Minimizing Phylogenetic Number to find Good Evolutionary Trees ⋆

Leslie Ann Goldberg [a,1,2], Paul W. Goldberg [b,1], Cynthia A. Phillips [c,1], Elizabeth Sweedyk [d], Tandy Warnow [e,3]

[a] *Department of Computer Science, University of Warwick, Coventry CV4 7AL, U.K.*

[b] *Department of Computer Science and Applied Mathematics, Aston University, Birmingham B4 7ET, U.K.*

[c] *Sandia National Laboratories, MS 1110, P.O. Box 5800, Albuquerque, NM 87185, U.S.A.*

[d] *593 Soda Hall, Dept. of Computer Science, UC Berkeley, Berkeley, CA 94720, U.S.A.*

[e] *Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, U.S.A.*

Abstract

Inferring phylogenetic trees is a fundamental problem in computational biology. We present a new objective criterion, the *phylogenetic number*, for evaluating evolutionary trees for species defined by biomolecular sequences or other qualitative characters. The phylogenetic number of a tree T is the maximum number of times that any given character state arises in T. By contrast, the classical *parsimony* criterion measures the total number of times that different character states arise in T. We consider the following related problems: finding the tree with minimum phylogenetic number, and computing the phylogenetic number of a given topology in which only the leaves are labeled by species. When the number of states is bounded (as is the case for biomolecular sequence characters), we can solve the second problem in polynomial time. Given the topology for an evolutionary tree, we can also compute a phylogeny with phylogenetic number 2 (when one exists) for an arbitrary number of states. This algorithm can be used to further distinguish trees that are equal under parsimony. We also consider a number of other related problems.

Preprint submitted to Elsevier Science *2 May 1996*

1 Introduction

The problem of evolutionary tree construction involves taking a given set of species, and constructing a tree which describes the evolutionary history of that set of species. We would expect a pair of species to be close together in the tree if they are closely related. Numerous variants of this general problem have been studied, the variants arising from the differing kinds of information that may be assumed to be available concerning the species.

In *character-based phylogeny*, the scenario is the following. A character c is a function from the species set S to some set R_c of *states*. For example, the character vertebrate-invertebrate has two states, so we can choose $R_c = \{0, 1\}$ and we can define c so that $c(s) = 0$ for every species s that is a vertebrate and $c(s) = 1$ for every species s that is an invertebrate. As another example, we could define a character c based on average life-span. In this case R_c might be a set of ranges such as $R_c = \{$0–10 years, 10–20 years, 20–60 years, more than 60 years$\}$. Then the function c could be defined to map each species s to the range containing its average life-span. We can think of a sequence of k characters $c_1, \ldots, c_k$ as mapping each species s in the species set to a vector $(c_1(s), \ldots, c_k(s))$ in $R_{c_1} \times \cdots \times R_{c_k}$. The species sets that we will consider will have the property that for any two distinct species, s and s', that are in a species set, $(c_1(s), \ldots, c_k(s)) \neq (c_1(s'), \ldots, c_k(s'))$. Thus, we will be able to identify each species s with a vector $(c_1(s), \ldots, c_k(s))$ in $R_{c_1} \times \cdots \times R_{c_k}$. Furthermore, we will think of the set $R_{c_1} \times \cdots \times R_{c_k}$ as being the set containing all *possible* species, including those in S.

The inputs to the phylogeny construction problem are the species set S (we will use n to denote the size of S) and a sequence of characters, $c_1, \ldots, c_k$. We will let r_{c_j} denote $|R_{c_j}|$, and r denote $\max_j r_{c_j}$. A *phylogenetic tree* for the input is a node-labeled tree in which every node of the tree is labeled with a vector in $R_{c_1} \times \cdots \times R_{c_k}$, and each species in S is the label of some node of the tree[4]. Thus, each character c_j can be extended to a function from the set

---

⋆ Full version of a paper presented at the 6th (1995) annual symposium on Combinatorial Pattern Matching

[1] This work was performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract DE-AC04-76AL85000.

[2] Part of this work was supported by the ESPRIT Basic Research Action Programme of the EC under contract 7141 (project ALCOM-IT).

[3] This work supported by a National Science Foundation Young Investigator Award under contract CCR-9457800, and by the U.S. Department of Energy under contract DE-AC04-76AL85000.

[4] A phylogenetic tree for the input $S, c_1, \ldots, c_k$ is sometimes defined to be a node-labeled tree in which every node of the tree is labeled with a vector in $R_{c_1} \times \cdots \times R_{c_k}$, and each species in S is the label of some *leaf* of the tree. It is clear that every

of vertices of T to R_{c_j}.

A species is naturally described using a string of length k over the alphabet $\{1, \ldots, r\}$. A phylogeny is a way of expressing similarity amongst a *set* of strings rather than expressing similarity between pairs of strings. Subsets of strings with strong similarities (as measured by matches in many locations) are located closer to each other in the tree than those that are more disparate. The output tree is the *pattern* of similarity amongst the entire set of input strings.

Classically, the quality of a phylogenetic tree is evaluated using optimization criteria. When the data are believed to be generated under a stochastic model, then the *likelihood* of the tree is calculated, and the tree with the highest likelihood score is considered optimal.

Other popular criteria do not explicitly presume a statistical model for the data. In parsimony, a tree is sought on which the total amount of evolutionary change is minimized. This can be calculated in several ways. One way is to count the total number of changes indicated over all the edges, where the evolutionary change indicated by an edge is quantified by the number of characters which change state over that edge. Another way to calculate this is to sum over all the characters the total number of times the character changes. The tree with the minimum total number of changes is called the maximum parsimony tree.

Although the parsimony criterion is very popular, there are data for which the evolutionary process produces characters which are very unlikely to have very many changes, or else few returns to states which have previously appeared on the tree. An example of such a character is the morphological character *vertebrate-invertebrate*; any proposed tree for this character in which the vertebrates and invertebrates are not clearly separated by a single edge would be rejected. Correspondingly, multi-state characters of this type would have each character state occupying a single connected subset of the tree; such characters are said to be *compatible* or *convex* on the tree. When working with data of this type, the parsimony criterion is inadequate because it does not express the constraint indicated by the characters. Instead, the *compatibility* criterion may be used; in this case, the tree on which the maximum number of characters are compatible is sought.

Thus, parsimony and compatibility each targets a different type of charac-

---

tree satisfying this alternative definition also satisfies our definition above. The alternative definition is equivalent to ours in the sense that we can convert a tree T satisfying our definition into a tree T' satisfying the alternative definition by adding extra leaves. Under all reasonable measures of fitness for phylogenetic trees, T and T' will have the same measure of fitness.

ter data and handle deviations from the assumptions differently. Parsimony targets the case where characters evolve slowly but not necessarily so as to produce compatible characters, and penalizes for each extra character state change without regard to how the extra changes are distributed. Compatibility targets the case where characters are presumed to evolve in such a way as to produce compatible characters, and penalizes for each *character* that is not compatible on the tree. Both criteria are used in practice for different types of datasets. Both criteria, compatibility and parsimony, result in NP-hard optimization problems[3,3]. An ideal tree is one in which all characters are compatible (i.e., all characters are convex on the tree). Such a tree is optimal under parsimony and compatibility criteria and is called a *perfect phylogeny*. The question of whether a perfect phylogeny exists for a given input is NP-Complete[3,3].

In this paper, we propose an alternative optimization criterion for evaluating phylogenetic trees which combine the good aspects of both parsimony and compatibility. Specifically, we allow the characters to be of varying types; thus, some can evolve quickly, and can potentially have many extra character state changes, while others may be compatible on the evolutionary tree, and others can fall between the two extremes. Our model presumes that for each character c and state i, we have a bound $\ell_{c,i}$, the number of times each state i of character c arises in the tree. Given these bounds, we would seek a tree T satisfying the constraints given by the bounds, if possible.

We will say that a phylogenetic tree T for an input consisting of a species set S and a sequence of characters $c_1, \ldots, c_k$ is an *ℓ-phylogeny* if, for every character c_j and every state $i \in R_{c_j}$, the set of vertices $c_j^{-1}(i)$ form at most ℓ connected components in T. (A 1-phylogeny is the same as a perfect phylogeny). The *ℓ-phylogeny problem* is the problem of determining whether an input has an ℓ-phylogeny. The *phylogenetic number* of an input is the minimum ℓ such that the input has an ℓ-phylogeny. The *phylogenetic number problem* is the problem of determining the phylogenetic number of an input.

The ℓ-phylogeny problem and the phylogenetic number problem both have fixed-topology versions which are defined as follows. The input is a species set S, a sequence of characters $c_1, \ldots, c_k$, and a tree T in which internal nodes are unlabeled and each leaf is labeled with a species $s \in S$. Each species $s \in S$ is the label of exactly one leaf of T. A phylogenetic tree for the input is formed by taking T and labeling the internal nodes of T with vectors in $R_{c_1} \times \cdots \times R_{c_k}$. The *fixed-topology ℓ-phylogeny problem* is the problem of determining whether the input has an ℓ-phylogeny. The fixed-topology phylogenetic number problem is defined analogously.

The ℓ-phylogeny problem and the phylogenetic number problem also have restricted versions in which new ancestral species may not be added, as in[3].

The restricted versions are defined as follows. The input is a species set S and a sequence of characters $c_1, \ldots, c_k$. A *restricted phylogenetic tree* for the input is a node-labeled tree in which every node of the tree is labeled with a vector in S, and each species in S is the label of some node of the tree. The *restricted ℓ-phylogeny problem* is the problem of determining whether the input has a restricted ℓ-phylogeny. The restricted phylogenetic number problem is defined analogously.

The ℓ-phylogeny problem can be generalized as follows. Fix positive integers $r, \ell_1, \ldots, \ell_r$. Suppose that $S, c_1, \ldots, c_k$ is a phylogeny input such that $\max_j r_{c_j} \leq r$. An $(\ell_1, \ldots, \ell_r)$-phylogeny for an input is defined to be a phylogenetic tree for the input such that, for each character c_j and each integer $i \leq |R_{c_j}|$, the set of vertices that are mapped to the ith state in R_{c_j} by c_j forms at most ℓ_i connected components in T. The *$(\ell_1, \ldots, \ell_r)$-phylogeny problem* is the problem of determining whether an input has an $(\ell_1, \ldots, \ell_r)$-phylogeny. A generalized version of the restricted ℓ-phylogeny problem is defined analogously.

1.1 Summary of Results and Outline of Paper

The 1-phylogeny problem is also known as the perfect phylogeny problem. It was shown to be NP-hard by Bodlaender, Fellows, Warnow, and (independently) Steel[3,3]. The hardness of 1-phylogeny implies that the phylogenetic number problem is NP-hard. In Section 2 of this paper we show that for any fixed $\ell > 1$ the ℓ-phylogeny problem is also NP-hard.

