Steiner Trees in Industries
D.-Z. Du and X. Cheng (Eds.) pp. – – –
©2000 Kluwer Academic Publishers

The Perfect Phylogeny Problem

David Fernández-Baca
Department of Computer Science and
Graduate Program in Bioinformatics and Computational Biology
Iowa State University, Ames, IA 50011
E-mail: `fernande@cs.iastate.edu`

Contents

1 Introduction

A *phylogeny* is a tree representation of the evolutionary history of a set of taxa (organisms, biological sequences, populations, or languages). Thus, phylogeny construction is among the basic computational problems in biology and linguistics. We will be concerned here with taxa described by the *states* they exhibit on a set of *characters*. A sample data set and a phylogeny

	a	b	c	d	e	f
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0

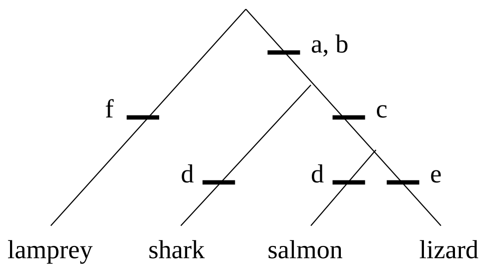

Figure 1: A data matrix and a phylogeny. The characters are: (a) paired fins, (b) jaws, (c) large dermal bones, (d) fin rays, (e) lungs, (f) rasping tongue.

for it (both adapted from [41]) are shown in Figure 1. The data is binary and in matrix form, rows are indexed by taxa and columns by characters. Entry (i,j) is "1" if taxon i has character j and it is "0" if the character is absent. The phylogeny shown is rooted and assumes that the taxa descend from a common ancestor where all characters are absent. Points at which characters emerge are indicated by labeled bars. [1]

The phylogeny of Figure 1 has the fewest state changes among all rooted trees for the given set of taxa. In evolutionary biology, this is referred to as a *most parsimonious tree*. While the validity of parsimony has been debated, it can be justified on biological grounds (see, e.g, [48]) and is widely used in practice, as attested by the popularity of software such as PAUP (which stands for "Phylogenetic analysis using parsimony") [54]. Finding a most parsimonious tree for a set of taxa is a Steiner tree problem: The given points are the taxa, which, if characters are binary, are represented by 0-1 vectors in a space of dimension equal to the number of characters. The Steiner points are the labels of the internal nodes, which are themselves 0-1 vectors (see Figure 2). The distance between two labels is the number of characters in which they differ; that is, it equals the Hamming distance between them. Hence the sum of the distances between neighbors in the tree is the total amount of evolutionary change implied by the tree.

In our sample phylogeny, fin rays emerge independently in two branches.

[1]The point of view reflected by this and much of the subsequent discussion — that is, trees as models of evolutionary history — has to some extent been abandoned by modern-day systematics (the field that studies the classification of taxa). Instead, trees of the sort considered here are viewed as models of character distribution, with no implicit time axis, and are often called *cladograms*, to distinguish them from evolutionary trees. For further discussion of this issue, see [41].

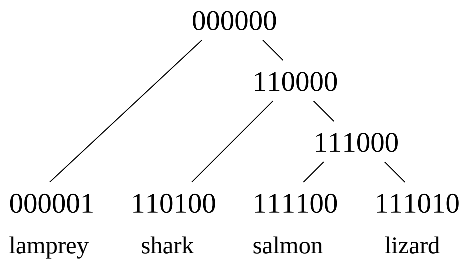

Figure 2: The phylogeny of Figure 1, with internal nodes labeled by the taxa implied by the character-state transitions in that tree.

This is called *homoplasy*, and is generally inescapable in real data. In the words of Page and Holmes [48], "homoplasy is a poor indicator of evolutionary relationships, because similarity does not reflect shared ancestry." Sets of characters that admit phylogenies without homoplasy are said to be *compatible*. The problem of determining whether a set of characters is compatible has deserved the attention of systematicians since the mid-60's. A good survey of the development of the field, with references to the early literature, has been written by Estabrook [20]. In the computer science literature, where the problem seems to have first been introduced by Gusfield [31, 32], phylogenies that avoid homoplasy are called *perfect* and the character compatibility problem is called the *perfect phylogeny problem*.

Our goal here is to give an overview of the elegant mathematical and algorithmic structure of the perfect phylogeny problem, identifying the common threads, such as tree compatibility and the notion of intersection graphs, that unify the field. Elegance comes at price: only in restricted cases does data admit a perfect phylogeny (for a case where the model fits well, see [47]). Coping with incompatibility is thus an important issue; we turn to it at the end of this chapter. To our knowledge, the only prior survey of scope similar to ours is Steel's lucid 1992 paper [52], which remains essential reading for anyone interested in the area. An update on the subject seems appropriate, now that the field has attained matured.

Perfect phylogenies are not the only approach to evolutionary tree construction. The reader interested in other methods, such as maximum likelihood, can consult the excellent references that have appeared in recent years [53, 41, 48]. In these same works, the reader can learn about other important topics that fall outside of the scope of this paper, such as the philosophical basis for parsimony, and the wider context within which phylogenetics and, in particular, character-based methods are studied.

Organization. In the next section, we provide basic problem definitions. Section 3 explains the relationship between perfect phylogenies and Steiner trees, showing the precise sense in which the former are a special case of the latter. Section 4 considers the polynomially-solvable special case of cladistic characters, where ordering relationships between states are specified. The discussion there centers on binary characters; multi-state characters are handled by reduction to binary ones. Section 5 discusses tree compatibility, focusing on the important polynomially-solvable special case of rooted trees and its relationship with testing the compatibility of incomplete directed characters. Section 6 describes the polynomial-time solution of the perfect phylogeny problem when the number of character states is fixed, a case that arises, for example, in molecular data (four or 20 states, depending on whether DNA or proteins are considered). Section 7 describes the relationship between restricted triangulations and perfect phylogenies and describes how this connection can be exploited to obtain a polynomial-time algorithm when the number of characters is fixed. Finally, Section 8 summarizes the main themes of the paper and discusses polymorphism and the problem of dealing with incompatibility.

2 Preliminaries

A *taxon* over a set of m characters $\mathcal{C}$ is a vector $\mathbf{s} \in \mathbb{Z}^m$ (where $\mathbb{Z}$ is the set of integers); $\mathbf{c}(\mathbf{s})$ is the *state of* $\mathbf{s}$ *on character* $\mathbf{c}$ or the *state of* $\mathbf{c}$ *for* $\mathbf{s}$. The set of allowed *states* for $\mathbf{c}(\mathbf{s})$ is denoted by $\mathcal{A}_{\mathbf{c}}$. We assume that $\mathcal{A}_{\mathbf{c}} = \{0, \ldots, r_{\mathbf{c}-1}\}$, for some integer $r_{\mathbf{c}}$. Let $r = \max_{\mathbf{c} \in \mathcal{C}} r_{\mathbf{c}}$. It is sometimes convenient to introduce two extra states outside of $\mathcal{A}_{\mathbf{c}}$: ?, which stands for an unknown state, and $*$, the dummy state.

Let $\mathcal{S}$ be a set of n taxa. The set can be represented by an $n \times m$ *character-state matrix* $M = [m_{ij}]$, where m_{ij} is the state of taxon i on character j. Taxa in $\mathcal{S}$ may have unknown states, but no dummy states. $\mathcal{C}$ is said to be *incomplete* if $\mathbf{c}(\mathbf{s}) = ?$ for some $\mathbf{s} \in \mathcal{S}$; otherwise, $\mathcal{C}$ is *complete*. For $\mathbf{c} \in \mathcal{C}$ and $\alpha \in \mathcal{A}_{\mathbf{c}} \cup \{?\}$, let $\mathcal{S}_{\mathbf{c},\alpha} = \{\mathbf{s} \in \mathcal{S} : \mathbf{c}(\mathbf{s}) = \alpha\}$. Then $\{\mathcal{S}_{\mathbf{c},\alpha} : \alpha \in \mathcal{A}_{\mathbf{c}} \cup \{?\}\}$ is a partition of $\mathcal{S}$. Characters of this sort, which specify no relationship between states, are sometimes called *qualitative characters*.

A *phylogeny* for $\mathcal{S}$ is a tree with exactly n leaves, each labeled by a distinct element of $\mathcal{S}$. An *internally-labeled phylogeny* for $\mathcal{S}$, is a phylogeny T for $\mathcal{S}$ where every node v is labeled by a taxon $\mathbf{s}_v$. Labels of internal

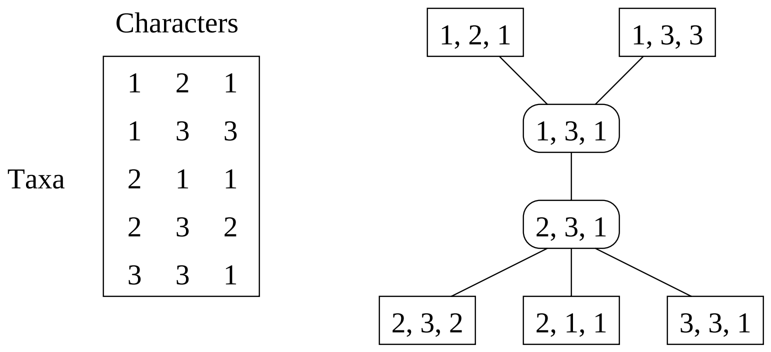

Figure 3: A character-state matrix and an internally-labeled perfect phylogeny.

nodes represent hypothetical ancestors to the elements of $\mathcal{S}$.

