DIMACS Technical Report 93-04
March 1993

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

by

Richa Agarwala[1] and David Fernández-Baca[2,3]
Department of Computer Science
Iowa State University
Ames, Iowa 50011

[1]Supported in part a College of Liberal Arts and Sciences Research Assistantship, Iowa State University.
[2]Visiting Member
[3]Supported in part by the National Science Foundation under grants CCR-8909626 and CCR-9211262.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

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

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 [6, 7]. In mathematical terms, the problem can be expressed as follows. Let $\mathcal{C} = \{1, \ldots, m\}$ be the *character set*, and for every $c \in \mathcal{C}$, let $\mathcal{A}_c = \{1, \ldots, r_c\}$ be the set of allowable *states* for character c. We write r to denote $\max_{c \in \mathcal{C}} r_c$. A *species* $\boldsymbol{s}$ is a vector $(s_1, \ldots, s_m)$ such that $\boldsymbol{s} \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$; s_c is referred to as the *state of character c for $\boldsymbol{s}$*. 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 U of all $\boldsymbol{u} \in V(T)$ such that $u_c = j$ induce a subtree of T.

The tree T, if it exists, is called a *phylogeny* for $\mathcal{S}$. We should note 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 $\mathcal{S}$.

The phylogeny problem was shown to be NP-complete by Bodlaender et al. [2] and, independently, by Steel [12]. 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, for every fixed m, the phylogeny problem is solvable in polynomial time [11]. In this paper, we pursue the second approach. When $r = 2$, the *binary* character case, an elegant characterization of the set of "yes" instances [5] leads to efficient algorithms [1, 8]. Dress and Steel [4] devised a $O(nm^2)$ algorithm for $r \leq 3$. Kannan and Warnow [10] gave a polynomial-time algorithm for $r \leq 4$ and conjectured the existence polynomial-time algorithms for any fixed r. Here we prove this conjecture by giving a $O(2^{3r}(nm^3 + m^4))$ algorithm for the perfect phylogeny problem.

2 Preliminaries

We now introduce some definitions and prove certain preliminary results.

Definition 1 Suppose T is a phylogeny for $\mathcal{S}$ and let $\boldsymbol{p}$ be some vertex in T. We shall say that the state of $\boldsymbol{p}$ on character c is *forced* if $\boldsymbol{p}$ lies on the path between vertices $\boldsymbol{a}$ and $\boldsymbol{b}$ 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 phylogeny if and only if every subset of $\mathcal{S}$ has one.*

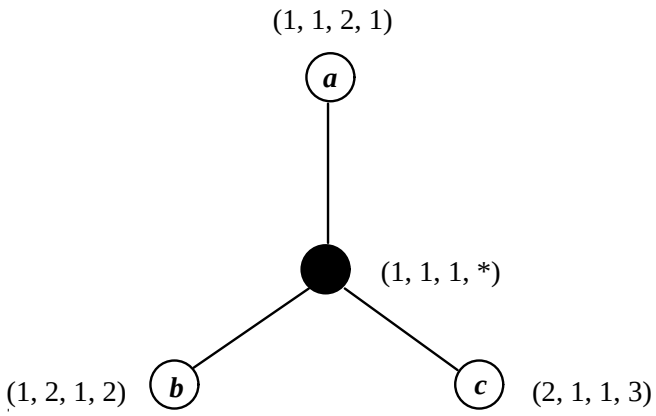

Figure 1: Forced and unforced states.

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 phylogeny of $\mathcal{S}$. Clearly, T satisfies (C1) and (C3) for $\mathcal{S}'$, but, possibly, not (C2). To obtain a 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 $\mathcal{S}'$. □

Definition 2 Two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are said to be *compatible* on a set of characters A if and only if $a_c = b_c$ for every $c \in A$.

Definition 3 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 $\boldsymbol{a} \in G$ and every $\boldsymbol{b} \in G'$, $a_c \neq b_c$. $\mathcal{M}(G)$, 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 4 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)*.

