PARSIMONY WITH GENERAL CHARACTER EVOLUTION

by

Chenchen Zhu

B.Sc., Shanghai TongJi University, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY

June 2003

APPROVAL

Name:	Chenchen Zhu
Degree:	Master of Science
Title of thesis:	Parsimony With General Character Evolution
Examining Committee:	Dr. Binay Bhattacharya Chair
	Dr. Arvind Gupta, Senior Supervisor, School of Computing Science, SFU
	Dr. Ladislav Stacho, Supervisor, Department of Mathematics, SFU
	Dr. Felix Breden, Associate Professor, Department of Biological Sciences, SFU External Examiner
Date Approved:	June 30/03

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENSE

Title of Thesis/Project/Extended Essay

Parsimony with General Character Evolution

Author: _____
(signature)

Zhu Chenchen
(name)

Aug. 01. 2003
(date)

:kj

Abstract

The questions of how to test hypotheses of character evolution, and incorporate character evolution into phylogenetic analysis are considered in this thesis. When a character transformation series is not linear, it is usually decoded into a set of binary characters by various binary coding methods. The disadvantages of such an approach have been discussed extensively in the literature. We propose a new approach together with the corresponding parsimony criteria for working with nonlinear transformation series. In particular, assuming a rooted character state tree is used to model the character evolution, the classical *small phylogeny problem* is extended by given not only the phylogenetic tree but also a character tree. Our techniques are based on finding tree minor embeddings of labeled trees. Three generalizations of *tree minor* are defined: *rooted tree minor*, *relax-minor* and *pseudo-minor*. Two new metrics, *bag cost* and *arc cost*, are also introduced as the target scoring functions of the problem. The *bag cost* is analogous to unweighted parsimony while the *arc cost* is analogous to weighted parsimony, that is, it allows a cost for each state transition. We show that the problem of finding minimum bag cost under relax-minor is **NP**-hard, however the problems of finding minimum pseudo-minor bag cost and minimum pseudo-minor arc cost can both be solved in linear time. Our algorithm for minimum pseudo-minor arc cost yields the same output for any character tree and phylogenetic tree as Sankoff's algorithm since a character tree can be transformed to a cost matrix. However our algorithm runs in linear time as opposed to the quadratic running time of Sankoff's. One application of our algorithms is the evaluation of multiple phylogenetic trees for a same set of species with the given character trees. Another application would be to test hypotheses for the evolution of a multistate character based on the given phylogenetic tree.

Acknowledgments

To my supervisors Arvind Gupta, Ladislav Stacho thank you for sharing your knowledge and ideas, for offering the financial support, and for all the countless hours you contributed which make this thesis possible. A big big thank you to Jano Manuch for the significant amount of time and effort he spent on helping with the thesis writing and proofreading. Special thanks to Felix Breden for being my external examiner. Your interest and advices are encouraging and valuable. My deepest appreciation goes to my dear family for their unconditional love and always being there for me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Phylogeny problem

Discovering patterns of evolution is receiving increasing attention amongst biologists, geologists, ecologists, and, most recently computer scientists. Traditionally, the approach to constructing phylogenies, or *trees of life* was through the study of fossil records. New techniques include constructing the best fit for a set of characters from matrices of characters, maximum likelihood constructions, and pair-wise distance constructions which assume a certain rate of mutation [FM67] [SN87] [Fel81]. This virtual explosion of techniques and algorithms has lead to the publication of many new phylogenies which can often be contradictory. Statistical approaches have been developed to assess their quality and closeness of their fit to the given data [SI89] [YTM94].

Recently, constructing phylogenetic trees using molecular data has achieved considerable prominence. One approach is to consider the character matrix for a set of extant species and set of characters for these species. This matrix gives the state of each character for each species. The problem is to construct a phylogenetic tree under an underlying *parsimony* assumption. Specifically, the internal nodes of the tree correspond to possibly hypothetical (extinct) species each labeled by a vector of character states. Parsimony dictates that the number of state changes for each character is minimized (here we count all state changes in moving from the root to the leaves).

The problem of constructing a phylogenetic tree from the character matrix, the *large* phylogeny problem, is **NP**-complete even when each character is binary (i.e. can take on only two values) [FL82] [DS86]. With the stronger assumption that the internal structure of the tree is known, the *small* phylogeny problem, there are polynomial time algorithms both for the case of uniform cost of each state change [Fit71] and non-uniform cost [San75].