Consider for the moment complete characters. Character $\mathbf{c}$ is *convex* in an internally-labeled phylogeny T if for every $\alpha \in \mathcal{A}_{\mathbf{c}}$, the set of nodes $\{v \in V(T) : \mathbf{c}(\mathbf{s}_v) = \alpha\}$ induces a subtree of T. An internally-labeled phylogeny T is *perfect* if every $\mathbf{c} \in \mathcal{C}$ is convex in T. If $\mathcal{S}$ admits a perfect phylogeny, $\mathcal{C}$ is said to be *compatible*. The *perfect phylogeny problem* (also known as the *character compatibility problem*) is to determine if a given set of taxa $\mathcal{S}$ on a set of characters $\mathcal{C}$ has a perfect phylogeny (see Figure 3).

For incomplete characters, we say that $\mathcal{S}$ admits a perfect phylogeny if there is some way to replace each question-mark by an allowed state, such that the resulting set of taxa has a perfect phylogeny.

We now give an alternative definition of convexity for phylogenies that are only leaf-labeled. This definition obviates the need to distinguish between complete and incomplete characters and is more appropriate for cladistic characters, to be introduced below. Given a phylogeny T, a character $\mathbf{c}$ and a state α of $\mathbf{c}$, let $T_{\mathbf{c},\alpha}$ denote the minimal subtree of T that connects the set $\mathcal{S}_{\mathbf{c},\alpha}$. Then, $\mathbf{c}$ is *convex* if for any two distinct states $\alpha, \beta \in \mathcal{A}_{\mathbf{c}}$, $T_{\mathbf{c},\alpha}$ and $T_{\mathbf{c},\beta}$ are vertex-disjoint. Clearly, any leaf-labeled phylogeny that satisfies this definition of convexity for $\mathbf{c}$ has an internal labeling that satisfies the original definition.

A perfect phylogeny T for $\mathcal{S}$ is *minimal* if, for every edge e in T, the tree that results from contracting e cannot be vertex-labeled to make it a perfect phylogeny for $\mathcal{S}$. Clearly, $\mathcal{S}$ has a perfect phylogeny if and only if it has a minimal perfect phylogeny.

A *cladistic character* $\mathbf{c}$ is a qualitative character together with a tree

	a	**b**
1	1	0
2	2	2
3	2	1
4	1	1
5	0	0
6	2	2
7	0	0
8	1	1

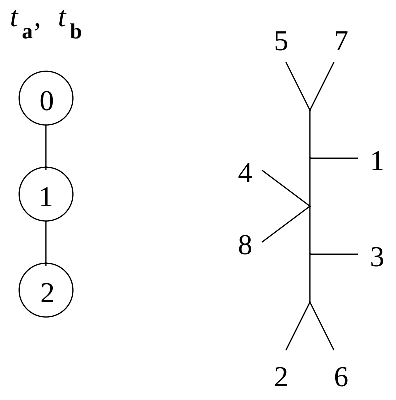

Figure 4: Two undirected cladistic characters with the same character-state tree and a phylogeny consistent with both.

$t_{\mathbf{c}}$ whose node set is $\mathcal{A}_{\mathbf{c}}$; $t_{\mathbf{c}}$ is called a *character-state tree* (see Figure 4). Character $\mathbf{c}$ is *directed* if $t_{\mathbf{c}}$ is rooted; otherwise, it *undirected* [19]. Character state trees specify an ordering among the states that must be respected by a phylogeny. More precisely, phylogeny T is *consistent* with a cladistic character $\mathbf{c}$ if the following holds for each edge (u, v) of $t_{\mathbf{c}}$. Let $t_{\mathbf{c},u}$ be the connected component of $t_{\mathbf{c}} - (u, v)$ containing u (v) and let A_u (A_v) be the set of all taxa $\mathbf{s}$ such that $\mathbf{c}(\mathbf{s}) = \alpha$, where α is a node of $t_{\mathbf{c},u}$. Then there exists an edge e in T such that the leaf label set of one component of $T - e$ contains A_u, while the corresponding set for the other component contains A_v (see Figure 4). Note that if T is consistent with $\mathbf{c}$, then $\mathbf{c}$'s underlying qualitative character is convex in T according to our second definition of convexity.

T is *consistent* with a set of characters $\mathcal{C}$ if it is consistent with every $\mathbf{c} \in \mathcal{C}$. A set $\mathcal{C}$ of cladistic characters is *compatible* if there is a phylogeny T that is consistent with $\mathcal{C}$. The problem of determining whether there exists a phylogeny consistent with a set of cladistic characters is the *compatibility problem for (directed or undirected) cladistic characters* or the *perfect phylogeny problem for (directed or undirected) cladistic characters.*

3 The Steiner Tree Problem in Phylogeny

Before embarking on the study of perfect phylogenies, let us examine their relationship with minimum Steiner trees. We need some definitions. For any two taxa $\mathbf{s}$ and $\mathbf{t}$ let $dist(\mathbf{s}, \mathbf{t})$ denote the Hamming distance between $\mathbf{s}$ and $\mathbf{t}$, that is the number of characters $\mathbf{c}$ such that $\mathbf{c}(\mathbf{s}) \neq \mathbf{c}(\mathbf{t})$. The *length* of an internally-labeled phylogeny T is

$$length(T) = \sum_{(u,v) \in E(T)} dist(\mathbf{s}_u, \mathbf{s}_v).$$

The *Steiner tree problem in phylogeny* is to find a minimum-length internally-labeled phylogeny for a set of taxa $\mathcal{S}$.

Observe that in *any* internally-labeled phylogeny T for $\mathcal{S}$, for every character $\mathbf{c}$ the number of edges (u, v) where $\mathbf{c}(\mathbf{s}_u) \neq \mathbf{c}(\mathbf{s}_v)$ is at least $r_\mathbf{c} - 1$ (assuming each state of $\mathbf{c}$ is exhibited by some taxon). Furthermore, if T is *perfect*, it must have an internal labeling with *exactly* $r_\mathbf{c} - 1$ edges where $\mathbf{c}(\mathbf{s}_u) \neq \mathbf{c}(\mathbf{s}_v)$. Thus, we can establish the following relationship between Steiner trees and perfect phylogenies.

Theorem 3.1 *$\mathcal{S}$ has a perfect phylogeny if and only if the minimum length of an internally-labeled phylogeny for $\mathcal{S}$ is $\sum_{\mathbf{c} \in \mathcal{C}} (r_\mathbf{c} - 1)$.*

Any character-state transition in a phylogeny beyond the lower bound of $\sum_{\mathbf{c} \in \mathcal{C}} (r_\mathbf{c} - 1)$ requires homoplasy. In the phylogenetics literature, such transitions are sometimes called *additional transitions*.

The Steiner tree problem in phylogeny is NP-hard even for binary characters [28]. In fact, it shares the non-approximability properties of its counterpart in arbitrary metric spaces [26]. On the positive side, given a phylogeny T for $\mathcal{S}$, an internal labeling that minimizes $length(T)$ can be found efficiently [27, 33]. Furthermore, moderately large instances can be solved in a reasonable amount of time by branch and bound (see, e.g., [24, 54]). Finally, known approximation algorithms for the Steiner tree problem in networks can be adapted to phylogenies [57, 35].

The above version of the Steiner tree problem in phylogeny corresponds to *Wagner parsimony* [55], in which all state transitions are equally likely. Two kinds of restrictions on the transitions are sometimes considered:

- In (binary) *Dollo parsimony* [16, 23], a character state can emerge exactly once in the tree; however, it may be lost multiple times.

- In *Camin-Sokal optimization* [13], a character state may emerge multiple times, but is never lost.

Even with these restrictions, the Steiner tree problem in phylogeny remains NP-hard [15]. However, in both cases, the optimal internal labeling for a given tree can be found in polynomial time [53].

4 Complete Cladistic Characters

The simplest case of the perfect phylogeny problem is when characters are complete and cladistic. Intuitively, this is because character-state trees and the absence of unclassified taxa significantly limit the number of possibilities to be considered.

To highlight the differences between complete and incomplete characters, consider the following well-known result [22].

Lemma 4.1 *A set $\mathcal{C}$ of complete cladistic characters is compatible if and only its elements are pairwise compatible.*

This lemma does *does not* hold for incomplete cladistic characters; that is, a set of incomplete directed binary characters may be incompatible even if its elements are pairwise compatible. For example, in the following matrix (taken from [49]), whose rows correspond to taxa and whose columns correspond to characters, every pair of characters is compatible, but the entire set is not:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ ? & 1 & 1 \\ 0 & ? & 1 \end{bmatrix}$$

In the remainder of this section we first study the most basic compatibility problem of all, the perfect phylogeny problem for binary characters, and show that it has a particularly elegant structure, which leads to a fast algorithm. We then argue that non-binary cladistic characters can be factored into binary ones, yielding a polynomial-time algorithm for cladistic character compatibility, regardless of the number of states. Lemma 4.1 follows implicitly from the results presented here.

$$\begin{array}{c|ccccc|}
 & a & b & c & d & e \\
\hline
1 & 1 & 1 & 0 & 1 & 0 \\
2 & 0 & 1 & 1 & 0 & 1 \\
3 & 0 & 1 & 1 & 0 & 0 \\
4 & 1 & 1 & 0 & 1 & 0 \\
5 & 0 & 1 & 1 & 0 & 1 \\
6 & 1 & 1 & 0 & 0 & 0 \\
\hline
\end{array}$$

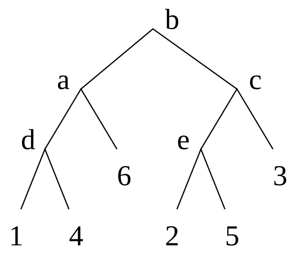

Figure 5: A compatible binary data set and the corresponding phylogeny. Nodes in the tree are labeled by the characters that emerge at them.

