Chapter 6

SUPERTREE METHODS FOR ANCESTRAL DIVERGENCE DATES AND OTHER APPLICATIONS

David Bryant, Charles Semple, and Mike Steel

Abstract: There are many ways to combine rooted phylogenetic trees with overlapping leaf sets into a single "supertree". The most widely used method is MRP (matrix representation with parsimony analysis), but other direct methods have been developed recently. However, all these methods utilize typically only the discrete topology of the input trees and ignore other information that might be available. Based, for example, on fossil data or molecular dating techniques, this information includes whether one particular divergence event occurred earlier or later than another, and actual time estimates for divergence events. The ability to include such information in supertree construction could allow for more accurate dating of certain species divergences. This is a topical problem in recent biological literature. In this chapter, we describe a way to incorporate divergence time information in a fast and exact supertree algorithm that extends the classic BUILD algorithm. The approach is somewhat flexible in that it allows any combination of relative and/or absolute divergence times. In addition to this extension, the last section of this chapter consists of applications of BUILD to problems in phylogenetics that are, in general, computationally challenging.

Keywords: BUILD; divergence dates; rooted phylogenetic tree; supertree

1. Introduction

We will follow mostly the notation of Semple and Steel (2003) and assume that the reader is familiar with the basic concepts of phylogenetic trees.

Bininda-Emonds, O. R. P. (ed.) Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, pp. 129–150. Computational Biology, volume 3 (Dress, A., series ed.).

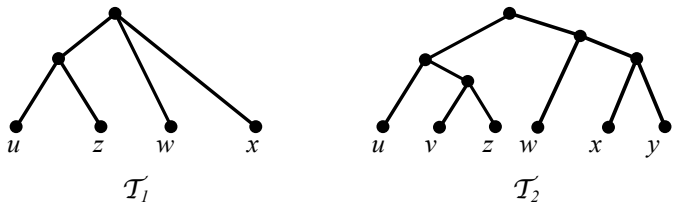

Figure 1. $\mathcal{T}_2$ displays $\mathcal{T}_1$.

Let $\mathcal{T}$ be a rooted phylogenetic X-tree. Thus, X refers to the leaf set of $\mathcal{T}$, here denoted $\mathcal{L}(\mathcal{T})$, which represents typically the set of extant species classified by $\mathcal{T}$. If $X' \subseteq X$, then the *restriction of* $\mathcal{T}$ *to* X', denoted $\mathcal{T}|X'$, is the rooted phylogenetic X'-tree that is obtained from the minimal rooted subtree of $\mathcal{T}$ containing X' by suppressing all non-root vertices of degree two.

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two rooted phylogenetic trees with $\mathcal{L}(\mathcal{T}_1) \subseteq \mathcal{L}(\mathcal{T}_2)$. We say that $\mathcal{T}_2$ *displays* $\mathcal{T}_1$ if $\mathcal{T}_2|\mathcal{L}(\mathcal{T}_1)$ is a refinement of $\mathcal{T}_1$. Informally, $\mathcal{T}_2$ displays $\mathcal{T}_1$ if, up to polytomies, all the ancestral relationships of $\mathcal{T}_1$ are preserved in $\mathcal{T}_2$. This notion is illustrated in Figure 1.

For a collection $\mathcal{R}$ of rooted phylogenetic trees, we denote the set $\cup_{\mathcal{T} \in \mathcal{R}} \mathcal{L}(\mathcal{T})$ by $\mathcal{L}(\mathcal{R})$. Extending the notion of display to $\mathcal{R}$, we say that a rooted phylogenetic tree $\mathcal{T}$ with $\mathcal{L}(\mathcal{R}) \subseteq \mathcal{L}(\mathcal{T})$ *displays* $\mathcal{R}$ if every member of $\mathcal{R}$ is displayed by $\mathcal{T}$. Via an algorithm called BUILD, Aho *et al.* (1981) showed that there is a polynomial-time algorithm to determine if a rooted phylogenetic tree exists that displays $\mathcal{R}$ and, if so, to construct such a rooted phylogenetic tree.

Informally, BUILD constructs clusters (subsets of $\mathcal{L}(\mathcal{R})$) that are broken down successively into disjoint subclusters according to the hierarchical relationships described by the trees in $\mathcal{R}$. If this process continues until all singleton clusters (i.e., clusters consisting of just a single species) are obtained, the rooted tree that corresponds to this resulting set of clusters is the tree returned by BUILD that displays $\mathcal{R}$. However, if the process stops before all singleton clusters are obtained, then there is no rooted phylogenetic tree that displays $\mathcal{R}$.

We remark here that BUILD was applied originally to relational databases and its application to phylogenetics appeared somewhat later (e.g., Steel, 1992; Constantinescu, 1995; Ng and Wormald, 1996; Semple, 2003). This fact might explain why BUILD is not that well known in this field.

A natural extension of BUILD is to include as input a collection of constraints representing the order in which the divergence events of certain different pairs of species occurred. To make this precise, we need several further definitions. Let $\mathcal{T}$ be a phylogenetic tree. For all $v_1, v_2 \in V(\mathcal{T})$, we write $v_1 \preceq_{\mathcal{T}} v_2$ if v_2 is a descendant of v_1. The relation $\preceq_{\mathcal{T}}$ induces a partial

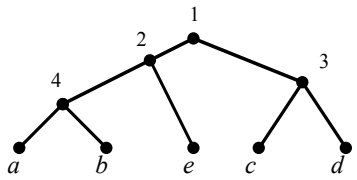

Figure 2. A ranked phylogenetic tree.

order on the vertices of $\mathcal{T}$. For a subset A of $\mathcal{L}(\mathcal{T})$, the unique vertex of $\mathcal{T}$ that is the greatest lower bound of A under $\leq_{\mathcal{T}}$ is referred to as the *most recent common ancestor* of A in $\mathcal{T}$. We denote this vertex as $\text{mrca}_{\mathcal{T}}(A)$. For simplicity, if $A = \{a, b\}$, we denote $\text{mrca}_{\mathcal{T}}(\{a, b\})$ by $\text{mrca}_{\mathcal{T}}(a, b)$.

Let $\mathcal{T}$ be a rooted phylogenetic tree. A *rank function* for $\mathcal{T}$ is a function r from the set V^o of interior vertices of $\mathcal{T}$ into the set of positive integers such that, for all $v_1, v_2 \in V^o$, $r(v_1) < r(v_2)$ if v_2 is a proper descendant of v_1. The pair $(\mathcal{T}, r)$ is a *ranked phylogenetic tree*. Again for simplicity, we denote $r(\text{mrca}_{\mathcal{T}}(A))$ by $r(A)$ for all subsets $A \in \mathcal{L}(\mathcal{T})$. In this chapter, a ranking of the interior vertices of $\mathcal{T}$ corresponds to an ordering of the occurrence of the associated speciation events. An example of a ranked phylogenetic tree is illustrated in Figure 2.

For species a, b, c, d, a *relative divergence date* is a statement of the form "div(c, d) predates div(a, b)", which is interpreted as "the divergence of c and d predates that of a and b". No specific dates are required to make this statement, just an ordering on the two associated speciation events.

Let $\mathcal{D}$ be a collection of relative divergence dates. We denote the set $\cup\{a, b, c, d\}$ over all statements "div(c, d) predates div(a, b)" in $\mathcal{D}$ by $\mathcal{L}(\mathcal{D})$. That is, $\mathcal{L}(\mathcal{D})$ is the set of all species that are mentioned by at least one statement of relative divergence. Also, we say that $\mathcal{D}$ is *preserved* by a ranked phylogenetic tree $(\mathcal{T}, r)$ with $\mathcal{L}(\mathcal{D}) \subseteq \mathcal{L}(\mathcal{T})$ if $r(c, d) < r(a, b)$ for all statements "div(c, d) predates div(a, b)" in $\mathcal{D}$.