In this thesis, we further investigate the small phylogeny problem where partial information of the evolutionary order of a multistate character is also given. In particular, we consider the case that such evolutionary order is represented as a rooted tree, called character tree. In what follows, we will review the history of character tree and explain the motivations of our work.

1.2 History of character trees

A character phylogeny [Hen66] or a character transformation series [Hen66], or character state tree [Far70] of a multistate character is a hypothesis that specifies which states of the character evolve directly into which other states. As explained by Mickevich [Mic82], to determine the character transformation series, both the character state polarity and character state order need to be known. The character order only describes which states are intermediate, but does not specify evolutionary direction. However character polarity explains which state is plesiomorphic or ancestral. Character polarity can be determined by using the outgroup comparison, parsimony analysis [Far82] [Fit71] [Mic82], fossil and stratigraphic data or ontogenetic criteria. To determine the character state order, various methods have been utilized. One direction is to impose a rule on how the character evolved. Examples include Haeckel's biogenetic law, Cope's rule and Bergmann's Rule in Morphocline analysis, Ontogenetic analysis. (See [MW90] for a review). The other direction is to maximize congruence among characters such as non-additive analysis [Fit71] or transformation series analysis (TSA) which runs in an iterative procedure [Mic82]. The congruence of a character with others in a phylogenetic tree means that species with similar states should be adjacent to each other.

In this thesis, we are not primarily concerned with transformation series inference. Instead, we focus on the methods of testing character transformation series and furthermore utilize it to study the relationships among the species possessing the character. Different approaches such as non-additive analysis, TSA and Morphocline analysis usually disagree on the transformation series for the same character because they are based on different optimization assumptions [Lip92]. Therefore, Lipscomb suggests that the transformation series should be viewed as a hierarchy of homologies, and then the methods phylogeneticists used to postulate and test homology should be used to test transformation series. In their method, the congruence of a transformation series with the other characters is used to test its support and as a means for choosing among several alternative transformation series for the same character. She is mainly concerned with the non-congruence caused by scattering and hierarchical discordance [ML91]. Scattering refers to the phenomenon that similar

states in different species are non-homologous with the result that a state appears in two or more species that are not adjacent on a phylogenetic tree. On the other hand, hierarchical discordance happens when the states that occur adjacently on the tree have a median state (or states) between them in the transformation series. It indicates that the order of the states in the transformation series is in conflict with the phylogenetic tree. The implementation of non-congruence detecting is easy when hypothesized transformation series is linearly ordered, but quite complicated and questionable when it is not linear. In the latter case, a character state tree needs to be recoded into multiple binary characters using additive binary coding [Far70] [CS65] since many program packages require linear variables (e.g. Hennig86, NTSYS, PAUP, PHYLIP). As Lipscomb mentioned, the recoding makes it more difficult to detect hierarchical discordance [Lip92].

It is not the first time that the disadvantage of recoding multistate characters is addressed. Mickevich already gave attention to this issue as early as 1982. He noted that "When data are restricted to distinct two state characters, the depth of cladistics as a theoretical approach is severely restricted. Such an approach clearly ignores evidence presented by multistate characters." [Mic82]. We also found the same issue when a known character transformation series need to be incorporated in the study of the phylogeny of species. In 1981, Brooks [Bro81] proposed the idea of using the phylogenetic tree of parasites as a character state tree to study the phylogeny of their hosts by assigning each host a multistate code associated with the parasite they harbor. A few years later, O'Grady and Deets [OD87] presented much more detailed illustration on the implementation of Brooks's proposal together with the coding schema for multistate characters. Their implementation requires the phylogeny of parasites to be transferred into a matrix by method, for example additive binary coding, redundant linear coding and nonredundant linear coding [OD87]. Again, a character tree are represented as binary characters.

Although it is common that multistate characters are disassembled into suites of binary characters for the purposes of analysis by existing methods in practice, more and more problematic results are now recognized [JAHS97] [OD87] [PM90]. Ogue and Mickevich demonstrated the inherent pitfalls of such practice in [PM90]. They showed that the disadvantages of representing multistate character as independent binary characters include the creation of artificial homoplasy, the obscuring relationships between species due to an arbitrary division of multiple states into two or more binary characters, and the ignorance of synapomorphic evidence offered by multistate transformation. Later Maddison showed that such disassembling may introduce inapplicable characters to some specific species, and then those character states must be treated as missing data [Mad93]. Therefore having a method of comparing character tree directly with phylogenetic tree of species without being coded

into binary characters is highly desirable as is the corresponding parsimony criterion.