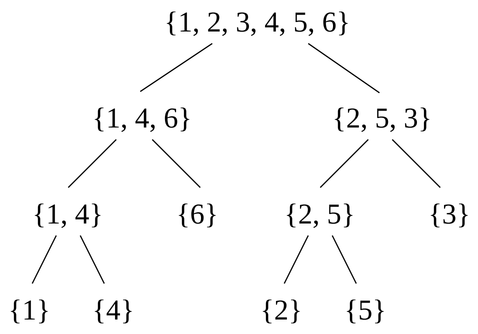

Figure 6: The n-tree representation of the phylogeny of Figure 5.

Binary characters. Assume, without loss of generality, that the root state is zero. The *1-state* of character $\mathbf{c}$, denoted $\mathbf{1_c}$, is the set of all taxa $\mathbf{s}$ such that $\mathbf{c}(\mathbf{s}) = 1$. A compatible binary data set and the corresponding perfect phylogeny T are shown in Figure 5. Since T is perfect, for each character $\mathbf{c}$, there is a node v such that the set of taxa labeling the leaves of the subtree rooted at v is exactly $\mathbf{1_c}$ (see Figure 5). If we label nodes by the 1-states of the corresponding characters we obtain what is known as the "n-tree" representation of the phylogeny: a collection of nested sets where children are subsets of their parents and sibling sets are disjoint [42] (see Figure 6). It is clear that a perfect phylogeny exists if the 1-sets can be arranged in an n-tree; moreover, this is the only case where it exists. We thus have the following classical result.

```
BinaryCompatibility($\mathcal{S}$, $\mathcal{C}$)
    sort characters by non-increasing cardinality of their 1-states
    build a tree $T$ with a single node $v$ labeled $\mathcal{S}$
    for each $\mathbf{s} \in \mathcal{S}$ do
        $lowest(\mathbf{s}) \leftarrow v$
    for each $\mathbf{c} \in \mathcal{C}$ do
        choose any $\mathbf{s} \in \mathbf{1_c}$ and let $v = lowest(\mathbf{s})$
        if there exists $\mathbf{t} \in \mathbf{c}$ such that $lowest(\mathbf{t}) \neq v$ then
            return nil
        else
            create a node $u$, labeled $\mathbf{1_c}$
            make $u$ a child of $v$
            for each $\mathbf{t} \in \mathbf{1_c}$ do
                $lowest(\mathbf{t}) \leftarrow u$
    return $T$
```

Figure 7: Testing compatibility of directed binary characters

Theorem 4.2 (Estabrook et al. [21]) *A set $\mathcal{C}$ of complete directed binary cladistic characters is compatible if and only if the 1-states of every pair of characters are either disjoint or one contains the other.*

Gusfield [32] used this result to obtain a $O(nm)$ algorithm for binary compatibility. Agarwala et al. [3] improved the running time to $O(k)$, where k is the number of ones in the data matrix; their algorithm is shown in Figure 7. The goal is to produce an n-tree representation of the 1-sets, if possible. We assume that all characters are distinct; duplicates can be handled easily. For every taxon $\mathbf{s}$ the algorithm maintains a pointer $lowest(\mathbf{s})$ to the lowest node in the current tree T containing $\mathbf{s}$. At the beginning of each iteration of the main loop (the second **for** loop), T is an n-tree representation of the 1-states of the characters examined so far. Since characters are processed by non-increasing cardinality, when $\mathbf{c}$ is considered in the main loop, there are only two possibilities: If $lowest(\mathbf{s})$ has the same value v for all $\mathbf{s} \in \mathbf{1_c}$, then the label of v is the smallest 1-set in the tree containing $\mathbf{1_c}$; thus, we make $\mathbf{1_c}$ a child of v. If there exist $\mathbf{s}, \mathbf{t} \in \mathbf{1_c}$ such that $lowest(\mathbf{s}) \neq lowest(\mathbf{t})$, then there must be a pair of characters violating the conditions of Theorem 4.2 and hence there is no perfect phylogeny. The approach can be extended to allow insertion and deletion of characters and taxa (see [3] for details).

Non-binary characters. Directed non-binary cladistic characters can be *factored* into binary characters. Let $\mathbf{c}$ be a directed cladistic character and let v be a node of $t_{\mathbf{c}}$. The *binary factor* of $\mathbf{c}$ associated with v is the directed binary character $\mathbf{d}$, whose 1-state consists of all taxa $\mathbf{s}$ such that $\mathbf{c}(\mathbf{s}) = \alpha$, where α is a node in the subtree of $t_{\mathbf{c}}$ rooted at v. One can now prove

Lemma 4.3 *A set $\mathcal{C}$ of complete directed cladistic characters is compatible if and only if the set of all binary factors of the characters in $\mathcal{C}$ is compatible.*

Since the number of factors for a character is at most one less than r, the maximum number of states, this lemma, in combination with the algorithm for directed binary characters, yields the following.

Theorem 4.4 *The compatibility of complete directed characters can be tested in $O(nmr)$ time.*

The result above gives an efficient implementation of Meacham's "tree-popping" algorithm [46]. It also gives a compatibility algorithm for complete *undirected* characters, thanks to the following fact.

Lemma 4.5 (McMorris [43]) *Let $\mathcal{C}$ be a set of complete undirected binary cladistic characters. Let $\mathcal{C}'$ be the set of complete directed binary characters obtained from $\mathcal{C}$ by rooting each $\mathbf{c} \in \mathcal{C}$ at whichever state has the most taxa, breaking ties arbitrarily. Then $\mathcal{C}$ is compatible if and only if $\mathcal{C}'$ is compatible.*

Given a complete undirected cladistic character $\mathbf{c}$, it is always possible to choose a root for $t_{\mathbf{c}}$ such that, for every binary factor of the resulting directed character, the ancestral state has at least as many taxa as the descendant state [22]. Such a root can be found in $O(n)$ time per character. After rooting the trees, we can test for compatibility of the resulting directed characters. We thus have the following.

Theorem 4.6 *The compatibility of complete undirected cladistic characters can be tested in $O(nmr)$ time.*

5 Tree Compatibility and Incomplete Characters

We use the following notation. Let T be an unrooted phylogeny for $\mathcal{S}$ and let $A \subseteq \mathcal{S}$. Then, $T|A$ denotes the tree obtained by suppressing all degree-two vertices from the minimal subtree of T connecting the nodes labeled A.

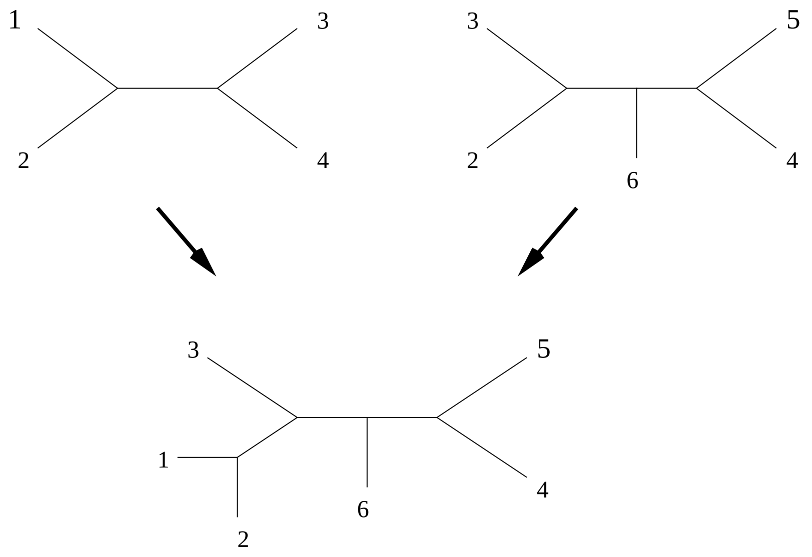

Figure 8: Top: A two-tree family $\mathcal{T}$. Bottom: A tree consistent with the elements of $\mathcal{T}$.

Let T and T' be phylogenies on $\mathcal{S}$ and $\mathcal{S}' \subseteq \mathcal{S}$, respectively. Then T is *consistent* with T' if T' can be obtained from $T|\mathcal{S}'$ by edge contractions. A set $\mathcal{T}$ of phylogenies on subsets of $\mathcal{S}$ is *compatible* if there exists a phylogeny that is consistent with every tree in $\mathcal{T}$ (see Figure 8). The *tree compatibility problem* is to determine if a collection $\mathcal{T}$ of phylogenies on subsets of $\mathcal{S}$ is compatible. An important special case of tree compatibility arises when the input is restricted to *quartets*, that is, binary phylogenies on four taxa (an example of a quartet is the first tree in Figure 8).

The compatibility of complete or incomplete cladistic characters can be formulated as a tree compatibility problem by representing each character as a tree (see Figure 9). Indeed, compatibility of cladistic or qualitative characters is polynomially reducible to quartet compatibility [52]. Unfortunately, we have the following result.

Theorem 5.1 (Steel [52]) *Tree compatibility is* NP*-complete even when the input consists of quartets.*

Since every quartet can be encoded as an incomplete binary character, we have the following (also observed by Steel).

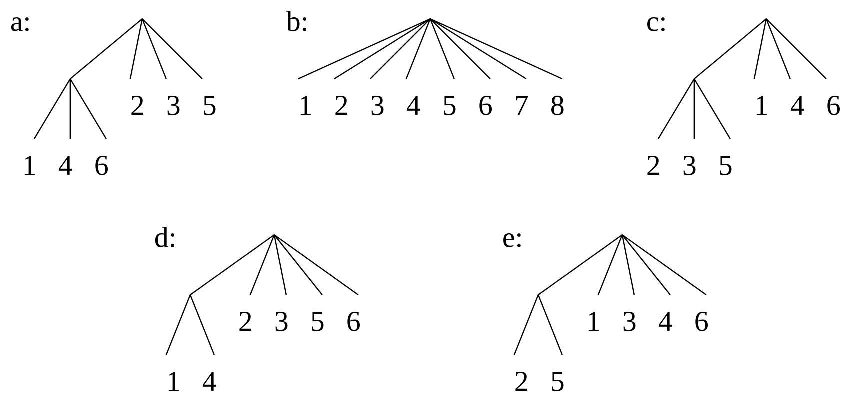

Figure 9: Tree representations of the binary directed characters of Figure 5. The phylogeny shown in that figure is consistent with all five trees shown here.

