Fast and Simple Algorithms for Perfect Phylogeny and Triangulating Colored Graphs

Richa Agarwala*and David Fernández-Baca†

Abstract

This paper presents an $O((r-n/m)^m rnm)$ algorithm for determining whether a set of n species has a perfect phylogeny, where m is the number of characters used to describe a species and r is the maximum number of states that a character can be in. The perfect phylogeny algorithm leads to an $O((2e/k)^k e^2 k)$ algorithm for triangulating a k-colored graph having e edges.

1 Introduction

A fundamental problem in biology is that of inferring the evolutionary history of a set of species $\mathcal{S}$, each of which is specified by the set of *traits* or *characters* that it exhibits [9, 10, 15, 16]. 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 $\mathcal{C} = \{1, \ldots, m\}$ be the *character set*, and for every $c \in \mathcal{C}$, let $\mathcal{A}_c$ be the set of allowable *states* for character c. We write r to denote $\max_{c\in\mathcal{C}} |\mathcal{A}_c|$. 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 $\boldsymbol{s}$*, or the *state of $\boldsymbol{s}$ on character c*. A *phylogeny* for $\mathcal{S}$ is a tree T such that $\mathcal{S} \subseteq V(T) \subseteq \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$ and where every leaf is in $\mathcal{S}$. T is *perfect* if for each $c \in \mathcal{C}$ and each state r_c of that character, the nodes having state r_c on character c form a subtree of T. The *perfect phylogeny problem* (more commonly known as the *character compatibility problem* in the biology literature [6]) is to determine whether a given set of species $\mathcal{S}$ has a perfect phylogeny. If $\mathcal{S}$ admits a perfect phylogeny, the set of characters $\mathcal{C}$ is said to be *compatible.*

The perfect phylogeny problem was shown to be NP-complete by Bodlaender et al. [4] and, independently, by Steel [18]. This fact suggests two lines of attack: The first is to restrict r, the number of character states. Algorithms for this version of the problem are important because biomolecular data (such as DNA, RNA, or amino-acid sequences) leads to characters whose number of states is bounded by a small constant ($r = 4$ for DNA or RNA, and $r = 20$ for amino-acid sequences). Gusfield [11] gave a $O(nm)$ algorithm

*DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), Rutgers University, Piscataway, NJ 08855 (*agarwala@dimacs.rutgers.edu*). Part of this work was done while this author was at Iowa State University and was supported by the National Science Foundation under grant CCR-9211262.

†Department of Computer Science, Iowa State University, Ames, IA 50011 (*fernande@cs.iastate.edu*). Supported in part by the National Science Foundation under grant CCR-9211262.

for $r = 2$, the *binary* character case. Agarwala, Fernández-Baca, and Slutzki [1] have shown that, under a suitable representation of the input, the run time for the binary character case 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 [12] gave a $O(n^2m)$ algorithm for $r \leq 4$ (*quaternary* characters). Agarwala and Fernández-Baca [2] gave an $O(2^{3r}(nm^3 + m^4))$ algorithm for the perfect phylogeny problem when the characters are restricted to have at most r states.

The line of attack we follow in this paper is to restrict m, the maximum number of characters. Efficient algorithms for small m are useful since, in practice, character sets tend to be incompatible and a standard tool for molecular biologists is checking small subsets of characters for character compatibility. Linear time algorithms have been found for $m = 3$ [5, 13]. Bodlaender, Fellows, and Hallett [3] have shown that the perfect phylogeny problem is hard for the complexity class $W[2]$, implying that it is unlikely to be solved by an algorithm whose running time is of the form $O(f(m)p(r,n,m))$ where f is an exponential function and p is a polynomial function. McMorris, Warnow, and Wimer [17] showed that the perfect phylogeny problem can be solved in $O((rm)^{m+1} + nm^2)$ time. Here we present a simpler, and thus more practical, $O((r - n/m)^m rnm)$ perfect phylogeny algorithm, which is based on the same structural properties of perfect phylogenies that earlier led us to an algorithm that was exponential only in r [2]. The procedure is also asymptotically faster than McMorris et al.'s for certain ranges of the parameter values.