1.3 Stratophenetics

There is considerable debate in the scientific community about the value of stratophenetics (the use of fossil records) in constructing phylogenies. In 1998, Nature held an on-line debate between many of the top paleontologists and geneticists to examine exactly this question [(mo98]. While it is the case that fossil records degrade over time, it is not so clear that they are less reliable. Almost every dating method exhibits similar behavior. Furthermore, in many cases fossil data is remarkably reliable and stands up to rigorous statistical analysis [MJBH00]. In particular, these tests show that the reliability of fossil data is uniformly consistent although partial information is lost over time.

There are two main types of information derived from stratophenetics. First, the age and duration of fossils can be determined by dating the sedimentary rock in which the fossil is found. Second, certain characters of the underlying species can be determined from the nature of the fossil itself. Molecular techniques also yield similar information. By performing alignment and assuming a certain rate of mutation, the evolutionary distance between species can be inferred while sequencing yields character state information.

A number of authors have suggested that phylogenetic trees incorporating fossil data may yield significantly better evolutionary trees (see for example [Ben01]). The emphasis is on determining consistency between trees constructed under both techniques with the result a "better" phylogeny. However, the problem of incorporating both fossil and molecular data has not been addressed. Here we present one such approach which we call *parsimony with general character evolution.*

1.4 Parsimony with general character evolution

Our starting point is to consider a set of character states whose transformation series (i.e. a partial ordering on the evolution of these states) is known. A natural representation of this information is a Hasse diagram. Since the phylogenetic information we consider is represented as a rooted tree, we will assume that the character state tree also occurs as a rooted tree.

The problem can be summarized as follows: We are given a *character tree* representing an evolution of some character. The vertices of the character tree represent states of this character. We are also given a set of species each taking on one state of the character and must find a parsimonious phylogenetic tree consistent with the character tree. If the internal structure of the phylogenetic tree

is not given, then for one character, it is trivial to construct a phylogenetic tree congruent with the character tree. However the problem is **NP**-complete for a general set of characters. Instead we consider the *small phylogeny* problem in which the internal structure of the phylogenetic tree is known. Since a transformation series is tested against a phylogenetic tree constructed from other characters, the *small phylogeny* problem also models Lipscomb's problem of testing transformation series. Regarding to the process of the character tree, we avoid the binary coding, and keep the original form of rooted tree. Thus not only the logical dependence and hierarchy between states are kept, which will be otherwise lost after a character tree is converted a set of binary characters [JAHS97], but also a lot of space can be saved since the binary coding matrix for all but the simplest trees can be quite large [OD87].

Our techniques are based on finding graph minor embeddings of labeled trees. Graph minors are generalizations of isomorphisms in which a vertex of the source graph is mapped to a connected component of the target graph preserving the adjacency relation of the source graph. Tree minors are the basis of the seminal work of Robertson and Seymour who used them to prove Wagner's conjecture [RS86] and the flavor of their techniques is carried forward here. We define three generalizations of graph minors, *rooted tree minor*, *relax-minor* and *pseudo-minor* which reflect structures arising in phylogenetic trees.

We will investigate the *small parsimony* problem under two different optimality criteria. In the first, the subgraph of the phylogenetic tree induced by a particular state has as few connected components as possible. It also reflects the non-congruence of scattering mentioned in [ML91] because less components implies less scattering. In the second, we allow a cost for state transitions (represented as edge costs in the phylogenetic tree) and look for trees that minimize the sum of these arc costs. Similarly, it reflects to the non-congruence of hierarchical discordance mentioned in [ML91] with less arc costs implying less hierarchical discordance. In both cases we find linear time algorithms for these problems. Finally, we show that certain variations of these problems (even when the internal structure of the phylogenetic tree is known) are **NP**-hard.

1.5 Thesis overview

In Chapter 2, we describe the standard parsimony optimization criteria and algorithms for small phylogeny problem. In particular, we describe Fitch's algorithm for unweighted parsimony and Sankoff's algorithm for weighted parsimony. Both algorithms are a basis for our results.