Note that, whereas the number of splits is at most $2^{n-1} - 1$, 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 that is not a c-cluster, then $\mathcal{D}(G) = \emptyset$.

Definition 5 Let (G, G') be a split. We say that (G, G') is of *type I* if there exists an $\boldsymbol{s} \in G$ such that for all $c \in \mathcal{M}(G)$, s_c equals the unique common state between G and G' on character c and $|G - \{s\}|, |G'| \geq 1$. If (G, G') is of type I, we refer to $\boldsymbol{s}$ as a *connecting species*. If (G, G') is not of type I, we say that it is of *type II*.

Lemma 2 *If all c-splits are of type II, then, in every phylogeny T of $\mathcal{S}$, every internal node of T is not in $\mathcal{S}$.*

Proof Suppose some phylogeny T of $\mathcal{S}$ has an internal node v in $\mathcal{S}$. Let T' be any connected component of $T - v$, let $G' = \mathcal{S} \cap V(T')$ 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 (G, G') is of type I, with v as a connecting species. □

Note that we can check whether or not a split (G_1, G_2) is of type I in $O(nm)$ time, since we simply need to check whether G_1 or G_2 contain a suitable species $\boldsymbol{s}$.

3 Subphylogenies

Definition 6 A *subphylogeny* T_G for a cluster G is a phylogeny for G containing a node $\boldsymbol{x}$ 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 $\boldsymbol{x}$ is referred to as the *connection* of T_G.

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

Lemma 3 *$\mathcal{S}$ has a 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 $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ be the connections of T_1 and T_2. We can obtain a phylogeny for $\mathcal{S}$ by taking T_1 and T_2 and connecting them as follows. If $\mathcal{D}(G_1) = \emptyset$, identify $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$. Otherwise, add an edge $(\boldsymbol{x}^1, \boldsymbol{x}^2)$. It is not hard to check that conditions (C1)–(C3) hold.

For the "only if" part, let T be a phylogeny for $\mathcal{S}$ and let $(\boldsymbol{u}, \boldsymbol{v})$ 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 $T - (\boldsymbol{u}, \boldsymbol{v})$ containing $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively, and let $G_1 = \mathcal{S} \cap V(T_1)$ and $G_2 = G - G_1$. T_1 and T_2 are obviously phylogenies for G_1 and G_2. We can construct a subphylogeny for G_1 from T_1 as follows. For each $c \in \mathcal{D}(G)$, let B_c be the set of all $\boldsymbol{b} \in V(T_1)$ such that $b_c = u_c$. Since T satisfies (C3), the states of these nodes on character c are unforced. Let $\boldsymbol{d}$ be any node in $V(T_1) - B_c$ that is adjacent to a node in B_c. Obviously, such a node must exist. Now, set $b_c = d_c$ for all $\boldsymbol{b} \in B_c$. Note that, since T_1 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 $\boldsymbol{u}$. An analogous construction can be used to obtain a subphylogeny for G_2. □

Definition 7 A cluster G is said to be *compatible* with a vector $\boldsymbol{s}$ 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 $G_1, \cdots, G_k$ and a vector $\boldsymbol{x}$ 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 $\boldsymbol{x}$ and has a subphylogeny.*

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

For the "only if " part, let T be a subphylogeny for G with connection $\boldsymbol{x}$. Without loss of generality, assume that all nodes in T are distinct. Let $\boldsymbol{x}^1, \cdots, \boldsymbol{x}^k$ be the neighbors of $\boldsymbol{x}$ in T and for $1 \leq i \leq k$, let T_k be the subtree of $T - \boldsymbol{x}$ containing $\boldsymbol{x}^i$ 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 $\boldsymbol{x}$. Also, as done in the proof of Lemma 4, we can insure that for every character c, the state of any $\boldsymbol{v} \in V(T_i)$ 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 $\boldsymbol{x}^i = \boldsymbol{x}$ 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. □

