaf set X , and suppose that S is monotone for T . Then for all $i \in X$, $h(i; S) \leq h(i; T)$.*

Proof. Let the vertices on the path from the root X to the leaf $\{i\}$ in S be

$$X = C_0 \supset C_1 \supset C_2 \supset \dots \supset C_k = \{i\}$$

so that $h(i; S) = k$. Let $C'_0 = X$ be the root of T . Since $C_0 \supseteq C'_0$ and S is monotone for T , it follows that in T , C'_0 has a child C'_1 that contains C_1 . Since C_1 is contained in C'_1 and C_2 is a child of C_1 in S , by monotonicity there is a child C'_2 of C'_1 in T that contains C_2 . Continuing in this manner, we see that for each j such that $0 \leq j < k$ there is a child C'_{j+1} of C'_j in T that contains C_{j+1} . Hence

$$X = C'_0 \supset C'_1 \supset C'_2 \supset \dots \supset C'_k$$

where each inclusion is strict since C'_{j+1} is a child of C'_j , and where C'_j contains C_j . If $C'_k = i$ then $h(i; T) = k = h(i; S)$. Otherwise, i must lie in a child of C'_k , so $h(i; T) > h(i; S)$. \square

Corollary 4.12. *Suppose S is monotone for T . Then $h(S) \leq h(T)$.*

Briefly, if S is monotone for T , Corollary 4.12 asserts that T has at least as much resolution as S , as measured by the total height. In particular, by Theorem 4.9, S_0 has at least as much resolution as the tree constructed by BUILD.

So far, only the rooted tree structure of $S_0(X, D, \lambda)$ has been defined. In order to find an additive rooted supertree, one needs to extend the procedure by defining a branch length for each edge. Suppose that in the algorithm, C is

a child of U in S_0 (so C is a component of $A(U)$). If there exist c and c' in C (possibly $c = c'$) and $u \in U - C$ such that (c, u) and (c, c') are in D , define

$$w(C, U) = \min\{\lambda(c, u) - \lambda(c, c') : c \in C, c' \in C, u \in U - C, (c, u) \in D, (c, c') \in D\};$$

otherwise set $w(C, U) = 0$. When S_0 is a rooted tree, these values make S_0 an additive rooted tree. The formula for $w(C, U)$ is suggested by Lemma 3.11.

It need not be true that $S_0(X, D, \lambda)$ with these branch lengths is an additive rooted supertree. The following result gives sufficient conditions under which S_0 is in fact a particular additive rooted supertree for \mathcal{F} . The sufficient conditions are quite strict, but many examples show that S_0 becomes an additive rooted supertree even when these conditions are not satisfied.

Proposition 4.13. *Let $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ be a positive additive tree family with lca-domain (X, D, λ) . Suppose there exists a positive additive rooted supertree (T, X, w', λ') for \mathcal{F} . Let S_0 be the null threshold supertree. Assume:*

(1) *For every $U \in T$ with child C that is a singleton $\{c\}$, there exists $u \in U - C$ with $(c, u) \in D$.*

(2) *For every $U \in T$ with child C that is not a singleton, there exists a child E of C , $u \in U - C$, $c \in C - E$, $e \in E$ such that (e, u) and (e, c) are in D .*

(3) *For every $U \in T$ with child C , for every proper subset C' of C , there exist $c' \in C'$, $c \in C - C'$, and $u \in U - C$ with (c', u) and (c', c) in D .*

If $U \in S_0$ and C is a child of U in S_0 , define

$$w(C, U; S_0) = \min\{\lambda(c, u) - \lambda(c, c') : c \in C, c' \in C, u \in U - C, (c, u) \in D, (c, c') \in D\}.$$

Define $\lambda'' : X \times X \rightarrow R$ by

$$\lambda''(x, y) = \sum[w(C, U; S_0) : x \in C, y \notin C, C \text{ a child of } U \text{ in } S_0].$$

Then $S_0 = T$ and $\lambda'' = \lambda'$, so S_0 is an additive rooted supertree for \mathcal{F} .

Proof. We first prove that $S_0 = T$. Suppose that $U \in T$ and $U \in S_0$. We show that each child of U in T forms a component of $A(U)$. Since the children of U form a partition of U , this will imply that the children of U in S_0 are the same as the children of U in T . Since X is in both T and S_0 , this will imply that $S_0 = T$.

If there is an edge between x and y in $A(U)$, then we may assume that there exists $z \in U$ such that $\lambda(x, z) > \lambda(x, y)$ with (x, z) and (x, y) in D . Since T is an additive rooted supertree, it follows that x and y are in the same child of U in T . Hence every child of U in S_0 is contained in a child of U in T . Suppose C is a child of U in S_0 , C' is the child of U in T that contains C , but C' contains C properly. By (3) select $u \in U - C'$, $c \in C$, $c' \in C' - C$ with (c, u) and (c, c') in D . Then $\lambda(c, c') < \lambda(c, u)$ by Theorem 2.1 since T is a positive additive rooted supertree and $\text{lca}(c, c')$ is contained in C' while $\text{lca}(c, u; T) = U$. Hence there is an edge between c and c' in $A(U)$, so $c' \in C$, a contradiction. This proves that $S_0 = T$.