In Chapter 3, we introduce our version of the small phylogeny problem. This is also based on

parsimony, but assumes a rooted character state tree is given, called character tree. Extending from two kinds of non-congruences considered by Lipscomb, we distinguish five inconsistencies between character tree and phylogenetic tree which are separation, transitivity, inversion, addition and negligence. Separation and transitivity corresponds to Lipscomb's scattering and hierarchical discordance respectively. Moreover, Lipscomb limits the scattering on the level of leave species, however separation extends it to also include the internal hypothetical ancestral species. Hence separation can detect the scattering invisible on leaves. Inversion follows from Camin and Sokal's assumption that evolution is irreversible. Addition indicates that the order of the states in the transformation series conflict with the cladogram but not caused by transitivity. Finally negligence indicates that the order of the states in the transformation series is not reflected on the phylogenetic tree. Our goal to the small phylogeny problem is then to label the phylogenetic tree so as to minimize inconsistencies, with the character tree. This approach is modeled on the tree minor structure. Extending the notion of tree minor, we define the rooted-minor as both phylogenetic tree and character trees are rooted.

To allow for inconsistencies, two relaxations of rooted-minor, relax-minor and pseudo-minor, are defined. We introduce two new cost functions, bag cost and arc cost , corresponding to the unweighted and weighted parsimony respectively. Using these in our scoring functions, with relax-minor and pseudo-minor yields the problems under study in this thesis.

Chapter 4 shows two **NP**-complete results, namely that the decision problem of rooted-minors is **NP**-complete when the leaves of the host tree are not prelabeled, and the problem of finding minimum relax-minor bag cost is also **NP**-hard irrespective of leaf labels.

In Chapter 5, we give linear time algorithms for the small phylogeny problems for both minimum pseudo-minor bag cost and minimum pseudo-minor arc cost. We contrast our algorithm for finding the labeling with minimum pseudo-minor arc cost with Sankoff's algorithm, which outputs the same labeling with the minimum arc cost since a cost matrix can be transfered from a character tree easily. For this limited case, our algorithm yields a linear time speeding. We also present a linear time algorithm for the decision problem of rooted-minor when the leaves of the host tree are prelabeled.

Our method can be applied to both character evolution analysis and phylogenetic tree inference. Chapter 6 describes three experiments we conducted on the data sets from previous work. In first experiment, following Lipscomb's idea of testing character trees while there are multiple phylogenetic trees [Lip92], we perform the test based on her data, and show that our metrics detect the same worst character tree as what she found. Second experiment is to test the hypothesis of the evolution of behavioral characters proposed by van Tets [vT65] on a best estimate phylogenetic tree of pelecaniforms followed the work of [MKG96]. It turns out that van Tets's hypothesis does not have the

best score compared with other alternative hypothesis under our metric. The last experiment aims to compare several phylogenetic trees of anura, which are independently constructed from different data source such as morphological data and molecular data [KF69] [FC93] [JMHM95], using three character trees from [KF69] and [Ing67]. The one constructed from molecular data is found to be mostly inconsistent with character trees.

We conclude in Chapter 7 with some summary remarks and open problems.

Chapter 2

Background and Definitions

2.1 Preliminary definition

Let $G = (V(G), A(G))$ be a directed tree with vertex set $V(G)$ and arc set $A(G)$. The symbol $\langle u, v\rangle$ will represent the arc from u to v.

Given two vertices u and v of G, if there is a directed path of length ≥ 1 from u to v, then we say u is an *ancestor* of v and v is an descendant of u; this is denoted by $u \prec v$, and the unique path from u to v in G is denoted by $u \rightarrow v$. In particular if $\langle u, v\rangle \in A(G)$, then we say that u is the *parent* of v and v is the child of u. If $u \not\prec v$ and $v \not\prec u$, then we say u and v are *incomparable*; this is denoted by $u \nsim v$.

A rooted tree is a directed tree with a unique vertex called *root* having no parent, and every other vertex having exactly one parent. The degree of a vertex $u \in V(G)$ is the number of children it has. The degree of G is the maximum degree of all of its vertices. The height of G is the length of the longest path from the root to a leaf. For each internal vertex $u \in V(G)$, let G_u denote the sub-tree of G rooted at u. The set of all leaves in G is denoted by $L(G)$.