Another reason why the new algorithm is of interest has to do with the close relation between perfect phylogeny and the problem of *triangulating colored graphs* (TCG). A graph is called *triangulated* if it has no induced cycles of length four or more. TCG is defined as follows. Given a graph G with a proper vertex coloring $l : V(G) \to \{1, 2, \cdots, k\}$, find whether G can be triangulated without introducing edges between vertices of the same color. (A coloring is *proper* if $(u, v) \in E(G)$ implies $l(u) \neq l(v)$.) TCG and the perfect phylogeny problem are polynomially equivalent [19]. McMorris et al.'s perfect phylogeny algorithm is in fact based on a $O((v + e(k-2))^{k+1})$ algorithm for TCG [17], where v and e are the number of vertices and edges of the graph. Our algorithm, together with a close examination of the reduction from TCG to perfect phylogeny (which is explained in Section 5) yields an $O((2e/k)^k e^2 k)$ algorithm for TCG.

In the next section, we present some preliminary definitions and results. Chief among the concepts reviewed there is the notion of a *c-cluster*. Additional properties of c-clusters required to find a perfect phylogeny are studied in Section 3. The perfect phylogeny algorithm is given in Section 4. Section 5 deals with TCG.

2 Preliminaries

We now state some definitions and preliminary results.

Lemma 1 [2] *A set of species $\mathcal{S}$ has a perfect phylogeny if and only if every subset of $\mathcal{S}$ has one.*

Definition 1 Let T be a perfect phylogeny for $\mathcal{S}$ and let $\boldsymbol{p}$ be some vertex in T. 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, we must have $p_c = a_c = b_c$.)

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 $\boldsymbol{a} \in G$ 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 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)*.

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 *good* 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 $|G - \{s\}|, |G'| \geq 1$.

Lemma 2 [2] *If $\mathcal{S}$ has no good c-splits, then, in every perfect phylogeny T of $\mathcal{S}$, every species $s \in \mathcal{S}$ is a leaf in T.*

Definition 5 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 $\mathcal{S} - G$.

Definition 6 A *subphylogeny* T_G for a cluster G is a perfect phylogeny for $G \cup \{\boldsymbol{x}\}$, where $\boldsymbol{x}$ is a species such that G is compatible with $\boldsymbol{x}$. Node $\boldsymbol{x}$ 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 [2] *Suppose $\mathcal{S}$ has no good c-splits. 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.*

Corollary 4 [2] *Suppose $\mathcal{S}$ has no good c-splits. $\mathcal{S}$ has a perfect phylogeny if and only if there exist pairwise disjoint c-clusters $G_1, \cdots, G_k$ and a vector $\boldsymbol{x}$ such that (i) $\cup_{i=1}^{k} G_i = \mathcal{S}$ and (ii) each G_i is compatible with $\boldsymbol{x}$ and has a subphylogeny.*

We shall assume that there are no duplicate species (i.e. for all $u, v \in \mathcal{S}, u \neq v$, there exists a character c such that $u_c \neq v_c$) and that at least two states of every character are exhibited by two or more species. The next lemma shows that the latter assumption can be made without loss of generality.

Lemma 5 *Suppose that there exists a character $c \in \mathcal{C}$ such that at most one of its states is exhibited by more than two species in $\mathcal{S}$. Then there exists a perfect phylogeny T for $\mathcal{S}$ on $\mathcal{C}$ if and only if there exists a perfect phylogeny T' for $\mathcal{S}$ on $\mathcal{C} - \{c\}$.*

Proof Assume without loss of generality that $c = m$ and let r_m be the only state of character c exhibited by more than one species. Deriving T' from T is easy, as we only have to suppress character c in each species in T. We can derive T from T' as follows. First, replace every node $(u_1, \cdots, u_{m-1}) \in V(T')$ by $(u_1, \cdots, u_{m-1}, r_m)$. Now the tree will contain nodes for every species in $\mathcal{S}$, except for those that do not have state r_m on character m. For each such species $\boldsymbol{s}$, there must be a species $\boldsymbol{s}'$ in the tree that is identical to $\boldsymbol{s}$, except for its state on character m. Thus, we add to T' a node for each such $\boldsymbol{s}$, along with an edge from $\boldsymbol{s}$ to the corresponding $\boldsymbol{s}'$. The final result will be a phylogeny T for $\mathcal{S}$ on $\mathcal{C}$. □