Having shown that the ℓ-phylogeny problem is NP-hard, we consider in Section 3 the fixed-topology ℓ-phylogeny problem. It is known that the fixed-topology 1-phylogeny problem can be solved in polynomial time[3]. We show that the fixed-topology 2-phylogeny problem can also be solved in polynomial time and that the fixed-topology ℓ-phylogeny problem is NP-hard for fixed $\ell > 2$. (We show that the fixed-topology ℓ-phylogeny problem is NP-hard for fixed $\ell > 2$ even when the input is guaranteed to have an $\ell + 1$-phylogeny and the degree of the topology is restricted to be at most 3.)

In Section 4 we consider the restricted ℓ-phylogeny problem. We show that there is a polynomial-time algorithm for the restricted 1-phylogeny problem, but the restricted ℓ-phylogeny problem is NP-hard for fixed $\ell \geq 2$.

Although the 1-phylogeny problem is NP-hard, it can be solved in polynomial time if the number, n, of species is fixed, or the number, k, of characters is fixed[3,3], or the quantity $r = \max_j r_{c_j}$ is fixed[3,3]. A full analysis of fixed parameter ℓ-phylogeny problems is outside the scope of this paper. However, we observe that all of the phylogeny problems can be solved in polynomial

time (by brute force) if n is fixed. In Section 5 we use interesting combinatorial techniques to show that for $k = 2$ the phylogenetic number problem can be solved in $O(n^2)$ time. The complexity of the ℓ-phylogeny problem remains open for fixed $\ell > 1$ and fixed $k > 2$. The difficulty of fixed-topology phylogeny problems does not change if k is fixed. In Section 6 we show that the fixed-topology phylogenetic number problem can be solved in polynomial time for fixed r. On a related note, we show that if r is fixed, there is a polynomial-delay algorithm for listing fixed-topology ℓ-phylogenies. We also show that for fixed $r \geq 2$ and fixed $\ell \geq 3$ the restricted ℓ-phylogeny problem is NP-hard. (This result follows from a more general result. Namely, we show that the restricted (ℓ_1, ℓ_2)-phylogeny problem is NP-hard for fixed $\ell_1 \geq 2$ and $\ell_2 \geq 2$ as long as one of ℓ_1, ℓ_2 is greater than 2.)

Finally, in section 7 we offer some concluding remarks and present some open problems.

1.2 Preliminary Facts

The following fact is used in some of the proofs and in the restricted 1-phylogeny algorithm.

Fact 1 *If an input S, $c_1, \ldots, c_k$ has an ℓ-phylogeny then it has an ℓ-phylogeny in which:*

(i) Each leaf has a label from S.
(ii) Each species is the label of at most one node.
(iii) Every node whose label is not in S has degree at least 3.
(iv) There are at most $\max(0, n-2)$ *nodes with labels that are not in S.*

Proof. It is easy to see that conditions (i–iii) can be satisfied. (One can convert an ℓ-phylogeny into one that satisfies conditions (i–iii) by removing leaves with labels that are not in S, combining branches of the tree to accomplish condition (ii), and then "splicing out" the appropriate degree 2 nodes to accomplish condition (iii).) To prove that condition (iv) can also be satisfied, suppose that T is an ℓ-phylogeny for the input that satisfies conditions (i–iii) and contains at least one node, w, with a label that is not in S. Let T' be the tree obtained from T by splicing out any nodes of degree 2. (Condition (iii) guarantees that no node with a label outside of S is spliced out in this process.) Consider T' to be rooted at w. We can add one or more new internal nodes to T' to obtain a complete binary tree T'' which is rooted at w and

has the same leaves as T'[5]. Conditions (i) and (ii) imply that T, and therefore T' and T'', have at most n leaves. Since T'' has at most n leaves, it has at most $n-1$ internal nodes. Therefore, T' has at most $n-2$ internal nodes, and T has at most $n-2$ nodes with labels that are not in S. $\Box$

Fact 1 implies that if an input has an ℓ-phylogeny then it has a polynomial-sized ℓ-phylogeny.

2 The Hardness of ℓ-Phylogeny

In this section we show that for any fixed $\ell > 1$, the ℓ-phylogeny problem is NP-hard. Our reduction is from the 1-phylogeny problem, which was shown to be NP-hard in [3,3].

We define the *weight* of an edge (v_1, v_2) in a phylogeny to be the number of characters c_j such that $c_j(v_1) \neq c_j(v_2)$. That is, the weight of (v_1, v_2) is the number of characters on which the species labeling v_1 and v_2 disagree, ie. the hamming distance between their vectors of character values. We define the *weight* of a phylogeny to be the sum of the weights of its edges. We start with the following observation.

Remark 2 *Let $S, c_1, \ldots, c_k$ be any input to the ℓ-phylogeny problem and let r denote $\max_j r_{c_j}$. Any ℓ-phylogeny for this input has weight at most $k(\ell r - 1)$.*

We will use the following lemma (in which species are referred to by strings over their character values).

Lemma 3 *For every integer ℓ there is an input $I_\ell = S, c_1, \ldots, c_{2\ell}$ in which $|S| = 2\ell^3 - 2\ell + 1$ and $R_{c_j} = \{0, \ldots, \ell - 1\}$ for $1 \leq j \leq 2\ell$ such that*

(i) For every state i in the range $0 \leq i < \ell$, the species $i^{2\ell}$ is in S.
(ii) I_ℓ has an ℓ-phylogeny
(iii) In any ℓ-phylogeny for I_ℓ the subgraph induced by all of the nodes with any given label is connected.
(iv) In any ℓ-phylogeny for I_ℓ all of the nodes are labeled by species in S. (That is, no new species are introduced.)

[5] To see how to construct T'', let the "level" of a vertex denote its distance from the root. Start with level 0 of T' and proceed through the levels of the tree in increasing order. Consider each vertex v on each level. If v has children $x_1, \ldots, x_j$ with $j > 2$ remove the edges $(v, x_2), \ldots, (v, x_j)$ and add a new node y which is a child of v and the parent of nodes $x_2, \ldots, x_j$. Note that at least one new internal node is added in the process, as w has at least three children in T'.

(v) In any ℓ-phylogeny for I_ℓ the path between the species $i^{2\ell}$ and $j^{2\ell}$ for $i \neq j$ passes through at least $2\ell - 1$ distinct species.

Example: The Input I_3

The species set S of input I_3 consists of 49 species. The values of the six characters on these species are defined as in figure 1:

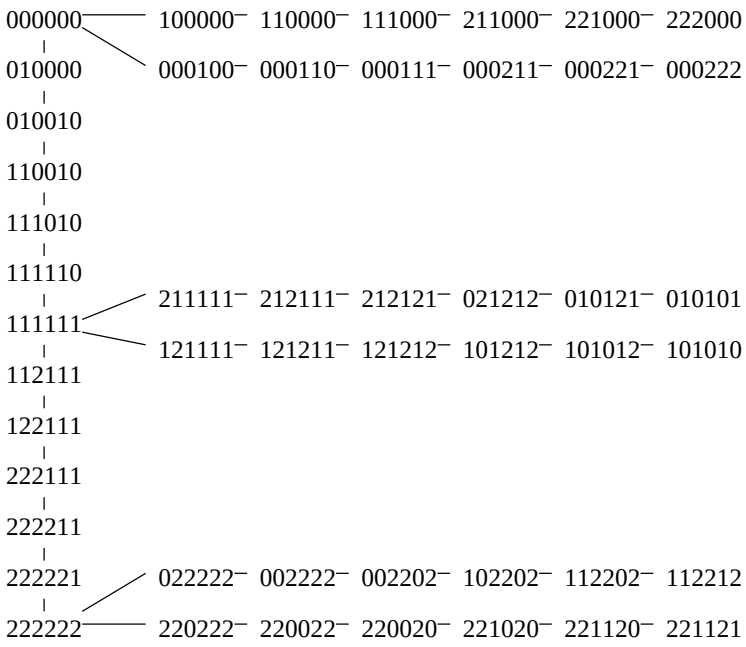

Fig. 1. The Input I_3

The Input I_3 has the 3-phylogeny shown in figure 1. By Remark 2 any 3-phylogeny for I_3 has weight at most 48. However, 48 edges with positive weight are needed just to hook up the 49 species in S into a tree. We conclude that any 3-phylogeny for I_3 consists of 48 edges with weight 1 plus possibly some edges with weight 0. Thus, the subgraph induced by all of the nodes with any given label forms a single connected component. Furthermore, no new species are introduced. Finally, since i^6 and j^6 differ in 6 characters (for $i \neq j$), any path between them in any 3-phylogeny for I_3 passes through at least 5 distinct species.

Construction of $I_\ell = S, c_1, \ldots, c_{2\ell}$:

For $1 \leq j \leq 2\ell$ we set $R_{c_j} = \{0, \ldots, \ell - 1\}$. For each state i in the range $0 \leq i < \ell$ we put the species $i^{2\ell}$ into S. The other species in S will be the species in the following phylogeny:

For each state i in the range $0 \leq i < \ell$ we will choose a unique partition P_i of the 2ℓ characters into two sets of size ℓ. (In the construction of I_3

above we used $P_0 = \{0,1,2\},\{3,4,5\}, P_1 = \{0,2,4\},\{1,3,5\}$, and $P_2 = \{0,1,4\},\{2,3,5\}$.)

We will use each of the parts of the partition P_i to form a "row" of species which will be connected to the species $i^{2\ell}$. To construct each row, consider the ordered list $c_{i_1},\ldots,c_{i_\ell}$ consisting of the characters in the appropriate part of the partition. From the species $i^{2\ell}$ form a new species by changing the state of character c_{i_1} to $(i+1) \bmod \ell$. Then form a new species by changing the state of character c_{i_2} to $(i+1) \bmod \ell$. Continue on until the state of character c_{i_ℓ} is changed to $(i+1) \bmod \ell$. Then change the state of character c_{i_1} to $(i+2) \bmod \ell$ and continue on in this manner until finally the state of character c_{i_ℓ} is changed to $(i+(\ell-1)) \bmod \ell$.

Finally, we will add species to connect the species $i^{2\ell}$ to the species $(i+1)^{2\ell}$ in the vertical *spine* (for i in the range $0 \leq i < \ell - 1$). Let c_λ be the second character in the first part of the partition corresponding to i and construct a new species from $i^{2\ell}$ by changing the state of character c_λ to $i+1$. Next, let c'_λ be the first character such that c_λ and c'_λ are in different parts of i's partition and c_λ and c'_λ are in different parts of $(i+1)$'s partition. Construct a new species by changing the state of character c'_λ to $i+1$. Now, construct $2\ell - 3$ more species by considering each remaining character in turn and changing it from state i to state $i+1$.