In the first part of this chapter, we present an algorithm called RANKEDTREE that provides a polynomial-time solution to the following classification problem.

Problem: PHYLOGENETIC RANKING

Instance: A collection $\mathcal{R}$ of rooted phylogenetic trees and a collection $\mathcal{D}$ of relative divergence dates.

Figure 3. Two rooted phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$. Applying RANKEDTREE to these trees with the two relative divergence dates "div(c, d) predates div(a, b)" and "div(a, e) predates div(c, d)" gives the ranked phylogenetic tree shown in Figure 2.

Question: Does a ranked phylogenetic tree on $\mathcal{L}(\mathcal{R}) \cup \mathcal{L}(\mathcal{D})$ exist that displays $\mathcal{R}$ and preserves $\mathcal{D}$ and, if so, can we construct such a ranked phylogenetic tree in polynomial time?

To illustrate PHYLOGENETIC RANKING, consider the phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$, and the two relative divergence dates shown in Figure 3. Applying RANKEDTREE to $\mathcal{T}_1$ and $\mathcal{T}_2$ and these relative divergence dates results in the ranked phylogenetic tree $(\mathcal{T}, r)$ shown in Figure 2. Observe that $\mathcal{T}$ displays both $\mathcal{T}_1$ and $\mathcal{T}_2$, and $\mathcal{T}$ preserves both relative divergence dates. In this case, $(\mathcal{T}, r)$ is unique; however, this does not generally happen. Note that it is possible for the interior vertices to have the same rank, in which case there is no particular ordering of the associated speciation events.

The chapter is organized as follows. In the next section, we describe briefly some necessary concepts. In Section 3, we present RANKEDTREE and show that it does indeed give a polynomial-time solution to PHYLOGENETIC RANKING. In Section 4, we extend the input of RANKEDTREE to include interval constraints on divergence dates. Lastly, in Section 5, we describe some applications and extensions of the BUILD algorithm.

2. Clusters, hierarchies, and precedence constraints

Let $\mathcal{T}$ be a rooted phylogenetic X-tree. A *cluster* of $\mathcal{T}$ is a subset of X whose elements are the set of descendants of a vertex of $\mathcal{T}$. Observe that X is a cluster of $\mathcal{T}$ and, for all $x \in X$, $\{x\}$ is a cluster of $\mathcal{T}$. We denote the set of clusters of $\mathcal{T}$ by $\mathcal{H}(\mathcal{T})$.

The set of clusters of a rooted phylogenetic X-tree is an example of a hierarchy $\mathcal{H}$ on X; that is, a collection of subsets of X that has the property that, for all $A, B \in \mathcal{H}$,

$$A \cap B \in \{\emptyset, A, B\}.$$

Hierarchies and rooted phylogenetic trees are related closely. In particular, given a hierarchy $\mathcal{H}$ on X that contains X and all 1-element subsets of X, there is a unique rooted phylogenetic X-tree the set of clusters of which is $\mathcal{H}$. Thus, a rooted phylogenetic tree $\mathcal{T}$ is determined by its set of clusters $\mathcal{H}(\mathcal{T})$. Indeed, $\mathcal{T}$ can be constructed quickly and easily from $\mathcal{H}(\mathcal{T})$. Furthermore, we can associate a rank function to a hierarchy in the same way we associate a rank function to the interior vertices of a rooted phylogenetic tree. Let r be a function from $\mathcal{H} - \{\{x\} : x \in X\}$ into the set of positive integers such that, for all $A, B \in \mathcal{H} - \{\{x\} : x \in X\}$, $r(A) < r(B)$ if B is a proper subset of A. If $\mathcal{H}$ contains X and all 1-element subsets of X, the pair $(\mathcal{H}, r)$ is called a *ranked hierarchy* on X and the function r is a *rank function for* $\mathcal{H}$. Observe that, by the remarks above, we can view ranked hierarchies on X as ranked phylogenetic X-trees. This viewpoint is used freely throughout this chapter.

Lastly, a *precedence constraint* is a pairwise relationship of the form

$$(c, d) \prec (a, b),$$

where a, b, c, and d are species. Note that a, b, c, d are not necessarily different because we might wish to allow constraints such as $(a, c) \prec (a, b)$ (i.e., where the same species is involved in both divergence events). In the context of this chapter, such a constraint denotes that the divergence of c and d predates that of a and b. For a collection $\mathcal{P}$ of precedence constraints, we denote the set $\cup\{a, b, c, d\}$ over all $(c, d) \prec (a, b) \in \mathcal{P}$ by $\mathcal{L}(\mathcal{P})$. A collection $\mathcal{P}$ of precedence constraints is *preserved* by a ranked phylogenetic tree $(\mathcal{T}, r)$ with $\mathcal{L}(\mathcal{P}) \subseteq \mathcal{L}(\mathcal{T})$ if $r(c, d) < r(a, b)$ for all $(c, d) \prec (a, b) \in \mathcal{P}$.

3. RANKEDTREE

In this section, we present RANKEDTREE, a polynomial-time solution to PHYLOGENETIC RANKING.

The input to RANKEDTREE is a collection of precedence constraints. These constraints are constructed from the collection $\mathcal{R}$ of rooted phylogenetic trees and the collection $\mathcal{D}$ of relative divergence dates. Evidently, a ranked phylogenetic tree preserves the relative divergence date "div(c, d) predates div(a, b)" if and only if it preserves $(c, d) \prec (a, b)$. We next show that a rooted phylogenetic tree $\mathcal{T}$ displays $\mathcal{R}$ if and only if $(\mathcal{T}, r)$ preserves a certain collection of precedence constraints for some rank function r for $\mathcal{T}$.

A *rooted triple* is a rooted binary phylogenetic tree $\mathcal{T}$ on three leaves. The rooted triple with leaves a, b, and c is denoted $ab \mid c$ if $\text{mrca}_{\mathcal{T}}(a, b)$ is a

descendant of $\mathrm{mrca}_{\mathcal{T}}(a, c)$ or, equivalently, a descendant of $\mathrm{mrca}_{\mathcal{T}}(b, c)$. The straightforward proof of the next lemma is omitted.

Lemma 3.1. *Let $\mathcal{T}$ be the rooted triple $ab \mid c$, let $\mathcal{T}'$ be a rooted phylogenetic tree with $\{a, b, c\} \subseteq \mathcal{L}(\mathcal{T}')$, and let r be a rank function for $\mathcal{T}'$. Then $\mathcal{T}'$ displays $\mathcal{T}$ if and only if $r(a, c) < r(a, b)$.*

For a rooted phylogenetic tree $\mathcal{T}$, let $\mathcal{R}(\mathcal{T})$ denote the set of rooted triples displayed by $\mathcal{T}$. It is well known that all rooted phylogenetic trees that display $\mathcal{R}(\mathcal{T})$ are refinements of $\mathcal{T}$ (e.g., see Theorem 1 of Bryant and Steel, 1995). Lemma 3.2 is an immediate consequence of this result and Lemma 3.1.

Lemma 3.2. *Let $\mathcal{R}$ be a collection of rooted phylogenetic trees and let*

$$\mathcal{P} = \{(a, c) \prec (a, b) : ab \mid c \in \mathcal{R}(\mathcal{T}) \text{ and } \mathcal{T} \in \mathcal{R}\}.$$

Let $\mathcal{T}'$ be a rooted phylogenetic tree with $\mathcal{L}(\mathcal{T}') = \mathcal{L}(\mathcal{R})$. Then $\mathcal{T}'$ displays $\mathcal{R}$ if and only if there is rank function r for $\mathcal{T}'$ such that $(\mathcal{T}', r)$ preserves $\mathcal{P}$.