There remains to show that the distances in S_0 are the same as those in T . Suppose first that C is a child of U which is a singleton $\{c\}$. Then $w(C, U; S_0) =$

$\min\{\lambda(c, u) : u \in U, u \neq c, (c, u) \in D\}$ since the set is nonempty by (1). But for $u \in U, u \neq c, \text{lca}(c, u; T) = U$ so $\lambda(c, u) = w(C, U; T)$ by Theorem 2.1.

Now suppose instead that C is a child of U which has a child E . Then $w(C, U; S_0) = \min\{\lambda(c, u) - \lambda(c, c') : c \in C, c' \in C, u \in U - C, (c, u) \in D, (c, c') \in D\}$. By (2) there exists a child E of $C, u \in U - C, c \in C - E, e \in E$ such that (e, u) and (e, c) are in D , whence $w(C, U; S_0) \leq \lambda(e, u) - \lambda(e, c)$. In T we have $w(C, U; T) = \lambda(e, u) - \lambda(e, c)$ by Theorem 2.1. Thus $w(C, U; S_0) \leq w(C, U; T)$. If $w(C, U; S_0) < w(C, U; T)$ there exist $c_0 \in C, c_1 \in C, u_0 \in U - C, (c_0, u_0) \in D$, and $(c_0, c_1) \in D$, with $\lambda(c_0, u_0) - \lambda(c_0, c_1) < w(C, U; T)$. But then $\text{lca}(c_0, c_1; T)$ is contained in C while $\text{lca}(c_0, u_0) = U$, whence $\lambda(c_0, u_0) - \lambda(c_0, c_1) \geq w(C, U; T)$. This proves $w(C, U; S_0) = w(C, U; T)$, so the branch lengths in S_0 agree with those in T . \square

Suppose there exists a positive additive rooted supertree T for \mathcal{F} . While S_0 will be a supertree for \mathcal{F} , it may not be an additive rooted supertree for \mathcal{F} . For example, if $X = \{1, 2, 3, 4\}$, let T have the nontrivial clusters $\{1, 2, 3\}$ and $\{1, 2\}$. Suppose every branch length in T is 1. Suppose that the input additive rooted trees are $T|_{\{1, 2\}}, T|_{\{1, 3\}}$, and $T|_{\{2, 4\}}$. Then BUILD-WITH-DISTANCES leads to the supertree S_0 with the single nontrivial cluster $\{1, 2\}$ and all branch lengths equal to 1. S_0 is not an additive supertree since $\lambda(2, 4; T) = 3$ is input while $\lambda(2, 4; S_0) = 2$. The difficulty arises from the failure of condition (2) in Proposition 4.13 when $U = \{1, 2, 3, 4\}, C = \{1, 2, 3\}, E = \{1, 2\}$.

On the other hand, the method often leads to additive rooted supertrees. Section 1 shows examples of additive rooted supertrees constructed by BUILD-WITH-DISTANCES. The reconstructions perfectly match the input data, including the branch lengths.

5 Minimal threshold trees

Let $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ be a positive additive tree family. In this section we seek a rooted tree on $X = \cup X_i$ that is closely related to the trees in \mathcal{F} in the case when no additive supertree exists.

Let T be a rooted tree on the leaf set X and let x, y , and z be distinct members of X . Write $xy|z$ in T and say $xy|z$ is a *rooted triple* in T if there exists U in T such that $x \in U$ and $y \in U$ but $z \notin U$. If T is a positive additive rooted tree with lca distance function λ , then $xy|z$ in T if and only if $\lambda(x, z) - \lambda(x, y) > 0$. From Theorem 2.1(vi), $\lambda(x, z) - \lambda(x, y) = \lambda(y, z) - \lambda(y, x)$ if $xy|z$ in T .

Analogously, let (X, D, λ) be an lca-domain. Suppose x, y , and z are members of X with $x \neq y$. Define the *primary evidence* $p(x, y, z)$ for $xy|z$ in (X, D, λ) as follows:

$$p(x, y, z) = \begin{cases} \max\{0, \lambda(x, z) - \lambda(x, y)\} & \text{if } (x, y) \in D \text{ and } (x, z) \in D, \\ 0 & \text{if either } (x, y) \text{ or } (x, z) \text{ is not in } D. \end{cases}$$

Note that if $p(x, y, z) > 0$, then z is distinct from x and y . Define the *confirmed evidence* $c(x, y, z)$ for $xy|z$ in (X, D, λ) by $c(x, y, z) = \min\{p(x, y, z), p(y, x, z)\}$. When $p(x, y, z) > 0$ or $p(y, x, z) > 0$, then we would hope that $xy|z$ in any tree to be constructed from (X, D, λ) . If we are concerned about errors, we might prefer the additional assurance that $c(x, y, z) > 0$ before we accept that $xy|z$ in a tree that we construct.