Corollary 5.2 *Testing the compatibility of incomplete undirected characters is* NP*-complete, even when the input characters are binary.*

On the other hand, if all input trees have at least one common taxon, the compatibility problem becomes polynomially-solvable, since the problem becomes a special case of *rooted compatibility*, to be discussed next.

The rooted case. An important result of Aho et al. [4] is that the compatibility of *rooted* trees can be determined in polynomial time. To describe this algorithm, we need some notation. Given a rooted phylogeny T for $\mathcal{S}$ and $A \subseteq \mathcal{S}$, $T|A$ has the same meaning as for unrooted trees, except that the root of the minimal subtree connecting A is suppressed only if it has just one child. Let $\mathcal{T}$ be a family of rooted phylogenies on subsets of $\mathcal{S}$. Given a set $X \subseteq \mathcal{S}$, $\mathcal{T}|X$ denotes $\{T|X : T \in \mathcal{T}\}$. $\mathcal{H}(\mathcal{T}, \mathcal{S})$ denotes the graph with vertex set $\mathcal{S}$ where $(\mathbf{s}, \mathbf{t})$ is an edge if and only if there is a tree $T \in \mathcal{T}$ such that $\mathbf{s}$ and $\mathbf{t}$ appear in the same connected component of $T - \{\mathit{root}(T)\}$.

The key insight is that if there exists a tree T^* that is consistent with every tree in $\mathcal{F}$, then each connected connected component of $\mathcal{H}(\mathcal{T}, \mathcal{S})$ (viewed as a set of taxa) is contained in a distinct subtree of the root of T^*. This gives us the first level of T^*; to build the rest, we proceed recursively on the components. This process is detailed in Figure 10.

We have the following, which is a restatement of results in [4], along the lines of [52].

```
RootedTreeCompatibility($\mathcal{T}, \mathcal{S}$)
    let $T$ be a tree with a single vertex $v$, labeled $\mathcal{S}$
    if $\mathcal{H}(\mathcal{T}, \mathcal{S})$ has more than one connected component then
        for each connected component $X$ of $\mathcal{H}(\mathcal{T}, \mathcal{S})$ do
            let $T_X$ = RootedTreeCompatibility($\mathcal{T}|X, X$)
            make $T_X$ a subtree of $v$ in $T$
    return $T$
```

Figure 10: The rooted tree compatibility algorithm.

Theorem 5.3 *Let $\mathcal{T}$ be a family of rooted phylogenies on the set of taxa $\mathcal{S}$ and let T be the tree returned by RootedTreeCompatibility($\mathcal{T}, \mathcal{S}$). Then $\mathcal{T}$ is compatible if and only if for each $\mathbf{s} \in \mathcal{S}$ there is a node labeled $\{\mathbf{s}\}$ in T.*

Assume that the trees in $\mathcal{T}$ are binary and have total size M. Aho et al. gave a $O(nM)$ implementation of their algorithm. Henzinger et al. improved the running time to $O(\min\{Mn^{1/2}, M + n^2 \log n\})$, also giving a $O(m \log^3 n)$ randomized version [34]. The speedup is achieved by improving the time needed to maintain the connected components of $\mathcal{H}(\mathcal{T}, \mathcal{S})$ under a sequence of deletions of edges in batches (which is how the graphs considered in recursive calls are obtained from $\mathcal{H}$).

Obviously, RootedTreeCompatibility can be used to test compatibility of a set $\mathcal{T}$ of unrooted trees in exponential time by enumerating all possible rootings of the trees in the set. There is one case where such enumeration is unnecessary. Suppose all the unrooted trees in $\mathcal{T}$ share some taxon $\mathbf{s}$. Then, the problem can be reduced to rooted compatibility by rooting each tree T in $\mathcal{T}$ at the neighbor of $\mathbf{s}$ in T [11].

Incomplete directed binary characters. Algorithm RootedTreeCompatibility gives a method for testing the compatibility of incomplete directed binary characters; however, the reduction to tree compatibility takes $\Omega(n^2 m)$ time. It is natural to ask whether the problem can be solved in time approaching the $O(nm)$ bound for compatibility of complete binary characters. This goal has nearly been attained by Pe'er, Shamir, and Sharan, who have devised an algorithm that solves the problem in time $O(nm + k \log^2(n+m))$, where k is the total number of taxa in the 1-states [49]. Their method can viewed as a clever adaptation of the procedure of Aho et al.

To explain the algorithm, we need some definitions. The *character-taxon graph* is the bipartite graph $CTG(\mathcal{S}, \mathcal{C}) = (\mathcal{S}, \mathcal{C}, E)$ with $E = \{(\mathbf{s}, \mathbf{c}) : \mathbf{s} \in$

```
IncompleteBinaryCompatibility($\mathcal{S}, \mathcal{C}$)
    let $G = CTG(\mathcal{S}, \mathcal{C})$, $T = \{\mathcal{S}, \emptyset\} \cup \{\{\mathbf{s}\} : \mathbf{s} \in \mathcal{S}\}$
    remove all $\mathcal{S}$-semi-universal and all null characters from $G$
    while $E(G) \neq \emptyset$ do
        for each connected component $A$ of $G$ such that $|E(A)| \geq 1$ do
            let $\mathcal{S}' = \mathcal{S} \cap A$
            compute the set $U$ of all $\mathcal{S}'$-semi-universal characters
            if $U = \emptyset$ then
                return nil
            else
                remove $U$ from $G$ and set $T \leftarrow T \cup \{S'\}$
    return $T$
```

Figure 11: Testing compatibility of incomplete directed binary characters

$\mathcal{S}, \mathbf{c} \in \mathcal{C}, \mathbf{c}(\mathbf{s}) = 1\}$. Let $\mathcal{S}'$ be a non-empty subset of $\mathcal{S}$. A character is *$\mathcal{S}'$-semi-universal* if its 0-state does not intersect $\mathcal{S}'$.

The procedure of Pe'er et al. is shown in Figure 11. CTG plays the role of graph $\mathcal{H}$ in the Aho et al. algorithm. In particular, the non-existence of a semi-universal character for a connected component of CTG is equivalent to the existence of a non-trivial unsplittable connected component in $\mathcal{H}$. Thus, if either condition is detected, it is correct to report that no perfect phylogeny exists. As in the Aho et al. algorithm, the time is dominated by the work needed to maintain connected components of CTG under edge deletions. Details of the analysis can be found in [49].

If the data is compatible, the value returned by the algorithm of Pe'er et al. is a collection of sets that can be viewed as the 1-sets of complete characters obtained by flipping the question-marks to 0's or 1's. These can be assembled into a phylogeny using the algorithm BinaryCompatibility of Figure 7.

General comments. From the preceding two algorithms emerges an important idea: that of decomposing $\mathcal{S}$ into equivalence classes according to the states of the root. We briefly elaborate on this here.

The fact that the root state of all characters is zero implies a partition of $\mathcal{S}$, each of whose elements is an equivalence class of the transitive closure of the relation R defined as follows: For $\mathbf{s}, \mathbf{t} \in \mathcal{S}$, $(\mathbf{s}, \mathbf{t}) \in R$ if and only if there exists some $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{c}(\mathbf{s}) = \mathbf{c}(\mathbf{t}) = 1$. These equivalence classes are precisely the connected components of the $\mathcal{H}$ graph of Aho et al. (and are closely related to the components of the CTG graph of Pe'er et al.). Let A be one of the equivalence classes. By construction, all taxa $\mathbf{s} \in A$ have $\mathbf{c}(\mathbf{s}) = 1$ for all characters $\mathbf{c}$ in some $\mathcal{C}' \subseteq \mathcal{C}$. We can further partition A into subsets, each of which is an equivalence class of the transitive closure of R', defined as: $(\mathbf{s}, \mathbf{t}) \in R'$ if and only if there exists some $\mathbf{c} \in \mathcal{C} - \mathcal{C}'$ such that $\mathbf{c}(\mathbf{s}) = \mathbf{c}(\mathbf{t}) = 1$. We can view this as the decomposition of A induced by the root of the subtree containing A; this node v must have $\mathbf{c}(\mathbf{s}_v) = 1$ for all $\mathbf{c} \in \mathcal{C}'$ and $\mathbf{c}(\mathbf{s}_v) = 0$ otherwise.

What makes testing compatibility of directed characters simpler than for undirected characters is, of course, knowledge of the root. Thus, we can build trees from the top down. Still, the idea of decomposing a set of taxa according to a given vector is important for two of the algorithms discussed in later sections, both of which use a bottom-up approach. In one case (Section 6), the space of potential roots is limited by convexity constraints; in the other (Section 7), it is limited by the fact that the number of characters is small.

