

A Polynomial-Time Algorithm for the Perfect Phylogeny Problem when the Number of Character States is Fixed

RICHA AGARWALA* AND DAVID FERNÁNDEZ-BACA†

Department of Computer Science, Iowa State University, Ames, IA 50011

July 23, 1993

Abstract

We present a polynomial-time algorithm for determining whether a set of species, described by the characters they exhibit, has a perfect phylogeny, assuming the maximum number of possible states for a character is fixed. This solves a longstanding open problem. Our result should be contrasted with the proof by Steel and Bodlaender, Fellows, and Warnow that the perfect phylogeny problem is NP-complete in general.

Key words. Algorithms, character compatibility, evolution, perfect phylogeny.

AMS subject classifications. 68Q25, 68R, 92A12, 92-08.

Abbreviated title. A polynomial algorithm for the perfect phylogeny problem.

1 Introduction

A fundamental problem in biology is that of inferring the evolutionary history of a set of species, each of which is specified by the set of *traits* or *characters* that it exhibits [9, 11, 17, 18]. Information about evolutionary history can be conveniently represented by an evolutionary or *phylogenetic* tree, often referred to simply as a *phylogeny*. In one of the standard models, the problem can be expressed mathematically as follows. Let $\mathcal{C} = \{1, \dots, m\}$ be the *character set*, and for every $c \in \mathcal{C}$, let $\mathcal{A}_c = \{1, \dots, r_c\}$ be the set of allowable *states* for character c . We write r to denote $\max_{c \in \mathcal{C}} r_c$. A *species* \mathbf{s} is a vector (s_1, \dots, s_m) such that $\mathbf{s} \in \mathcal{A}_1 \times \dots \times \mathcal{A}_m$; s_c is referred to as the *state of character c* for

*Supported in part by a College of Liberal Arts and Sciences Research Assistantship, Iowa State University. E-mail address: agarwala@iastate.edu.

†Supported in part by the National Science Foundation under grants CCR-8909626 and CCR-9211262. This author gratefully acknowledges the support of DIMACS, at Rutgers University, where much of this work was carried out. E-mail address: fernande@cs.iastate.edu.

\mathbf{s} , or the *state of \mathbf{s} on character c* . We assume that if $i \in \mathcal{A}_c$, then there exists a species $\mathbf{s} \in \mathcal{S}$ such that $s_c = i$. The *perfect phylogeny problem* is to determine whether a given set of n distinct species \mathcal{S} has a tree T with the following properties:

(C1) $\mathcal{S} \subseteq V(T) \subseteq \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$,

(C2) Every leaf in T is in \mathcal{S} .

(C3) For every $c \in \mathcal{C}$ and every $j \in \mathcal{A}_c$, the set of all $\mathbf{u} \in V(T)$ such that $u_c = j$ induce a subtree of T .

The tree T , if it exists, is called a *perfect phylogeny* for \mathcal{S} and the set of characters \mathcal{C} is said to be *compatible*. We should point out that in the biology literature, the perfect phylogeny problem is more commonly known as the *character compatibility problem* [6]. In this context, one is frequently interested in computing a maximal set of compatible characters, since, in practice, character sets tend to be incompatible. Note also that instances of the phylogeny problem are often expressed in matrix form, by giving the set of species \mathcal{S} as an $n \times m$ matrix M whose rows are the species in \mathcal{S} [5, 14].

The perfect phylogeny problem was shown to be NP-complete by Bodlaender et al. [2] and, independently, by Steel [20]. This fact suggests at least two lines of attack: one is to restrict m , the number of characters; the other is to restrict r . Pursuing the first approach, McMorris, Warnow, and Wimer have shown that the perfect phylogeny problem is solvable in $O(n^{m+1})$ time [19], which is polynomial for every fixed m ; linear time algorithms have been found for $m = 3$ [3, 15]. In this paper, we pursue the second approach. Gusfield [14] gave a $O(nm)$ algorithm for $r = 2$, the *binary* character case; the procedure is optimal if the input is given as a matrix of zeros and ones. In [1] it is shown that, under a suitable representation of the input, the run time can be reduced to $O(C)$ where C is the number of ones in the matrix. This reference also presents efficient algorithms for dynamic insertion and deletion of species and characters. All of the above results for the binary case are based on an elegant and well-known characterization of the set of “yes” instances [8]. Dress and Steel [7] devised a $O(nm^2)$ algorithm for $r \leq 3$. Kannan and Warnow [16] gave a $O(n^2m)$ algorithm for $r \leq 4$ (*quaternary* characters) and conjectured the existence of a $O(r^{r-2}n^2m)$ algorithm, which is polynomial for every fixed r . Here we prove the existence of a polynomial-time algorithm for any fixed r by giving a $O(2^{3r}(nm^3 + m^4))$ algorithm for the perfect phylogeny problem.