Suppose that (X, D, λ) is an lca-domain. A *support function* is a function s which assigns to each nonempty subset U of X and each pair (x, y) of distinct members of U a number $s(x, y, U) \geq 0$ called the *support* of $\{x, y\}$ on U , such that

- (1) $s(x, y, U) = s(y, x, U)$ for all x and y in U ;
- (2) when $s(x, y, U) > 0$, there exists $z \in U$ such that either $p(x, y, z) > 0$ or $p(y, x, z) > 0$;
- (3) when there exists $z \in U$ such that both $p(x, y, z) > 0$ and $p(y, x, z) > 0$, then $s(x, y, U) > 0$.

Intuitively, $s(x, y, U)$ summarizes the evidence for a clustering of the members of U so that x and y are clustered together.

We shall utilize the following natural support functions, which depend on (X, D, λ) : Suppose U is a nonempty subset of X and x and y are distinct members of U .

- (1) The *primary* support function is

$$p(x, y, U) = \max\{p(x, y, z), p(y, x, z) : z \in U\}.$$

- (2) The *confirmed* support function is

$$c(x, y, U) = \max\{c(x, y, z) : z \in U\}.$$

- (3) The *accumulated primary* support function is

$$a_p(x, y, U) = \Sigma[p(x, y, z) + p(y, x, z) : z \in U].$$

- (4) The *accumulated confirmed* support function is

$$a_c(x, y, U) = \Sigma[c(x, y, z) : z \in U].$$

It is easy to check that these are support functions. Their uses will be compared in Section 6. The accumulated support functions allow different kinds of evidence to reinforce each other.

Suppose (X, D, λ) is an lca-domain. If no supertree exists, then by Theorem 4.8 there exists U which is not a singleton set yet such that $A(U)$ is connected. For each edge $\{x, y\}$ of $A(U)$, we may assume (interchanging x and y if necessary) that there exists $z \in U$ such that $p(x, y, z) > 0$. If there were an additive supertree T , then since $\lambda(x, z) - \lambda(x, y) = p(x, y, z) > 0$ we would have $xy|z$ in T . Since $A(U)$ is connected, not all these assertions can simultaneously be true, and we must pick which ones to believe. In practical applications, the distance function is estimated from data, so small errors are likely, and it is natural to expect that a statement $xy|z$ with greater support $p(x, y, z)$ is more reliable

than a statement $x'y'|z'$ with lesser support $p(x', y', z')$. (This supposition may, however, be debated; see Wilkinson *et al.* 2003.) Similar arguments apply to other methods of measuring the support. Thus if $c(x, y, z) > c(x', y', z')$, we have stronger evidence for $xy|z$ than we have for $x'y'|z'$.

Here we present an algorithm which gives more credence to an edge that has higher support. We introduce a generalization of the graph $A(U)$, in which edges are required to have at least a certain degree of support.

Let (X, D, λ) be an lca-domain and s be a support function. If U is a subset of X with more than one point and $t \geq 0$, define the *generalized Aho graph* $A^s(U, t)$ with threshold t and support function s as follows: The vertices of $A^s(U, t)$ are the members of U . If x and y are distinct members of U there is an edge $\{x, y\}$ in $A^s(U, t)$ iff $s(x, y, U) > t$. Note that for the primary support function p , $A^p(U, 0) = A(U)$.

Lemma 5.1. *Let (X, D, λ) be an lca-domain and let s be a support function. Let U be a nonempty subset of X .*

- (1) *For sufficiently large t , $A^s(U, t)$ has no edges.*
- (2) *If $v > t$ and $A^s(U, t)$ is disconnected, then $A^s(U, v)$ is disconnected.*
- (3) *Suppose U has more than one point, and let*

$$t_0 = \inf\{t \geq 0 : A^s(U, t) \text{ is disconnected}\}.$$

Then $A^s(U, t_0)$ is disconnected.