The algorithm RANKEDTREE is shown in Figure 4. Note that, for a graph G and a subset V' of the vertex set of G, $G[V']$ denotes the subgraph of G induced by V'; that is, the subgraph of G that has vertex set V' and edge set $\{\{a, b\} \in E(G) : a, b \in V'\}$.

Briefly, RANKEDTREE works as follows. The input to RANKEDTREE is a collection $\mathcal{P}$ of precedence constraints. If there exists a ranked phylogenetic tree that preserves $\mathcal{P}$, then RANKEDTREE builds a ranked hierarchy $(\mathcal{H}, r)$ on $\mathcal{L}(\mathcal{P})$ recursively beginning with the hierarchy $\{\mathcal{L}(\mathcal{P})\}$. At iteration k, a hierarchy $\mathcal{H}_k$ on $\mathcal{L}(\mathcal{P})$ is constructed by adding the blocks of particular partitions of minimal members of $\mathcal{H}_{k-1}$. Which blocks are added is determined by the components of a certain graph that is constructed at each iteration. This process ends when the constructed hierarchy contains $\{x\}$ for all $x \in \mathcal{L}(\mathcal{P})$. If there is no such ranked phylogenetic tree, then at some iteration, i say, no new blocks are added to $\mathcal{H}_{i-1}$ and RANKEDTREE returns "not compatible", indicating that there is no such ranked phylogenetic tree.

Theorem 3.3 is the main result of this section.

Theorem 3.3. *Suppose that* RANKEDTREE *is applied to a collection $\mathcal{P}$ of precedence constraints.*

(i) *If* RANKEDTREE *returns a ranked hierarchy on $\mathcal{L}(\mathcal{P})$, then this ranked hierarchy preserves $\mathcal{P}$.*

RANKEDTREE($\mathcal{P}$**)**

Input:
A collection $\mathcal{P}$ of precedence constraints
Output:
A ranked hierarchy $(\mathcal{H}, r)$ on $\mathcal{L}(\mathcal{P})$ that preserves $\mathcal{P}$ **or** the phrase "*not compatible*" if no such hierarchy exists
Data structures:
Partitions $\pi_1, \pi_2, \ldots$ of $\mathcal{L}(\mathcal{P})$
Hierarchies $\mathcal{H}_1, \mathcal{H}_2, \ldots$ on $\mathcal{L}(\mathcal{P})$
Graphs $G_1 = (\mathcal{L}(\mathcal{P}), E_1)$, $G_2 = (\mathcal{L}(\mathcal{P}), E_2), \ldots$
Rank function r

begin
$k \leftarrow 1$
$\mathcal{H}_k \leftarrow \{\mathcal{L}(\mathcal{P})\}$
$\pi_k \leftarrow \{\mathcal{L}(\mathcal{P})\}$
repeat while π_k contains a block with at least two elements
$E_k \leftarrow \{\{a, b\}$: there exists $B \in \pi_k$ and $c, d \in B$ with $(c, d) \prec (a, b) \in \mathcal{P}\}$
$G_k \leftarrow (\mathcal{L}(\mathcal{P}), E_k)$
if $G_k[B]$ is connected for all $B \in \pi_k$, **then**
return "not compatible" and **halt**
else
Let π_{k+1} be the partition of X given by the components of G_k.
$\mathcal{H}_{k+1} \leftarrow \mathcal{H}_k \cup \pi_{k+1}$
for all blocks B in π_k that are not blocks of π_{k+1} **do**
$r(B) \leftarrow k$
end (for)
$k \leftarrow k + 1$
end (if-else)
end (repeat)
$\mathcal{H} \leftarrow \mathcal{H}_k$
return $\mathcal{H}$ and r
end.

Figure 4. RANKEDTREE.

(ii) *If* RANKEDTREE *returns "not compatible", then there is no ranked hierarchy that preserves* $\mathcal{P}$.

Proof. To prove (i), suppose that RANKEDTREE returns a ranked hierarchy on $\mathcal{L}(\mathcal{P})$. Furthermore, suppose that $(c, d) \prec (a, b) \in \mathcal{P}$ and $r(c, d) = l$. Then c and d are elements of the same block of π_k for all $k \leq l$, and therefore $\{a, b\} \in E_k$ for all $k \leq l$. It follows that a and b are elements of the same block of π_k for all $k \leq l + 1$. Thus $r(c, d) < l + 1 \leq r(a, b)$. It follows that the returned rank hierarchy preserves $\mathcal{P}$.

Now consider (ii). Suppose that there exists a ranked hierarchy $(\mathcal{H}^*, r^*)$ that preserves $\mathcal{P}$, but RANKEDTREE returns "not compatible" at iteration k. Then, for all blocks B of π_k, the graph $G_k[B]$ is connected.

Let B be a block of π_k that minimizes

$$\max\{r^*(A) : A \in \mathcal{H}^*, B \subseteq A\},$$

and let A be the member of $\mathcal{H}^*$ that corresponds to this minimization. By the minimality of A, there exist disjoint members $A_1, A_2 \in \mathcal{H}^*$ such that $A_1, A_2 \subset A$, and $A_1 \cap B$ and $A_2 \cap B$ are both non-empty. Choose A_1 and A_2 to be maximal with these properties.

Because $G_k[B]$ is connected, there is an edge in this graph joining a vertex $a \in A_1 \cap B$ and a vertex $b \in A_2 \cap B$. This implies that there is a precedence constraint $(c, d) \prec (a, b) \in \mathcal{P}$ such that c and d are elements of the same block B' of π_k. Because $(\mathcal{H}^*, r^*)$ preserves $\mathcal{P}$, we have $r^*(c, d) < r^*(a, b)$. By the choice of a and b, we have $r^*(a, b) = r^*(A) = r^*(B)$. But then

$$r^*(B') \leq r^*(c, d) < r^*(a, b) = r^*(B),$$

contradicting the choice of B. This completes the proof of (ii).

Proposition 3.4. *For a collection $\mathcal{P}$ of precedence constraints,* RANKEDTREE *can be implemented to run in $O(|\mathcal{P}| + n^3)$ time, where $n = |\mathcal{L}(\mathcal{P})|$.*

Proof. The running time of RANKEDTREE is dominated by the time taken to complete the main loop. Provided the statement "not compatible" is not returned, each pass through this loop adds at least one cluster to the hierarchy being built. Because there are at most $2n - 2$ clusters that can be added, there are at most $O(n)$ iterations of this loop. Now, for each iteration k of this loop, we construct a graph, determine its components, and update the hierarchy assigning a ranking to certain members of the resulting hierarchy. For the first iteration, the graph G_1 takes $O(|\mathcal{P}|)$ time to construct. After that, for all k, we can construct G_{k+1} from G_k by deleting the appropriate edges. Over all iterations, the total time required to construct all these graphs is $O(|\mathcal{P}|)$. Lastly, for each iteration, it takes $O(n^2)$ time to determine the components, update the hierarchy, and assign a ranking as above.

Combining the remarks at the beginning of this section with Theorem 3.3 and Proposition 3.4, we get the following corollary immediately.

Corollary 3.5. *The algorithm* RANKEDTREE *provides a polynomial-time solution to* PHYLOGENETIC RANKING.

Figure 5. Two rooted phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$.

Remarks.