Lemma 6 *If $n > (r-1)m+1$, then $\mathcal{S}$ does not have a perfect phylogeny.*

Proof By definition, in a perfect phylogeny nodes having the same character state on any character form a subtree. Thus, the maximum number of edges in any perfect phylogeny with no duplicate nodes is at most $(r-1)m$. The maximum number of distinct nodes in any phylogeny is therefore $(r-1)m+1$. □

3 Finding c-clusters

In combination with some additional results, Lemma 3 and its corollary pave the way for a perfect phylogeny algorithm that is exponential only in r. A key observation is that the number of c-clusters is $O(2^r m)$ [2]. We shall use Lemma 3 in a different way to obtain an algorithm that is exponential only in m. One of the ideas is the straightforward observation that the number of distinct choices for nodes in a phylogeny is $O(r^m)$. The other ingredient is the fact that the choice of an internal node in some sense fixes the kinds of subtrees that node may have in a phylogeny. To formalize this, we need a new notion. Given a species $\boldsymbol{x}$, Kannan and Warnow [14] defined an equivalence relation $E_{\boldsymbol{x}}$ as the transitive closure of the following relation $\mathcal{R}$ on $\mathcal{S} - \{x\}$:

$$(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R} \text{ if there exists } c \in \mathcal{C} \text{ such that } a_c = b_c \neq x_c.$$

It is clear from this definition that two species in $\mathcal{S}$ which are in the same equivalence class must be in the same component of $T - \{\boldsymbol{x}\}$, for any perfect phylogeny T of $\mathcal{S} \cup \{\boldsymbol{x}\}$. The set of equivalence classes is denoted by $(\mathcal{S} - \{\boldsymbol{x}\})/\boldsymbol{x}$ and can be computed in $O(nm)$ time [14]. Note that if $\boldsymbol{x}$ is an internal node in any perfect phylogeny of $\mathcal{S}$, then $|(\mathcal{S} - \{\boldsymbol{x}\})/\boldsymbol{x}| \geq 2$. In particular, we make the following remark.

Lemma 7 *If $|(\mathcal{S} - \{\boldsymbol{x}\})/\boldsymbol{x}| = 1$ for every $\boldsymbol{x} \in \mathcal{S}$, then $\mathcal{S}$ has no good splits.*

We now reformulate and extend a result in [14].

Lemma 8 *Let $G \in (\mathcal{S} - \{\boldsymbol{x}\})/\boldsymbol{x}$. If $\mathcal{S} \cup \{\boldsymbol{x}\}$ has a perfect phylogeny, then G is a c-cluster that is compatible with $\boldsymbol{x}$.*

Proof From the definition of $E_{\boldsymbol{x}}$, it follows that G is a cluster compatible with $\boldsymbol{x}$. We now show that G is a c-cluster. The claim is trivially true if $G = \mathcal{S}$ as $\mathcal{D}(G) = \mathcal{C}$. Otherwise, it suffices to show that there exists a perfect phylogeny with no duplicate nodes for $\mathcal{S} \cup \{\boldsymbol{x}\}$ such that the species in each component of $T - \{\boldsymbol{x}\}$ give us the equivalence classes of $(\mathcal{S} - \{\boldsymbol{x}\})/\boldsymbol{x}$.

Since $\mathcal{S} \cup \boldsymbol{x}$ has a perfect phylogeny, then, by Lemma 1, so does $G \cup \{\boldsymbol{x}\}$. Specifically, let T_G be a perfect phylogeny with no duplicate nodes for G. To prove that G is a c-cluster, it is enough to show that $\boldsymbol{x}$ is a leaf in T_G, since this will establish that every species in G differs from species in $\mathcal{S} - G$ in at least one character. Suppose $\boldsymbol{x}$ is not a leaf. Let $H_1, \cdots, H_l$ be the species in components of $T_G - \{\boldsymbol{x}\}$. Since G is an equivalence class of $(\mathcal{S} - \{\boldsymbol{x}\})/\boldsymbol{x}$, there exist species $\boldsymbol{p} \in H_j, \boldsymbol{q} \in H_{j'}$ for some $j \neq j'$, and a character $c \in \mathcal{C}$ such that $p_c = q_c \neq x_c$. But this implies that T_G is not a perfect phylogeny, which is a contradiction. □

The previous lemma gives a way of finding the c-clusters that are compatible with a vector $\boldsymbol{x}$. We can assemble c-clusters into a perfect phylogeny for $\mathcal{S}$ by using Corollary 4 and the next result.