Proof. (1) is obvious since there are only finitely many values $s(x, y, U)$. Also, (2) is obvious since the edges in $A^s(U, v)$ will be a subset of the edges in $A^s(U, t)$. For (3), if t_0 is as described, then for each $t > t_0$, $A^s(U, t)$ is disconnected. For every edge $\{x, y\}$ in $A^s(U, t_0)$ we know that $s(x, y, U) > t_0$, so we may choose a number $t(x, y)$ such that $s(x, y, U) > t(x, y) > t_0$. Let

$$\hat{t} = \min\{t(x, y) : x \in U, y \in U, s(x, y, U) > t_0\}.$$

Then every edge in $A^s(U, t_0)$ is in $A^s(U, \hat{t})$. Yet since $\hat{t} > t_0$, every edge in $A^s(U, \hat{t})$ is in $A^s(U, t_0)$. Hence $A^s(U, t_0) = A^s(U, \hat{t})$. Since $\hat{t} > t_0$, $A^s(U, \hat{t})$ is disconnected by (2), whence so is $A^s(U, t_0)$. \square

Let (X, D, λ) be an lca-domain, and let s be a support function. Assume $U \subseteq X$ contains at least two points. Define the *threshold* for U with support function s to be

$$\text{thr}^s(U) = \text{thr}^s(U; X, D, \lambda) := \inf\{t \geq 0 : A^s(U, t) \text{ is disconnected}\}.$$

By Lemma 5.1, $A^s(U, \text{thr}^s(U))$ is disconnected.

Let (X, D, λ) be an lca-domain, and let s be a support function. Define the *minimal threshold tree* $S^s = S^s(X, D, \lambda) = S^s(\lambda)$ to be the smallest collection of subsets of X such that:

- (1) $X \in S^s$.
- (2) If $U \in S^s$ and is not a singleton, then each component C of $A^s(U, \text{thr}^s(U))$ is in S^s .

Theorem 5.2. *Let (X, D, λ) be an lca-domain, and let s be a support function. Then S^s is a rooted tree.*

Proof. By Lemma 5.1, whenever $U \in S^s$ is not a singleton, it follows that U has at least two children since $A^s(U, \text{thr}^s(U))$ is disconnected. Hence all singleton sets from X are in S^s . The other properties of being a rooted tree are obvious. \square

Theorem 5.3. *Suppose $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ is an additive rooted tree family and (X, D, λ) is its lca-domain. Let s be a support function. If there exists a positive additive rooted supertree for \mathcal{F} then $\text{thr}^s(U) = 0$ for all U with at least two members, and S^s is a supertree for \mathcal{F} .*

Proof. Let (T, X, w, λ') be a positive additive rooted supertree for \mathcal{F} . Let U in S^s contain at least two points. I claim that $A^s(U, 0)$ is disconnected, so that $\text{thr}^s(U) = 0$. Let $V = \text{lca}(U; T)$. I will show that if there is an edge $\{x, y\}$ in $A^s(U, 0)$, then x and y lie in the same child of V in T . This will prove that $\text{thr}^s(U) = 0$, since otherwise $A^s(U, 0)$ is connected, so all members of U will lie in the same child of V in T , contradicting the definition of V .

If there is an edge $\{x, y\}$ in $A^s(U, 0)$, then $s(x, y, U) > 0$, and by (2) in the definition of a support function we may assume that there exists $z \in U$ such that $(x, z) \in D, (x, y) \in D, \lambda(x, z) - \lambda(x, y) > 0$. Since T is an additive supertree, it follows that its lca distance function λ' agrees with λ on D , so by Theorem 2.1 $\text{lca}(x, y; T)$ is a proper descendent of $\text{lca}(x, z)$ in T . In particular, since x and z are in V , then $\text{lca}(x, y; T)$ must be contained in a child of V , so x and y are in the same child of V .

To see that S^s is a supertree for \mathcal{F} , let U be a cluster in T_i . Let $W = \text{lca}(U; S^s)$. We show that $U = W \cap X_i$, which will prove that S^s is a supertree. It is immediate that $W \cap X_i$ contains U . Suppose that $w \in W \cap X_i - U$. Since T is a supertree for \mathcal{F} , there exists W' in T with $W' \cap X_i = U$. Then $\text{lca}(w, W'; T)$ is a proper ancestor of W' because $w \notin W'$. Since T is a positive additive supertree, for every x and y in U it follows $\lambda'(x, w) - \lambda'(x, y) > 0$ and $\lambda'(y, w) - \lambda'(y, x) > 0$. Since x, y , and w are all in X_i , $(x, y), (y, x), (y, w)$, and (x, w) are in D . But $\lambda = \lambda'$ on D because T is an additive supertree, whence $p(x, y, w) > 0$ and $p(y, x, w) > 0$. By property (3) of a support function, it follows $s(x, y, W) > 0$, so there is an edge $\{x, y\}$ in $A^s(W, 0) = A^s(W, \text{thr}^s(W))$ for each x and y in U . It follows that all members of U are contained in a single child of W , contradicting the definition of W . This contradiction shows that no such w can exist, so $U = W \cap X_i$. \square

If there exists a positive additive supertree, then the supertree S_0 from Section 4 is the same as S^p , where p is the primary support function, since $A(U) = A^p(U, 0) = A^p(U, \text{thr}^p(U))$ for each U by Theorem 5.3. In that case, the following result together with Corollary 4.12 shows that S^p has the most resolution among the various supertrees S^s . This fact is the reason that Section 4 dealt only with S_0 .