Instances of the perfect phylogeny problem where there is a known upper bound on the number of states per character are of interest to biologists. For instance, quaternary characters arise when using DNA to describe species; each possible state of a character corresponds to a nucleotide, A, G, C, or T. Other instances of interest are those where $r = 20$, which occur when species are described by protein sequences. These can be viewed as strings from a 20 letter alphabet, each letter corresponding to one amino acid. While, admittedly, the running time of our algorithm grows quickly with r , it is a worst-case estimate which, in practice may not pose too large a problem for current computational technology. Furthermore, the dependency on r can be reduced by a factor of 2^r using recent ideas due to Kannan and Warnow (see Section 6).

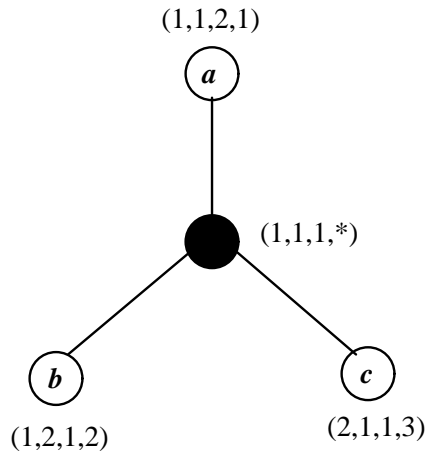


Figure 1: Forced and unforced states.

2 Preliminaries

We now introduce some definitions and prove certain preliminary results.

Definition 1 Suppose T is a perfect phylogeny for \mathcal{S} and let \mathbf{p} be some vertex in T . We shall say that the state of \mathbf{p} on character c is *forced* if \mathbf{p} lies on the path between vertices \mathbf{a} and \mathbf{b} in \mathcal{S} such that $a_c = b_c$. (Observe that if this is the case, in order to satisfy condition (C3) we must have $p_c = a_c = b_c$.)

If the state of a character of a node is unforced, several assignments may be possible. In Figure 1, for example, the state of the fourth character of the internal node is unforced and we could assign it a value of 1, 2 or 3.

Lemma 1 *A set of species \mathcal{S} has a perfect phylogeny if and only if every subset of \mathcal{S} has one.*

Proof The “if” part is trivial. For the “only if” part, let \mathcal{S}' be any subset of \mathcal{S} and let T be a perfect phylogeny of \mathcal{S} . Clearly, T satisfies (C1) and (C3) for \mathcal{S}' , but, possibly, not (C2). To obtain a perfect phylogeny for \mathcal{S}' , repeatedly delete from T any leaf that is not in \mathcal{S}' , until this operation is no longer possible. Since each deletion preserves properties (C1) and (C3) for \mathcal{S}' , the final tree will also satisfy (C2) for \mathcal{S}' . \square

Definition 2 Suppose $G \subset \mathcal{S}$ and let $G' = \mathcal{S} - G$. $\mathcal{D}(G)$, the set of *distinguishing characters* of G , is the set of all $c \in \mathcal{C}$ such that for every $\mathbf{a} \in G$ and every $\mathbf{b} \in G'$, $a_c \neq b_c$. $\mathcal{M}(G)$, the set of *common characters*, is $\mathcal{C} - \mathcal{D}(G)$.

Obviously, $\mathcal{D}(G) = \mathcal{D}(G')$ and $\mathcal{M}(G) = \mathcal{M}(G')$.

Definition 3 A pair (G, G') where $G \subset \mathcal{S}$ and $G' = \mathcal{S} - G$ is called a *split* if, for every character, the number of common character states between G and G' is at most one. A split (G, G') is a *c-split* if $\mathcal{D}(G) \neq \emptyset$. If (G, G') is a split (c-split), G and G' are called *clusters* (*c-clusters*).

Whereas there can be up to $2^{n-1} - 1$ splits, the total number of c-splits is at most $(2^{r-1} - 1) \cdot m$. Observe that we can determine whether a partition (G, G') of \mathcal{S} is a split in $O(nm)$ time. Note also that if G is a cluster but not a c-cluster, then $\mathcal{M}(G) = \mathcal{C}$.

Definition 4 Let (G, G') be a split. We say that (G, G') is of *type I* if there exists an $\mathbf{s} \in G$ such that for all $c \in \mathcal{M}(G)$, s_c equals the unique common state between G and G' on character c and $|G - \{\mathbf{s}\}|, |G'| \geq 1$. If (G, G') is of type I, we refer to \mathbf{s} as a *connecting species*. If (G, G') is not of type I, we say that it is of *type II*.