(i) Those readers familiar with BUILD will observe that RANKEDTREE works in a similar way. Indeed, if the input to RANKEDTREE arises from just a collection $\mathcal{R}$ of rooted phylogenetic trees and includes no relative divergence dates, then its output is identical to that returned by BUILD applied to $\mathcal{R}$. However, there is one significant difference. In BUILD, one considers a single minimal block of the current hierarchy at each iteration. Here, we need to consider all such blocks to guarantee an appropriate ranking.

(ii) The rooted phylogenetic tree that is associated with the ranked phylogenetic tree returned by RANKEDTREE when applied to a collection $\mathcal{R}$ of rooted phylogenetic trees and a collection of relative divergence dates is not necessarily a refinement of the rooted phylogenetic tree returned by BUILD when applied to $\mathcal{R}$. For example, consider the two rooted phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$ shown in Figure 5 and the relative divergence date "div(a, c) predates div(a, b)". Applying BUILD to $\mathcal{T}_1$ and $\mathcal{T}_2$ returns the rooted phylogenetic tree shown in Figure 6(a). However, applying RANKEDTREE to $\mathcal{T}_1$ and $\mathcal{T}_2$ as well as the relative divergence date, returns the ranked phylogenetic tree shown in Figure 6(b).

4. Divergence time intervals

In this section, we extend the input of RANKEDTREE to include time bounds on speciation events.

Let $\mathcal{T}$ be a rooted phylogenetic X-tree. A *divergence time function* for $\mathcal{T}$ is a function f from the set V^o of interior vertices of $\mathcal{T}$ into the set $\mathbf{R}^{>0}$ of positive reals, so that, if $v_1, v_2 \in V^o$ and v_1 is a proper ancestor of v_2, then $f(v_1) > f(v_2)$. The pair $(\mathcal{T}, f)$ is a *dated phylogenetic tree*. For a subset A of X of size at least two, we denote $f(\text{mrca}_{\mathcal{T}}(A))$ by $f(A)$. In the context of this

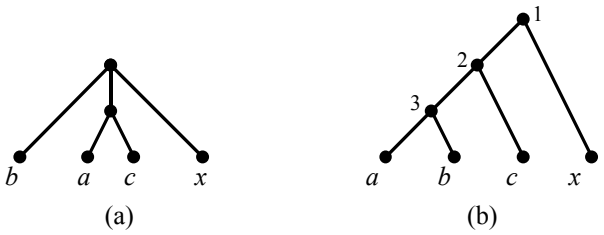

Figure 6. Rooted phylogenetic trees produced by (a) BUILD when applied to the trees $\mathcal{T}_1$ and $\mathcal{T}_2$ in Figure 5 and (b) RANKEDTREE when applied to $\mathcal{T}_1$ and $\mathcal{T}_2$ in Figure 5 as well as the relative divergence date "div(a, c) predates div(a, b)".

chapter, the values assigned to the interior vertices of $\mathcal{T}$ in this way represent the number of years ago that the corresponding speciation events occurred.

For species a and b, a *divergence time bound* on a and b is a lower or upper bound, denoted by $l(a, b)$ and $u(a, b)$, respectively, on the number of years ago that a and b diverged. If a and b have both a lower and upper bound, then $l(a, b) < u(a, b)$. If no information is provided concerning a lower or upper bound on the divergence time of a and b, we can always take $l(a, b) = 0$ and $u(a, b) = \infty$, respectively.

Let $\mathcal{I}$ be a collection of divergence time bounds. We denote the set $\cup\{a, b\}$ over all $l(a, b)$, $u(a, b) \in \mathcal{I}$ by $\mathcal{L}(\mathcal{I})$. Furthermore, $\mathcal{I}$ is *preserved* by a dated phylogenetic tree $(\mathcal{T}, f)$ if $l(a, b) < f(a, b)$ and $f(a, b) < u(a, b)$ for all $l(a, b)$, $u(a, b) \in \mathcal{I}$.

Farach *et al.* (1995) showed that, given a collection $\mathcal{I}$ of divergence time bounds, there is a polynomial-time solution for determining and, if possible, constructing a dated phylogenetic tree with leaf set $\mathcal{L}(\mathcal{I})$ that preserves $\mathcal{I}$. They refer to this problem as the "sandwich-to-ultrametric" problem, and the running time of the algorithm is $O(|\mathcal{I}| + n \log(n))$ where $n = |\mathcal{L}(\mathcal{I})|$. In this section, we show that there is a polynomial-time solution to the following extension of this problem.

Problem: PHYLOGENETIC DIVERGENCE TIMES

Instance: A collection $\mathcal{R}$ of rooted phylogenetic trees, a collection $\mathcal{D}$ of relative divergence dates, and a collection $\mathcal{I}$ of divergence time bounds.
Question: Does a dated phylogenetic tree with leaf set $\mathcal{L}(\mathcal{R}) \cup \mathcal{L}(\mathcal{D}) \cup \mathcal{L}(\mathcal{I})$ exist that displays $\mathcal{R}$ and preserves $\mathcal{D}$ and $\mathcal{I}$ and, if so, can we construct such a tree in polynomial time?

Like PHYLOGENETIC RANKING, one can obtain a polynomial-time solution to PHYLOGENETIC DIVERGENCE TIMES via RANKEDTREE. Indeed,

by transforming $\mathcal{I}$ to the collection of precedence constraints in the statement of Theorem 4.1, such a solution immediately follows from Theorems 3.3 and 4.1.

Theorem 4.1. *Let $\mathcal{I}$ be a collection of divergence time bounds and let*

$$\mathcal{P}(l, u) = \{(c, d) \prec (a, b) : l(c, d) \geq u(a, b), \text{ where } l(c, d), u(a, b) \in \mathcal{I}\}.$$

Let $\mathcal{T}$ be a rooted phylogenetic tree on $\mathcal{L}(\mathcal{I})$. Then there is a divergence time function f for $\mathcal{T}$ with *$(\mathcal{T}, f)$ preserving $\mathcal{I}$ if and only if there is a rank function r for $\mathcal{T}$ with $(\mathcal{T}, r)$ preserving $\mathcal{P}$.*

Proof. First suppose that there is divergence time function f for $\mathcal{T}$ such that $(\mathcal{T}, f)$ preserves $\mathcal{I}$. Let $v_1, v_2, \ldots, v_n$ be an ordering of the interior vertices of $\mathcal{T}$ such that

$$f(v_1) \geq f(v_2) \geq \ldots \geq f(v_n).$$

Let r be the function from the set V^o of interior vertices of $\mathcal{T}$ into the set of positive integers defined, for all i, by $r(v_i) = i$. We next show that $(\mathcal{T}, r)$ preserves $\mathcal{P}(l, u)$.

Let $(c, d) \prec (a, b) \in \mathcal{P}(l, u)$, and let $v_i = \text{mrca}_{\mathcal{T}}(c, d)$ and $v_j = \text{mrca}_{\mathcal{T}}(a, b)$. Because $(c, d) \prec (a, b) \in \mathcal{P}(l, u)$, we have $u(a, b) \leq l(c, d)$. Therefore,

$$f(a, b) < u(a, b) \leq l(c, d) < f(c, d)$$

and so $f(v_i) > f(v_j)$. This implies that $i < j$ and so $r(c, d) = r(v_i) < r(v_j) = r(a, b)$. It follows that $(\mathcal{T}, r)$ preserves $\mathcal{P}(l, u)$.