Proposition 5.4. *Let (X, D, λ) be an lca-domain and s be a support function. Suppose there exists a positive additive supertree. Let p be the primary support function. Then S^s is monotone for S^p .*

Proof. Let $U \in S^s$, $U' \in S^p$, and $U \subseteq U'$. We may assume that U contains at least two members. Let C be a child of U in S^s . Since $\text{thr}^s(U) = 0$ by Theorem 5.3, C is a component of $A^s(U, 0)$. We must show that there is a child of U' in S^p that contains C . It suffices to show that, when there is an edge $\{x, y\}$ in $A^s(U, 0)$, then there is an edge $\{x, y\}$ in $A^p(U', 0)$ since $\text{thr}^p(U') = 0$ by Theorem 5.3. Suppose there is an edge $\{x, y\}$ in $A^s(U, 0)$. Then $s(x, y, U) > 0$. By property (2) of a support function, there exists $z \in U$ such that either $p(x, y, z) > 0$ or $p(y, x, z) > 0$. Either way, there is an edge $\{x, y\}$ in $A^p(U', 0)$. \square

Theorem 5.5. *Let (X, D, λ) be an lca-domain. Let s be a support function which can be computed in polynomial time. Then there exists an algorithm to compute S^s in polynomial time.*

Proof. Suppose that X contains n taxa. Since S^s is a rooted tree with n leaves, it contains at most $2n - 2$ edges and $n - 1$ vertices that are not leaves. Hence there are at most $n - 1$ different clusters U that are not singleton sets. If a cluster U has exactly two points, its children are the corresponding singleton sets. It therefore suffices to prove that, when $U \in S^s$ has at least three points, then $\text{thr}^s(U)$ can be computed in polynomial time.

For such U , let $E = \{s(x, y, U) : x \in U, y \in U\}$. Then E can be computed in polynomial time. I claim that, if $\text{thr}^s(U) > 0$, then $\text{thr}^s(U) \in E$. To see this, suppose $\text{thr}^s(U) > 0$ and $\text{thr}^s(U) \notin E$. Since E is a finite closed set, there exists $\epsilon > 0$, $\epsilon < \text{thr}^s(U)$, such that no member $h \in E$ satisfies $\text{thr}^s(U) - \epsilon \leq h \leq \text{thr}^s(U)$. Let $g = \text{thr}^s(U) - \epsilon$. Then each edge of $A^s(U, \text{thr}^s(U))$ trivially is an edge of $A^s(U, g)$ since $g < \text{thr}^s(U)$. Moreover, for any edge $\{x, y\}$ of $A^s(U, g)$ since $s(x, y, U) > g$ and $s(x, y, U) \in E$, it follows that $s(x, y, U) > \text{thr}^s(U)$, so $\{x, y\}$ is an edge of $A^s(U, \text{thr}^s(U))$ as well. Hence $A^s(U, g) = A^s(U, \text{thr}^s(U))$ and $A^s(U, g)$ is disconnected by Lemma 5.1. This contradicts the definition of $\text{thr}^s(U)$, showing that $\text{thr}^s(U) \in E$.

Hence, $\text{thr}^s(U) = \min\{v \in E \cup \{0\} : A^s(U, v) \text{ is disconnected}\}$. Since there are fewer than n^2 members of E , for each U one may compute $\text{thr}^s(U)$ in polynomial time. \square

Let (X, D, λ) be an lca-domain, and let s be a support function. Let U be a subset of X that contains at least two points. In fact, there is a faster procedure to compute $\text{thr}^s(U)$ and the children of U . If U contains exactly two points x and y , then clearly the children of U are the singleton sets $\{x\}$ and $\{y\}$. If U contains more than two points, proceed as follows:

If $A^s(U, 0)$ is disconnected, then $\text{thr}^s(U) = 0$ and output the components of $A^s(U, 0)$. Otherwise, assume that $A^s(U, 0)$ is connected. Let $m_0 = 0$ and let M_0 be the largest value $s(x, y, U)$ for a and b in U , so that trivially $A^s(U, M_0)$ is disconnected. Hence $\text{thr}^s(U)$ lies in the interval $(m_0, M_0]$. We now use the bisection method to narrow in on $\text{thr}^s(U)$. Given an interval $(m_i, M_i]$ such that

$A^s(U, m_i)$ is connected but $A^s(U, M_i)$ is disconnected, let $t_i = (m_i + M_i)/2$. If $A^s(U, t_i)$ is disconnected, then $\text{thr}^s(U)$ lies in the interval $(m_i, t_i]$ and we set $m_{i+1} = m_i, M_{i+1} = t_i$. If $A^s(U, t_i)$ is connected, then $\text{thr}^s(U)$ lies in the interval $(t_i, M_i]$ and we set $m_{i+1} = t_i, M_{i+1} = M_i$. With each iteration, the length of the interval $(m_i, M_i]$ is halved. Hence M_i converges rapidly to $\text{thr}^s(U)$.