Lemma 9 *Suppose that $\mathcal{S}$ has no good c-splits and that $\mathcal{S} \cup \{\boldsymbol{x}\}$ has a perfect phylogeny for some $\boldsymbol{x} \notin \mathcal{S}$. Then $\boldsymbol{x}$ and $\mathcal{S}/\boldsymbol{x}$ are, respectively, a species and a set of c-clusters satisfying the conditions of Corollary 4.*

Proof From Lemma 8 and the fact that $E_{\boldsymbol{x}}$ is an equivalence relation, the elements of $\mathcal{S}/\boldsymbol{x}$ are pairwise disjoint c-clusters compatible with $\boldsymbol{x}$ whose union equals $\mathcal{S}$. Let $G \in \mathcal{S}/\boldsymbol{x}$. By Lemma 1 $G \cup \{\boldsymbol{x}\}$ has a perfect phylogeny T_G, because by assumption $\mathcal{S} \cup \{\boldsymbol{x}\}$ has one. Since G is compatible with $\boldsymbol{x}$, T_G is a subphylogeny for G. □

4 The Perfect Phylogeny Algorithm

In describing the perfect phylogeny algorithm, we will assume that $n \leq (r-1)m + 1$. Note that if this does not hold, we can use Lemma 6 to immediately conclude that $\mathcal{S}$ does not have a perfect phylogeny. Furthermore, we will assume that there are no good splits. To see that no generality is lost because of this, observe that, by Lemma 7, the presence of good splits implies that there exists a species $\boldsymbol{x} \in \mathcal{S}$ such that $|(\mathcal{S} - \{\boldsymbol{x}\})/\boldsymbol{x}| \geq 2$. For each $G \in (\mathcal{S} - \{\boldsymbol{x}\})/\boldsymbol{x}$, we can recursively search for a phylogeny for the set $G \cup \{\boldsymbol{x}\}$. By Lemma 1, if $\mathcal{S}$ has a perfect phylogeny then each of these subproblems must have one. Moreover, the phylogenies for the subproblems (assuming they all have one), can be combined into a phylogeny for the entire set by identifying the nodes for $\boldsymbol{x}$ in the various trees. (More details are given in [2].) Thus, any problem with good splits can be broken down into smaller subproblems. By doing this repeatedly, we will be left with subproblems without good splits. The time needed to do the reduction can be shown to be dominated by the rest of the work done by the algorithm (see [2]).

The search graph. Our algorithm associates with $\mathcal{S}$ a directed search graph W. Let us refer to any species $\boldsymbol{x} \notin \mathcal{S}$ such that $|\mathcal{S}/\boldsymbol{x}| \geq 2$ as a *useful* species. (Note that by Lemma 2 the absence of good splits implies that in any perfect phylogeny, every internal