Checking whether a split (G_1, G_2) is of type I can be achieved $O(nm)$ time as follows. First, compute $\mathcal{M}(G_1)$ and the common state between G_1 and G_2 on each $c \in \mathcal{M}(G_1)$. This can be done in $O(nm)$ time by considering characters one at a time. Now, it suffices to search for a species \mathbf{s} such that for all $c \in \mathcal{M}(G_1)$, s_c equals the common state between G_1 and G_2 . This takes $O(m)$ time per species, for a total of $O(nm)$ time.

If there is a type I c-split (G_1, G_2) where \mathbf{s} is a connecting species, the problem can be divided into constructing perfect phylogenies T_1 and T_2 for $G_1 \cup \{\mathbf{s}\}$ and $G_2 \cup \{\mathbf{s}\}$. If one or both of the latter sets has no perfect phylogeny then, by Lemma 1, neither does \mathcal{S} . If both sets have perfect phylogenies, then, a perfect phylogeny for \mathcal{S} is obtained by identifying the nodes for \mathbf{s} in T_1 and T_2 . We now consider the case where all c-splits are of type II.

Lemma 2 *If all c-splits are of type II, then, in every perfect phylogeny T of \mathcal{S} , every species $s \in \mathcal{S}$ is a leaf in T .*

Proof Suppose there exists a species $\mathbf{s} \in \mathcal{S}$ such that \mathbf{s} is an internal node in some perfect phylogeny T of \mathcal{S} . Let T' be any connected component of $T - \mathbf{s}$, let G' be the set of species in T' and $G = \mathcal{S} - G'$. Clearly, (G, G') is a c-split with $|G'| \geq 1$ and $|\mathcal{S} - G'| \geq 2$. One can also readily verify that c-split (G, G') is of type I, with \mathbf{s} as a connecting species. \square

3 Subphylogenies

In this section and the next we assume that \mathcal{S} has no type I splits.

Definition 5 A *subphylogeny* T_G for a cluster G is a perfect phylogeny for G containing a node \mathbf{x} such that for every $c \in \mathcal{M}(G)$, x_c equals the (unique) common state for character c between G and $\mathcal{S} - G$ and for every $c \in \mathcal{D}(G)$, x_c is the state of some species in G on character c . Node \mathbf{x} is referred to as the *connection* of T_G .

The next result implies that, in searching for a perfect phylogeny for \mathcal{S} , we can restrict our attention to perfect phylogenies constructed entirely from subphylogenies.

Lemma 3 *\mathcal{S} has a perfect phylogeny if and only if there exists a split (G_1, G_2) such that both G_1 and G_2 have subphylogenies.*

Proof For the “if” part, let (G_1, G_2) be a split satisfying the requirements of the lemma and let T_1 and T_2 be subphylogenies for G_1 and G_2 , respectively. Let \mathbf{x}^1 and \mathbf{x}^2 be the connections of T_1 and T_2 . We can obtain a perfect phylogeny for \mathcal{S} by taking T_1 and T_2 and connecting them as follows. If $\mathcal{D}(G_1) = \emptyset$, identify \mathbf{x}^1 and \mathbf{x}^2 . Otherwise, add an edge $(\mathbf{x}^1, \mathbf{x}^2)$. It is not hard to check that conditions (C1)–(C3) hold.

For the “only if” part, let T be a perfect phylogeny for \mathcal{S} and let (\mathbf{u}, \mathbf{v}) be any edge in T . Without loss of generality, assume that every node in T that is not in \mathcal{S} has degree at least three. Let T_1 and T_2 be the subtrees of $T - (\mathbf{u}, \mathbf{v})$ containing \mathbf{u} and \mathbf{v} , respectively, and let $G_1 = \mathcal{S} \cap V(T_1)$ and $G_2 = \mathcal{S} - G_1$. T_1 and T_2 are obviously perfect phylogenies for G_1 and G_2 . We can construct a subphylogeny for G_1 from T_1 as follows. For each $c \in \mathcal{D}(G_1)$ such that u_c is not the state of some species in G_1 , let B_c be the set of all $\mathbf{b} \in V(T_1)$ such that $b_c = u_c$. Let \mathbf{d} be any node in $V(T_1) - B_c$ that is adjacent to a node in B_c . Such a \mathbf{d} must exist, for otherwise we would have $a_c = u_c$ for every $\mathbf{a} \in G_1$ contradicting the assumption that u_c is not the state of some species in G_1 . We must have $d_c = a_c$ for some $\mathbf{a} \in G_1$, for otherwise (C3) would be violated in T , as we would have $a_c = b_c$ for some $\mathbf{b} \in T_2$ and the node \mathbf{u} which is on the path between \mathbf{a} and \mathbf{b} in T is such that $u_c \neq a_c$. Now, set $b_c = d_c$ for all $\mathbf{b} \in B_c$. Since T satisfies (C3), for every $c \in \mathcal{M}(G_1)$, u_c equals the unique common state between G_1 and G_2 . The resulting modification of T_1 is therefore a subphylogeny for G_1 with connection \mathbf{u} . An analogous construction can be used to obtain a subphylogeny for G_2 . \square

Definition 6 A cluster G is said to be *compatible* with a vector \mathbf{s} if for every $c \in \mathcal{M}(G)$, s_c equals the unique common state for character c between G and $\mathcal{S} - G$.