For the converse, suppose that there is a rank function r for $\mathcal{T}$ such that $(\mathcal{T}, r)$ preserves $\mathcal{P}(l, u)$. For each $v \in V^o$, let

$$(1) \qquad f_l(v) = \max\{\{0\} \cup \{l(a, b) : r(a, b) \geq r(v) \text{ and } l(a, b) \in \mathcal{I}\}\}$$

and

$$(2) \qquad f_u(v) = \min\{\{\infty\} \cup \{u(c, d) : r(c, d) \leq r(v) \text{ and } u(a, b) \in \mathcal{I}\}\}.$$

We show first that $f_l(v) < f_u(v)$ for all v. Suppose there exists an interior vertex v such that $f_l(v) \geq f_u(v)$. Then there are elements $a, b, c, d \in X$ such that $l(a, b) \geq u(c, d)$, $r(a, b) \geq r(v)$, and $r(c, d) \leq r(v)$. This implies that $(a, b) \prec (c, d) \in \mathcal{P}(l, u)$ and so $r(a, b) < r(c, d)$; a contradiction. Thus,

$f_l(v) < f_u(v)$ for all $v \in V^o$. Furthermore, by construction, if $r(v) < r(v')$ for vertices $v, v' \in V^o$, then $f_l(v) \geq f_l(v')$ and $f_u(v) \geq f_u(v')$.

Now let $v_1, v_2, \ldots, v_n$ be an ordering of the interior vertices of $\mathcal{T}$ such that

$$r(v_1) \leq r(v_2) \leq \ldots \leq r(v_n).$$

Let f be the function from V^o into the set of positive reals that is defined recursively as follows:

(i) set $f(v_1)$ so that $f_l(v_1) < f(v_1) < f_u(v_1)$, and
(ii) for all $i \in \{2, \ldots, n\}$, set $f(v_i)$ so that $f_l(v_i) < f(v_i) < \min\{f(v_{i-1}), f_u(v_i)\}$.

Observe that, if $v = \mathrm{mrca}_{\mathcal{T}}(a, b)$ for some $a, b \in X$, then

$$l(a, b) \leq f_l(v) < f(v) < f_u(v) \leq u(a, b).$$

Moreover, if v' is a proper descendent of v, then, by the minimality condition in (ii), $f(v') < f(v)$. We conclude that $(\mathcal{T}, f)$ preserves $\mathcal{I}$.

Example 4.2. To illustrate PHYLOGENETIC DIVERGENCE TIMES, suppose that we have as our instance $\mathcal{R} = \{\mathcal{T}_1, \mathcal{T}_2\}$, where $\mathcal{T}_1$ and $\mathcal{T}_2$ are as shown in Figure 7; $\mathcal{D}$ consisting of the statement "div(a, e) predates div(c, f)"; and $\mathcal{I}$ consisting of the divergence time bounds (in millions of years) $l(a, d) = 1$ and $u(a, d) = 3.5$, $l(a, b) = 4$ and $u(a, b) = 6$, and $l(c, f) = 3$ and $u(c, f) = 5$. Treating this instance as input, RANKEDTREE returns the ranked phylogenetic tree $(\mathcal{T}, r)$ shown in Figure 8.

Figure 9(a) shows $\mathcal{T}$ together with the values $f_l(v)$ and $f_u(v)$ for each interior vertex v given by (1) and (2), respectively. Furthermore, Figure 9(b) shows $\mathcal{T}$ together with a divergence time function f for $\mathcal{T}$ given by (i) and (ii) in the proof of Theorem 4.1.

Suppose that RANKEDTREE applied to collections $\mathcal{R}$, $\mathcal{D}$, and $\mathcal{I}$ of rooted phylogenetic trees, relative divergence dates, and divergence time bounds returns a ranked phylogenetic tree. Let $a, b \in \mathcal{L}(\mathcal{R}) \cup \mathcal{L}(\mathcal{D}) \cup \mathcal{L}(\mathcal{I})$. We would like to find the *most recent* (respectively, *most ancient*) *admissible dates* of the divergence of a and b. This is the smallest (respectively, largest) date measured from the present into the past at which a and b could have diverged so that RANKEDTREE applied to these collections together with the lower bound (respectively, upper bound) corresponding to this date returns a ranked phylogenetic tree. Note that these values are not given immediately by the output of RANKEDTREE applied to $\mathcal{R}$, $\mathcal{D}$ and $\mathcal{I}$. There are two reasons

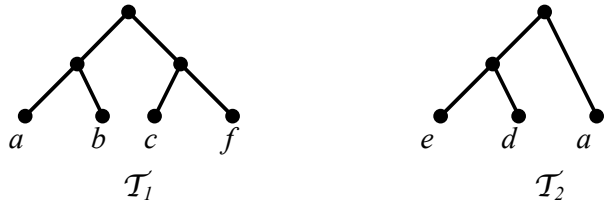

Figure 7. Two rooted phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$.

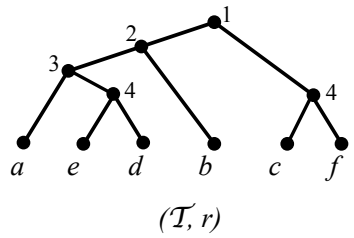

Figure 8. The ranked phylogenetic tree produced by RANKEDTREE when applied to $\mathcal{T}_1$ and $\mathcal{T}_2$ in Figure 7 together with the relative divergence dates and divergence time bounds given in Example 4.2.

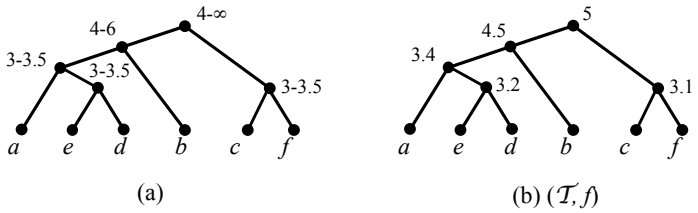

Figure 9. Assigning a divergence time function to the ranked phylogenetic tree shown in Figure 8. (a) The assignment of intervals as given by (1) and (2) in the proof of Theorem 4.1. (b) One allowable choice of divergence dates.

for this. First, for any interior vertex v, the interval $[f_l(v), f_u(v)]$ described in the proof of Theorem 4.1 does not necessarily contain all possible admissible dates for v in the tree returned by the algorithm. As a simple illustration of this, consider the pair d and e in Example 4.2. The most recent admissible divergence date for d and e is just before the present time (technically, $0 + \varepsilon$, where $\varepsilon > 0$) and not the value 3 as illustrated in Figure 9(a). Second, there might exist many rooted phylogenetic trees that display $\mathcal{R}$ and preserve $\mathcal{D}$ and $\mathcal{I}$ for which there is a divergence time function that allows more ancient or more recent divergence times for a particular pair of species than that given by the tree returned by RANKEDTREE. To avoid having to search a possibly exponential set of trees, it is therefore comforting to have the

following result, which shows that the problem can be solved in polynomial time.

Corollary 4.3. *Suppose that* RANKEDTREE *applied to collections* $\mathcal{R}$, $\mathcal{D}$, *and* $\mathcal{I}$ *returns a ranked phylogenetic tree*. Let $a, b \in \mathcal{L}(\mathcal{R}) \cup \mathcal{L}(\mathcal{D}) \cup \mathcal{L}(\mathcal{I})$. *Then there is a polynomial-time algorithm for determining the most ancient and most recent admissible dates for the divergence of a and b over all possible rooted phylogenetic trees and divergence time functions that display* $\mathcal{R}$ *and preserve* $\mathcal{D}$ *and* $\mathcal{I}$.

Proof. We will describe a simple polynomial algorithm that uses RANKEDTREE as a subroutine. It is possible that a more direct algorithm can be developed, but we do not explore this here.