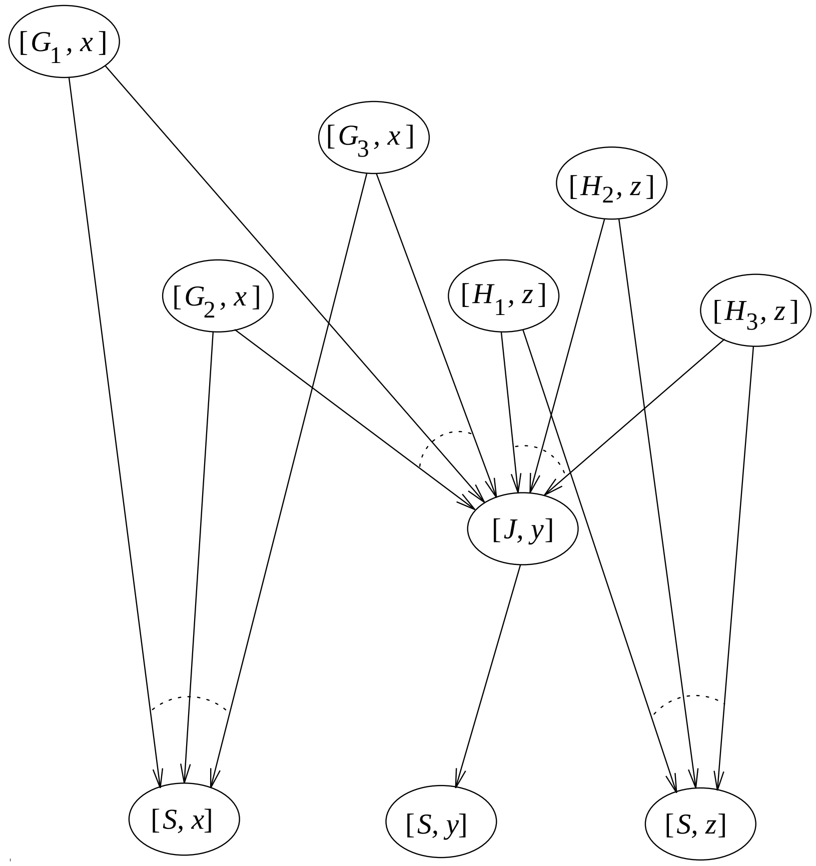


Figure 1: A portion of the search graph associated with $\mathcal{S}$.

node must be useful.) $V(W)$ consists of all nodes $[\mathcal{S}, \boldsymbol{x}]$ such that $\boldsymbol{x}$ is a useful species and all nodes $[G, \boldsymbol{x}]$ such that $\boldsymbol{x}$ is a useful species and $G \in \mathcal{S}/\boldsymbol{x}$. $E(W)$ consists of all edges $([G, \boldsymbol{x}], [\mathcal{S}, \boldsymbol{x}])$ such that $[G, \boldsymbol{x}] \in V(W)$ together with all edges $([G_1, \boldsymbol{x}_1], [G_2, \boldsymbol{x}_2])$ such that $G_1 \subseteq G_2$, and $\boldsymbol{x}_1$ differs from $\boldsymbol{x}_2$ in exactly one character state. A portion of such a search graph is shown in Figure 1. The total number of nodes of type (ii) in W is $O(r^m n)$, since there are $\prod_{i=1}^m r_i$ choices for $\boldsymbol{x}$ and $\mathcal{S}/\boldsymbol{x}$ can have at most n classes; all such nodes can be generated in $O(r^m nm)$ time. Thus, W has $O(r^m n)$ vertices. There are $O(r^{m+1} nm)$ edges because the outdegree of any node is $O(rm)$.

Edges into a node $w \in V(W)$ are grouped into *bundles*, each of which consists of all edges whose starting nodes have the same second component. Bundles of edges are indicated by dotted lines on Figure 1. Intuitively, W considers all possible candidates for internal nodes and the ways in which each candidate divides $\mathcal{S}$ into equivalence classes. Different bundles of edges into a node $[G, \boldsymbol{x}]$ give different ways to express G as the union of c-clusters. (For example, Figure 1 shows that J can be expressed as the union of G_1, G_2, and G_3 or as the union of H_1, H_2, and H_3.) Observe that, by Lemma 8, $\boldsymbol{x}$ is compatible with G and hence can serve as a connection for a subphylogeny of G, assuming one exists.

The algorithm. Algorithm PHYLOGENY employs W to try to find pairwise disjoint c-clusters $G_1, \cdots, G_k$ and a vector $\boldsymbol{x}$ satisfying the conditions of Corollary 4. By Lemma 9, determining the right partition involves identifying a suitable internal node $\boldsymbol{x}$. PHY-

LOGENY also searches for a way to express each G_i as the union of disjoint c-clusters satisfying the conditions of Lemma 3. The algorithm achieves this goal by starting with singleton sets of species, building up to a set containing all of $\mathcal{S}$. For this purpose, PHYLOGENY treats the search graph as a circuit where truth values are propagated from node to node along edges. As soon as a node gets a truth value, that value is placed on all of its outgoing edges. In the description below, a node whose incoming edges have a truth value but which has not itself been assigned a truth value shall be called *active*.

PHYLOGENY($\mathcal{S}$)

```
Generate the search graph W
Assign the value true to every [G, x] ∈ V(W) such that |G| = 1
while there is an active node
    do Choose any active node w
        if there is some bundle B of edges directed into w
            all of whose members are true and ∪{H : ([H, y], w) ∈ B} = G
            then Make w true
            else Make w false
if there exists a node [S, x] which is true
    then return success
    else return failure
```

Theorem 10 PHYLOGENY *runs in $O(r^{m+1}nm)$ time (which is linear in the size of W) and returns* success *if and only if $\mathcal{S}$ has a perfect phylogeny.*

The proof of Theorem 10 will require one additional result.