The following result demonstrates that a subphylogeny for a cluster can always be assembled from subphylogenies for c-clusters.

Lemma 4 *Let G be a cluster. Then, G has a subphylogeny if and only if there exist pairwise disjoint c-clusters G_1, \dots, G_k and a vector \mathbf{x} such that (i) for every $c \in \mathcal{M}(G)$, x_c equals the (unique) common state for character c between G and $\mathcal{S} - G$, (ii) $\cup_{i=1}^k G_i = G$, and (iii) each G_i is compatible with \mathbf{x} and has a subphylogeny.*

Proof For the “if” part, let T_1, \dots, T_k be the subphylogenies for G_1, \dots, G_k with roots $\mathbf{x}^1, \dots, \mathbf{x}^k$. Clearly, the tree T consisting of a node for \mathbf{x} and the trees T_1, \dots, T_k connected to \mathbf{x} by edges $(\mathbf{x}^1, \mathbf{x}), \dots, (\mathbf{x}^k, \mathbf{x})$ is a subphylogeny for G .

For the “only if” part, let T be a subphylogeny for G with connection \mathbf{x} . Without loss of generality, assume that all nodes in T are distinct. Let $\mathbf{x}^1, \dots, \mathbf{x}^k$ be the neighbors of \mathbf{x} in T and for $1 \leq i \leq k$, let T_i be the subtree of $T - \mathbf{x}$ containing \mathbf{x}^i and let $G_i = V(T_i) \cap \mathcal{S}$. For each $c \in \mathcal{M}(G_i)$, x_c equals the unique common state between G_i and $\mathcal{S} - G_i$. This is because either this state is shared with some species in G_j , for some $j \neq i$, or it is shared with some species in $\mathcal{S} - G$. In either case, due to condition (C3), the value of x_c^i must equal the common state and, hence, G_i is compatible with \mathbf{x} . Also, as in the proof of Lemma 3, we can insure that for every character c , the state of any $\mathbf{v} \in V(T_i)$ on character c will be that of some species in G_i on c , by altering unforced states, if needed. Thus, T_i can be transformed into a subphylogeny for G_i . All that is left is to verify that each G_i is indeed a c-split; i.e., that $\mathcal{D}(G_i) \neq \emptyset$. Suppose $\mathcal{D}(G_i) = \emptyset$. Then we must have had $\mathbf{x}^i = \mathbf{x}$ in T , contradicting our earlier assumption that all nodes

are distinct, since for every character c , there is one common character state between G and $\mathcal{S} - G$ and condition (C3) must be satisfied in T . \square

To find a perfect phylogeny for \mathcal{S} , we shall rely on certain properties of subphylogenies which allow them to be combined into larger subphylogenies. These properties are discussed next.

Lemma 5 *Let G, G_1, G_2 be clusters such that $G = G_1 \cup G_2$ and $G_1 \cap G_2 = \emptyset$. If G_1 and G_2 have subphylogenies, then there exists a subphylogeny T for G .*

Proof Let T_1 and T_2 be subphylogenies for G_1 and G_2 respectively. Let \mathbf{x}^1 and \mathbf{x}^2 be the connections of T_1 and T_2 . Let \mathbf{x} be a node where for each $c \in \mathcal{M}(G)$, x_c equals the unique common state between G and $\mathcal{S} - G$ and for each $c \in \mathcal{D}(G)$, x_c equals x_c^1 . Construct T by adding a node \mathbf{x} and connecting the trees T_1 and T_2 to \mathbf{x} by edges $(\mathbf{x}, \mathbf{x}^1)$ and $(\mathbf{x}, \mathbf{x}^2)$ respectively; \mathbf{x} will be the connection of T . Since \mathbf{x} has the necessary states to be a connection for a subphylogeny of G , it suffices to prove that T is a perfect phylogeny for G . For this, we need to verify that T satisfies conditions (C1)–(C3). Since T_1 and T_2 are subphylogenies, and for every character c , x_c is the state of some species in G_1 or G_2 , it is clear that T satisfies (C1) and (C2) for G . To verify that T satisfies (C3), we must show that if $x_c^1 = x_c^2 = j$ for any character $c \in \mathcal{C}$, then $x_c = j$. This is trivially true when $c \in \mathcal{D}(G)$, since $x_c = x_c^1$. If $c \in \mathcal{M}(G)$, x_c equals the unique common state for character c between G and $\mathcal{S} - G$. We must have $x_c = x_c^1$ or $x_c = x_c^2$ because the species in G sharing the common character state with a species in $\mathcal{S} - G$ belongs to either G_1 or G_2 . Hence, T satisfies (C3).