Proposition 5.6. *Let (X, D, λ) be an lca-domain, and let s be a support function. Let x, y , and z be distinct elements of X . Let $U = \text{lca}(x, y, z; S^s)$. If $s(x, y, U) > \text{thr}^s(U)$, then $xy|z$ in S^s .*

Proof. Since $s(x, y, U) > \text{thr}^s(U)$, it follows that there is an edge $\{x, y\}$ in $A^s(U, \text{thr}^s(U))$. Since $A^s(U, \text{thr}^s(U))$ is disconnected, x and y are in the same child W of U in S^s . Yet z is not in W since $U = \text{lca}(x, y, z)$. Hence $xy|z$ in S^s . \square

Theorem 5.7. *Let (X, D, λ) be an lca-domain, and let s be a support function. Let T be a rooted tree with leaf set X . Assume that, whenever x, y , and z in X satisfy $s(x, y, \text{lca}(x, y, z; S^s)) > \text{thr}^s(\text{lca}(x, y, z; S^s))$, then $xy|z$ in T . Then S^s is monotone for T .*

Proof. Let $U \in S^s$ have child C in S^s . Assume that $V \in T$ contains U . We must see that there exists a child C' of V in T that contains C . It suffices to show that, if there is an edge $\{x, y\}$ in $A^s(U, \text{thr}^s(U))$, then x and y are in the same child of V in T . Choose $u \in U - C$, so that $U = \text{lca}(x, y, u; S^s)$. Since there is an edge $\{x, y\}$, it follows $s(x, y, U) > \text{thr}^s(U)$. From the hypotheses it follows that $xy|u$ in T . Since x, y , and u are in V , it follows that either x, y , and u are all in the same child of V or else x and y are in one child and u is in a different child. In either case both x and y are in the same child of V . \square

Theorem 5.7 shows that, when T is a rooted tree, either T fails to have $xy|z$ in T for some triple which is highly supported, or else S^s is monotone for T .

S^s can be made into an additive rooted tree in the same manner as S_0 : Suppose that in the algorithm, C is a child of U in S^s (so C is a component of $A^s(U, \text{thr}^s(U))$). If there exist c and c' in C (possibly $c = c'$) and $u \in U - C$ such that (c, u) and (c, c') are in D , define

$$w(C, U) = \min\{\lambda(c, u) - \lambda(c, c') : c \in C, c' \in C, u \in U - C, (c, u) \in D, (c, c') \in D\};$$

otherwise set $w(C, U) = 0$.

6 Comparison of minimal threshold trees

If (X, D, λ) is an lca-domain and (T, X, w, λ') is an additive rooted tree, then the *root-square deviation* between them is

$$\Delta((X, D, \lambda), (T, X, w, \lambda')) = \sqrt{\sum [(\lambda(x, y) - \lambda'(x, y))^2 : (x, y) \in D]}$$

We may write it more briefly as $\Delta(\lambda, T)$. If T is an additive supertree then clearly $\Delta(\lambda, T) = 0$. A good approximate supertree should have small root-square deviation.

In this section we will make a comparison between the minimal threshold trees for the primary, the confirmed, the accumulated primary, and the accumulated confirmed support functions p , c , a_p , and a_c respectively.

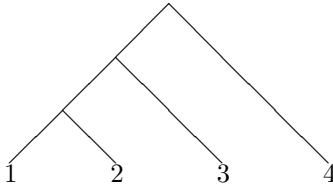


Figure 6: A tree T . All branch lengths are 1.

First, suppose that the correct tree is T as in Figure 6, in which all edges have unit branch length. Note there are only two nontrivial clusters— $\{1, 2\}$ and $\{1, 2, 3\}$. Suppose λ is the lca-distance function for T . Then BUILD-WITH-DISTANCES reconstructs T from λ .

We consider the effect of perturbing λ slightly. Suppose λ' agrees with λ except that $\lambda'(1, 3) = x$ where the correct value in T is $x = 2$. One finds that $S^p(\lambda') = T$ for $0 < x < 4$ but generally with incorrect branch lengths. For $x > 4$, $S^p(\lambda')$ has the nontrivial clusters $\{1, 2\}$ and $\{1, 2, 4\}$, and when $x = 4$, $S^p(\lambda')$ has the unique nontrivial cluster $\{1, 2\}$. Similarly, $S^c(\lambda') = S^{a_c}(\lambda') = T$ for $1 < x$, while $S^c(\lambda') = S^{a_c}(\lambda')$ has the unique nontrivial cluster $\{1, 2, 3\}$ for $0 < x \leq 1$. Finally, $S^{a_p}(\lambda') = T$ for $0 < x < 5$, while $S^{a_p}(\lambda')$ has the nontrivial clusters $\{1, 2\}$ and $\{1, 2, 4\}$ for $x > 5$ and has the unique nontrivial cluster $\{1, 2\}$ when $x = 5$. In this example, $S^c = S^{a_c}$ yields the correct tree when x is too large, but fails when x is too small by 1. Moreover, when it fails, $S^c = S^{a_c}$ is a proper subtree of T and merely lacks some of the resolution of T . S^p yields the correct tree when x is too small, but fails when x is too large by 2, and its failure yields a cluster not in T . S^{a_p} yields the correct tree when x is too small but fails when x is too large by 3; its failure also yields a cluster not in T .