Proof of Lemma 3. By construction, S contains the species $i^{2\ell}$ for every state i in the range $0 \leq i < \ell$. To see that the phylogeny constructed above is indeed an ℓ-phylogeny for I_ℓ note that for each state i and for each state $j \neq i$ a character c_λ only has state i in one of the two rows connected to $j^{2\ell}$ and the species with c_λ in state i are connected in this row. Furthermore, there is a single connected component with character c_λ in state i in the rows connected to $i^{2\ell}$ and this connected component contains all species on the vertical spine with character c_λ in state i. We now wish to show that all of the species introduced in the construction are distinct. Suppose that instead two species s_1 and s_2 have identical labels. Note that, by construction, s_1 and s_2 could not be of the form $i^{2\ell}$. Furthermore, they could not be on the same horizontal row and they could not both be on the vertical spine. There are three cases to consider:

(i) s_1 and s_2 are on different rows, both of which are attached to $i^{2\ell}$.
In this case s_1 has state i for all of the characters in one part of the partition P_i and s_2 has state i for all of the characters in the other part of the partition P_i so it must be the case that $s_1 = s_2 = i^{2\ell}$ which is a contradiction.

(ii) s_1 is on a horizontal row connected to $i^{2\ell}$ and s_2 is on a horizontal row connected to $j^{2\ell}$ for some $j \neq i$.

In this case s_1 has state i for all of the characters in some part of the partition P_i so s_2 must have character i for all of the characters in that part of the partition P_i and character j on all other characters. But then the partition P_j is the same as the partition P_i, which is not true by construction.

(iii) s_1 is on the vertical spine between $i^{2\ell}$ and $(i+1)^{2\ell}$ and s_2 is on a horizontal row.

By construction s_2 must be on a row attached to i or on a row attached to $i+1$. However, the choice of c_λ and c'_λ ensures that s_2 cannot be on either of these rows.

Now that we know that the species are distinct, we count them. There are ℓ species of the form $i^{2\ell}$. Each of the 2ℓ horizontal rows has $\ell(\ell-1)$ species. Finally, there are $(\ell-1)(2\ell-1)$ additional species on the vertical spine. We conclude that S has $2\ell^3-2\ell+1$ distinct species. By Remark 2, any ℓ-phylogeny for I_ℓ has weight at most $2\ell(\ell^2-1) = 2\ell^3-2\ell$. However, $2\ell^3-2\ell$ edges with positive weight are needed just to hook up the $2\ell^3-2\ell+1$ species in S into a tree. We conclude that any ℓ-phylogeny for I_ℓ consists of $2\ell^3-2\ell$ edges with weight 1 plus possibly some edges with weight 0. Thus, the subgraph induced by all of the nodes with any given label forms a single connected component. Furthermore, no new species are introduced. Finally, since $i^{2\ell}$ and $j^{2\ell}$ differ in 2ℓ characters, any path between them in any ℓ-phylogeny for I_ℓ passes through at least $2\ell-1$ distinct species. □

We will use Lemma 3 to prove the following theorem

Theorem 4 *For any fixed $\ell > 1$ the ℓ-phylogeny problem is NP-hard.*

Proof. The reduction is from the 1-phylogeny problem. Let $S, c_1, \ldots, c_k$ be an input to the 1-phylogeny problem such that $R_{c_j} \subseteq \{0, \ldots, r-1\}$ for $1 \leq j \leq k$. Let $S', c'_1, \ldots, c'_{2\ell}$ be an input to the ℓ-phylogeny problem satisfying the conditions in Lemma 3. Let $S^* = \{ sr^k \mid s \in S' \}$. For each i in the range $0 \leq i < \ell$ let $S_i = \{ i^{2\ell}y \mid y \in S \}$. Let $S'' = S^* \cup \bigcup_{0 \leq i < \ell} S_i$. Let I be the input to the ℓ-phylogeny problem with species set S'' and characters $c'_1, \ldots, c'_{2\ell}, c_1, \ldots, c_k$. (Note that in input I the range of c_j has been extended from R_{c_j} to $R_{c_j} \cup \{r\}$.)

$\rightarrow$ Suppose that T is a 1-phylogeny for $S, c_1, \ldots, c_k$. For each i in the range $0 \leq i < \ell$ let T_i be a copy of T in which each label y has been changed to $i^{2\ell}y$. (T_i is a 1-phylogeny for $S_i, c'_1, \ldots, c'_{2\ell}, c_1, \ldots, c_k$.) Let T^* be an ℓ-phylogeny for $S^*, c'_1, \ldots, c'_{2\ell}, c_1, \ldots, c_k$. (Part (ii) of Lemma 3 guarantees that T^* exists.) Now for each i in the range $0 \leq i < \ell$ connect an arbitrary node in T_i to the node $i^{2\ell}r^k$ in T^*. (The construction, together with Part (i) of Lemma 3

guarantees that there is a vertex of T^* labeled $i^{2\ell}r^k$.) The resulting tree is an ℓ-phylogeny for I.

$\leftarrow$ Suppose that T is an ℓ-phylogeny for I. If we restrict our attention to characters $c'_1, \ldots, c'_{2\ell}$, we still have an ℓ-phylogeny. Therefore, by Part (iii) of Lemma 3, the subgraph induced by all of the species which have some particular set of states for characters $c'_1, \ldots, c'_{2\ell}$ is connected. We will use the notation T_i to refer to the induced subtree of T containing those species that have state i for characters $c'_1, \ldots, c'_{2\ell}$.

We claim that for any j in the range $1 \leq j \leq k$ any path in T between a node $t_i \in T_i$ and a node $t_h \in T_h$ (for $h \neq i$) contains some species s with $c_j(s) = r$. Clearly, this claim implies that T_0 is a 1-phylogeny for $S_0, c'_1, \ldots, c'_{2\ell}, c_1, \ldots, c_k$. Hence, $S, c_1, \ldots, c_k$ has a 1-phylogeny.

To prove the claim note that by Part (v) of Lemma 3 the path between T_i and T_h passes through at least $2\ell - 1$ nodes $v_1, \ldots, v_{2\ell-1}$, no two of which agree on all of characters $c'_1, \ldots, c'_{2\ell}$. By construction and by Part (i) of Lemma 3, S'' contains the species $i^{2\ell}r^k$ and by Part (iii) of Lemma 3 it is part of T_i. Similarly, S'' contains the species $h^{2\ell}r^k$ and it is part of T_h. Furthermore, (by construction and by Part (iv) of Lemma 3), for each node v_m, S'' contains a species v'_m that agrees with v_m on characters $c'_1, \ldots, c'_{2\ell}$ and has characters $c_1, \ldots, c_k$ in state r. By Part (iii) of Lemma 3 v'_m is in the connected subgraph of T induced by species which agree with v_m on characters $c'_1, \ldots, c'_{2\ell}$. Now suppose that none of $v_1, \ldots, v_{2\ell-1}$ has character c_j in state r. Then the sub-graph of T induced by those nodes that have character c_j in state r has $2\ell + 1$ connected components, which contradicts the fact that T is an ℓ-phylogeny. $\square$

3 The Fixed-Topology ℓ-Phylogeny Problem

It is known that the fixed-topology 1-phylogeny problem can be solved in polynomial time[3]. In Subsection 3.1, we show that the fixed-topology 2-phylogeny problem can also be solved in polynomial time. In Subsection 3.2 we show that the fixed-topology ℓ-phylogeny problem is NP-hard for fixed $\ell > 2$. (We show that the fixed-topology ℓ-phylogeny problem is NP-hard for fixed $\ell > 2$ even when the input is guaranteed to have an $\ell + 1$-phylogeny and the degree of the topology is restricted to be at most 3.)

3.1 The Fixed-Topology 2-Phylogeny Problem

In this subsection, we show that the fixed-topology 2-phylogeny problem can be solved in polynomial time. The algorithm runs in time $O(nrk)$ where n is the number of species, r is the maximum number of states in any character, and k is the number of characters. If a 2-phylogeny exists, then our algorithm computes a labeling that achieves a 2-phylogeny.

Since the topology is fixed, the characters are independent and can be handled one at a time. We will now show how to compute the labels for a single character in time $O(nr)$, where in this case r is the number of states for this character. The overall bound then follows.

Although the input tree is unrooted, for this algorithm, we root this tree from an arbitrary internal node. The choice of root does not affect the existence of a 2-phylogeny, but it may affect the labeling.

Let T be the input tree with leaves labeled by states $1, 2, \ldots, r$. Consider a single state i and let T_i be the subtree of tree T consisting of all the leaves labeled i and the unique set of paths connecting this set of leaves. For state i to have a single connected component in tree T, every node in T_i must be labeled i. For state i to have at most two connected components, every node in tree T_i with degree greater than 2 must be labeled i (otherwise state i would be split into at least 3 components). We call such nodes *branch points* of tree T_i. The branch points and the leaves already labeled i are the *forced points* of tree T_i. At most one path of degree-2 nodes between two forced points can be labeled something other than i.

We begin by computing T_i for $i = 1, \ldots, r$. Each branch point of T_i is labeled as such, each path between two forced points is given a unique label, and each degree-2 node in T_i is labeled with its path label. Note that the root of tree T_i need not be a branch point. If each node of tree T is given a length-r vector, then information for all r trees T_i can be stored in this vector. For example, node v could be a branch point for tree T_i (ith slot of the vector indicates branch point), on the lth path for tree T_j (the jth slot of the vector has the number l), and not in tree T_h (the hth slot is null). We can compute all r trees in time $O(nr)$ using depth-first search.

The first phase of the algorithm (the forced phase) computes all forced labels. For each tree T_i, each branch point of T_i is labeled i and a pointer to the node is placed into a queue. If at any time we try to label a node that is already labeled with something else, then we stop and report that there is no 2-phylogeny for this topology.

Now all path conflicts have to be settled for the labeled nodes. We remove the

first node from the queue. Suppose it is node v and it is labeled i. If this node is also in path l of tree T_j for some $j \neq i$, then tree T_j must give up path l. Once path l is broken, then in order to achieve 2 connected components for state j, every other path in tree T_j must be labeled j. We traverse tree T_j, clearing path l (setting slot j to null for all nodes on path l of tree T_j) and labeling all other nodes j. If we attempt to label a node that is already labeled, then we stop. There can be no 2-phylogeny. Otherwise, the newly-labeled nodes are added to the queue. We do this for all paths that go through node v, then clear path conflicts on all the other nodes in the queue. Because each node can be labeled, enqueued, dequeued, and processed at most once, and each tree can be traversed at most once, this phase can be completed in time $O(nr)$.