Lemma 11 *If there exists a perfect phylogeny for $\mathcal{S} \cup \{\boldsymbol{x}\}$ then* PHYLOGENY *assigns* true *to $[G, \boldsymbol{x}]$ for each $G \in \mathcal{S}/\boldsymbol{x}$.*

Proof The proof is by induction on $|G|$. The claim is trivially true for nodes $[G', \boldsymbol{x}']$ where $|G'| = 1$. Suppose the claim holds for all nodes $[G', \boldsymbol{x}']$ where $|G'| < k$. Consider a node $[G, \boldsymbol{x}]$ where $|G| = k$. Let T be a perfect phylogeny for $G \cup \{\boldsymbol{x}\}$. By Lemma 8, G is a c-cluster; moreover, by the proof of the same lemma, $\boldsymbol{x}$ must be a leaf in T. Let $\boldsymbol{y}$ be the first node on the path starting at $\boldsymbol{x}$ in T with degree at least 3. There exists at least one such node as there are no good splits and $|G| \geq 2$. Obviously, there exists a phylogeny for $\mathcal{S} \cup \boldsymbol{y}$ and there must exist pairwise disjoint clusters in $\mathcal{S}/\boldsymbol{y}$ whose union gives G, such that each c-cluster has a subphylogeny. By induction hypothesis, the corresponding nodes of W will be assigned true. Thus, $[G, \boldsymbol{y}]$ will be assigned the value true, which will be propagated to every node $[G, \boldsymbol{z}]$ such that $\boldsymbol{z}$ is on the path between $\boldsymbol{y}$ and $\boldsymbol{x}$ in T. Hence, $[G, \boldsymbol{x}]$ will become true. □

Proof of Theorem 10: From Lemmas 9 and 11, it follows that if $\mathcal{S}$ has a perfect phylogeny, then PHYLOGENY will assign the value true to some node $[\mathcal{S}, \boldsymbol{x}] \in V(W)$ and hence the algorithm returns success. The reader may also verify that, unless $\mathcal{S}$ has a perfect phylogeny, at termination no node $[\mathcal{S}, \boldsymbol{x}] \in V(W)$ will be true, and the algorithm will return failure. Therefore, PHYLOGENY is correct.

The search graph can be constructed in time $O(r^{m+1}nm)$ as follows. Create a table of size $O(r^m)$, with one entry per species not in $\mathcal{S}$. Every entry will have pointers to all nodes that differ in exactly one character. This can be done in $O(r^m(rm))$ time. For each entry $\boldsymbol{x}$, we compute the equivalence classes of $\mathcal{S}/\boldsymbol{x}$ and record them using a vector of length n whose ith entry points to the class containing species i.

To find the neighbors of node $[G, x]$ in W, we use the table to find a $\boldsymbol{y}$ such that $\boldsymbol{x}$ and $\boldsymbol{y}$ differ in 1 character state. Use the vector associated with $\boldsymbol{y}$ to find the equivalence class containing each $\boldsymbol{s} \in G$ with respect to $\boldsymbol{y}$. Hence, putting edges between any $\boldsymbol{x}$ and $\boldsymbol{y}$ pair takes time $O(n)$, for a total of $O(r^m rnm)$ time.

All the remaining steps of PHYLOGENY can be done in time linear in size of the search graph. Hence, the total running time is $O(r^{m+1}nm)$. □

Constructing the perfect phylogeny. The description of PHYLOGENY omits any mention of how to construct a phylogeny once it has been determined that one exists. We now explain how to do so. If the algorithm returns `success`, traverse W, constructing an in-tree D as follows. Choose any node $[\mathcal{S}, \boldsymbol{a}]$ having value `true` as the root of D. Next, do the following until every leaf $[G, \boldsymbol{x}]$ in D has $|G| = 1$: Pick any leaf $w = [G, \boldsymbol{x}]$ in D such that $|G| \geq 2$ and add to D any bundle B of edges into w whose elements all have the value `true`.