Definition 1. A vertex u is the *least common ancestor* of $v_1, v_2, ..., v_k$, written

$u = \mathrm{LCA}(v_1, v_2, ..., v_k)$, if $u \prec v_j$ for $j = 1, 2, \ldots, k$, and for any other vertex u' so that $u' \prec v_j$ for $j = 1, 2, \ldots, k$, we also have $u' \prec u$.

Definition 2. Given two vertices u and v of G, if $u \prec v$, then the distance $d(u, v)$ from u to v is the number of arcs on the path $(u, \cdots, v)$ in G; otherwise $d(u, v) = \infty$.

2.2 Small phylogeny problem

Given a rooted tree G and a finite set C together with a function $p := L(G) \to C \cup \{\varepsilon\}$, where C is called the set of all possible states of a character, and $\varepsilon \notin C$, we say that a function l on $V(G)$ is *p-constrained* if either for every $u \in L(G)$, we have $p(u) = \varepsilon$, or for every $u \in L(G)$, we have $p(u) \neq \varepsilon$ and $l(u) = p(u)$. Since for a given phylogenetic tree in the *small parsimony problem*, the character state of the species represented by the leaves are given, we will mainly consider p for which $p(u) \neq \varepsilon$ for every $u \in L(G)$. However to better study the complexity of the underlying problems, we allow $p = \varepsilon$.

The pair (G, p) is called a *phylogenetic tree* if leaves of G represent a set of extant species, internal vertices of G represent hypothetical ancestors, and for each leaf $v \in L(G)$ the character of the species v is in the state $p(v)$. A triple (G, p, l) is a *fully labeled phylogenetic tree* if (G, p) is a phylogenetic tree and $l : V(G) \to C$ is a p-constrained labeling.

Given phylogenetic tree (G, p), the classical small phylogeny problem asks to find a labeling l such that (G, p, l) is a fully labeled phylogenetic tree and a certain score function (involving l) is minimized. Note that the labeling l determines the state of the character for each species in G; i.e. as a result we assume that $l(v) \in C$ is the character state of the species $v \in V(G)$.

2.3 Previous work

Parsimony provides one approach to the small phylogeny problem. There are various types of parsimony criteria in which transformations between character states may be constrained. They differ primarily in their optimality criteria, the weighting of the transformations permitted, and in the actual algorithms utilized to find the minimal cost for the given phylogenetic tree. However they all assume that each character develop independently. In this chapter, we will review five most commonly used parsimony criteria and explain two implementations.

Wagner parsimony was based upon the work of Wagner(1961) and formalized by Frarris(1970) [Far70]. The states of a character are measured on an interval scale, i.e. a transformation from one state to another must pass through all intermediate ordered states which are presumably known. For example, if $C = \{a, b, c, d\}$ and the order of four states is a, b, c and d, then a transformation from a to b would be one step, from a to c two steps, and from a to d three steps. However free reversibility of states is allowed, e.g. a transformation from c to a is permissable and has the same steps as the one from a to c.

Fitch parsimony (Fitch 1971) [Fit71] is a generalized version of Wagner parsimony by allowing unordered states. Once again free reversibility is allowed and transformation from any state to any other state has constant cost. In the above Wagner parsimony example, a transformation from a to b, from b to a, from a to c, from c to a, from a to d and from d to a would all take one step.

Both Wagner and Fitch parsimony allow free reversibility of states, but there are situations in which character states may be constrained in such a way that certain transformations are considered either highly unlikely or impossible. Dollo parsimony (Farris 1977) [Far77] was introduced to accommodate those evolutionary scenarios. It is especially useful for restriction sites data where a site is difficult to gain but easy to lose, therefore the two transformations have to be weighted accordingly. However Dollo parsimony requires the state polarity to be prespecified.

Camin-Sokal parsimony (Camin and Sokal 1965) [CS65] assumes evolution is irreversible, i.e. once a state has been acquired it may never be lost. A *priori* knowledge about state evolution is required.

Finally generalized parsimony [SO90] [SC83] assigns a cost to every possible transformation of states, often represented as a $k * k$ matrix M, where $k = |C|$ and M_{ij} represents the cost of the transformation from state i to state j. All the above parsimony criteria can be treated as special cases of a generalized parsimony since the matrix can be weighted to correspond to Fitch, Wagner, Dollo and Camin-Sokal parsimony.

Each of the above five optimization criteria has its own implementation when applied to the small phylogeny problem. In what follows, we present Fitch's algorithm for Fitch's parsimony and Sankoff's algorithm for generalized parsimony. Our techniques borrow heavily from them. Originally, both algorithms consider characters separately and assume the given phylogenetic tree is a binary tree. In this thesis, both algorithms are generalized to also accept trees that are not binary.