We ensure first that u and l are defined for all pairs of species by extending $\mathcal{I}$ as follows: if $u(x, y) \notin \mathcal{I}$, then set $u(x, y) = \infty$ and, if $l(x, y) \notin \mathcal{I}$, then set $l(x, y) = 0$.

We describe a method for determining the most ancient admissible date of the divergence of a and b; an analogous result for most recent admissible date is similar. Let

$$T_{ab} = \{t : t \leq u(a, b) \text{ and } t = u(x, y) \text{ for some } u(x, y) \in \mathcal{I} \text{ with } u(x, y) > l(a, b)\}.$$

Then, for any $\varepsilon > 0$, the most ancient admissible date for the divergence of a and b is the maximum value of $t - \varepsilon$ over all $t \in T_{ab}$ with the property that RANKEDTREE applied to $\mathcal{R}$, $\mathcal{D}$, and $\mathcal{I}$ together with $l(a, b) = t - \varepsilon$ returns a ranked phylogenetic tree. Clearly, the running time of this method is polynomial in $|\mathcal{L}(\mathcal{P})|$.

5. Methodological applications of BUILD

The aim of this section is to illustrate how some problems in phylogenetics that, in general, appear computationally intractable (NP-hard) can be solved efficiently in certain cases by applying the simple supertree method BUILD. We have chosen four problems to illustrate the scope of this approach. In each of these problems, the computational part of the solution is done using BUILD. The first two consider the supertree problem for unrooted phylogenetic trees and the second two consider the compatibility of two-state characters.

5.1 Combining unrooted trees

Let $\mathcal{T}$ be a phylogenetic X-tree and let $X' \subseteq X$. Analogous to the rooted case, the *restriction of* $\mathcal{T}$ *to* X' is the phylogenetic X'-tree obtained by suppressing all degree-two vertices of the minimal subtree of $\mathcal{T}$ containing X'. We denote this restriction by $\mathcal{T}|X'$. For two phylogenetic trees $\mathcal{T}_1$ and $\mathcal{T}_2$ with $\mathcal{L}(\mathcal{T}_1) \subseteq \mathcal{L}(\mathcal{T}_2)$, we say that $\mathcal{T}_2$ *displays* $\mathcal{T}_1$ if $\mathcal{T}_2|\mathcal{L}(\mathcal{T}_1)$ is a refinement of $\mathcal{T}_1$. A collection $\mathcal{U}$ of phylogenetic trees is *compatible* if there is a phylogenetic tree $\mathcal{T}$ that displays every member of $\mathcal{U}$, in which case we say that $\mathcal{T}$ *displays* $\mathcal{U}$.

For a collection of phylogenetic trees, determining the compatibility of this collection is NP-complete in general (Bodlaender *et al.*, 1992; Steel, 1992). However, the first two problems show that particular cases can be solved efficiently.

5.1.1 Specified split

For our first problem, suppose we have a collection $\mathcal{U} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k\}$ of unrooted phylogenetic trees together with a split $A|B$ of $\cup_{i=1}^{k} \mathcal{L}(\mathcal{T}_i)$. We wish to determine if there is a phylogenetic tree that displays $\mathcal{U}$ and induces the split $A|B$. For the biologist, $A|B$ should be thought of as some "reliable" split of the taxa that is expected to be present in all acceptable supertrees of $\mathcal{U}$. The introduction of this split is important computationally because it can allow for the fast solution of the problem of determining the compatibility of $\mathcal{U}$. This is described in Theorem 5.1, the proof of which provides a polynomial-time algorithm.

Theorem 5.1. *Let* $\mathcal{U} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k\}$ *be a collection of phylogenetic trees and let* $A|B$ *be a split of* $X = \cup_{i=1}^{k} \mathcal{L}(\mathcal{T}_i)$. *Suppose that, for all* i, $\mathcal{L}(\mathcal{T}_i) \cap A$ *and* $\mathcal{L}(\mathcal{T}_i) \cap B$ *are both non-empty. Then there is a polynomial-time algorithm to determine if a phylogenetic* X*-tree that displays* $\mathcal{U}$ *and induces the split* $A|B$ *exists and, if so, to construct such a phylogenetic tree.*

Proof. If, for some i, the split $(A \cap \mathcal{L}(\mathcal{T}_i))|(B \cap \mathcal{L}(\mathcal{T}_i))$ is not a split of $\mathcal{T}_i$, then there is no phylogenetic tree satisfying the statement of the theorem. Thus, we can assume, for all i, that $\sigma_i = (A \cap \mathcal{L}(\mathcal{T}_i))|(B \cap \mathcal{L}(\mathcal{T}_i))$ is a split of $\mathcal{T}_i$. Now, for each i, let $\mathcal{T}_i^A$ and $\mathcal{T}_i^B$ denote the two rooted phylogenetic trees, with $\mathcal{L}(\mathcal{T}_i^A) = \mathcal{L}(\mathcal{T}_i) \cap A$ and $\mathcal{L}(\mathcal{T}_i^B) = \mathcal{L}(\mathcal{T}_i) \cap B$, obtained by deleting the unique edge of $\mathcal{T}$ corresponding to σ_i and distinguishing the end vertices of this edge as root vertices. Let $\mathcal{R}^A = \{\mathcal{T}_1^A, \mathcal{T}_2^A, \ldots, \mathcal{T}_k^A\}$ and $\mathcal{R}^B = \{\mathcal{T}_1^B, \mathcal{T}_2^B, \ldots, \mathcal{T}_k^B\}$.

If the application of BUILD to either $\mathcal{R}^A$ or $\mathcal{R}^B$ detects that either of these is incompatible, then no phylogenetic X-tree exists that displays $\mathcal{U}$ and that also induces the split $A \mid B$. Otherwise, we will let $\mathcal{T}^A$ and $\mathcal{T}^B$ denote the rooted phylogenetic trees returned by BUILD when applied to $\mathcal{R}^A$ and $\mathcal{R}^B$, respectively. Thus, assume that two such trees are returned. Let $\mathcal{T}$ be the phylogenetic X-tree obtained by adjoining the roots of $\mathcal{T}^A$ and $\mathcal{T}^B$ with a new edge. It now follows that $\mathcal{T}$ displays $\mathcal{U}$, and $A \mid B$ is an induced split of $\mathcal{T}$.

5.1.2 Quartet compatibility

Binary phylogenetic trees with just four leaves — *quartet trees* — play a special role in phylogenetics. For example, the problem of determining the compatibility of a set of unrooted phylogenetic trees can be reduced to determining the compatibility of a set of associated quartet trees (see Steel, 1992). Furthermore, many techniques for reconstructing phylogenies (such as the "quartet puzzling" approach of Strimmer and von Haeseler, 1996) are based on quartet trees. Supertree techniques based on quartet methods have also been proposed (e.g., Piaggio-Talice *et al*., 2004).

We will write $xy \mid uv$ to denote the quartet tree in which the interior edge separates the pair of leaves x, y from u, v. For a set $\mathcal{Q}$ of quartet trees, we let

$$\mathcal{L}(\mathcal{Q}) = \bigcup_{\mathcal{T} \in \mathcal{Q}} \mathcal{L}(\mathcal{T}).$$

The proof of Theorem 5.2 describes an algorithm to determine the compatibility of an arbitrary set of quartet trees. Although not in polynomial time, this method might be practical when $\mathcal{Q}$ is reasonably small. McMorris *et al.* (1994) provide a more general algorithm for this problem, but it has complexity $O(n^{k+1})$ where $k = |\mathcal{Q}|$ and $n = |\mathcal{L}(\mathcal{Q})|$, which, in general, will be larger than the complexity of the following approach.