There are simple examples of additive tree families in which each of the minimal threshold trees described in this paper appears superior. Perhaps, for example, $\Delta(\lambda, S^{a_p}) < \Delta(\lambda, S^{a_c}) < \Delta(\lambda, S^c) < \Delta(\lambda, S^p)$ for superiority of S^{a_p} . A rule of thumb, however, is that, just as in the example, S^{a_p} appears to be correct for a broader range of moderate errors. S^c and S^{a_c} appear more conservative and most likely to fail, as in the example, by merely giving a tree that is less resolved but contained in the correct tree. S^p appears to be the most sensitive at finding additional resolution when the incompatibilities are less serious, yet also the most erratic.

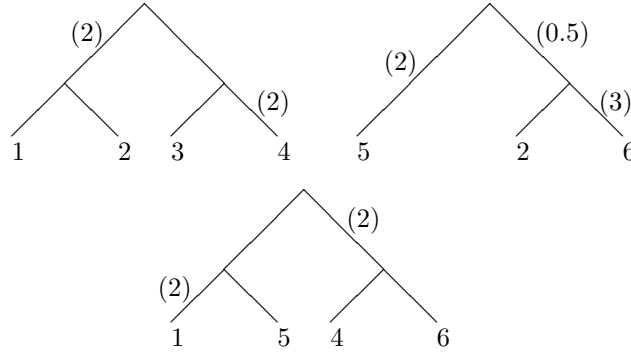


Figure 7: Some input trees. All unspecified branch lengths are 1.

Next consider the family \mathcal{F} of input trees shown in Figure 7 and the lead-domain (X, D, λ) obtained from \mathcal{F} . S_0 is not a rooted tree and there is no supertree. The nontrivial clusters of S^{ap} are $\{1, 2, 5\}$, $\{1, 2\}$, $\{3, 4, 6\}$, $\{4, 6\}$, while $\Delta(\lambda, S^{ap}) = 0.430$. The nontrivial clusters of $S^c = S^{ac}$ are $\{1, 2, 5\}$, $\{1, 2\}$, $\{3, 4, 6\}$, while $\Delta(\lambda, S^c) = \Delta(\lambda, S^{ac}) = 0.688$. The nontrivial clusters of S^p are $\{1, 2, 4, 5, 6\}$ and $\{1, 2, 4, 6\}$ with $\Delta(\lambda, S^p) = 1.269$. Hence S^{ap} is best both because it has the most resolution and because it has the lowest root-square deviation. $S^c = S^{ac}$ is next best, but lacks some of the resolution. In this case S^p seems substantially inferior. For this family \mathcal{F} , MRP using PAUP* finds 18 maximum parsimony trees. The strict consensus of these is the star tree; the only nontrivial cluster in the Adams consensus is $\{3, 4\}$, and the only nontrivial cluster in the 50% majority rule tree is $\{3, 4, 6\}$. The MinCut supertree of Semple and Steel (2000) is the star tree. These low resolution trees indicate the presence of serious incompatibilities. Both S^{ap} and S^c appear substantially superior to the trees found by MRP.

7 Discussion

In the problem of describing the Tree of Life, supertree methods appear well positioned to combine the rooted trees carefully chosen by many researchers into a single tree containing all the taxa. The results in this paper suggest that, if the input trees are additive rooted trees, additional resolution may be obtained over topological methods by using algorithms that take this additional distance information into account. Even if the conditions for combining these into an additive rooted supertree are not met, the procedures lead often to rooted trees

that show more resolution than trees obtained by other methods such as MRP. The algorithm BUILD-WITH-DISTANCES and its modifications for the case when no supertree exists are fast and easy to implement.

In applied problems it is likely that the distance information in different input trees will not be immediately compatible. Suppose $\mathcal{F} = \{(T_i, X_i, w_i, \lambda_i)\}$ is an additive rooted tree family. If a and b are in both X_i and X_j , typically it will not be true that $\lambda_i(x, y) = \lambda_j(x, y)$. Clearly a procedure will need to be provided to reconcile these values. The current code by the author merely averages the different values $\lambda_i(x, y)$ obtained from different i when x and y are in X_i , and this works moderately well. Better reconciliation methods should be the subject of further research. The same cautions described in Lapointe and Cucumel (1997) apply here.