2.4 Fitch's algorithm

Fitch's Algorithm is used to find the minimum number of state changes for the given phylogenetic tree.

Input: A phylogenetic tree (G, p) with degree δ, and a single character with a state set C of k possible values.

Output: A labeling l such that (G, p, l) is a full labeled phylogenetic tree with the minimum number of state changes.

Description: There are two steps. In the first step, the tree is traversed in postorder to assign a set

of possible states $S_v \subseteq A$ to each vertex $v \in V(G)$. If v is a leaf, then let $S_v := \{p(v)\}$. Otherwise, let $u_1, u_2, \cdots, u_j$ be v's children where j is the degree of v ($j \leq \delta$). If $S_{u_1} \cap S_{u_2} \cdots \cap S_{u_j} \neq \emptyset$ then $S_v := S_{u_1} \cap S_{u_2} \cdots \cap S_{u_j}$, else $S_v := S_{u_1} \cup S_{u_2} \cdots \cup S_{u_j}$.

In the second step, the tree is traversed in preorder to assign values of l to each internal vertex $v \in V(G)$. Let u be the parent of v. If $l(u) \in S_v$, then $l(v)$ is assigned $l(u)$. Otherwise the algorithm arbitrarily assigns any $t \in S_v$ to $l(v)$ (including the root). Finally the total number of state changes equals the total number of union operations in the first step.

Complexity: For each vertex $v \in V(G)$, it takes $O(k \cdot \delta)$ time to compute S_v in the first step, and again $O(k)$ time to computer $l(v)$ in second step. Therefore the total running time is $O(|V(G)| \cdot k \cdot \delta)$. When G is a binary tree, then the running time is $O(|V(G)| \cdot k)$.

Example:

Figure 2.1 shows an example ran Fitch's algorithm on a four species binary phylogenetic tree and a character with a states set $C = \{a, c, g\}$. (a) is the input phylogenetic tree (G, p), (b) is the intermediate result after first step, and (c) is the full labeled tree (G, p, l) after the second step. The asterisks in (b) mark the vertices where $S_u \cap S_w = \emptyset$. The minimum state changes of this tree is two.

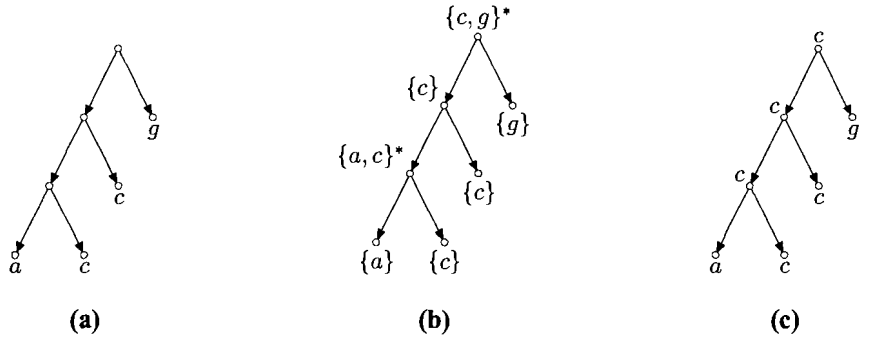


Figure 2.1: An example of Fitch's Algorithm for a 4-species binary phylogenetic tree.

2.5 Sankoff's algorithm

Sankoffs algorithm [SC83] is a generalization of Fitch's algorithm that allows different costs for transformations between different states.

Input: A phylogenetic tree (G, p) with degree δ, and a $k * k$ cost matrix M for a single character with a state set C of k possible values.

Output: A labeling l such that (G, p, l) is a full labeled phylogenetic tree with minimum cost of state changes.