Theorem 5.2. *Let $\mathcal{Q}$ be a set of k quartet trees. Then there is an $O(k^2 2^k)$-time algorithm for determining the compatibility of $\mathcal{Q}$.*

Proof. First note that, for a collection $\mathcal{R}$ of k rooted triples, BUILD can be implemented to run on $\mathcal{R}$ in $O(k^2)$ time (see Aho *et al.*, 1981).

To describe an algorithm that satisfies the statement of the theorem, let x be an element not in $\mathcal{L}(\mathcal{Q})$ and, for each $q = ab \mid cd$ in $\mathcal{Q}$, consider the collections of rooted triples $S_1(q) = \{cd \mid a, cd \mid b\}$ and $S_2(q) = \{ab \mid c, ab \mid d\}$. For each $q \in \mathcal{Q}$, every rooted phylogenetic tree $\mathcal{T}$ that displays either $S_1(q)$ or $S_2(q)$ has the property that the phylogenetic tree obtained from $\mathcal{T}$ by not distinguishing the root and suppressing this vertex if it has degree two,

displays q. Now, for each of the 2^k functions $\pi : \mathcal{Q} \to \{1, 2\}$, BUILD can determine the compatibility of $\cup_{q\in\mathcal{Q}} S_{\pi(q)}(q)$ in $O(nk)$ time. Moreover, it is checked easily that $\mathcal{Q}$ is compatible if and only if $\cup_{q\in\mathcal{Q}} S_{\pi(q)}(q)$ is compatible for some choice of π. The theorem now follows.

We note in passing that the running time of the algorithm in Theorem 5.2 could be improved slightly by invoking the approach of Henzinger *et al.* (1999).

5.2 Two-state character compatibility

In the next two problems, we consider characters that assign one of two states to some or all of the species. More precisely, a *two-state character* on X is a function $\chi : X' \to \{0, 1\}$, where X' is some subset of X. Allowing X' to be a strict subset of X allows for uncertainty or ambiguity of the character state of certain species in X. The two-state character χ is *convex* on a phylogenetic X-tree if there exists a split $A \mid B$ of $\mathcal{T}$ for which $\chi^{-1}(0) \subseteq A$ and $\chi^{-1}(1) \subseteq B$. Furthermore, a collection C of two-state characters on X is *compatible* if there exists a phylogenetic X-tree on which all the characters in C are convex, in which case we say that $\mathcal{T}$ *displays* C. The biological relevance of these concepts, in particular their close connection with the concept of homoplasy, is described by Semple and Steel (2002). In general, determining the compatibility of C is an NP-complete problem.

In both problems presented here, we consider the following collection of rooted phylogenetic trees. For a two-state character $\chi : X' \to \{0, 1\}$, let $\mathcal{T}_\chi$ denote the rooted phylogenetic X'-tree that has exactly two interior vertices with the non-root vertex adjacent to the leaves in $\chi^{-1}(1)$ and the root vertex adjacent to the leaves in $\chi^{-1}(0)$. For a collection C of two-state characters on X, let

$$\mathcal{R}(C) = \{\mathcal{T}_\chi : \chi \in C\}. \tag{3}$$

5.2.1 Directed case

Let χ be a two-state character on X and let $\mathcal{T}$ be a rooted phylogenetic X-tree. We say that χ is *convex on* $\mathcal{T}$ *relative to* $0 \to 1$ if there is a function $f : V(\mathcal{T}) \to \{0, 1\}$ that extends χ so that

(i) there is no arc (u, v) of $\mathcal{T}$ with $f(u) = 1$ and $f(v) = 0$, and
(ii) there is at most one arc (u, v) of $\mathcal{T}$ with $f(u) = 0$ and $f(v) = 1$.

Here, we view the edges of a rooted phylogenetic tree as arcs directed away from the root. A collection C of two-state characters on X is *compatible relative to* $0 \rightarrow 1$ if there is a rooted phylogenetic X-tree $\mathcal{T}$ on which all the characters in C are convex relative to $0 \rightarrow 1$, in which case we say that $\mathcal{T}$ *displays* C *relative to* $0 \rightarrow 1$.

The setup of the previous paragraph is useful for modeling situations in which 0 represents some specific "ancestral" state and 1 represents a specific "derived" state, and where transitions are rare and always proceed from the ancestral to the derived state. If there is uncertainty as to whether the state for species x is ancestral or derived, no state is assigned to that species. This formulation of compatibility has been applied, for example, to certain molecular genetic data known as SINEs ("short interspersed nuclear elements"), where the states 0 and 1 denote the absence and presence, respectively, of a particular sequence inserted into a particular region of a genome (see Pe'er *et al.*, 2000). The present setting is also relevant to a modification of the MRP technique for supertree construction (Baum, 1992; Ragan, 1992) described by Bininda-Emonds and Bryant (1998), where reversals (i.e., $1 \rightarrow 0$ transitions) are prohibited during the parsimony optimization step.

The question of determining the compatibility of a collection of two-state, directed characters has been shown to have a polynomial time solution by Pe'er *et al.* (2000), and, as a special case of a more general result, by Benham *et al.* (1995). The following theorem provides a further polynomial-time approach to the problem, showing that it can be regarded as a special case of the supertree problem for rooted phylogenetic trees.

Theorem 5.3.

(i) *Let* $\chi : X' \rightarrow \{0, 1\}$ *be a two-state character on* X *and let* $\mathcal{T}$ *be a rooted phylogenetic* X*-tree. Then* χ *is convex on* $\mathcal{T}$ *relative to* $0 \rightarrow 1$ *if and only if* $\mathcal{T}$ *displays* $\mathcal{T}_\chi$.

(ii) *Let* C *be a collection of two-state characters on* X*. Then a rooted phylogenetic* X*-tree* $\mathcal{T}$ *displays* C *relative to* $0 \rightarrow 1$ *if and only if* $\mathcal{T}$ *displays* $\mathcal{R}(C)$

Proof. Now χ is convex on $\mathcal{T}$ relative to $0 \rightarrow 1$ if and only if there exists a cluster A of $\mathcal{T}$ such that

$$\chi^{-1}(1) \subseteq A \text{ and } \chi^{-1}(0) \subseteq X - A. \tag{4}$$

Furthermore, (4) holds if and only if $\chi^{-1}(1)$ is a cluster of $\mathcal{T} \mid X'$. Because $\chi^{-1}(1)$ is the only -non-trivial cluster of $\mathcal{T}_\chi$, this last condition holds if and

only if $\mathcal{T}$ displays $\mathcal{T}_\chi$. This completes the proof of (i). Part (ii) is an immediate consequence of (i).

5.2.2 Undirected case

In this problem, we again consider two-state characters taking values in the set {0, 1} except that we no longer regard 0 as "ancestral". The notion of convexity for a character on a phylogenetic tree $\mathcal{T}$ as defined before the previous problem is consistent with the concept of characters evolving without homoplasy — it allows at most one occurrence of either the transition $0 \rightarrow 1$ or the transition $1 \rightarrow 0$ on $\mathcal{T}$.

Theorem 5.4. *Let $C = \{\chi_1, \chi_2, \ldots, \chi_k\}$ be a collection of two-state characters on X. Suppose that, for all $i, j \in \{1, 2, \ldots, k\}$,*

$$\chi_i^{-1}(0) \cap \chi_j^{-1}(0) \neq \emptyset.$$

Then there is a polynomial-time algorithm to determine if C is compatible and, if so, to construct a phylogenetic X-tree that displays C.