To find a 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 $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ be the connections of T_1 and T_2. Construct T by adding a node $\boldsymbol{x}$ and the edges $(\boldsymbol{x}, \boldsymbol{x}^1)$ and $(\boldsymbol{x}, \boldsymbol{x}^2)$, $\boldsymbol{x}$ will be the connection of T. The character states of $\boldsymbol{x}$ are defined as follows. For each $c \in \mathcal{M}(G)$, set x_c equal to the common state between G and $\mathcal{S} - G$. For $c \in \mathcal{D}(G)$, set $x_c = x_c^1$. Since $\boldsymbol{x}$ has the required states on characters in $\mathcal{M}(G)$, it suffices to prove that T is a phylogeny for G. For this we must show that $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are compatible with $\boldsymbol{x}$ on $\mathcal{M}(G_1)$ and $\mathcal{M}(G_2)$ respectively. We have two cases to consider:

$c \in \mathcal{M}(G)$**:** Then x_c equals the unique common state for character c between G and $\mathcal{S} - G$ and c must be in $\mathcal{M}(G_1) \cup \mathcal{M}(G_2)$. There are two possibilities:

$c \in \mathcal{M}(G_1) \cap \mathcal{D}(G_2)$ or $c \in \mathcal{M}(G_2) \cap \mathcal{D}(G_1)$. Consider the first of these two cases — the other can be dealt with analogously. Then there is no species in G_2 that has a common character state with a species in $\mathcal{S} - G_2$. Now since $G = G_1 \cup G_2$ and G

has a common character state with $\mathcal{S} - G$, the common state must be with respect to a species in G_1 and hence $x_c^1 = x_c$.

$c \in \mathcal{M}(G_1) \cap \mathcal{M}(G_2)$. Then we claim that $x_c^1 = x_c^2 = x_c$. First, note that we must have $x_c^1 = x_c$ or $x_c^2 = x_c$ 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.

We now show that $x_c^1 = x_c^2$. Assume the contrary. Since $c \in \mathcal{M}(G_1)$, there exists a species in G_1 that shares the character state x_c^1 with a species $\boldsymbol{a}$ in $\mathcal{S} - G_1$. Similarly since $c \in \mathcal{M}(G_2)$, there exists a species in G_2 that shares the character state x_c^2 with a species $\boldsymbol{b}$ in $\mathcal{S} - G_2$. Then $\boldsymbol{b} \notin G_1$, for if $\boldsymbol{b} \in G_1$, G_1 and $\mathcal{S} - G_1$ would share two character states on character c, contradicting the assumption that G_1 is a cluster. Similarly, $\boldsymbol{a} \notin G_2$. Hence, $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{S} - G$ and there are two common character states between G and $\mathcal{S} - G$ for character c. This contradicts the assumption that G is a cluster.

$c \in \mathcal{D}(G)$**:** There are three subcases:

$c \in \mathcal{D}(G_1) \cap \mathcal{D}(G_2)$. Then, setting $x_c = x_c^1$ will not affect the compatibility of $\boldsymbol{x}$ with $\boldsymbol{x}^2$.

$c \in \mathcal{M}(G_1) \cap \mathcal{M}(G_2)$. Then $x_c = x_c^1 = x_c^2$ as desired.

$(c \in \mathcal{M}(G_1) \cap \mathcal{D}(G_2))$ or $(c \in \mathcal{D}(G_1) \cap \mathcal{M}(G_2))$. Impossible, since $c \in \mathcal{D}(G)$.