The final phase completes the labeling of the tree. If we succeed in emptying the queue without encountering a fatal conflict, it is still possible that some nodes remain unlabeled. We show that there is always a 2-phylogeny. Let trees T_i and T_j be left undetermined by the forced phase of the algorithm. If the intersection of these two trees is empty, there is no conflict between them. Otherwise, the intersection is connected[6] and contains exactly one path from each tree[7]. Furthermore, the root of one of the trees (possibly both) is in the intersection[8]. Suppose that the root of T_i is contained in $T_i \cap T_j$. Then tree T_i gives up the path through its root (if both roots are contained in $T_i \cap T_j$, one of the trees chosen arbitrarily will give up the path through its root). By the structure of the intersection, this clears the conflict between tree T_i and T_j. We can solve all conflicts between pairs of trees in a similar manner. Since each tree was not forced to give up a path in the forced phase of the algorithm (otherwise it would have been fully determined then), it is free to give up one path in this phase. Each tree will give up at most one path, namely the one through its root. Therefore, all conflicts are resolved and we have a 2-phylogeny. This phase of the algorithm can be implemented in $O(nr)$ time by processing each remaining tree in order (determining whether it must relinquish the path through its root, and claiming all other paths).

Thus we have shown how to compute the labelings of the internal nodes of the input tree T in time $O(nr)$ per character for an overall time of $O(nrk)$. Thus, we have proved the following theorem.

Theorem 5 *The fixed-topology 2-phylogeny problem can be solved in polyno-*

[6] If two nodes v_1 and v_2 are both in T_i and both in T_j, then every node on the unique path in T between v_1 and v_2 must also be in both trees.

[7] If the intersection contained pieces of two paths from tree T_i, then it must contain a branch point for tree T_i and therefore tree T_j would have been forced to relinquish a path and left completely determined by the forced phase.

[8] Consider a node in the intersection. If its parent in T is in the intersection, move up to it. Continue until some parent is no longer in the intersection. That node is the root of at least one of T_i and T_j

mial time.

3.2 The Fixed-Topology ℓ-Phylogeny Problem for $\ell > 2$

In this subsection we prove the following theorem.

Theorem 6 *The fixed-topology ℓ-phylogeny problem is NP-hard for fixed $\ell > 2$.*

Proof. The proof is by reduction from 3SAT. Let $\ell > 2$ be fixed. Suppose that we are given an input to 3SAT. We will show how to construct a one-character input S, c, T to the fixed-topology ℓ-phylogeny problem such that the phylogeny input has an ℓ-phylogeny if and only if the input to 3SAT is satisfiable.

The species set S, the set R_c of states, and the character c are constructed as follows. For each of the n variables, x, in the satisfiability input we have states s_x and $s_{\overline{x}}$ and species $s_{(x,1)}, \ldots, s_{(x,\ell+1)}$ and $s_{(\overline{x},1)}, \ldots, s_{(\overline{x},\ell+1)}$ where $c(s_{(x,j)}) = s_x$ and $c(s_{(\overline{x},j)}) = s_{\overline{x}}$. For each of the m clauses, C, in the satisfiability input we have state s_C and species $s_{(C,1)}, \ldots, s_{(C,\ell+1)}$ where $c(s_{(C,j)}) = s_C$. For the ith occurrence of the literal x in the satisfiability input, we have state s_{x_i} and species $s_{(x_i,1)}, \ldots, s_{(x_i,\ell+1)}$ where $c(s_{(x_i,j)}) = s_{x_i}$. Similarly, for the ith occurance of the literal $\overline{x}$ in the satisfiability input, we have state $s_{\overline{x}_i}$ and species $s_{(\overline{x}_i,1)}, \ldots, s_{(\overline{x}_i,\ell+1)}$ where $c(s_{(\overline{x}_i,j)}) = s_{\overline{x}_i}$. Let N denote $n(2\ell-3) + m(4\ell-11)$. For each h in the range $1 \leq h < N$ we have a state s'_h and species $s'_{(h,1)}, \ldots, s'_{(h,\ell+1)}$ where $c(s'_{(h,j)}) = s'_h$.

We will show how to construct a tree T in which internal nodes are unlabeled and each leaf is labeled with a species in S. Each species in S will be the label of exactly one leaf of T. To construct T we will first construct trees $T_1, \ldots, T_N$. Finally, we will hook T_i to T_{i+1} for $1 \leq i < N$

We start by showing how to hook tree T_i to tree T_{i+1}. Let t_i be an internal node in T_i of degree at most 2 and let t_{i+1} be an internal node in T_{i+1} of degree at most 2 (it will be clear from the construction that such small-degree internal nodes exist in T_i and T_{i+1}). Connect t_i and t_{i+1} with a chain of $\ell + 1$ new internal nodes. Finally, give each of the internal nodes in the chain a leaf and label the new leaves with the species $s'_{(i,1)}, \ldots, s'_{(i,\ell+1)}$. For example, if $\ell = 3$ then connect t_i and t_{i+1} as in figure 2:

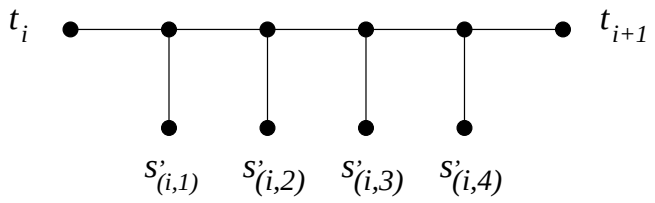

Fig. 2. Example for $\ell = 3$

Note that in any ℓ-phylogeny for the input, at least one of the internal nodes in the chain will be labeled with a species s such that $c(s) = s'_i$. Since we have now used all $\ell + 1$ species s with $c(s) = s'_i$, neither T_i nor T_{i+1} contains a leaf s such that $c(s) = s'_i$. Therefore when T_i is hooked to T_{i+1} as above, any leaves $\ell_i \in T_i$ and $\ell_{i+1} \in T_{i+1}$ with $c(\ell_i) = c(\ell_{i+1})$ are in different connected components in the subgraph induced by $c^{-1}(c(\ell_i))$.

We next show how to construct the trees $T_1, \ldots, T_N$. Trees $T_1, \ldots, T_{N-n-m}$ will each consist of a single internal node connected to a single leaf. In particular, we will construct one such tree for each of the following species: for each variable x, species $s_{(x,1)}, \ldots, s_{(x,\ell-2)}$ and $s_{(\overline{x},1)}, \ldots, s_{(\overline{x},\ell-2)}$; for each clause C, species $s_{(C,1)}, \ldots, s_{(C,\ell-3)}$; for the ith occurance of the literal x, species $s_{(x_i,1)}, \ldots, s_{(x_i,\ell-3)}$; for the ith occurance of the literal $\overline{x}$, species $s_{(\overline{x}_i,1)}$, $\ldots, s_{(\overline{x}_i,\ell-3)}$.

Trees $T_{N-n-m+1}, \ldots, T_{N-m}$ will be used for truth-setting. For each variable x in the satisfiability input we will construct a tree as follows. Suppose that the literal x appears i times in the satisfiability input and that the literal $\overline{x}$ appears j times in the satisfiability input. Construct a tree consisting of a chain of $2i + 2j + 6$ internal nodes. Each internal node will have one leaf, and the species at the leaves will be (in order): first, $s_{(x,\ell-1)}$; then, $s_{(x_1,\ell-2)}$, $s_{(x_1,\ell-1)}$, $s_{(x_2,\ell-2)}$, $s_{(x_2,\ell-1)}, \ldots, s_{(x_i,\ell-2)}$, $s_{(x_i,\ell-1)}$; then $s_{(x,\ell)}, s_{(\overline{x},\ell-1)}$, $s_{(x,\ell+1)}$, $s_{(\overline{x},\ell)}$; then $s_{(\overline{x}_1,\ell-2)}$, $s_{(\overline{x}_1,\ell-1)}, \ldots, s_{(\overline{x}_j,\ell-2)}$, $s_{(\overline{x}_j,\ell-1)}$; finally, $s_{(\overline{x},\ell+1)}$. For example, if $\ell = 3$, $i = 1$, and $j = 2$ construct a tree as in figure 3:

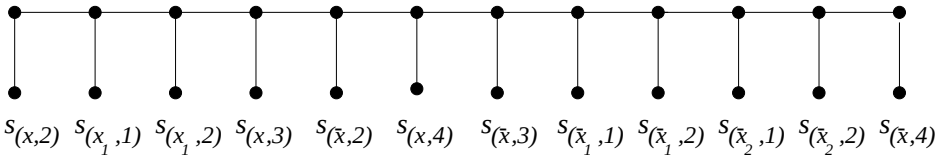

Fig. 3. Example for $\ell = 3$, $i = 1$, $j = 2$

Because we have already introduced single-leaf trees for the species $s_{(x,1)}, \ldots$, $s_{(x,\ell-2)}$ and $s_{(\overline{x},1)}, \ldots, s_{(\overline{x},\ell-2)}$, we observe that in any ℓ-phylogeny, the truth-setting tree for variable x must have at most 2 connected components for each of the states s_x and $s_{\overline{x}}$. We will say that an ℓ-phylogeny sets the satisfiability variable x to "true" if and only if the leaves $s_{(x,\ell)}$ and $s_{(x,\ell+1)}$ are in the same connected component for state s_x. If the variable x is set to "true" then the leaf $s_{(x,\ell-1)}$ can be in a different connected component for state s_x.

Therefore, for $1 \leq h \leq i$, state s_{x_h} can form a single connected component in the truth-setting tree for x. Otherwise, state s_{x_h} must have two connected components in the truth-setting tree for x. Similarly, if x is set to "false" then leaves $s_{(\overline{x},\ell-1)}$ and $s_{(\overline{x},\ell)}$ can be in the same connected component for state $s_{\overline{x}}$ and leaf $s_{(\overline{x},\ell+1)}$ can be in a different connected component. Therefore, for $1 \leq h \leq j$, state $s_{\overline{x}_h}$ can form a single connected component in the truth-setting tree for x. Otherwise, state $s_{\overline{x}_h}$ must have two connected components in the truth-setting tree for x.