Proof. We establish Theorem 5.4 by showing that C is compatible if and only if the associated collection $\mathcal{R}(C)$ of rooted phylogenetic trees (described by (3)) is compatible, and that, when this occurs, the rooted phylogenetic tree returned by BUILD when applied to $\mathcal{R}(C)$ immediately gives a phylogenetic tree that displays C.

First, suppose that $\mathcal{R}(C)$ is compatible and that $\mathcal{T}$ is a rooted phylogenetic X-tree that displays $\mathcal{R}(C)$ (e.g., the rooted phylogenetic tree produced by BUILD when applied to $\mathcal{R}(C)$). Let $\mathcal{T}^{-\rho}$ be the phylogenetic X-tree obtained from $\mathcal{T}$ by not distinguishing the root ρ. Note that if ρ has degree two, then this vertex is also suppressed. By Theorem 5.3, for each $\chi \in C$, χ is convex on $\mathcal{T}$ relative to $0 \rightarrow 1$. But this implies that χ is convex on $\mathcal{T}^{-\rho}$. It follows that $\mathcal{T}^{-\rho}$ displays C.

Now suppose that C is compatible and that $\mathcal{T}$ is a phylogenetic X-tree that displays C. For all $i \in \{1, 2, \ldots, k\}$, let V_i denote the vertex set of the minimal subtree of $\mathcal{T}_i$ that contains the leaves of $\chi_i^{-1}(0)$. Because $\chi_i^{-1}(0) \cap \chi_j^{-1}(0) \neq \emptyset$, for all $i, j \in \{1, 2, \ldots, k\}$, it follows that

$$V_i \cap V_j \neq \emptyset$$

for all i, j. Thus, by the Helly intersection property of subtrees of a tree (see Golumbic, 1980), there exists a vertex v_ρ of $\mathcal{T}$ such that

$$v_\rho \in \bigcap_{i=1}^{k} V_i .$$

If v_ρ is an interior vertex of $\mathcal{T}$, then it is checked easily that the rooted phylogenetic X-tree obtained by rooting $\mathcal{T}$ on v_ρ displays $\mathcal{R}(C)$. If v_ρ is not an interior vertex, then the rooted phylogenetic tree obtained by rooting $\mathcal{T}$ on the vertex adjacent to v_ρ displays $\mathcal{R}(C)$. In both cases, the resulting rooted phylogenetic X-tree displays $\mathcal{R}(C)$. This completes the proof of the theorem.

Note that a particular case where the intersection condition in Theorem 5.4 applies is when

$$|\chi_i^{-1}(0)| > {}^1\!/_2 |X|$$

for all $i \in \{1, 2, \ldots, k\}$. In particular, we have the following corollary.

Corollary 5.5. *Let C be a collection of two-state characters on X. If each character in C assigns a strict majority of elements of X to some particular state, then there is a polynomial-time algorithm for determining the compatibility of C.*

6. Conclusion

Supertree methods continue to be extended and applied in various ways to study interesting problems in phylogenetics. In this chapter, we have considered how the BUILD algorithm can be extended to account for relative and absolute divergence date information. In the last section, we also described applications of BUILD to some other problems arising in phylogeny reconstruction.

One of the limitations of these approaches is that they are essentially "all-or-nothing"; that is, they return a tree (and dates) only if the input trees (and input dates) are compatible. But, incompatibility tends to be the rule rather than the exception for real biological data. However, the point in developing "exact" methods such as BUILD and the extensions described in this paper is that they form the basis for methods that apply in the general (incompatible) setting. Indeed, the BUILD approach was extended recently to the supertree method MINCUTSUPERTREE (Semple and Steel, 2000; also Page, 2002) that returns a supertree for every input of rooted phylogenetic trees. We believe that similar techniques can be applied to extend algorithms such as RANKEDTREE, and we hope to explore this in further work.

Acknowledgements

We thank the New Zealand Marsden Fund (UOC–MIS–005) for supporting this research, and Mike Charleston, Olaf Bininda-Emonds and an anonymous referee for helpful comments on an earlier version of this chapter.

References

AHO, A. V., SAGIV, 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. *SIAM Journal on Computing* 10:405–421.

BARTHÉLEMY, J.-P. AND GUÉNOCHE, A. 1991. *Trees and Proximity Representations. John Wiley and Sons, United Kingdom.*

BAUM, B. R. 1992. Combining trees as a way of combining datasets for phylogenetic inference, and the desirability of combining gene trees. *Taxon* 41:3–10.

BENHAM, C., KANNAN, S., PATERSON, M., AND WARNOW, T. 1995. Hen's teeth and whale's feet: generalized characters and their compatibility. *Journal of Computational Biology* 2:515–525.

BININDA-EMONDS, O. R. P. AND BRYANT H. N. 1998. Properties of matrix representation with parsimony analyses. *Systematic Biology* 47:497–508.

BODLAENDER, H. L., FELLOWS, M. R., AND WARNOW, T. J. 1992. Two strikes against perfect phylogeny. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, volume 623, pp. 273–283. Springer-Verlag, Berlin.

BRYANT, D. AND STEEL, M. 1995. Extension operations on sets of leaf-labelled trees. *Advances in Applied Mathematics* 16:425–453.

CONSTANTINESCU, M. AND SANKOFF, D. 1995. An efficient algorithm for supertrees. *Journal of Classification* 12:101–112.

FARACH, M., KANNAN, S., AND WARNOW, T. 1995. A robust model for finding optimal evolutionary trees. *Algorithmica* 13:155–179.

GOLUMBIC, M. C. 1980. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York.

HENZINGER, M. R., KING, V., AND WARNOW, T. 1999. Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. *Algorithmica*, 24:1–13.

MCMORRIS, F. R., WARNOW, T. J., AND WIMER, T. 1994. Triangulating vertex-colored graphs. *SIAM Journal on Discrete Mathematics* 7:296–306.

NG, M. P. AND WORMALD, N. C. 1996. Reconstruction of rooted trees from subtrees. *Discrete Applied Mathematics*, 69:19–31.

PAGE, R. D. M. 2002. Modified mincut supertrees. In R. Guigó and D. Gusfield (eds), *Proceedings of the Second International Workshop on Algorithms in Bioinformatics WABI 2002*, pp. 537–552, Springer-Verlag, New York.

PE'ER, I., SHAMIR, R., AND SHARAN, R. 2000. Incomplete directed perfect phylogeny. In D. Sankoff (ed.), *Proceedings of the Eleventh Symposium on Combinatorial Pattern Matching CPM*, Lecture Notes in Computer Science 1848:143–153. Springer, New York.

PIAGGIO-TALICE, R., BURLEIGH, J. G., AND EULENSTEIN, E. 2004. Quartet supertrees. In O. R. P. Bininda-Emonds (ed). *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pp. 173–191. Kluwer Academic, Dordrecht, the Netherlands.

RAGAN, M. A. 1992. Phylogenetic inference based on matrix representation of trees. *Molecular Phylogenetics and Evolution* 1:53–58.

SEMPLE, C. 2003. Reconstructing minimal rooted trees. *Discrete Applied Mathematics* 127:489–503.

SEMPLE, C. AND STEEL, M. 2000. A supertree method for rooted trees. *Discrete Applied Mathematics* 105:147–158.

SEMPLE, C. AND STEEL, M. 2002. Tree reconstruction from multi-state characters. *Advances in Applied Mathematics* 28:169–184.

SEMPLE, C. AND STEEL, M. 2003. *Phylogenetics.* Oxford University Press, Oxford.

STEEL, M. 1992. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification* 9:91–116.

STRIMMER, K. AND VON HAESELER, A. 1996. Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Molecular Biology and Evolution* 13:964–969.