Note that if $\mathcal{D}(G_1) = \emptyset$, then, rather than adding an edge $(\mathbf{x}, \mathbf{x}^1)$, we can simply identify nodes \mathbf{x}^1 and \mathbf{x} . A similar situation arises when $\mathcal{D}(G_2) = \emptyset$. \square

Lemma 6 *Let G, G_1, G_2 be clusters such that $G = G_1 \cup G_2$ and $G_1 \cap G_2 = \emptyset$. Suppose that G_1 has a subphylogeny T_1 and that there exists a subphylogeny T for G with T_1 as a subtree at the connection \mathbf{x} of T . Then if G_2 is not a c -cluster, the value of x_c on every $c \in \mathcal{D}(G)$ is forced.*

Proof If G_2 is not a c -cluster, $\mathcal{M}(G_2) = \mathcal{C}$; i.e., for every $c \in \mathcal{C}$, there exists a common state between a species in G_2 and one in $\mathcal{S} - G_2$. Consider any character $c \in \mathcal{D}(G)$. We claim that G_1 and G_2 share a state on c . This implies that x_c is forced. To prove the claim, assume the contrary. Since $c \in \mathcal{M}(G_2)$, G_2 must share a state with $\mathcal{S} - G$ on character c . But this would imply that $c \in \mathcal{M}(G)$, a contradiction. \square

4 Building a Subphylogeny

The heart of our perfect phylogeny algorithm is a procedure SUBPHYLOGENY that determines whether a cluster G has a subphylogeny and, if so, constructs one. It assumes that for every c -cluster $G' \subset G$, a subphylogeny has been constructed, if one exists.

Algorithm SUBPHYLOGENY(G)

```

begin
  if  $|G| = 1$  then
(S1)   return the tree  $T_G$  consisting of the single species  $\mathbf{a} \in G$ 
(S2)   for each c-cluster  $G_1 \subset G$  such that  $G_1$  has a subphylogeny  $T_1$  do
         $G_2 = G - G_1$ 
        if  $G_2$  is a c-cluster having a subphylogeny then
            Use Lemma 5 to construct a subphylogeny  $T_G$  for  $G$ 
(S3)   return  $T_G$ 
        else /*  $G_2$  is not a c-cluster */
            Use Lemma 6 to compute the states of the connection  $\mathbf{x}$  of  $T_G$ 
            Initialize  $T_G$  to consist of  $\mathbf{x}$  together with  $T_1$  as its subtree
(S4)   for each c-cluster  $H \subseteq G_2$  do
            if  $H$  has a subphylogeny  $T_H$  and  $H$  is compatible with  $\mathbf{x}$  then
                 $G_2 = G_2 - H$ 
                Make  $T_H$  a subtree of  $\mathbf{x}$  in  $T_G$ 
(S5)   if  $G_2 = \emptyset$  then return  $T_G$ 
        endif
    endfor
    return FAILURE
end

```

Lemma 7 *Let G be a cluster and suppose that for every c-cluster G' such that $G' \subset G$, we have determined whether G' has a subphylogeny and, if so, one has been constructed. Then, if G has a subphylogeny, SUBPHYLOGENY(G) constructs it. Otherwise, the procedure returns FAILURE.*

Proof When $|G| = 1$, a node for the single species $\mathbf{s} \in G$ is indeed a subphylogeny for G . Hence, the tree returned in (S1) is correct.

Suppose $|G| > 1$ and that G has a subphylogeny T_G with connection \mathbf{x} . Then, there must exist a c-cluster $G_1 \subseteq G$ having a subphylogeny T_1 such that T_1 is a subtree in some subphylogeny T_G of G . At some point during the execution of **for** loop (S2), we will consider one such G_1 . If $G_2 = G - G_1$ is a c-cluster having a subphylogeny, then, by Lemma 5, (S3) returns a subphylogeny for G .