6 Bounding the Number of States

In this and the next section, we consider only complete qualitative characters. As we saw in Section 4, the compatibility of qualitative binary characters can be determined in $O(nm)$ time by reducing the problem to directed compatibility. However, the complexities of cladistic and qualitative character compatibility diverge for characters with three or more states. If no bound is placed on the number of states, determining the compatibility of qualitative characters is NP-complete [8], whereas the same problem for cladistic characters is polynomially-solvable (Theorem 4.6). Dress and Steel gave a $O(nm^2)$ algorithm for testing the compatibility of ternary characters [18]. Later, Kannan and Warnow gave a $O(n^2m)$ algorithm for quaternary characters [37]. The problem was shown to be polynomially-solvable for all fixed r by Agarwala and Fernández-Baca, who gave a $O(2^{3r}(nm^3 + m^4))$ algorithm [1]. This bound that was improved to $O(2^{2r}nm^2)$ by Kannan and Warnow [39]. In this section we give an account of the last two algorithms.

Let us say that subsets A, B of $\mathcal{S}$ *share state* α on character $\mathbf{c}$ if there exist $\mathbf{s} \in A$, $\mathbf{t} \in B$ such that $\mathbf{c}(\mathbf{s}) = \mathbf{c}(\mathbf{t}) = \alpha$. A subset G of $\mathcal{S}$ is a *cluster* if G and $\mathcal{S} - G$ share at most one state α on each character. A cluster is *proper* if there exists some character $\mathbf{c}$ on which no state is shared between G and $\mathcal{S} - G$.

For example, consider the set of taxa

$$\mathcal{S} = \{(1,2,1),(1,3,3),(2,1,1),(2,3,2),(2,4,1),(3,3,3),(4,4,1)\}. \quad (1)$$

Then $H = \{(2,1,1),(2,3,2)\}$ is a cluster, but not a proper one, since state 2 is shared with $\mathcal{S} - H$ on the first character, state 3 on the second, and state 1 on the third. On the other hand, $G = \{(2,1,1),(2,3,2),(2,4,1),(4,4,1)\}$ is a proper cluster, since no states are shared between G and $\mathcal{S} - G$ on the first character.

A simple but important observation is that there are at most $2^r m$ proper clusters, since at most 2^r are defined by the states of any given character. This bound is polynomial when r is fixed.

Removing any edge in a minimal perfect phylogeny partitions the taxa into disjoint sets such that there is at least one character on which no state is shared; otherwise, the edge could be contracted and the phylogeny would not be minimal. Thus, we have the next lemma [1].

Lemma 6.1 *Let T be a minimal perfect phylogeny for $\mathcal{S}$, let e be an edge in T, and let $\mathcal{S}'$ be the subset of $\mathcal{S}$ labeling the leaves of a component of $T - \{e\}$. Then $\mathcal{S}'$ is a proper cluster.*

Thus, minimal perfect phylogenies can be assembled from phylogenies for proper clusters. This and the polynomial bound on the number of proper clusters motivate the dynamic programming approach that we are about to develop. The idea is to enumerate the proper clusters G by increasing cardinality, testing whether each G has a perfect phylogeny made up of phylogenies for smaller proper clusters. Since our goal is to compose the phylogenies by linking roots through edges, the permissible states for the root of a phylogeny for a cluster G are partially determined by convexity. More precisely, the *splitting vector* of G, which is a partial specification of this root, is the taxon $Sv(G)$ where, for each $\mathbf{c} \in \mathcal{C}$, $\mathbf{c}(Sv(G)) = \alpha$, if G and $\mathcal{S} - G$ share state α on character $\mathbf{c}$ and $\mathbf{c}(Sv(G)) = *$ otherwise.

Let G, G_1 be clusters such that $G_1 \subset G$. G and G_1 are *compatible* if for every character $\mathbf{c}$ such that $\mathbf{c}(Sv(G)), \mathbf{c}(Sv(G_1)) \neq *$, $\mathbf{c}(Sv(G)) = \mathbf{c}(Sv(G_1))$. Intuitively, if G and G_1 are compatible, there conceivably exists a phylogeny for $G \cup \{Sv(G)\}$ such that one of the subtrees of $Sv(G)$ is a phylogeny for $G_1 \cup \{Sv(G_1)\}$. In such a phylogeny, some of the $*$-states of $Sv(G)$ may have to take on specific values, since some states may be shared between G_1 and $S - G_1$ that are not shared between G and $S - G$. Define the *splitting vector* for (G, G_1) to be the taxon $Sv(G, G_1)$ such that for each character $\mathbf{c}$ on which a state α is shared between G and $S - G$ or between G_1 and $S - G_1$, $\mathbf{c}(Sv(G, G_1)) = \alpha$; otherwise $\mathbf{c}(Sv(G, G_1)) = *$.

To illustrate these definitions, consider the set of taxa (1). Sets

$$G = \{(2,1,1), (2,3,2), (2,4,1), (4,4,1)\} \text{ and } G_1 = \{(2,4,1), (4,4,1)\}$$

are proper clusters with $G_1 \subset G$. $Sv(G) = (*, 3, 1)$ and $Sv(G_1) = (2, *, 1)$; thus, G and G_1 are compatible and $Sv(G, G_1) = (2, 3, 1)$.

The final ingredient in the algorithm is a way to handle the following problem: Given a cluster G and a taxon $\mathbf{x}$, determine how to split G into subsets $H_1, \ldots, H_l$ such that, for each i, all taxa in H_i must lie in the same subtree of $\mathbf{x}$ in any phylogeny for $G \cup \{\mathbf{x}\}$.

Definition 6.2 *Given a taxon* $\mathbf{x}$, *let* $\sim_{\mathbf{x}}$ *be the equivalence relation on* $\mathcal{S}$ *defined as the transitive closure of the following relation* $R_{\mathbf{x}}$: *For* $\mathbf{s}, \mathbf{t} \in \mathcal{S}$, $(\mathbf{s}, \mathbf{t}) \in R_{\mathbf{x}}$ *if there exists a character* $\mathbf{c}$ *such that* $\mathbf{c}(\mathbf{s}) = \mathbf{c}(\mathbf{t}) \neq \mathbf{c}(\mathbf{x}) \neq *$. *Denote by* $\mathcal{S}/\mathbf{x}$ *the collection of equivalence classes of* $\sim_{\mathbf{x}}$.

Clearly, each of the sets in $\mathcal{S}/\mathbf{x}$ must be in the same connected component of $T - \{\mathbf{x}\}$ for any perfect phylogeny of $\mathcal{S} \cup \{\mathbf{x}\}$. (Compare this with the comments at the end of Section 5.)

For example, consider again the set of taxa in (1). Then $\mathcal{S}/(1,3,1)$ has three classes:

$$H_1 = \{(1,2,1)\}, \qquad H_2 = \{(1,3,3), (3,3,3)\},$$

$$H_3 = \{(2,3,2), (2,1,1), (2,4,1), (4,4,1)\}.$$

A proper cluster G is *good* if $G \cup \{Sv(G)\}$ has a perfect phylogeny. A pair of proper clusters (G, G_1) where $G_1 \subset G$ and G_1 is compatible with G is *good* if there exists a perfect phylogeny for G with an internal node v labeled by $Sv(G, G_1)$ such that the removal of v partitions G into subsets some of which union to G_1. We now have the following result [39] (see also [1]).

Theorem 6.3 *Suppose that* G *is a proper cluster and that* $G_1 \subset G$ *is a good proper cluster. Let* $G_2 = G - G_1$. *Then,* (G, G_1) *is good if and only if (i)* G_2 *is a good proper cluster or (ii) each* $H \in G_2/Sv(G, G_1)$ *is a good proper cluster. In case (ii), every state of* $Sv(G, G_1)$ *is defined.*

This leads to the algorithm of Figure 12. The procedure iterates over all $O(2^r m)$ proper clusters G. For each of these, it considers $O(2^r m)$ choices

```
PerfectPhylogeny($\mathcal{S}$)
    identify all proper clusters and
        sort them by nondecreasing cardinality
    mark all 1-element clusters as "good"
    **for each** proper cluster $G_1$ with at least 2 elements **do**
        **for each** proper cluster $G_1 \subset G$ compatible with $G$ **do**
            **let** $G_2 = G - G_1$
            **if** $G_2$ is a good cluster **then**
                mark $G$ as "good"
            **else**
                **let** $\mathbf{x} = Sv(G, G_1)$
                **if** every $H \in G_2/\mathbf{x}$ is good **then**
                    mark $G$ as "good"
                **else**
                    mark $G$ as "bad"
```

Figure 12: Perfect phylogeny when the number of states is fixed.

of G_1. Kannan and Warnow show how to find the equivalence classes of $G_2/Sv(G, G_1)$ in $O(n)$ time at the expense of precomputing, in $O(2^r nm^2)$ time, the equivalence classes of $\mathcal{S}/Sv(G)$ for every proper cluster G (see [39]). A $O(2^{2r} nm^2)$ bound follows. Kannan and Warnow also describe how to extend this algorithm to generate all minimal phylogenies [39].

7 Bounding the Number of Characters

We now describe two perfect phylogeny algorithms that run in polynomial time when the number of characters is fixed, even if the number of states is unbounded. The first relies on the connection between character compatibility and triangulations of colored graphs, while the second avoids triangulations entirely, relying instead on the equivalence relation defined in the previous section. Interestingly, the second scheme gives an alternative solution to the triangulation problem.

Restricted triangulations and phylogenies. The *intersection graph* of a collection of sets $\mathcal{F}$, denoted $IG(\mathcal{F})$, is the graph with vertex set $\mathcal{F}$ such that there is an edge between $A, B \in \mathcal{F}$ if and only if $A \cap B \neq \emptyset$ [30, 44].