Hence, $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are compatible with $\boldsymbol{x}$ on $\mathcal{M}(G_1)$ and $\mathcal{M}(G_2)$ respectively. Note that if $\mathcal{D}(G_1) = \emptyset$, then, rather than adding an edge $(\boldsymbol{x}, \boldsymbol{x}^1)$, we can simply identify nodes $\boldsymbol{x}^1$ and $\boldsymbol{x}$. A similar situation arises when $\mathcal{D}(G_2) = \emptyset$. □

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 $\boldsymbol{x}$ 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 We first show that $\mathcal{M}(G_1) \cup \mathcal{M}(G) = \mathcal{C}$. Suppose there exists some $c \notin \mathcal{M}(G_1) \cup \mathcal{M}(G)$. We claim that there can be no common state between G_2 and $\mathcal{S} - G_2$ on c. Suppose the common state is with a species in G_1. Then $c \in \mathcal{M}(G_1)$, a contradiction. We also arrive at a contradiction if we assume that the common state is with a species in $\mathcal{S} - G$. Thus, since there is no common state on c, we must have $c \in \mathcal{D}(G_2)$, contradicting the assumption that G_2 is not a c-cluster.

Hence, for every $c \in \mathcal{D}(G)$, we must have that $c \in \mathcal{M}(G_1)$ and thus G_1 and G_2 share a common state on c. Therefore, x_c must equal this common state. □

4 The Algorithm

We now describe the algorithm PHYLOGENY, which constructs a 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 $\boldsymbol{s}$ is a connecting species, the algorithm recursively attempts to construct phylogenies T_1 and T_2 for $G_1 \cup \{\boldsymbol{s}\}$ and $G_2 \cup \{\boldsymbol{s}\}$. If one or both of the latter sets has no phylogeny, by Lemma 1, neither does $\mathcal{S}$. If both of the sets have phylogenies, then, a phylogeny for $\mathcal{S}$ is obtained by identifying the nodes for $\boldsymbol{s}$ in T_1 and T_2.

If there is no type I c-split, by Lemma 2, none of the species appears as an internal node in any phylogeny for $\mathcal{S}$. PHYLOGENY deals with this situation as a special case, by invoking a procedure called PHASE-II, whose description shall occupy most of the remainder of this section. Before proceeding, however, we now summarize the steps of PHYLOGENY.

Algorithm PHYLOGENY($\mathcal{S}$)

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

Implementing PHASE-II

The heart of PHASE-II 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 it exists.

Algorithm SUBPHYLOGENY(G)

Step 1. If $|G| = 1$, then return the tree T_G where $V(T_G) = G$, whose connection is the single node in T_G.

Step 2. Consider each c-cluster G_1 such that $G_1 \subset G$ and G_1 has a subphylogeny. Let $G_2 = G - G_1$. For each G_1, apply the following steps to find if there exists a subphylogeny for G whose connection $\boldsymbol{x}$ has a subphylogeny for G_1 as one of its subtrees.

Step 2a. If G_2 is a c-cluster having a subphylogeny, construct a subphylogeny T_G for G by applying the construction in the proof of Lemma 5. Return T_G.

Step 2b. If G_2 is not a c-cluster, compute the states of the connection $\boldsymbol{x}$ as explained in Lemma 6. Let $R = \{H : H \subseteq G_2$ is a c-cluster that is compatible with $\boldsymbol{x}$ and has a subphylogeny$\}$. Initialize T_G to consist of $\boldsymbol{x}$ with a subphylogeny for G_1 as its subtree and repeat the following step while both R and G_2 are not empty.

ADD-TREE. Choose any $H \in R$ and set $R = R - H$. If $H \subseteq G_2$, set $G_2 \leftarrow G_2 - H$ and make T_H a subtree of $\boldsymbol{x}$ in T_G, where T_H is a subphylogeny for H.

Now, if $G_2 = \emptyset$, return T_G.

If all possible G_1's have been exhausted, return FAILURE.

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 $\boldsymbol{s} \in G$ is indeed a subphylogeny for G. Hence, the tree returned in Step 1 is correct.