If G_2 is not a c-cluster, then, by Lemma 6, the states of the connection \mathbf{x} are completely determined and, by Lemma 4, there exists a collection of pairwise disjoint c-clusters having subphylogenies such that their union is G and each c-cluster in the collection is compatible with \mathbf{x} . In the **for** loop (S4), a subphylogeny for a c-cluster $H \subseteq G$ is added to the current T_G only if H is compatible with \mathbf{x} and every species in H is not yet in T_G , and a tree is returned only when the union of all the c-clusters added is $G - G_1$; i.e., $G_2 = \emptyset$. Therefore, any tree T_G returned in (S5) is a subphylogeny for G . We now argue that if there exists a subphylogeny for G with T_1 as a subtree, then one such subphylogeny will be constructed by the loop (S4) and G_2 will become empty. We shall do this by showing that if there is such a subphylogeny for G , then at the beginning of each iteration of loop (S4), there is always at least one c-cluster $H' \subseteq G_2$ compatible with \mathbf{x} that has not yet been considered.

By Lemma 1, if G has a subphylogeny, then for every $J \subseteq G$, the set of species $J \cup \{\mathbf{x}\}$ has a perfect phylogeny. We claim that there exists a c-cluster $J' \subseteq J$ such that J' has a subphylogeny. To prove this, suppose T_J is a perfect phylogeny for $J \cup \{\mathbf{x}\}$, let \mathbf{y} be any neighbor of \mathbf{x} in T_J , and let J' be the set of species in the subtree T_y of $T_J - \mathbf{x}$ containing \mathbf{y} . Clearly, J' is a cluster; moreover, J' is a c-cluster since we can assume that all the nodes in T_J are distinct. Furthermore, T_y can be modified to obtain a subphylogeny for J' as in the proof of Lemma 3.

In particular, if we take $J = G_2$ in the above argument, we have that in each iteration of (S4), there exists a c-cluster $J' \subseteq G_2$ such that J' has a subphylogeny. Since, at the beginning of each iteration, for all c-clusters J'' considered up to this point, $J'' \not\subseteq G_2$, either H itself is a subset of G_2 and has a subphylogeny, or, some yet to be considered c-cluster H' is a subset of G_2 and has a subphylogeny.

Subphylogeny returns FAILURE only if no choice of G_1 led to the construction of a subphylogeny for G . In this event, there was certainly no subphylogeny for G . \square

Running time of SUBPHYLOGENY. SUBPHYLOGENY(G) considers each of the $O(2^r m)$ c-clusters G_1 such that $|G_1| < |G|$. For each such c-cluster, it verifies that $G_1 \subset G$, which can be done in $O(n)$ time. With a particular G_1 , the algorithm goes through $O(2^r m)$ c-clusters, checking in $O(n + m)$ time whether they are subsets of G_2 that are compatible with \mathbf{x} . The total time taken by SUBPHYLOGENY is therefore $O(2^{2r}(nm^2 + m^3))$.

5 Building a perfect phylogeny

We now describe the algorithm PHYLOGENY, which constructs a perfect phylogeny for \mathcal{S} , if it has one. The algorithm first tries to find if one of the $O(2^{r-1} \cdot m)$ c-splits is of type I. If there is a type I c-split (G_1, G_2) where \mathbf{s} is a connecting species, the algorithm recursively attempts to construct perfect phylogenies T_1 and T_2 for $G_1 \cup \{\mathbf{s}\}$ and $G_2 \cup \{\mathbf{s}\}$. As stated earlier, if one or both of the latter sets has no perfect phylogeny then, by Lemma 1, neither does \mathcal{S} . Otherwise, a perfect phylogeny for \mathcal{S} is obtained by identifying the nodes for \mathbf{s} in T_1 and T_2 .

If there is no type I c-split then, by Lemma 2, none of the species appears as an internal node in any perfect phylogeny for \mathcal{S} . In this case, PHYLOGENY considers each possible c-cluster G such that $|G| \leq n - 1$ and attempts to build a subphylogeny for it using SUBPHYLOGENY. It then uses this information to build a perfect phylogeny for \mathcal{S} , if possible. The steps of PHYLOGENY are as follows.

Algorithm PHYLOGENY(\mathcal{S})

```

begin
  if  $|\mathcal{S}| = 1$  then
    return the tree  $T$  consisting of the single species  $\mathbf{a} \in \mathcal{S}$ 
  if there exists a type I c-split  $(G_1, G_2)$  then
    Let  $\mathbf{s}$  be the connecting species
    Call PHYLOGENY( $G_1 \cup \{\mathbf{s}\}$ ) and PHYLOGENY( $G_2 \cup \{\mathbf{s}\}$ )
    if both calls succeed then
      Combine the resulting trees into a perfect phylogeny for  $\mathcal{S}$ 