A *chord* in a cycle is an edge between a pair of non-adjacent vertices. A

a	b	c
1	2	1
1	3	3
2	1	1
2	3	2
3	3	1

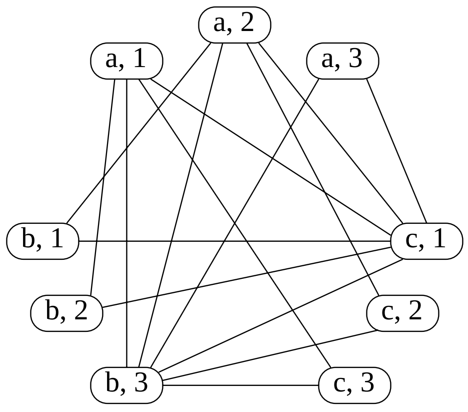

Figure 13: A character state matrix and its character-state intersection graph.

graph is *triangulated* (or *chordal*) if every cycle of length greater than three has a chord. The following fundamental theorem was proved independently by Gavril [29] and Buneman [12].

Theorem 7.1 *A graph is chordal if and only if it is the intersection graph of a family of subtrees of a tree.*

A graph G is *properly* m*-colored*, if it has a mapping $f : V(G) \mapsto \{1, \ldots, m\}$ such that there is no edge (u, v) with $f(u) = f(v)$. A *restricted triangulation* of an m-colored graph G is a properly m-colored triangulated graph obtained by adding edges to G.

Assume that each $\mathbf{c} \in \mathcal{C}$ is assigned a unique color $g(\mathbf{c}) \in \{1, \ldots, m\}$, and define the *character-state intersection graph* of $\mathcal{C}$, denoted $CIG(\mathcal{C})$, to be $IG(\{\mathcal{S}_{\mathbf{c},\alpha} : \mathbf{c} \in \mathcal{C}, \alpha \in \mathcal{A}_{\mathbf{c}}\})$, where vertices $\mathcal{S}_{\mathbf{c},\alpha}$ are assigned color $g(\mathbf{c})$ (see Figure 13). Clearly, $CIG(\mathcal{C})$ is properly m-colored; furthermore, it is a *partition intersection graph*, that is a graph whose edge-set is covered by m-cliques.

The next result is due to Buneman [12].

Theorem 7.2 *$\mathcal{C}$ is compatible if an only if its character-state intersection graph has a restricted triangulation.*

To see this, suppose that $\mathcal{S}$ has a perfect phylogeny T. Let $T_{\mathbf{c},\alpha}$ be the minimal subtree of T containing all taxa in $\mathcal{S}_{\mathbf{c},\alpha}$. Then, by Theorem 7.1,

$IG(\{T_{\mathbf{c},\alpha} : \mathbf{c} \in \mathcal{C}, \alpha \in \mathcal{A}_{\mathbf{c}}\})$, where $T_{\mathbf{c},\alpha}$ is assigned color $g(\mathbf{c})$, is chordal. In fact, this intersection graph is a restricted triangulation of $CIG(\mathcal{C})$, since, by convexity, for each $\mathbf{c} \in \mathcal{C}$ and every two distinct states α and β of $\mathbf{c}$, $T_{\mathbf{c},\alpha}$ and $T_{\mathbf{c},\beta}$ are vertex disjoint.

We now sketch the other direction. If CIG has a restricted triangulation, then, by Theorem 7.1, this triangulation is the intersection graph of a family of subtrees of some tree T. Each such subtree corresponds to the minimal tree connecting all taxa in $\mathcal{S}_{\mathbf{c},\alpha}$ for some $\mathbf{c} \in \mathcal{C}, \alpha \in \mathcal{A}_{\mathbf{c}}$ and thus T corresponds to a phylogeny for $\mathcal{S}$. For details, see [12].

A triangulation algorithm. We now outline the main ideas behind McMorris, Warnow and Wimer's $O(V^{m+1})$ algorithm to determine if a V-node m-colored partition intersection graph has a restricted triangulation [45]. This gives a $O(r^{m+1}m^{k+1} + nm^2)$-time perfect phylogeny algorithm, since the character state intersection graph for $\mathcal{C}$ has rm vertices and can be built in $O(nm^2)$ time.

We need a definition and some notation. A *separator* in a graph G is a set of vertices S such that the graph $G - S$ is disconnected. Given a separator S for G, and a connected component C of $G - S$, write $C \cup cl(S)$ to denote the graph obtained by adding enough edges to the subgraph induced by $C \cup S$ so as to make S a clique. McMorris et al. show the following:

Lemma 7.3 *Let G be a properly m-colored graph. Then, G has a restricted triangulation if and only if G has an $(m-1)$-vertex separator S such that, for every connected component C of $G - S$, $C \cup cl(S)$ has a restricted triangulation.*

Theorem 7.4 *Let G be a properly m-colored graph with at least $m+1$ vertices. Let S be an $(m-1)$-vertex separator of G whose vertices have distinct colors and let C be a connected component of $G - S$. Then $C \cup cl(S)$ has a restricted triangulation if and only if there exists some vertex v in C and a family $\mathcal{M}$ of $(m-1)$-vertex subsets of $S \cup \{v\}$ such that:*

1. *each $M \in \mathcal{M}$ is a separator for both $C \cup cl(S)$ and G;*

2. *for each vertex $u \in S - \{v\}$, there is a set $M_u \in \mathcal{M}$ and a component C_u of $G - M_u$ and of $C \cup cl(S) - M_u$ such that C_u has fewer vertices than C and $C_u \cup cl(M_u)$ has a restricted triangulation; and*

3. *every edge of C is in exactly one of the above C_u's.*

The preceding theorem implies that we can determine whether or not $C \cup cl(S)$ has a restricted triangulation if we know the answer to the restricted triangulation problem for every graph $C' \cup cl(S')$ with fewer vertices than $C \cup cl(S)$, where S' is an $(m-1)$-vertex separator of G whose vertices have distinct colors, and C' is a component of $G - S'$. This leads to the following scheme for determining if graph G has a restricted triangulation: Find all $O(V^{m-1})$ $(m-1)$-vertex separators S of G, compute all graphs $C \cup cl(S)$ for the components C of $G - S$. Now process each such graph H in order of nondecreasing size, using the answers for the smaller graphs to determine whether H has a restricted triangulation. The process can be implemented to run in $O(V^{m+1})$ time [45].

The algorithm of McMorris et al. uses the close connection between restricted triangulations and the problem of determining whether a graph has tree-width at most k, for some fixed k (for a definition of tree-width, see [50]). In fact, the algorithm is nearly identical to the procedure devised by Arnborg et al. to determine whether a graph has tree-width k [5]. It turns out, however, that for fixed k, the tree-width problem can be solved in linear time [7]. On the other hand, while the triangulation approach leads to a linear-time algorithm for three-colored graphs [9, 38], a linear-time algorithm for m-colored graphs, m fixed, seems unlikely, since this problem is not *finite state*, in the sense described by Bodlaender et al. [8] (see also [6]).

An alternative approach. It is possible to solve the perfect phylogeny problem with a bounded number of characters *without* resorting to triangulations. Here we outline this approach, proposed by Agarwala and Fernández-Baca [2], which leads to a $O((r - n/m)^m rnm)$ algorithm.

The method attempts to find a perfect phylogeny in which the Hamming distance between the labels of any two adjacent nodes is one; such a tree exists whenever *some* perfect phylogeny exists. Let $\mathcal{S}^*$ denote the set $\mathcal{A}_{\mathbf{c}_1} \times \cdots \times \mathcal{A}_{\mathbf{c}_m}$ of all possible taxa. Define a directed search graph $SG(\mathcal{S}, \mathcal{C})$ whose vertex set consists of all $[G, \mathbf{x}]$ such that $\mathbf{x} \in \mathcal{S}^*$ and $G = \mathcal{S}$ or $G \in (\mathcal{S} - \mathbf{x})/\mathbf{x}$, where, as in the previous section (Definition 6.2), $(\mathcal{S} - \mathbf{x})/\mathbf{x}$ is the set of equivalence classes of the relation "$\sim_{\mathbf{x}}$." Node $[G, \mathbf{x}]$ represents the question of whether or not $G \cup \{\mathbf{x}\}$ has a perfect phylogeny.

The edge set of SG consists of all pairs of the form $([G, \mathbf{x}], [\mathcal{S}, \mathbf{x}])$, and all pairs of the form $([G_1, \mathbf{x}_1], [G_2, \mathbf{x}_2])$ such that $G_1 \subseteq G_2$ and the Hamming distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ is one. Let v be a node in SG. The set of all edges $(u_1, v), \ldots, (u_l, v)$ such that $u_i = [H_i, \mathbf{y}]$ and $\bigcup_i H_i = G$ is called a *bundle*. If each $[H_i, \mathbf{y}]$ has a "yes" answer, then there must exist a phylogeny for $G \cup \{\mathbf{x}\}$ with distinct subtrees for the H_i's. Note that there can be multiple bundles into a node and that there might be one-edge bundles $([G, \mathbf{y}], [G, \mathbf{x}])$. SG has $O(r^m n)$ nodes, since $|\mathcal{S}^*| = O(r^m)$, and any $\mathbf{x} \in \mathcal{S}^*$ defines at most n equivalence classes. There are $O(r^{m+1}nm)$ edges, because the out-degree of any node is $O(rm)$. SG encodes all possible ways in which clusters can be expressed in terms of smaller subfamiles; thus, if the characters are compatible, a perfect phylogeny must be embedded within SG.

To determine if a perfect phylogeny exists, the algorithm propagates truth values from node to node along the edges of SG. As soon as a node gets a truth value, the value is placed on all outgoing edges. To initialize the process, we assign a value of true to every vertex $[G, \mathbf{x}]$ such that $|G| = 1$. During the propagation, a node v becomes *active* when all its incoming edges have been assigned a truth value. An active node is assigned the value true if there is some bundle whose edges are all true; otherwise, the node gets a false value. It can be shown that a perfect phylogeny exists if and only if, at the end of the propagation process, there is some node $[\mathcal{S}, \mathbf{x}]$ which is true. The propagation time is linear in the size of the graph, which yields a $O(r^{m+1}nm)$ algorithm. Through a more careful analysis, this bound can be improved to $O((r - n/m)^m rnm)$ — see [2].