Trees $T_{N-m+1}, \ldots, T_N$ will be used for clause-checking. For each clause $C = x_i \vee \overline{y}_j \vee z_k$ in the satisfiability input we will construct a tree consisting of a chain of 10 internal nodes. Each internal node will have one leaf, and the species at the leaves will be (in order): $s_{(C,\ell-2)}, s_{(x_i,\ell)}, s_{(x_i,\ell+1)}, s_{(C,\ell-1)}, s_{(\overline{y}_j,\ell)}, s_{(\overline{y}_j,\ell+1)}, s_{(C,\ell)}, s_{(z_k,\ell)}, s_{(z_k,\ell+1)}, s_{(C,\ell+1)}$. For example, if $\ell = 3$, construct a tree as in figure 4:

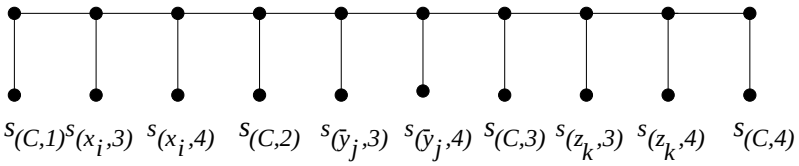

Fig. 4. Example for $\ell = 3$

Because we have already introduced single-leaf trees for the species $s_{(C,1)}, \ldots, s_{(C,\ell-3)}$, we observe that in any ℓ-phylogeny, the clause-checking component for clause C must have at most 3 connected components for the state s_C. This is possible if one of the literals in the clause has been set to "true" by the truth checking component and not otherwise. The correctness of the reduction follows. □

The input to the fixed-topology ℓ-phylogeny problem that is constructed in the proof of Theorem 6 had two notable features. First, (because there are only $\ell + 1$ species with each state), the input is guaranteed to have an $\ell + 1$-phylogeny. Second, the degree of the tree T is at most 3. Therefore, the fixed-topology ℓ-phylogeny problem is NP-hard for fixed $\ell > 2$ even when the input is guaranteed to have an $\ell + 1$-phylogeny and the degree of the topology is restricted to be at most 3.

4 The Restricted ℓ-Phylogeny Problem

In this section we show that there is a polynomial-time algorithm for the restricted 1-phylogeny problem. We then show that the restricted ℓ-phylogeny problem is NP-hard for fixed $\ell \geq 2$.

We start by describing the algorithm for solving the restricted 1-phylogeny problem. Suppose that $S, c_1, \ldots, c_k$ is an input to the restricted 1-phylogeny problem. If the input has a restricted 1-phylogeny, it has one in which each species in S is the label of exactly one node (if not, combine branches).

We define the *weight* of an edge (v_1, v_2) in a phylogeny to be the number of characters c_j such that $c_j(v_1) \neq c_j(v_2)$. That is, the weight of (v_1, v_2) is the number of characters on which the species labeling v_1 and v_2 disagree. We define the *weight* of a phylogeny to be the sum of the weights of its edges.

Let G denote the complete graph with vertex set S. We seek a spanning tree T of G in which, for every character c_j and every state $i \in R_{c_j}$, the set of vertices $c_j^{-1}(i)$ form a connected component in T. Let the *weight* of an edge (s, s') in G be the number of characters c_j such that $c_j(s) \neq c_j(s')$. Then a spanning tree of G is a 1-phylogeny for the input if and only if its weight is $\sum_{j=1}^{k}(r_{c_j} - 1)$, and any spanning tree that is not a 1-phylogeny will have a greater weight. Therefore, the restricted 1-phylogeny problem reduces to the minimum weight spanning tree problem, which can be solved in polynomial time[3]. We have proved the following theorem

Theorem 7 *The restricted 1-phylogeny problem can be solved in polynomial time.*

In the remainder of this section, we prove the following theorem.

Theorem 8 *The restricted ℓ-phylogeny problem is NP-hard for fixed $\ell \geq 2$.*

Proof. The reduction is from the *ℓ-consecutive ones problem,* which is defined as follows:

INSTANCE: A (0,1)-matrix M.

QUESTION: Can the rows of M be permuted in such a way that for each column in the resulting matrix, there are at most ℓ sequences of consecutive ones.

The ℓ-consecutive ones problem is known to be solvable in polynomial time for $\ell = 1$[3]. However, it is NP-complete for fixed $\ell > 1$[3].

Let ℓ be a positive integer that is greater than or equal to 2. Suppose that we are given an input M to the ℓ-consecutive ones problem with n rows and m columns. (We will assume that $n \geq 3\ell$.) We will show how to construct an input $S, c_1, \ldots, c_{m+\binom{n}{\ell-1}}$ to the restricted ℓ-phylogeny problem such that the phylogeny input has a restricted ℓ-phylogeny if and only if the rows of M can be permuted in such a way that for each column in the resulting matrix there

are at most ℓ sequences of consecutive ones.

The phylogeny input is constructed as follows. Let M' be a matrix derived from M by replacing the zeroes in each column of M with integers in the range $2, ..., n+1$ in such a way that each column of M' has at most one occurence of each integer in the range $2, ..., n+1$. The species set S will have n species — one for each row of M'. For j in the range $1 \leq j \leq m$ character c_j will map the species corresponding to row r of M to the entry in column j of row r of M'. We will define the remaining $\binom{n}{\ell-1}$ characters as follows. For j in the range $1, \ldots, \binom{n}{\ell-1}$ we will have $R_{c_{j+m}} = \{0, 1\}$. We will let S_j denote the jth size-$(\ell-1)$ subset of S and we will set $c_{j+m}(s) = 1$ for $s \in S_j$ and $c_{j+m}(s) = 0$ for $s \notin S_j$.

$\rightarrow$ Suppose that T is a restricted ℓ-phylogeny for $S, c_1, \ldots, c_{m+\binom{n}{\ell-1}}$. Using Fact 1, we can assume that each species in S is the label of exactly one node in T. Let $V = \{v_1, \ldots, v_{\ell-1}\}$ be any set of $\ell - 1$ vertices of T and let j be the integer such that the species labeling the vertices in V correspond to the set S_j. Observe that the graph obtained by removing the vertices in V from T has at most ℓ connected components (otherwise, the set of vertices $c_{j+m}^{-1}(0)$ form more than ℓ connected components in T, so T is not an ℓ-phylogeny). We will show that every node in T has degree at most 2. Suppose instead that T has a vertex, v_1, of degree greater than or equal to 3. We will show that there are $\ell - 2$ other vertices, $v_2, \ldots, v_{\ell-1}$ such that the graph obtained by removing the vertices in $V = \{v_1, \ldots, v_{\ell-1}\}$ from T has at least $\ell + 1$ connected components. This will be a contradiction, so we will conclude that every node in T has degree at most 2. To show that $v_2, \ldots, v_{\ell-1}$ exist, note that the subgraph of T formed by removing vertex v_1 has at least 3 connected components. Furthermore, if any subgraph T' of T that is formed by removing up to $\ell - 1$ vertices has fewer than $\ell + 1$ connected components, it is possible to remove a vertex so as to increase the number of connected components [9]. Let v_2 be a vertex such that removing v_2 from $T - v_1$ increases the number of connected components. Similarly, let v_3 be a vertex such that removing v_3 from $T - \{v_1, v_2\}$ increases the number of connected components. Continuing this process we identify $v_2, \ldots, v_{\ell-1}$. We have now shown that T is a path. It follows that we can arrange the rows of M in the order that the corresponding species occur on path T and that, in such an arrangement, each column has at most ℓ sequences of consecutive ones.

$\leftarrow$ Suppose that $p = \{p_1, ..., p_n\}$ is a permutation of $\{1, ..., n\}$ such that when the rows of M are permuted according to p each column has at most ℓ sequences of consecutive ones. Let T be a path consisting of the species in S, arranged according to permutation p. Then T is a restricted ℓ-phylogeny for

[9] To see this, note that (since $n \geq 3\ell$) T' has some connected component with more than 2 vertices.

$S, c_1, \ldots, c_{m+\binom{n}{\ell-1}}$. $\square$

5 Two-Character Phylogeny

In this section we show that for $k = 2$ the phylogenetic number problem can be solved in $O(n^2)$ time, where n is the number of species. We start by proving the following fact.

Fact 9 *If a phylogeny input S, c_1, c_2 has an ℓ-phylogeny then it has a restricted ℓ-phylogeny T in which each species in S is the label of exactly one node and for each character $j \in \{1, 2\}$ and each state $i \in R_{c_j}$, at most one of the connected components in the subgraph of T induced by the set of vertices $c_j^{-1}(i)$ has more than one vertex.*

Proof. Suppose that T' is a an ℓ-phylogeny for S, c_1, c_2. We start by showing that S, c_1, c_2 has a restricted ℓ-phylogeny in which each species in S is the label of exactly one node. We can assume that each species is the label of at most one node of T' (if not, combine branches). Now, suppose that a species $s \notin S$ is the label of some node of T'. We can assume that this node, v, is an internal node of T' (otherwise delete it). Let U_1 be the set of neighbors u of v such that $c_1(u) = c_1(v)$. Let U_2 be the the set of neighbors u of v such that $c_2(u) = c_2(v)$. Note that $U_1 \cap U_2 = \emptyset$ since s is the only species that can label a node in $U_1 \cap U_2$ and v is the only node with label s. Let U_3 be the set of neighbors of v that are not in U_1 or U_2. We can form a new ℓ-phylogeny for S, c_1, c_2 by deleting node v, connecting the vertices in U_1 in a path, connecting the vertices in U_2 in a path, connecting the vertices in U_3 in a path, and connecting some node from U_1 to some node in U_2 and some node from U_2 to some node from U_3.

We have now shown that S, c_1, c_2 has a restricted ℓ-phylogeny in which each species in S is the label of exactly one node. Let T be such an ℓ-phylogeny. Suppose that for character $j \in \{1, 2\}$ and state $i \in R_{c_j}$, C and C' are two non-singleton connected components in the subgraph of T induced by the set of vertices $c_j^{-1}(i)$. Let $c \in C$ and $c' \in C'$ be vertices such that the path connecting c to c' in T does not include any other vertices in C or C'. (Note that c and c' are uniquely defined.) For every $v \in C$ which is adjacent to c note that the path between v and c' passes through vertex c. Remove the edge (v, c) from T and add the edge (v, c'). Note that the resulting tree is an ℓ-phylogeny for S, c_1, c_2. (To see this, note that since the species labeling v is different from the species labeling c, the character other than character j disagrees on v and c.) $\square$

In this section, we represent the phylogeny input S, c_1, c_2 as a bipartite graph. One set of vertices in the graph will be the set R_{c_1} and the other set of vertices in the graph will be the set R_{c_2}. For $i \in R_{c_1}$ and $j \in R_{c_2}$ the edge (i, j) will be present in the graph if and only if S contains a species s such that $c_1(s) = i$ and $c_2(s) = j$. (This is the partition intersection graph, [3,3].) Let $d(u)$ denote the degree of a vertex u in this graph. We will define a *special ℓ-coloring* of the graph to be a coloring of the edges with the colors white, blue, red, and purple such that each vertex i in R_{c_1} has $\max(0, d(i) - \ell + 1)$ of its neighboring edges colored either red or purple and the rest of its neighboring edges colored either white or blue and each vertex j in R_{c_2} has $\max(0, d(j) - \ell + 1)$ of its neighboring edges colored either blue or purple and the rest of its neighboring edges colored either white or red. (Intuitively, think of each edge as starting out white. Then each vertex i in R_{c_1} adds red color to $\max(0, d(i) - \ell + 1)$ of its neighboring edges and each vertex j in R_{c_2} adds blue color to $\max(0, d(j) - \ell + 1)$ of its neighboring edges. Edges that get colored both red and blue in this process become purple.) We will prove the following lemma.