Suppose $|G| > 1$ and that G has a subphylogeny T_G with connection $\boldsymbol{x}$. Then, there must exist a c-cluster $A \subseteq G$ having a subphylogeny T_A such that T_A is a subtree of $\boldsymbol{x}$ in T_G. At some point during the execution of Step 2, G_1 will equal one such A. If $G_2 = G - G_1$ is a c-cluster having a subphylogeny, then, by Lemma 5, Step 2a returns a subphylogeny for G. If G_2 is not a c-cluster, then, by Lemma 6, the states of the connection $\boldsymbol{x}$ are completely determined and, by Lemma 4, there exists a set of pairwise disjoint c-clusters $H_1, \cdots, H_k$ having subphylogenies such that (i) $G_1 \cup \bigcup_{i=1}^{k} H_i = G$ and (ii) $H_1, \cdots, H_k$ are compatible with $\boldsymbol{x}$. By assumption, subphylogenies $T_1, \cdots, T_k$ for $H_1, \cdots, H_k$ have already been constructed. Furthermore, all the H_j's will be contained initially in the set R of Step 2b.

It is clear that at all times during the execution of Step 2b R will contain every c-cluster B such that $B \subseteq G_2$, B has a subphylogeny, and B is compatible with $\boldsymbol{x}$. We shall show that in addition R contains a set $\mathcal{N}$ of pairwise disjoint c-clusters such that $\bigcup\{N : N \in \mathcal{N}\} = G_2$. As stated above, the initial R has the desired properties. We now prove that ADD-TREE preserves the invariant. During each execution of ADD-TREE, some H is removed from R. If $H \not\subseteq G_2$, then clearly it is not in $\mathcal{N}$, and the invariant is maintained. Otherwise, H is subtracted from G_2. Let R^0, G_2^0 and R^1, G_2^1 be the sets R and G_2 immediately before and immediately after an application of ADD-TREE. Then, $R^1 = R^0 - \{H\}$ and $G_2^1 = G_2^0 - H$. Let the subset $\mathcal{N}$ corresponding to R^0 be $\mathcal{N}^0$; we shall show that R^1 will have a corresponding subset $\mathcal{N}^1$. Now if $H \in \mathcal{N}^0$, R^1 clearly contains a subset $\mathcal{N}^1$ with the desired properties, namely $\mathcal{N}^1 = \mathcal{N}^0 - H$.

If $H \notin \mathcal{N}^0$, then, since $\bigcup\{N : N \in \mathcal{N}^0\} = G_2^0$, there exists $\mathcal{G} \subseteq \mathcal{N}^0$ such that for each $J \in \mathcal{G}$, $J \cap H \neq \emptyset$ and $\bigcup\{J : J \in \mathcal{G}\} \supseteq H$. We claim that for every $J \in \mathcal{G}$ there exist disjoint c-clusters compatible with $\boldsymbol{x}$ whose union is $J - H$, such that each c-cluster has a subphylogeny.

Since J is compatible with $\boldsymbol{x}$, there exists a phylogeny for $J \cup \{\boldsymbol{x}\}$. Therefore, by Lemma 1, there exists a phylogeny T' for $(J \cup \{\boldsymbol{x}\}) - H$. Now, let $\boldsymbol{x}^1, \cdots, \boldsymbol{x}^l$ be the neighbors of $\boldsymbol{x}$ in T' and for $1 \leq i \leq l$, let T_l be the subtree of $T' - \boldsymbol{x}$ containing $\boldsymbol{x}^i$ and let $J_i = V(T_i) \cap \mathcal{S}$. Then, a subphylogeny can be constructed for each J_i in much the same way as was done in

the proofs of Lemmas 3 and 4. Observe too that each J_i must be in R^0, since it is compatible with $\boldsymbol{x}$. Let $\mathcal{N}'$ consist of all these c-clusters for all $J \in \mathcal{G}$. Then $\mathcal{N}^1 = (\mathcal{N}^0 - \mathcal{G}) \cup \mathcal{N}'$ has the required properties.