Via Warnow's polynomial-time reduction from restricted triangulation to perfect phylogeny [56], the preceding algorithm yields a $O((2E/m)^m(E^2m))$ algorithm for restricted triangulation problem of graphs with E edges [2]. In fact, further inspection of Theorem 7.4 reveals a close relationship with the search graph approach. In particular, separators play the role of the taxa $\mathbf{x} \in \mathcal{S}^*$ used to partition $\mathcal{S}$ in the search graph.

8 Discussion and Further Topics

Except for the algorithms for binary compatibility of Section 4, which are, in a sense, optimal (see [32]), it is not clear whether any of the time bounds presented here is the best possible. It would be especially interesting to know whether the time bound of the algorithm for perfect phylogeny when the number of states is fixed (Section 6) can be improved to $O(2^{2r}nm)$. This would be appealing from a theoretical point of view, since it would bring the result for $r \geq 3$ in line with the binary case, and would also have practical significance, since for many data sets, m is much larger than n. A more realistic, and still relevant, goal is to achieve $O(2^{2r}n^2m)$ running time. This would match Kannan and Warnow's bound for quaternary characters [37] and may be achievable by finding a better approach for selecting the "initial" cluster G_1 in the algorithm of Figure 12. Regarding the algorithms for the case where the number of characters is fixed (Section 7), it is unclear what sort of improvements one could expect, although perhaps a combination of the two approaches we have presented might yield better results. Improvements on the speed of rooted tree compatibility (Section 5) seem to rest on faster algorithms for maintaining connected components of graphs under edge deletion. For the unrooted case, it would be useful to identify other polynomially-solvable special cases. That this might hinge on the degree of overlap among the leaf sets of the input trees is suggested by Steel's proof of Theorem 5.1 [52], which requires minimal sharing of taxa.

In what is left of this section, we first discuss two important subjects related to perfect phylogenies: polymorphism and dealing with incompatibility. We then give some final comments.

Polymorphism. A character is *polymorphic* if it can have more than one state on a given taxon. Characters of this sort are commonly encountered in molecular genetics and linguistics (see [10] for references). As before, let $\mathcal{A}_{\mathbf{c}} = \{0, 1, \ldots, r_{\mathbf{c}} - 1\}$ denote the set of allowed states for character $\mathbf{c}$. For a taxon $\mathbf{s}$, $\mathbf{c}(\mathbf{s})$ is a subset of $2^{\mathcal{A}_{\mathbf{c}}} - \emptyset$. Let T be a phylogeny for a set of taxa $\mathcal{S}$ over a set of polymorphic characters $\mathcal{C}$, where every internal node v is labeled by a taxon $\mathbf{s}_v$ over the same set of characters. T is *perfect* if for each $\mathbf{c} \in \mathcal{C}$ and every $\alpha \in \mathcal{A}_{\mathbf{c}}$, the set of all nodes v such that $\alpha \in \mathbf{c}(\mathbf{s}_v)$ induces a subtree of T.

A character $\mathbf{c}$ has *load* l in a phylogeny T if for every node v of T, $|\mathbf{c}(\mathbf{s}_v)| \leq l$. The load of T is the maximum load on any character. Obviously, the problem of finding a minimum-load perfect phylogeny is NP-hard, since it provides an answer to the perfect phylogeny problem for ordinary (*monomorphic*) characters.

Bonet et al. [10] have shown various positive and negative results about the problem; we mention some of these. The compatibility of polymorphic binary characters can be determined in polynomial time using the techniques of Section 4. The minimum-load problem is NP-hard whether one fixes the number of characters or the maximum number of states per character. On the other hand, if one fixes both the number of characters and the maximum load, determining if there exists a perfect phylogeny of load at most l is solvable in polynomial time by generalizing either the triangulation approach or the search graph approach of Section 7.

Dealing with incompatibility. It is an unavoidable fact that most data sets are incompatible. This leaves us with at least three options: (i) find a maximum-cardinality set of compatible characters, (ii) attempt to minimize the amount of homoplasy, (iii) construct a phylogeny that matches as closely as possible the partitions implied by the characters. We briefly comment on each possibility.

Regarding (i), Lemma 4.1 states that, for complete cladistic characters, it suffices to find a maximum-cardinality subset of *pairwise* compatible characters. Unfortunately, this is equivalent to finding a maximum clique and is thus NP-hard. On the positive side, the problem of determining whether the removal of k characters yields a compatible data set is equivalent to vertex cover. This problem is *fixed-parameter tractable* [17], that is, on an m-vertex graph (each vertex corresponding to one of the characters) it can be solved in time $O(f(k)m^a)$, where a is independent of k. In fact, a bound of $O(km + 1.271^k k^2)$ can be achieved [14].

Regarding (ii), we first note that this is essentially the Steiner tree problem in phylogeny, discussed in Section 3. As an intermediate option between perfect phylogenies and Steiner trees, we can formulate a parameterized version of the problem, which is motivated by Theorem 3.1. Define the *imperfection* q of an internally labeled phylogeny T for $\mathcal{S}$ as $\mathit{length}(T) - \sum_{c \in \mathcal{C}}(r_c - 1)$. It is shown in [25] that the problem of determining whether a set of taxa admits a phylogeny with imperfection at most q is solvable in polynomial time if q and the number of states are fixed. The algorithm builds on the techniques described in Section 6.

Problem (iii) can be viewed as achieving consensus among conflicting characters by smoothing over the incompatibility in the data. This addresses a criticism leveled at character compatibility methods: that they unrealistically require strict compliance to the partitions implied by characters. The method suggested for problem (i) of finding a maximum-cardinality set of compatible characters is not quite satisfactory, since it may discard partitions that, while imperfect, are still informative.

We now review two relatively recent developments. The first is due to Semple and Steel [51] and arises in the context of *supertrees*, that is phylogenies built by combining trees on overlapping sets of taxa. Thus, the supertree problem is just the tree compatibility problem of Section 5. By Theorem 5.1, the unrooted supertree problem is NP-complete. On the other hand the RootedTreeCompatibility algorithm of Aho et al. described in Section 5 (Figure 10) solves the rooted case efficiently. However, this by itself is unsatisfactory, since in practice few sets of rooted trees are compatible. Recall that incompatibility is reflected in RootedTreeCompatibility in the presence at some level of the recursion of a connected $\mathcal{H}$ graph with two or more vertices. Semple and Steel give one method for breaking up this graph by deleting certain edges so as to allow the computation to proceed. Edges to remove are chosen using minimum cuts in a weighted graph closely related to $\mathcal{H}$; this graph is designed so as to preserve nestings of taxa that are present in all input trees. The procedure can be implemented in polynomial time and the tree produced has some desirable properties. A disadvantage, however, is that the local optimality criterion used seems too strict, and thus valuable information in the trees may be discarded.

An interesting approach has been suggested by Kearney et al. [40], who define an optimization version of character compatibility that they call *fractional character compatibility.* This work is related to earlier results on quartet compatibility by three of the same authors [36]. Intuitively, the goal is to find a phylogeny that matches the given characters as closely as possible. More formally, let $\mathbf{c}$ be a complete undirected binary character and let T be an unrooted phylogeny for $\mathcal{S}$. Then $\mathbf{c}$ defines a bipartition $(\mathbf{0_c}, \mathbf{1_c})$ of $\mathcal{S}$, where $\mathbf{0_c}$ $(\mathbf{1_c})$ is the set of all taxa exhibiting state zero (one) on $\mathbf{c}$. Any edge $e = (u, v)$ in T also defines a bipartition $(\mathcal{S}_u, \mathcal{S}_v)$ of $\mathcal{S}$, where $\mathcal{S}_u$ $(\mathcal{S}_v)$ is the set all taxa that lie in the connected component of $T - (u, v)$ containing u (v). The *similarity* between $\mathbf{c}$ and e is

$$sim(\mathbf{c}, e) = \max\{|\mathbf{0_c} \cap \mathcal{S}_u| + |\mathbf{1_c} \cap \mathcal{S}_v|, |\mathbf{0_c} \cap \mathcal{S}_v| + |\mathbf{1_c} \cap \mathcal{S}_u|\}.$$

Notice that e defines the same partition as $\mathbf{c}$ if and only if $sim(\mathbf{c}, e) = |\mathcal{S}|$. The similarity between a character $\mathbf{c}$ and T is

$$sim(\mathbf{c}, T) = \max_{e \in T} sim(\mathbf{c}, e).$$

Given a set of complete undirected binary characters $\mathcal{C}$, define the similarity between $\mathcal{C}$ and T as

$$sim(\mathcal{C}, T) = \sum_{\mathbf{c} \in \mathcal{C}} sim(\mathbf{c}, T).$$

Given a set of complete undirected binary characters $\mathcal{C}$ on $\mathcal{S}$, the *fractional character compatibility problem* asks to find a tree T that maximizes $sim(T, \mathcal{C})$. This problem formulation is attractive because of the well-defined and intuitively reasonable global optimality criterion on which it is based. Note, for instance, that the value of the optimum solution is at most $|\mathcal{C}||\mathcal{S}|$, and that the value is precisely this number if and only if the instance has a perfect phylogeny. Although Kerney et al. show that fractional character compatibility is NP-complete, they also prove that it has a polynomial-time approximation scheme when $|\mathcal{C}| = \Theta(|\mathcal{S}|)$. This algorithm, while theoretically significant, is too slow to be of practical use; whether its time bound can be improved is important open problem.