Lemma 10 *A phylogeny input S, c_1, c_2 has an ℓ-phylogeny if and only if the corresponding bipartite graph has a special ℓ-coloring with no purple cycle.*

Proof. First, suppose that the input S, c_1, c_2 has an ℓ-phylogeny. By Fact 9 it has a restricted ℓ-phylogeny T in which each species in S is the label of exactly one node and for each character $h \in \{1, 2\}$ and each state $i \in R_{c_h}$, at most one of the connected components in the subgraph of T induced by the set of vertices $c_h^{-1}(i)$ has more than one vertex. Construct a special ℓ-coloring as follows. For each vertex $i \in R_{c_1}$ let C_i be the largest connected component in the subgraph of T induced by the set of vertices $c_1^{-1}(i)$. Arbitrarily choose $\max(0, d(i) - \ell + 1)$ of the vertices in C_i and add red color to the corresponding edges in the graph. For each vertex $j \in R_{c_2}$ let C_j be the largest connected component in the subgraph of T induced by the set of vertices $c_2^{-1}(j)$. Arbitrarily choose $\max(0, d(j) - \ell + 1)$ of the vertices in C_j and add blue color to the corresponding edges in the graph. We will now argue that the special ℓ-colored graph has no purple cycle. Suppose instead that the special ℓ-colored graph has a purple cycle consisting of the edges $(i_1, j_1), (i_2, j_1), (i_2, j_2), \ldots, (i_m, j_m), (i_1, j_m)$. Then, by construction, there is a path in T between the species (i_1, j_1) and the species (i_2, j_1) which is contained in C_{j_1}. Similarly, there is a path in T between the species (i_2, j_1) and the species (i_2, j_2) which is contained in C_{i_2}. These paths intersect exactly at the species (i_2, j_1). Continuing in this manner, we construct a cycle in T, which contradicts the fact that T is a phylogeny.

Next, suppose that the graph has a special ℓ-coloring with no purple cycle. Construct an ℓ-phylogeny T as follows. The nodes of T are the species in S. For each vertex $i \in R_{c_1}$, let C_i be the set of species in $c_1^{-1}(i)$ such that the

corresponding edges in the graph have red color. Add a path to T which traverses the nodes in C_i. All of the species on this path have the same state in character 1. Also, these species correspond to red edges in the special ℓ-coloring. For the purpose of the proof, we will think of the corresponding nodes in the path as having red color. For each vertex $j \in R_{c_2}$, let C_j be the set of species in $c_2^{-1}(j)$ such that the corresponding edges in the graph have blue color. Add a path to T which traverses the nodes in C_j. All of the species on this path have the same state in character 2. Also, these species correspond to blue edges in the special ℓ-coloring. For the purpose of the proof, we will think of the corresponding nodes in the path as having blue color. We will now argue that T has no cycle. Suppose instead that T has a cycle. Note by construction that every edge in the cycle either fixes character 1 or fixes character 2 (but not both). For example, the cycle might look like $(i_1, j_1), (i_1, j_2), (i_1, j_3), (i_2, j_3), (i_2, j_4), (i_2, j_5), (i_1, j_5)$. Let $(x_1, y_1), \ldots, (x_m, y_m)$ be the sequence of nodes that we get when we traverse the nodes in the cycle in order, skipping any node such that the edge into the node fixes the same character as the edge out of the node. (For the above example, we get the sequence $(i_1, j_1), (i_1, j_3), (i_2, j_3), (i_2, j_5), (i_1, j_5)$.) Each species (x_a, y_a) is colored purple in T, so each edge (x_a, y_a) is colored purple in the graph. (To see that species (x_a, y_a) is colored purple in T, note that it is part of a path fixing the state of character 1 (hence, red color is added). It is also part of a path fixing the state of character 2 (hence, blue color is added).) Finally, we observe that the edges (x_a, y_a) form a cycle in the graph, which contradicts the fact that the graph has no purple cycle. We conclude that T has no cycle. If T is disconnected, we arbitrarily add edges making it into a tree. $\Box$

We now present a polynomial-time algorithm that takes as input an integer ℓ and a bipartite graph G and determines whether G has a special ℓ-coloring with no purple cycle. The algorithm proceeds by considering a sequence of special ℓ-colored graphs $\{G_0, G_1, \ldots\}$. Graph G_0 is an arbitrary special ℓ-coloring of the graph G. For $t \geq 1$, G_t is constructed by modifying the coloring in G_{t-1}. We will use the notation $\mathcal{E}(G_t)$ to denote the set of edges that are contained in some purple cycle in G_t. When the algorithm considers the graph G_{t-1} it will either produce a graph G_t such that $\mathcal{E}(G_t) \subset \mathcal{E}(G_{t-1})$ or it will terminate with the answer "no". If the algorithm ever produces a graph G_t such that $\mathcal{E}(G_t) = \emptyset$ it will terminate with the answer "yes".

We now show how to construct the graph G_t from G_{t-1} (or to terminate with the answer "no"). Fix an edge $e \in \mathcal{E}(G_{t-1})$. The procedure will consider a sequence of special ℓ-colored graphs $G'_0 = \{G_{t-1}, G'_1, G'_2, \ldots\}$. For each graph G'_j in the sequence, e will be a member of $\mathcal{E}(G'_j)$. For each graph G'_j, let $P(G'_j)$ be the graph that is obtained by considering all of the purple edges in G'_j (and no other edges) and let $(U_e(G'_j), V_e(G'_j))$ be the vertices of the connected component in $P(G'_j)$ that contains e. Let $M_e(G'_j)$ be the set of edges in the

connected component in $P(G'_j)$ that contains e. To transform G'_j into G'_{j+1} the algorithm may make one *e-move* in which it either selects a vertex $u \in U_e(G'_j)$ and transfers the red color from one edge adjacent to u to another edge adjacent to u that does not already have red color or the procedure selects a vertex $v \in V_e(G'_j)$ and transfers the blue color from one edge adjacent to v to another edge adjacent to v that does not already have blue color. The move is *legal* if and only if $\mathcal{E}(G'_{j+1}) \subseteq \mathcal{E}(G'_j)$. Such a move is called a *finishing* move if $\mathcal{E}(G'_{j+1}) \subset \mathcal{E}(G'_j)$. It is called an *e-continuing* move if it is not finishing, but $M_e(G'_{j+1}) \subset M_e(G'_j)$. When it considers the special ℓ-colored graph G'_j, the algorithm checks every possible e-move. If it finds a legal e-move, it constructs G'_{j+1} by making this move. If the move is finishing, then the procedure returns the graph $G_t = G'_{j+1}$. If the move is not finishing, but it is e-continuing, the procedure now considers the graph G'_{j+1}. (Note that in this case $\mathcal{E}(G'_{j+1}) = \mathcal{E}(G'_j)$ so $e \in \mathcal{E}(G'_j)$.) If there are no legal e-moves that are finishing or e-continuing, the algorithm terminates with the answer "no". Note that at most $|M_e(G'_0)|$ continuing moves can be made, so the procedure terminates in polynomial time.

The correctness of the algorithm follows from the following lemma.

Lemma 11 *If a bipartite graph G has a special ℓ-coloring with no purple cycle and H is a special ℓ-coloring of G with $e \in \mathcal{E}(H)$ then there is a legal e-move from H that is either finishing or e-continuing.*

Proof. Let G_e be the subgraph of G induced by $U_e(H) \cup V_e(H)$ and let S_e denote the set of edges in G_e. We wish to compute an upper bound for $|S_e|$. To do so, let $d'(w)$ denote the degree of vertex w in graph G_e. Since G has a special ℓ-coloring with no purple cycle, G_e has a special ℓ-coloring with no purple cycle. Let H'_e be such a special ℓ-coloring of G_e. The number of edges with red color added in H'_e is at least $\sum_{u \in U_e(H)}(d'(u) - \ell + 1)$. The number of edges with blue color added is at least $\sum_{v \in V_e(H)}(d'(v) - \ell + 1)$. The number of purple edges (which have both red color and blue color) is at most $|U_e(H)| + |V_e(H)| - 1$. Hence,

$$|S_e| \geq \sum_{u \in U_e(H)} (d'(u) - \ell + 1) + \sum_{v \in V_e(H)} (d'(v) - \ell + 1) - (|U_e(H)| + |V_e(H)| - 1)$$

and therefore $|S_e| < \ell(|U_e(H)| + |V_e(H)|)$.

Now consider H. Let S_1 be the set of edges that are adjacent to vertices in U_e and do not have red color. Let S_2 be the set of edges that are adjacent to vertices in V_e and do not have blue color. Suppose that some edge e' is in $S_1 \cap S_2$. Let e'' be a purple edge that is adjacent to e'. Clearly, the e-move that transfers color from e'' to e' is legal. Suppose that it is not finishing and let H' be the

graph obtained from H by making this move. Then $M_e(H') \subseteq M_e(H) - \{e''\}$. Hence, the move is e-continuing.

Suppose instead that $S_1 \cap S_2 = \emptyset$. Every vertex $w \in U_e(H) \cup V_e(H)$ has $d(w) \geq \ell$. (If $d(w) < \ell$ then w will not add color to its neighboring edges in any special ℓ-coloring of G so w will not be in the connected component containing e in $P(H)$.) Therefore $|S_1| \geq \sum_{u \in U_e(H)}(\ell - 1)$ and $|S_2| \geq \sum_{v \in V_e(H)}(\ell - 1)$. Let S_3 be the set of purple edges with endpoints in $U_e(H) \cup V_e(H)$. Note that $|S_3| \geq |U_e| + |V_e|$. S_3 is disjoint from S_1 and S_2 so $|S_1 \cup S_2 \cup S_3| \geq \ell(|U_e(H)| + |V_e(H)|)$. We conclude that some edge in S_1 or S_2 must have an endpoint outside of G_e.