After D has been constructed, we can build a perfect phylogeny T by disregarding the orientation of the edges of D and replacing each node $[G, \boldsymbol{x}]$ of D with a node labeled $\boldsymbol{x}$. Observe that the phylogeny produced by the algorithm is one where adjacent nodes differ in exactly one character state.

An improvement. We shall say that a character c *distinguishes* a species $\boldsymbol{s} \in \mathcal{S}$ if $c \in \mathcal{D}(\{\boldsymbol{s}\})$ (i.e., s_c is unique to species $\boldsymbol{s}$). The following lemma gives a way of reducing the number of internal nodes that need to be considered for building a perfect phylogeny.

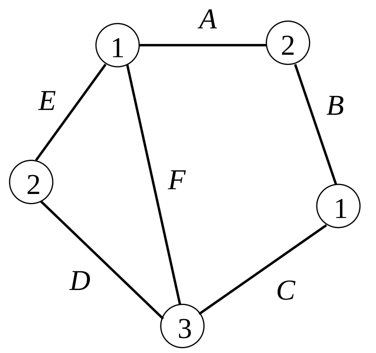
Lemma 12 *For every $c \in \mathcal{C}$, let k_c be the number of species in $\mathcal{S}$ that are distinguished by c. Then, the number of species not in $\mathcal{S}$ that can be internal nodes in a phylogeny for $\mathcal{S}$ is $O((r - \sum_{i=1}^{m} k_i/m)^m)$.*

Proof We shall prove that the number of possible internal nodes is $O(\prod_{i=1}^{m}(r - k_i))$. The lemma follows, since $\prod_{i=1}^{m}(r - k_i) \leq (r - \sum_{i=1}^{m} k_i/m)^m$.

Consider any character c and let $\boldsymbol{s} \in \mathcal{S}$ be such that $c \in \mathcal{D}(\{s\})$. We claim that if $\mathcal{S}$ has a perfect phylogeny, then $\mathcal{S}$ also has a perfect phylogeny where no node except $\boldsymbol{s}$ has state s_c for character c. The reason is that state s_c is never forced for any node $\boldsymbol{x} \neq \boldsymbol{s}$, so a different assignment is possible for x_c if at least one state of character c is exhibited by more than one species.

The claim implies that the number of choices for state c for any internal node in a phylogeny is $r - k_c$. Therefore, the number of candidate internal nodes is $O(\prod_{i=1}^{m}(r - k_i))$. □

In constructing W, we only need to consider c-clusters derived from the $O((r - \sum_{i=1}^{m} k_i/m)^m)$ species mentioned in Lemma 12. Furthermore, if $\mathcal{S}$ has no good splits, then for every species $\boldsymbol{s} \in \mathcal{S}$, $\mathcal{D}(\{\boldsymbol{s}\}) \geq 1$ and, thus, $\sum_{i=1}^{m} k_i \geq n$. Hence, the number of nodes in the search graph is reduced to $O((r - n/m)^m n)$. Since PHYLOGENY takes time



	1	2	3
A	1	1	*
B	2	1	*
C	2	*	1
D	*	2	1
E	1	2	*
F	1	*	1

Figure 2: Reducing TCG to perfect phylogeny. Numbers inside nodes are colors.

linear in the size of W, and the outdegree of any node of W is $O(rm)$, the running time is $O((r - n/m)^m rnm)$.

5 Triangulating Colored Graphs

We now turn our attention to TCG. We shall assume the following about the input graph G.

1. There are at least two nodes in G having any given color. There is no loss of generality in assuming this: If a node $u \in V(G)$ is such that no other node has the same color, then we can add edges $\{(u, v) : v \in V(G), v \neq u\}$. After introducing these edges, there can be no cycle of length four or more containing u in G (and in any triangulation of G).

2. Every node in G has degree at least two. Again, no generality is lost: If a node u has degree one, then it is not in any cycle in G. Hence no edge having u as an endpoint needs to be added to triangulate G.