Final comments. It has been our goal to provide a unified view of the core topics in perfect phylogenies. Certain recurring themes are evident, perhaps none as important as intersection graphs and their relatives. While intersection graphs only appear explicitly in Section 7, variations are encountered in the tree compatibility algorithm of Aho et al. and the incomplete directed binary compatibility procedure of Pe'er et al. (Section 5). The idea is also used implicitly in Section 4. Even the useful concept of the decomposition of $\mathcal{S}$ induced by a taxon (Sections 5–7) carries at its heart the notion of intersection.

Acknowledgments

This work was supported in part by the National Science Foundation under grants CCR-9520946 and CCR-9988348. Thanks to Duhong Chen for useful discussions and references.

References

[1] Richa Agarwala and David Fernández-Baca. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM J. Computing*, 23:1216–1224, 1994.

[2] Richa Agarwala and David Fernández-Baca. Simple algorithms for perfect phylogeny and triangulating colored graphs. *International Journal of Foundations of Computer Science*, 7(1):11–21, 1996.

[3] Richa Agarwala, David Fernández-Baca, and Giora Slutzki. Fast algorithms for inferring evolutionary trees. *Journal of Computational Biology*, 2(3):397–408, 1995.

[4] A.V. Aho, Y. Sagiv, T.G. Szymanski, and J.D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J. Computing*, 10(3), 1981.

[5] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Meth.*, 8:277–284, 1987.

[6] H. L. Bodlaender, M. R. Fellows, M. T. Hallet, H. T. Wareham, and T. J. Warnow. The hardness of perfect phylogeny, feasible register assignment, and other problems on thin colored graphs. *Theoretical Computer Science*, 244:167–188, 2000.

[7] Hans Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Computing*, 25:1305–1317, 1996.

[8] Hans Bodlaender, Michael Fellows, and Tandy Warnow. Two strikes against perfect phylogeny. In *19th International Colloquium on Automata, Languages, and Programming*, pages 273–283. Springer Verlag, Lecture Notes in Computer Science, 1992.

[9] Hans L. Bodlaender and Ton Kloks. A simple linear-time algorithm for triangulating three-colored graphs. *Journal of Algorithms*, 15(1):160–172, 1993.

[10] Maria Bonet, Cynthia Phillips, Tandy Warnow, and Shibu Yooseph. Constructing evolutionary trees in the presence of polymorphic characters. *SIAM J. Computing*, 29(1):103–131, 1999.

[11] David Bryant. *Building trees, hunting for trees, and comparing trees: Theory and methods in phylogenetic analysis.* PhD thesis, Department of Mathematics, University of Canterbury, New Zealand, 1997.

[12] P. Buneman. A characterisation of rigid circuit graphs. *Discrete Math.*, 9:205–212, 1974.

[13] J. H. Camin and R. R. Sokal. A method for deducing branching sequences in phylogeny. *Evolution*, 19:311–326, 1965.

[14] J. Chen, J. Kanj, and W. Jia. Vertex cover: further observations and further improvements. In *WGG '99*, volume 1665 of *Lecture Notes in Computer Science*, pages 313–324. Springer-Verlag, 1999.

[15] W. H. E. Day, D. S. Johnson, and D. Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences*, 81:33–42, 1986.

[16] L. Dollo. Les lois de l'évolution. *Bulletin de la Societé Belge de Géologie de Paleontologie et d'Hydrologie*, 7:164–167, 1893.

[17] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, New York, NY, 1999.

[18] A. Dress and M. Steel. Convex tree realizations of partitions. *Appl. Math. Letters*, 5(3):3–6, 1992.

[19] G. F. Estabrook. Phylogenetic trees and character state trees. In T. Duncan and T. F. Stuessy, editors, *Cladistics*, pages 131–151. Columbia University Press, New York, NY, 1984.

[20] G. F. Estabrook. Ancestor-descendant relations and incompatible data: Motivation for research in discrete mathematics. In B. Mirkin, F. R. McMorris, F. S. Roberts, and A. Rzhetsky, editors, *Mathematical Hierarchies and Biology*, volume 37 of *DIMACS Series in Discrete Mathematics*, pages 1–28. American Mathematical Society, 1997.

[21] G. F. Estabrook, C. S. Johnson Jr., and F. R. McMorris. An idealized concept of the true cladistic character. *Mathematical Biosciences*, 23:263–272, 1975.

[22] G. F. Estabrook and C. A. Meacham. How to determine the compatibility of undirected character state trees. *Mathematical Biosciences*, 46:251–256, 1979.

[23] J. S. Farris. Phylogenetic analysis under Dollo's law. *Syst. Zool.*, 26:77–88, 1977.

[24] J. Felsenstein. PHYLIP homepage. `http://evolution.genetics.washington.edu/phylip.html`.

[25] David Fernández-Baca and Jens Lagergren. A polynomial-time algorithm for near-perfect phylogeny. In *Proc. 23rd International Conference on Automata, Languages, and Programming*, Lecture Notes in Computer Science, pages 670–680. Springer-Verlag, 1996.

[26] David Fernández-Baca and Jens Lagergren. On the approximability of the steiner tree problem in phylogeny. *Discrete Applied Mathematics*, 88(1):127–143, 1999. Special issue on Computational Biology.

[27] W. Fitch. Towards defining the course of evolution: minimum change for a specified tree topology. *Syst. Zool.*, 20:406–416, 1971.

[28] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Adv. Appl. Math.*, 3:43–49, 1982.

[29] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Combin. Theory Ser. B*, 16:47–56, 1974.

[30] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, San Diego, 1980.

[31] Dan Gusfield. The Steiner tree problem in phylogeny. Technical Report 334, Computer Science Department, Yale University, New Haven, Conn., 1984.

[32] Dan Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28, 1991.

[33] J. A. Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29:53–65, 1973.

[34] Monika Rauch Henzinger, Valerie King, and Tandy Warnow. Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. *Algorithmica*, 24:1–13, 1999.

[35] S. Hougardy and H. J. Prömel. A 1.598 approximation for the Steiner problem in graphs. In *Proceedings of the Tenth Annual Symposium on Discrete Algorithms*, pages 448–453, 1999.

[36] Tao Jiang, Paul Kerney, and Ming Li. Orchestrating quartets: approximation and error correction. In *Proc. 39th IEEE Symp. on Foundations of Computer Science*, pages 416–425, 1998.

[37] Sampath Kannan and Tandy Warnow. Inferring evolutionary history from DNA sequences. In *Proceedings of the 31st Annual Symposium on the Foundations of Computer Science, St. Louis, Missouri, 1990*, pages 362–378, 1990.

[38] Sampath Kannan and Tandy Warnow. Triangulating three-colored graphs. *SIAM J. on Discrete Mathematics*, 5:249–258, 1992.

[39] Sampath Kannan and Tandy Warnow. A fast algorithm for the computation and enumeration of perfect phylogenies. *SIAM J. Computing*, 26(6):1749–1763, 1997.

[40] Paul Kearney, Ming Li, John Tsang, and Tao Jiang. Recovering branches on the tree of life: an approximation algorithm. In *Proc. 10th ACM-SIAM Symp. Discrete Algorithms*, pages 537–546, 1999.

[41] I. J. Kitching, P. L. Forey, C. J. Humphries, and David M. Williams. *Cladistics: The theory and practice of parsimony analysis.* Oxford University Press, Oxford, New York, 1998.

[42] T Margush and F. R. McMorris. Consensus n-trees. *Bulletin of Mathematical Biology*, 43:239–244, 1981.

[43] F. R. McMorris. On the compatibility of binary qualitative taxonomic characters. *Bull. Math. Biol.*, 39:133–138, 1977.

[44] F. R. McMorris and McKee T. A. *Topics in Intersection Graph Theory.* SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA, 1999.

[45] F. R. McMorris, T. J. Warnow, and T. Wimer. Triangulating vertex colored graphs. *SIAM Journal on Discrete Mathematics*, 7(2), 1994. Preliminary version in 4th Annual Symposium on Discrete Algorithms, Austin, Texas, 1993.

[46] Christopher A. Meacham. A manual method for character compatibility analysis. *Taxon*, 30(3):591–600, 1981.

[47] M. Nikaido, A. P. Rooney, and N. Okada. Phylogenetic relationships among certiodactyls based on insertions of short and long interspersed elements: Hyppopotamuses are the closest extant relatives of whales. *Proc. Natl. Acad. Sci. USA*, 96:10261–10266, 1999.

[48] R. D. M. Page and E. C. Holmes. *Molecular Evolution: A phylogenetic approach.* Blackwell Science, Oxford, 1998.

[49] I. Pe'er, R. Shamir, and R. Sharan. Incomplete directed perfect phylogeny. In R. Giancarlo and D. Sankoff, editors, *Combinatorial Pattern Matching*, volume 1848 of *Lecture Notes in Computer Science*, pages 143–153. Springer-Verlag, 2000.

[50] Neal Robertson and P. D. Seymour. Graph minors II: Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[51] Charles Semple and Mike Steel. A supertree method for rooted trees. *Discrete Applied Mathematics*, 105:147–158, 2000.

[52] M. A. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.

[53] D. L. Swofford, G. J. Olsen, P. J. Waddel, and D. M Hillis. Phylogenetic inference. In D. M. Hillis, C. Moritz, and B. K. Mable, editors, *Molecular Systematics*, chapter 11, pages 407–509. Sinauer Assoc., Sunderland, Mass, 1996.

[54] David Swofford. PAUP* homepage. `http://www.lms.si.edu/PAUP`.

[55] W. H. Wagner. Problems in the classification of ferns. *Recent Advances in Botany*, 1:841–844, 1961.

[56] Tandy J. Warnow. *Combinatorial algorithms for constructing phylogenetic trees.* PhD thesis, University of California, Berkeley, May 1991.

[57] A. Zelikovsky. An 11/6-approximation for the network Steiner problem. *Algorithmica*, 9:463–470, 1993.