Without loss of generality, assume that there is an edge $e' \in S_1$ that has endpoint $u \in U_e(H)$ and its other endpoint, v, outside of $V_e(H)$. There are two cases. Suppose that u is contained in a purple cycle in H. Let (u, w) be an edge in such a cycle. Consider the e-move that transfers color from (u, w) to e'. This move is legal. (Since v is not in $V_e(H)$ no purple cycles are created by the move.) Let H' be the graph obtained from H by making this move. $\mathcal{E}(H') \subseteq \mathcal{E}(H) - \{(u, w)\}$, so the move is finishing. Suppose instead that u is not contained in a purple cycle in H. Let (u, w) be the first edge on the unique path from u to e in $P(H)$. Consider the legal e-move that transfers color from (u, w) to e'. Suppose that it is not finishing and let H' be the graph obtained from H by making this move. Then $M_e(H') \subseteq M_e(H) - \{(u, w)\}$. Hence, the move is e-continuing. □

In Lemma 10 we showed that a phylogeny input S, c_1, c_2 has an ℓ-phylogeny if and only if the corresponding bipartite graph has a special ℓ-coloring with no purple cycle. We then described a polynomial-time algorithm that takes as input an integer ℓ and a bipartite graph G and determines whether G has a special ℓ-coloring with no purple cycle. Hence, we have shown that there is a polynomial-time algorithm that takes input ℓ and a phylogeny input S, c_1, c_2 and determines whether the phylogeny input has an ℓ-phylogeny. (In fact, our algorithm constructs an ℓ-phylogeny if one exists.) Using binary search (or even linear search) on ℓ, we obtain a polynomial-time algorithm that takes as input a phylogeny input S, c_1, c_2 and determines the phylogenetic number of the input. Hence, we have proved the following theorem.

Theorem 12 *The phylogenetic number problem can be solved in polynomial time for $k = 2$.*

Unfortunately, Fact 9 no longer holds if we add a third character c_3. Hence, our approach does not solve the phylogenetic number problem (or even the ℓ-phylogeny problem) for fixed $k \geq 3$. (To see that Fact 9 does not hold for $k > 2$, consider the 3-species 3-character input $\{100, 010, 001\}$. One can construct a 1-phylogeny for this input by attaching each species to the new

species 000. However, the input does not have a restricted 1-phylogeny.)

We now show how to implement the two-character algorithm just described in time $O(n^2)$, where n is the number of species in the input set (hence the number of edges in the bipartite partition-intersection graph). In particular, given a special ℓ-coloring of the graph, we give an $O(n)$-time algorithm to perform an e-finishing move, perhaps through a series of e-continuing moves. Because there can be at most $O(n)$ e-finishing moves, the $O(n^2)$-result follows.

Let G be the partition-intersection graph for input S, c_1, c_2. Construct a special ℓ-coloring in $O(n)$ time by having each node i choose $\max(0, d(i) - \ell + 1)$ neighbors to color red (if node i is in R_{c_1}) or blue (if node i is in R_{c_2}). Compute the biconnected components of graph $P(G)$ (purple edges only) in time $O(n)$[3]. If all biconnected components are isolated nodes, then there are no purple cycles and we are done.

Otherwise, mark all the *active* vertices: those which belong to a biconnected component of size greater than one and therefore participate in a purple cycle. Pick an arbitrary active node v. In $O(n)$ time, compute the connected component of node v in $P(G)$ using depth-first search. We call these nodes *inside* nodes and all other nodes *outside nodes*. An edge (v_i, v_o) is *useful* if v_i is an inside node, v_o is an outside node, and it is not colored by node v_i. For example, if $v_i \in R_{c_1}$, then it controls the color red, so a useful edge going outward from v_i is white or blue.

If any active node v_a is adjacent to a useful edge, then we can make a finishing move by transferring color from an edge in a purple cycle adjacent to v_a to the useful edge (see the proof of lemma 11), and we are done. If there are no such edges, find all the useful edges adjacent to the remaining inside nodes. Place them in a *continuing* list and keep a pointer from the inside node to the corresponding record in this list. Pick the first edge on the continuing list (v_i, v_o). Let v_p be the inside node that is v_i's parent in the depth-first search tree created above. Then (v_p, v_i) is the first edge on the unique path to the distinguished node v. We make an e-continuing move (where e is any purple-cycle edge adjacent to node v) by transferring color from (v_p, v_i) to (v_i, v_o). This breaks the component of node v in $P(G)$ into two pieces. Node v_i and all its descendants in the depth-first search tree are now no longer part of node v's component.

We update the continuing list as follows. Starting at node v_i, trace each newly-severed node by walking the old depth-first tree. For each node x, consider all adjacent nodes y. If y is an inside node, then edge (y, x) is now useful. If node y is active, then add it to a second *finishing list* with a pointer from node y. Otherwise add it to the continuing list. If node y is outside, then this edge is no longer useful, so remove it from whatever list it is part of. Note that if

node y is inside now, but is moved outside in this tracing, then the edge (y, x) will be added and then removed from a list.

If there is an edge in the finishing list, we can make a finishing move and be done. Otherwise, we pick the first edge on the continuing list and iterate until we find a finishing move. Heuristically it would be better to pick a useful (continuing) edge from the node that is closest to node v.

The process of finding a finishing move requires time $O(n)$. Each edge that was purple at the start of the phase is traced (perhaps in each direction) at most twice: once when the first connected component is determined and once when the piece containing the edge is severed by an ϵ-continuing move. Each edge that was originally not purple is considered at most four times: once for each endpoint that is initially inside, and once as these endpoints move outside. Each of these edges can be added to a list once and removed once.

6 Phylogeny With a Fixed Number of States

In subsection 6.1 we show that the fixed-topology phylogenetic number problem can be solved in polynomial time for fixed r. On a related note, we show that if r is fixed, there is a polynomial-delay algorithm for listing fixed-topology ℓ-phylogenies. In subsection 6.2 we show that for fixed $r \geq 2$ and fixed $\ell \geq 3$ the restricted ℓ-phylogeny problem is NP-hard. (This result follows from a more general result. Namely, we show that the restricted (ℓ_1, ℓ_2)-phylogeny problem is NP-hard for fixed $\ell_1 \geq 2$ and $\ell_2 \geq 2$ as long as one of ℓ_1, ℓ_2 is greater than 2.)

6.1 Fixed-Topology Phylogeny with a Fixed Number of States

In this subsection we prove the following theorem.

Theorem 13 *The fixed-topology phylogenetic number problem can be solved in polynomial time for fixed r.*

It suffices to consider each character independently. We are given an input tree T with each of its n leaves labeled by a state in the range $\{1, \ldots, r\}$. We wish to label the internal nodes of T to construct a phylogeny with the smallest possible phylogenetic number. We root the tree at an arbitrary node, constructing the child and parent pointers. The choice of root will not affect the phylogenetic number of the tree.

For a given character, this problem can be solved by a two-pass algorithm: once up the tree and once down. In the upward phase, for each node v, and for each vector in the set

$$\{\, (i, \ell_1, \ldots, \ell_r) \mid \ (1 \leq i \leq r) \text{ and } 0 \leq \ell_j \leq n \text{ for } 1 \leq j \leq r \ \}$$

we construct, if possible, a labeling of the nodes in the subtree rooted at v such that v is labeled with state i and, in the subtree rooted at v, the subgraph induced by nodes labeled j has exactly ℓ_j connected components. We call such a labeling a *configuration* of the subtree rooted at v, or a configuration of v for short. If there are no leaves in this subtree labeled j for some $j \in \{1, \ldots, r\}$, then we have $\ell_j = 0$ for all configurations (there are no connected components labeled j in the subtree rooted at v).

There are $O(rn^r)$ possible configurations for the subtree rooted at any node, with one possible configuration for each leaf. Once the possible configurations have been constructed for the children of a node, we can construct the possible configurations for the parent by combining configurations of the children incrementally. Consider the first two children v_1 and v_2 of parent node v. For each pairing of a configuration for v_1 with a configuration for v_2, we construct r configurations for the subtree consisting of parent node v and the subtrees rooted at children v_1 and v_2, one configuration for each possible labeling of the parent v. If node v is labeled i, and the configurations of v_1 and v_2 are represented by the vectors $(i_1, \ell_{11}, \ell_{12}, \ldots, \ell_{1r})$ and $(i_2, \ell_{21}, \ell_{22}, \ldots, \ell_{2r})$ respectively, then the resulting configuration is $(i, \ell_1, \ell_2, \ldots, \ell_r)$ where $\ell_j = \ell_{1j} + \ell_{2j}$ for all $j \neq i$, and $\ell_i = \ell_{1i} + \ell_{2i} + 1 - m$, where $m \in \{0, 1, 2\}$ is the number of children (considering only v_1 and v_2) which are labeled i. That is, the number of components of state j is the sum of the number of components in each child for most states. The only state that can differ is the state with which node v is labeled (i). In this case, if neither v_1 nor v_2 is labeled i, then we create a new component of state i (the node v) in addition to the components present in the children. If exactly one child is labeled i, then the label of node v becomes part of that component. If both v_1 and v_2 are labeled i, then one component of state i from each child can merge through node v, and the number of components in the combination is one fewer than the sum.

Whenever a new possible configuration is achieved through a combination of configurations in the two children, it is recorded along with pointers to the configurations of v_1 and v_2 that achieve this phylogenetic configuration. Although there are $r^2 n^{2r}$ ways to pair up the configurations of two children, there can be at most rn^r configurations for the parent. If a configuration is achieved multiple ways, we only remember one way.

After computing the $O(rn^r)$ configurations for the subtree consisting of node v with the subtrees rooted at v_1 and v_2 (call this tree T'), we now add

child v_3. The compuation is almost the same as before. Let possible configurations for T' and the subtree rooted at v_3 be represented by vectors $(i, \ell'_1, \ell'_2, \ldots, \ell'_r)$ and $(j, \ell_{31}, \ell_{32}, \ldots, \ell_{3r})$ respectively. Then the combined configuration is $(i, \ell_1, \ell_2, \ldots, \ell_r)$ where $\ell_k = \ell'_k + \ell_{3k}$ for all k, unless $i = j$. In this case, we have $\ell_i = \ell'_i + \ell_{3i} - 1$ because one component of state i from the subtree rooted at v_3 can connect to components of state i from the other children through the parent v.

Each child of node v is added in this way until we have computed the $O(rn^r)$ possible configurations for the entire subtree rooted at node v. We continue up the tree until we have computed all possible configurations for the root. This computation takes $O(r^2 n^{2r+1})$ time. We then pick a possible configuration with the minimum phylogenetic number and go down the tree generating labels by following the pointers to the subconfigurations that achieve the optimal configuration.

The above algorithm makes it clear that if r is fixed, there is a polynomial-delay algorithm for listing fixed-topology ℓ-phylogenies.

6.2 Restricted Phylogeny with a Fixed Number of States

In this subsection we show that for fixed $r \geq 2$ and fixed $\ell \geq 3$ the restricted ℓ-phylogeny problem is NP-hard.

We start by proving the following more general theorem.

Theorem 14 *The restricted (ℓ_1, ℓ_2)-phylogeny problem is NP-hard for fixed $\ell_1 \geq 2$ and $\ell_2 \geq 2$ as long as one of ℓ_1, ℓ_2 is greater than 2.*

Proof. Without loss of generality, assume that $\ell_1 \geq \ell_2$. The reduction is from the 2-consecutive ones problem.