Thus, assuming we have chosen the right G_1, a subphylogeny for G will be found. If no choice of G_1 yields a subphylogeny for G, then G has no subphylogeny and Step 2 correctly returns FAILURE. □

SUBPHYLOGENY is invoked by the following algorithm.

Algorithm PHASE-II

Step 1. Enumerate all c-clusters of size at most $n-1$ by non-decreasing order of cardinality. For each c-cluster G, determine whether G has a subphylogeny T_G. If so, record T_G, and its connection.

Step 2. For each $\boldsymbol{s} \in \mathcal{S}$, determine if $G = \mathcal{S} - \{\boldsymbol{s}\}$ has a subphylogeny. If so, let T_G be the tree and $\boldsymbol{x}$ be its connection. Return the tree obtained by adding a node $\boldsymbol{s}$ and the edge $(\boldsymbol{s}, \boldsymbol{x})$ to T_G.

Theorem 8 *If $\mathcal{S}$ has a phylogeny, then* PHASE-II *finds one.*

Proof Suppose there exists a phylogenetic tree T for $\mathcal{S}$. Let $\boldsymbol{s}$ be any species that is a leaf in T. Then $\mathcal{S} - \{\boldsymbol{s}\}$ is a c-cluster and has a subphylogeny. By Lemma 7, PHASE-II computes subphylogenies for all c-clusters G such that $|G| \leq n-1$ and, in particular, it finds a subphylogeny for $\mathcal{S} - \{\boldsymbol{s}\}$. Hence, at some point, Step 2 of PHASE-II obtains a phylogeny for $\mathcal{S}$. □

Analysis

We first establish the correctness of our algorithm.

Theorem 9 PHYLOGENY *correctly determines whether or not $\mathcal{S}$ has a phylogeny and, if so, constructs one.*

Proof Follows from Theorem 8, and the fact that we are dealing with type I c-splits properly. □

Next, we consider the running time of PHYLOGENY. First, we analyze the running time of a call to SUBPHYLOGENY(G). This procedure will consider 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 $\boldsymbol{x}$. The total time of SUBPHYLOGENY is therefore $O(2^{2r}(nm^2 + m^3))$.

In PHASE-II, we generate all c-clusters, which takes $O(2^r nm^2)$ time. Since SUBPHYLOGENY is applied to each c-cluster, PHASE-II takes $O(2^{3r}(nm^3 + m^4))$ time.

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. Hence, the running time of PHYLOGENY is $O(2^{3r}(nm^3 + m^4))$.

5 Remarks

Our algorithm constructs a phylogeny by working from the bottom up, following what is essentially a dynamic programming approach. One can use *memoization* (a technique described in some detail in pp. 312–314 of [3]) 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, identifying such splits and exploiting their presence may tend to reduce the running time of the algorithm when the phylogeny has one or more elements of $\mathcal{S}$ as internal nodes.

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 2 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 [10] and [4] 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.

Acknowledgements

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

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. To appear in *Proceedings of the 30th Allerton Conference on Comm., Control, and Comput.*, 1992.

[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] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Mass., 1990.

[4] A. Dress and M. Steel. Convex tree realizations of partitions. *Appl. Math. Letters*, Vol. 5, No. 3, 3–6, 1992.

[5] 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.

[6] 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.

[7] W. M. Fitch. Aspects of Molecular Evolution. *Annual Reviews of Genetics*, Vol. 7, 343–380, 1973.

[8] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, Vol. 21, 19–28, 1991.

[9] S. Kannan and T. Warnow. Triangulating three-colored graphs. *SIAM J. on Discrete Mathematics*, Vol. 5, 249–258, 1992.

[10] 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.

[11] F.R. McMorris, T.J. Warnow, and T. Wimer. Triangulating vertex colored graphs. To appear in *Proceedings of the 4th Annual Symposium on Discrete Algorithms*, Austin, Texas, 1993.

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