```

```

    else return FAILURE
else /* All c-splits are of type-II */
    size = 1
    while size ≤ n - 1 do
(P1)   for each c-cluster  $G$  such that  $|G| = size$  do
        Call SUBPHYLOGENY( $G$ )
        if  $G$  has a subphylogeny  $T_G$ , record  $T_G$  and its connection
        endwhile
        size = size + 1
    endwhile
    Pick any  $\mathbf{s} \in \mathcal{S}$ 
    if  $G = \mathcal{S} - \{\mathbf{s}\}$  has a subphylogeny  $T_G$  then
        let  $\mathbf{x}$  be the connection of  $T_G$ 
(P2)   return the tree obtained by adding a node  $\mathbf{s}$  and the edge  $(\mathbf{s}, \mathbf{x})$  to  $T_G$ 
(P3)   else return FAILURE
    endif
end

```

Theorem 8 PHYLOGENY *correctly determines whether or not \mathcal{S} has a perfect phylogeny and, if so, constructs one.*

Proof The correctness of the algorithm hinges on how it deals with the case where no c-split is of type I. We first note that by the proof of Lemma 3 and because for any $\mathbf{s} \in \mathcal{S}$ the tree consisting of node \mathbf{s} is a subphylogeny for $\{\mathbf{s}\}$, if a tree is returned by our algorithm in (P2), then the tree is a perfect phylogeny for \mathcal{S} . Thus, it suffices to argue that PHYLOGENY will never return FAILURE if \mathcal{S} has a perfect phylogeny.

If \mathcal{S} has a perfect phylogeny and all c-splits are of type II, then every $\mathbf{s} \in \mathcal{S}$ will be a leaf in any perfect phylogeny of \mathcal{S} . Hence, both $\{\mathbf{s}\}$ and $\mathcal{S} - \{\mathbf{s}\}$ must be c-clusters. As we noted above, a subphylogeny for $\{\mathbf{s}\}$ is \mathbf{s} itself, while the subphylogeny of $\mathcal{S} - \{\mathbf{s}\}$ must have been constructed in some iteration of (P1). Therefore, (P2) will not return FAILURE. \square

Running time of PHYLOGENY. PHYLOGENY spends $O(2^r nm^2)$ time generating c-clusters and testing each of these to find out whether it is of type I. It is clear that, in the worst case, the running time of PHYLOGENY is dominated by the time required to deal with the case where all c-clusters are of type II. SUBPHYLOGENY is applied to each of the $O(2^r m)$ c-clusters, which requires $O(2^{3r}(nm^3 + m^4))$ total time. Hence, the running time of PHYLOGENY is $O(2^{3r}(nm^3 + m^4))$ as well.

6 Concluding remarks

Our algorithm uses dynamic programming to construct perfect phylogenies by working from the bottom up. One can use *memoization* (a technique described in some detail in pp. 312–314 of [4]) to obtain an equivalent top-down recursive algorithm with the same running time. Such a procedure has been proposed to us by E.L. Lawler (personal communication).

Algorithm PHYLOGENY can be modified to work correctly and within the same time bounds even if instances with type I c-splits are not treated separately. However, in practice, exploiting the presence of such splits may tend to reduce the running time of the algorithm.

Kannan and Warnow (personal communication) have discovered a clever way to reduce the running time of our algorithm by a factor of 2^r . Their technique speeds up step (S4) of SUBPHYLOGENY by providing a way to determine in $O(nm)$ time whether there exists a subphylogeny for G having a subphylogeny for a given c-cluster G_1 as a subtree. Even with this improvement, the algorithms presented in [16] and [7] are faster than ours for the cases where $r \leq 4$ and $r \leq 3$, respectively. It is an open problem whether our algorithm can be improved to match those bounds on those special cases.

Other methodologies for reconstructing phylogenies have been proposed in the past; these have been ably surveyed by Felsenstein [10]. We shall limit ourselves here to discussing one problem that is closely related to perfect phylogeny: the *Steiner tree problem in phylogeny* [12, 13]. A Steiner tree for a set of species \mathcal{S} is a tree T satisfying (C1) and (C2). Obviously, a perfect phylogeny for \mathcal{S} is also a Steiner tree for \mathcal{S} . The length of T is the sum of the lengths of its edges, where the length of an edge $(\mathbf{u}, \mathbf{v}) \in E(T)$ is the Hamming distance between \mathbf{u} and \mathbf{v} (i.e., the number of characters in which \mathbf{u} and \mathbf{v} differ). The Steiner tree problem in phylogeny is to compute a minimum length Steiner tree for a given set of species. Minimum length Steiner trees satisfy the *parsimony* criterion [10], as they give phylogenies in which species evolve with the least number of character changes. The following theorem establishes a relationship between the perfect phylogeny problem and the Steiner tree problem in phylogeny.