References

- Aho, A.V., Y. Sagiv, T.G. Szymanski, and J.D. Ullman (1981). Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J. Comput* 10 (3), 405-421.
- Baum, B.R. (1992). Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon* 41, 3-10.
- Bininda-Emonds, O. and M. J. Sanderson (2001). Assessment of the accuracy of matrix representation with parsimony analysis supertree construction. *Syst. Biol.* 50(4), 565-579.
- Bininda-Emonds, O., J. L. Gittleman, and M. Steel (2002). The (super)tree of life: procedures, problems, and prospects. *Ann. Rev. Ecol. Syst.* 33, 265-289.
- Bryant, D., C. Semple, and M. Steel (2004). Supertree methods for ancestral dates and other applications, in *Phylogenetic Supertrees: Combining information to reveal the tree of life.* (O.R.P. Bininda-Emonds ed.) Kluwer Academic, Dordrecht, the Netherlands, pp. 129-150.
- Colonus, H. and H.H. Schulze (1981). Tree structures for proximity data. *British Journal of Mathematical and Statistical Psychology* 34, 167-180.
- Dress, W.M. and Péter L. Erdős (2003). X-trees and weighted quartet systems, *Ann. Combin.* 7, 155-169.
- Felsenstein, J. (1993). PHYLIP (Phylogeny Inference Package) version 3.5c. Distributed by the author. Department of Genetics, University of Washington, Seattle.
- Fitch, W.M., and E. Margoliash (1967). Construction of phylogenetic trees. *Science* 155, 279-284.
- Hartigan, J.A. (1967). Representation of similarity matrices by trees. *Journal of the American Statistical Association* 62, 1140-1158.
- Hubert, L. (1972). Some extensions of Johnson's hierarchical clustering algorithms. *Psychometrika* 37, 261-274.

- Hubert, L. (1973). Monotone invariant clustering procedures. *Psychometrika* 38, 47-62.
- Janowitz, M.F., F.-J. Lapointe, F.R. McMorris, B. Mirkin, F.S. Roberts, eds. (2003). *Bioconsensus: DIMACS Working Group Meetings on Bioconsensus, October 25-26, 2000 and October 2-5, 2001*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 61, American Mathematical Society.
- Johnson, S.C. (1967). Hierarchical clustering schemes. *Psychometrika* 32, 241-254.
- Lapointe, F.-J. and G. Cucumel (1997). The average consensus procedure: combination of weighted trees containing identical or overlapping sets of taxa. *Syst. Biol.* 46(2), 306-312.
- Lapointe, F.-J., M. Wilkinson, and D. Bryant (2003). Matrix representations with parsimony or with distances: Two sides of the same coin?. *Syst. Biol.* 52(6), 865-868.
- Lapointe, F.-J. and C. Levasseur (2004). Everything you always wanted to know about the average consensus, and more, in *Phylogenetic supertrees: combining information to reveal the Tree of Life* (O.R.P. Bininda-Emonds, ed.) Kluwer Academic, Dordrecht, the Netherlands, pp. 87-105.
- Page, R.D.M. (2003). Modified mincut supertrees, in R. Guig and D. Gusfield (Eds), *Algorithms in Bioinformatics, Proceedings of the second international workshop, WABI 2002, Rome, Italy, September 17-21, 2002*, Lecture Notes in Computer Science 2452, Springer-Verlag, Berlin, pp. 537-552.
- Ragan, M.A. (1992). Phylogenetic inference based on matrix representation of trees. *Mol. Phylogenet. Evol.* 1, 53-58.
- Sanderson, M.J., A. Purvis, and C. Henze (1998). Phylogenetic supertrees: Assembling the trees of life. *Trends Ecol Evol.* 13, 105-109.
- Semple, C. and M. Steel (2000). A supertree method for rooted trees. *Discrete Applied Mathematics* 105, 147-158.
- Semple, C. and M. Steel (2003). *Phylogenetics*. Oxford University Press, Oxford.
- Swofford, D. L. (2002). *PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods)*. Version 4. Sinauer Associates, Sunderland, Massachusetts.
- Swofford, D.L., G.J. Olsen, P.J. Waddell, and D.M.Hillis (1996). Phylogenetic inference, 407-514, in Hillis, D., Moritz, C. and Mable, B. (Eds), *Molecular Systematics*, second edition, Sinauer, Sunderland, MA.
- Thorley, J.L. and R.D.M. Page (2000). RADCON: Phylogenetic tree comparison and consensus. *Bioinformatics* 16, 486-487.
- Wilkinson, M., F.-J. Lapointe, and D. J. Gower (2003). Branch lengths and support. *Syst. Biol.* 52(1), 127-130.