Any instance $\langle G, l\rangle$ of TCG (where G is the graph and l the coloring function) can be reduced to an instance of perfect phylogeny with $\mathcal{C} = \{1, \cdots, k\}$ as follows [19]. Define a function $f : V(G) \to \mathbf{Z}$ such that $f(u) \neq f(v)$ whenever $l(u) = l(v)$. For each edge $(u, v) \in E(G)$, introduce a species $\boldsymbol{s}$ with $s_{l(u)} = f(u)$ and $s_{l(v)} = f(v)$. For each character $c \in \mathcal{C} - \{l(u), l(v)\}$ introduce a new state (unique to $\boldsymbol{s}$) for c and assign it to s_c. An example of this reduction is shown in Figure 2. The instance of perfect phylogeny is represented by a character matrix whose rows are indexed by species and whose columns are indexed by characters. Entries of the matrix are character states; a star represents a character state that is unique to a species. Let $v = |V(G)|$ and $e = |E(G)|$. Then, $|\mathcal{S}| = e$, $|\mathcal{C}| = k$, and $r \leq e - 2$. Hence, by Theorem 10, we have an $O(e^{k+2}k)$ procedure for triangulating colored graphs. Note, however, that the reduction from TCG to perfect phylogeny introduces many states that are unique to a species; specifically, for each species $\boldsymbol{s}$ there are $k-2$ states that are unique to $\boldsymbol{s}$. Hence (using the notation of the Lemma 12), $\sum_{i=1}^{m} k_i \geq e(k-2)$. Lemma 12 then implies that the running time of the algorithm can be reduced to $O((e - e(k-2)/k)^k(e^2k)) = O((2e/k)^k(e^2k))$.

References

[1] R. Agarwala, D. Fernández-Baca, and G. Slutzki, *Fast algorithms for inferring evolutionary trees*, Submitted to Journal of Computational Biology. This is a corrected version of the paper that appeared in Proceedings of the 30th Allerton Conference on Comm., Control, and Comput. (1992), pp. 594–603.

[2] R. Agarwala and D. Fernández-Baca, *A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed*, In Proceedings of the 34th Annual Symposium on Foundations of Computer Science (1993), pp. 140–147. To appear in *SIAM J. on Computing.*

[3] H. Bodlaender, M. Fellows, and M. T. Hallett, *The parameterized complexity of DNA mapping, perfect phylogeny and other problems of "bounded width"*, manuscript.

[4] H. Bodlaender, M. Fellows, and T. Warnow, *Two strikes against perfect phylogeny*, In Proceedings of the 19th International Colloquium on Automata, Languages, and Programming, Springer Verlag, Lecture Notes in Computer Science (1992), pp. 273–283.

[5] 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 (1992), pp. 415–423.

[6] W. H. E. Day and D. Sankoff, *Computational complexity of inferring phylogenies by compatibility*, Syst. Zool., Vol. 35, No. 2 (1986), pp. 224–229.

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

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

[9] G. F. Estabrook, *Cladistic methodology: A discussion of the theoretical basis for the induction of evolutionary history*, Annual Review of Ecology and Systematics, 3 (1972), pp. 427–456.

[10] W. M. Fitch, *Aspects of molecular evolution*, Annual Reviews of Genetics, 7 (1973), pp. 343–380.

[11] D. Gusfield, *Efficient algorithms for inferring evolutionary trees*, Networks, 21 (1991), pp. 19–28.

[12] S. Kannan and T. Warnow, *Inferring evolutionary history from DNA sequences*, to appear, SIAM J. on Computing. A preliminary version of this paper appeared in the Proceedings of the 31st Annual Symposium on the Foundations of Computer Science (1990), St. Louis, Missouri, pp. 362–378.

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

[14] S. Kannan and T. Warnow, *A fast algorithm for the computation and enumeration of perfect phylogenies when the number of character states is fixed*, To appear in Proceedings of the Symposium on Discrete Algorithms (1995).

[15] W. J. Le Quesne, *A method of selection of characters in numerical taxonomy*, Syst. Zool., 18 (1969), pp. 201–205.

[16] W. J. Le Quesne, *Further studies based on the uniquely derived character concept*, Syst. Zool., 21 (1972), pp. 281–288.

[17] F. R. McMorris, T. Warnow, and T. Wimer, *Triangulating vertex colored graphs*, In Proceedings of the 4th Annual Symposium on Discrete Algorithms (1993), Austin, Texas, pp. 120–127.

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

[19] T. J. Warnow, *Combinatorial Algorithms for Constructing Phylogenetic Trees*, Ph.D. Thesis, University of California, Berkeley, May 1991.