Theorem 9 \mathcal{S} has a perfect phylogeny if and only if the minimum length of a Steiner tree for \mathcal{S} is $\sum_{i=1}^m (r_i - 1)$.

Proof Suppose T is a Steiner tree. It follows from the definition that the length of T is the sum of contributions of the individual characters, where the contribution of character c to the length of T is the number of edges $(\mathbf{u}, \mathbf{v}) \in E(T)$ such that $u_c \neq v_c$. Since there are at least r_c species which differ from each other on any $c \in \mathcal{C}$, the contribution of character c to the length of T must be at least $r_c - 1$, implying that the length of any Steiner tree for \mathcal{S} is at least $\sum_{i=1}^m (r_i - 1)$.

A Steiner tree T for \mathcal{S} has length *exactly* $\sum_{i=1}^m (r_i - 1)$ if and only if for every $c \in \mathcal{C}$, T can be partitioned into exactly r_c subtrees T_1, \dots, T_{r_c} such that $u_c = i$ for every $\mathbf{u} \in T_i$. If such a partition is possible, T will satisfy (C3), in addition to (C1) and (C2), and must be a perfect phylogeny for \mathcal{S} . \square

It does not seem possible to generalize our algorithm in any straightforward way to produce minimum length Steiner trees for sets of species. To illustrate the difficulty in doing so, note that whereas for binary characters, the perfect phylogeny problem can be solved in time linear in the size of the input, the Steiner tree problem in phylogeny remains NP-complete [13].

Acknowledgements

We thank Martin Farach, Sampath Kannan, and Tandy Warnow for their encouragement, Mike Steel for bringing his results to our attention, and the referees for their useful suggestions. Special thanks are due to Eugene Lawler, who carefully read our manuscript and suggested numerous improvements in terminology and exposition.

References

- [1] R. Agarwala, D. Fernández-Baca, and G. Slutzki. Fast algorithms for inferring evolutionary trees. Technical Report 92-19, Department of Computer Science, Iowa State University, Ames, IA, 1992. In *Proceedings of the 30th Allerton Conference on Comm., Control, and Comput.*, 1992, pp. 594–603.
- [2] H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming*, pp. 273–283, Springer Verlag, Lecture Notes in Computer Science, 1992.
- [3] H. Bodlaender and T. Kloks. A simple linear time algorithm for triangulating three-colored graphs. In *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 415–423, 1992.
- [4] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Mass., 1990.
- [5] J. Camin and R. Sokal. A method for deducing branching sequences in phylogeny. *Evolution* 19, pp. 311–326, 1965.
- [6] W. H. E. Day and D. Sankoff. Computational complexity of inferring phylogenies by compatibility. *Syst. Zool.*, Vol. 35, No. 2, 224–229, 1986.
- [7] A. Dress and M. Steel. Convex tree realizations of partitions. *Appl. Math. Letters*, Vol. 5, No. 3, 3–6, 1992.
- [8] 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.
- [9] G. F. Estabrook. Cladistic methodology: A discussion of the theoretical basis for the induction of evolutionary history. *Annual Review of Ecology and Systematics*, Vol. 3, 427–456, 1972.
- [10] J. S. Felsenstein. Phylogenies from molecular sequences: Inference and Reliability. *Annual Reviews of Genetics*, Vol. 22, 521–565, 1988.
- [11] W. M. Fitch. Aspects of molecular evolution. *Annual Reviews of Genetics*, Vol. 7, 343–380, 1973.
- [12] L. R. Foulds. Maximum savings in the Steiner problem in phylogeny. *J. theoretical Biology*, Vol. 107, 471–474, 1983.
- [13] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, Vol. 3, 43–49, 1982.

- [14] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, Vol. 21, 19–28, 1991.
- [15] S. Kannan and T. Warnow. Triangulating three-colored graphs. *SIAM J. on Discrete Mathematics*, Vol. 5, 249–258, 1992.
- [16] S. Kannan and T. Warnow. Inferring evolutionary history from DNA sequences. In *Proceedings of the 31st Annual Symposium on the Foundations of Computer Science*, pp. 362–378, St. Louis, Missouri, 1990.
- [17] W. J. Le Quesne. A method of selection of characters in numerical taxonomy. *Syst. Zool.*, 18, 201–205, 1969.
- [18] W. J. Le Quesne. Further studies based on the uniquely derived character concept. *Syst. Zool.*, 21, 281–288, 1972.
- [19] F.R. McMorris, T.J. Warnow, and T. Wimer. Triangulating vertex colored graphs. In *Proceedings of the 4th Annual Symposium on Discrete Algorithms*, Austin, Texas, 1993.
- [20] M.A. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, Vol. 9, 91–116, 1992.