Let M be the matrix in the input to the 2-consecutive ones problem. Let n' denote the number of rows of M and m denote the number of columns of M. (We will assume that $n' \geq 3\ell_2$.) We will show how to construct an input to the restricted (ℓ_1, ℓ_2)-phylogeny problem such that the phylogeny input has a restricted (ℓ_1, ℓ_2)-phylogeny if and only if the rows of M can be permuted in such a way that for each column in the resulting matrix there are at most 2 sequences of consecutive ones.

The phylogeny input is constructed as follows. Let M' be a matrix derived from M by adding $2(\ell_2 - 2)$ rows to the bottom of M. The entries in the $(n' + i)$th row are equal to 0 for odd $i > 0$ and are equal to one for even

$i > 0$. Let n denote $n' + 2(\ell_2 - 2)$. Note that M' has n rows. The species set $S = \{s_1, \ldots, s_n\}$ will have n species. Species s_i will correspond to row i of M'. Let k_1 denote $\binom{n}{\ell_2 - 1}$. Let k_2 denote $\binom{n'}{\ell_2 - 1}$. Let k_3 denote $\max(0, n - n' - 1)$. Let k denote $m + k_1 + k_2 + k_2 k_3$. The input to the phylogeny problem will be $S, c_1, \ldots, c_k$. The characters $c_1, \ldots, c_k$ will be defined as follows:

(i) (Characters that describe M') For j in the range $1 \leq j \leq m$ character c_j will map species s_i to the entry in column j of row i of M'.

(ii) (Characters that make every phylogeny a path) For j in the range $1 \leq j \leq k_1$ let S_j denote the jth size-$(\ell_2 - 1)$ subset of S. We set $c_{m+j}(s) = 0$ for $s \in S_j$ and $c_{m+j}(s) = 1$ for $s \notin S_j$.

(iii) (Characters that place s_n at one end of the path) For j in the range $1 \leq j \leq k_2$ let S'_j denote the jth size-$(\ell_2 - 1)$ subset of $\{s_1, \ldots, s_{n'}\}$. We set $c_{m+k_1+j}(s) = 0$ for $s \in S'_j$ and $c_{m+k_1+j}(s_n) = 0$ and $c_{m+k_1+j}(s) = 1$ for every other species s.

(iv) (Characters that place $s_{n'+1}, \ldots, s_n$ consecutively at the end of the path) For j in the range $1 \leq j \leq k_2$ and i in the range $1 \leq i \leq k_3$ let m' denote $m + k_1 + k_2 + (i-1)k_2 + j$. We set $c_{m'}(s_r) = 0$ for $s_r \in S'_j$ and $c_{m'}(s_r) = 1$ for $s_r \in \{s_1, \ldots, s_{n'}\} - S'_j$. Furthermore, we set $c_{m'}(s_{n'+1}) = \cdots = c_{m'}(s_{n-i-1}) = 1$ and we set $c_{m'}(s_{n-i}) = \cdots = c_{m'}(s_n) = 0$.

$\rightarrow$ Suppose that T is a restricted (ℓ_1, ℓ_2)-phylogeny for $S, c_1, \ldots, c_k$. Using Fact 1, we can assume that each species in S is the label of exactly one node in T. Following the proof of Theorem 8, we can show that every node in T has degree at most 2. That is, T is a path. If $n = n'$ (i.e., $\ell_2 = 2$) then it follows that we can arrange the rows of M in the order that the species occur in path T and that, in such an arrangement, each column has at most 2 sequences of consecutive ones. Suppose instead that $n > n'$. We will now show that the node labeled s_n has degree 1. Suppose instead that it has degree 2. We argue as in the proof of Theorem 8 that there is a size-$(\ell_2 - 1)$ set $S' \subseteq \{s_1, \ldots, s_{n'}\}$ such that if s_n and the species in S' are removed from T, the resulting subgraph has at least $\ell_2 + 1$ connected components. Let j be the integer such that $S' = S'_j$. Then the set of vertices $c^{-1}_{m+k_1+j}(1)$ form more than ℓ_2 connected components in T, which is a contradiction. We conclude that the node labeled s_n is an endpoint of the path. For i in the range $1 \leq i \leq k_3$ we will now argue that the node labeled s_{n-i} is adjacent to a node with a label in $\{s_{n-i+1}, \ldots, s_n\}$. Suppose that this is not the case. We argue as in the proof of Theorem 8 that there is a size-$(\ell_2 - 1)$ set $S' \subseteq \{s_1, \ldots, s_{n'}\}$ such that if the species in $S' \cup \{s_{n-i}, \ldots, s_n\}$ are removed from T then the resulting subgraph has at least $\ell_2 + 1$ connected components. Let j be the integer such that $S' = S'_j$. Then the set of vertices $c^{-1}_{m+k_1+k_2+(i-1)k_2+j}(1)$ form more than ℓ_2 connected components in T, which is a contradiction. We conclude that T is a path consisting of the species in $\{s_1, \ldots, s_{n'}\}$ (in some order) followed by $s_{n'+1}, \ldots, s_n$. It follows that we can arrange the rows of M in the order that the species occur in path T and that, in such an arrangement, each column has at most 2 sequences of

consecutive ones.

$\leftarrow$ Suppose that $p = \{p_1, ..., p'_n\}$ is a permutation of $\{1, ..., n'\}$ such that when the rows of M are permuted according to p each column has at most 2 sequences of consecutive ones. If $\ell_2 = 2$ then let T be the path consisting of the species in $\{s_1, \ldots, s_{n'}\}$, arranged according to p. T is a restricted (3,2)-phylogeny for $S, c_1, \ldots, c_k$. Hence, T is a restricted (ℓ_1, ℓ_2)-phylogeny for $S, c_1, \ldots, c_k$. Suppose instead that $\ell_2 > 2$. Let T be a path consisting of the species in $\{s_1, \ldots, s_{n'}\}$, arranged according to permutation p, followed by $s_{n'+1}, \ldots, s_n$. Then T is a restricted (ℓ_2, ℓ_2)-phylogeny for $S, c_1, \ldots, c_k$. Hence, T is a restricted (ℓ_1, ℓ_2)-phylogeny for $S, c_1, \ldots, c_k$. $\square$

Note that Theorem 14 has the following corollary.

Corollary 15 *For fixed $r \geq 2$ and fixed $\ell \geq 3$ the restricted ℓ-phylogeny problem is NP-hard.*

7 Conclusions

In this section we present some open problems. There are several restrictions of the parameters which yield problems for which the complexity is still open. Recall that k is the number of characters, r is the maximum number of states for any character, and ℓ is the phylogenetic number. It is unknown whether the following restricted versions of the ℓ-phylogeny problem can be solved by polynomial-time algorithms:

(i) Finding an ℓ-phylogeny where the number k of characters is a constant greater than 2 (for $\ell > 1$),
(ii) Finding an ℓ-phylogeny where the number r of states per character is a constant.
(iii) For the case where $r = 2$, determining whether an input has a $(1, 2)$-phylogeny or a $(2, 2)$-phylogeny. Recall that for $r = 2$, the problem of finding a $(1, 1)$-phylogeny is in $\mathcal{P}$. Finding a $(2, 3)$-phylogeny is $\mathcal{NP}$-complete in the restricted case.

This paper also leaves open the problems of randomly generating phylogenies with constraints upon their phylogenetic number and approximation algorithms for the $\mathcal{NP}$-complete versions of the ℓ-phylogeny problem. In particular, suppose that there exists a perfect phylogeny. For what ℓ can we find an ℓ-phylogeny in polynomial time (with ℓ possibly a function of k and r)?

References

[1] R. Agarwala and D. Fernandez-Baca, "Fast and Simple Algorithms for Perfect Phylogeny and Triangulating Colored Graphs", *DIMACS Tech. Rept.* #94-51, 1994.

[2] R. Agarwala, D. Fernández-Baca, "A Polynomial-Time Algorithm for the Perfect Phylogeny Problem when the Number of Character States is Fixed", *procs. of the 34th annual Symposium on Foundations of Computer Science*, 1993.

[3] A. Aho, J. Hopcroft, J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[4] H. Bodlaender, M. Fellows, T. Warnow, "Two Strikes Against Perfect Phylogeny", *procs. of the 19th International Congress on Automata, Languages and Programming (ICALP)*, pp. 273-287, Springer-Verlag Lecture Notes in Computer Science, 1992.

[5] P. Buneman, "A Characterization of Rigid Circuit Graphs", *Discrete Math.* **9**: pp. 205-212, 1974.

[6] W.H.E. Day, "Computationally difficult parsimony problems in phylogenetic systematics," *Journal of theoretical biology*, 103: 429-438.

[7] W.H.E. Day and D. Sankoff, "Computational complexity of inferring phylogenies by compatibility", *Systematic Zoology*, 35(2): 224-229, 1986.

[8] L. Exdoffier and P.E. Smouse, "Using allele frequencies and geographic subdivision to reconstruct gene trees within a species: molecular variance parsimony" *Genetics* **136**: pp. 343-359, (1994).

[9] Wm. Fitch, "Toward defining the course of evolution: minimum change for a specified tree topology", *Syst. Zool.*, 20:406-416, 1971.

[10] D.R. Fulkerson, D.A. Gross, "Incidence matrices and interval graphs", *Pacific J. Math.*, **15** (3), 1965.

[11] P.W. Goldberg, M.C. Golumbic, H. Kaplan, R. Shamir, "Four Strikes Against Physical Mapping of DNA", *Journal of Computational Biology*, **2**(1), pp. 139-152, 1995.

[12] S. Kannan and T. Warnow, "Inferring Evolutionary History from DNA Sequences", *SIAM J. on Computing*, Vol. 23, No. 4, August 1994.

[13] S. Kannan and T. Warnow, "A Fast Algorithm for the Computation and Enumeration of Perfect Phylogenies when the Number of Character States is Fixed", *6th ACM/SIAM Symposium on Discrete Algorithms*, pp. 595-603, 1995.

[14] F.R. McMorris and C.A. Meacham, "Partition intersection graphs" *Ars Combinatorica* **16** pp. 135-138, 1983.

[15] F.R. McMorris, T. Warnow, T. Wimer, "Triangulating Colored Graphs", *SIAM J. on Discrete Mathematics*, Vol. 7, No. 2, pp. 296-306, 1994.

[16] R.C. Prim, "Shortest Connection Networks and Some Generalisations", *Bell System Tech. J.*, **36** 1389-1401, 1957.

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

[18] T. Warnow, "Efficient Algorithms for The Character Compatibility Problem", *New Zealand Journal of Botany*, Vol. 31, (1993), pp. 239-248.