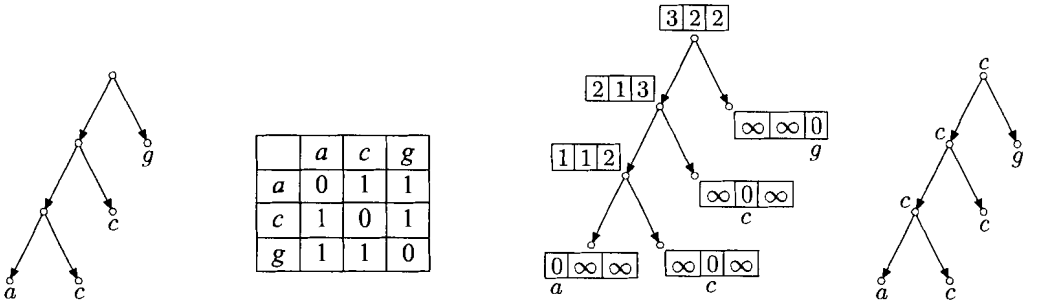
Description: There are two steps. In the first step, the tree is traversed in postorder. For each vertex $v \in V(G)$ and each state $t \in C$, compute a quantity $S_t(v)$ which is the minimum cost of the subtree G_v when $l(v) = t$. If v is a leaf, then $S_t(v) := 0$ for $t = p(v)$, and $S_t(v) := \infty$ for $t \neq p(v)$. Otherwise $S_t(v) = \sum(\min_{x \in C}\{M_{tx} + S_x(u_1)\} + \cdots + \min_{y \in C}\{M_{ty} + S_y(u_j)\}$ where $u_1, \cdots, u_j$ are children of v ($j \leq \delta$).

In the second step, the tree is traversed in preorder to determine the value of l for internal vertices $v \in V(G)$. If v is the root, then $l(v) := \text{argmin}_{t \in C} S_t(v)$. Otherwise, let u be v's parent, $l(v) := \text{argmin}_{x \in C}(M_{l(u)x} + S_x(v))$. It is easy to see that the minimum cost of G is $\min_{t \in C} S_t(g)$ where g is the root of G.

Complexity: For each vertex $v \in V(G)$, it takes $O(k \cdot \delta)$ steps to compute $l(v)$, so the total running time is $O(|V(G)| \cdot k \cdot \delta)$. Again if G is a binary tree, then the running time is $O(|V(G)| \cdot k)$.

Example:

Figure 2.2 shows an example ran Sankoff's algorithm on a four species binary phylogenetic tree and a cost matrix on a character with the states set $C = \{a, c, g\}$. In Figure 2.2, (a) is the input phylogenetic tree (G, p), (b) is the cost matrix, (c) is the intermediate result after the first step, and (d) is the full labeled tree (G, p, l) after the second step. In (c), three cells besides each vertex $v \in V(G)$ show $S_a(v)$, $S_c(v)$ and $S_g(v)$ from the left to right respectively. The minimum cost of this tree is two and the root of G can be labeled as either c or g with the same cost.



	a	c	g
a	0	1	1
c	1	0	1
g	1	1	0

Figure 2.2: An example of Sankoff's Algorithm for a 4-species binary phylogenetic tree.

Chapter 3

Parsimony With General Character Evolution

For C the set of states of a character, we assume that partial information about the evolution of the states is known and is represented as a rooted tree. For two vertices a and b of the tree, $a \prec b$, if and only if the state b is derived from the state a.

Given two rooted trees, the character tree H and the phylogenetic tree (G, p), we must find a labeling l such that (G, p, l) will be a fully labeled phylogenetic tree such that for all two states $a, b \in C$ the following two conditions are satisfied.

(i) If $a \prec b$, then for every pair of vertices $u, v \in V(G)$ with $l(v) = a$ and $l(u) = b$, $u \not\prec v$;

(ii) If $a \nsim b$, then for every pair of vertices $u, v \in V(G)$ with $l(v) = a$ and $l(u) = b$, $u \nsim v$.

Extending from Lipscomb's two non-incongruences, one can distinguish five types of inconsistencies between the evolutionary order of species given by (G, p, l) with the order of states given by H. (see Figure 3.1.)

Definition 3. The following are five types of *inconsistencies* between the evolutionary order of species given by (G, p, l) and the order of states given by H:

- A *transitivity* occurs if for some $a, b \in V(H)$ $a \prec b$, $\langle a, b \rangle \notin A(H)$, and for some $\langle u, v \rangle \in A(G)$, $l(u) = a$ and $l(v) = b$.
- An *addition* occurs if for some $a, b \in V(H)$ $a \nsim b$, but for some $u, v \in V(G)$ with $l(u) = a$ and $l(v) = b$ either $u \prec v$ or $v \prec u$.

- A *separation* occurs if there exist three vertices $u, v, w \in V(G)$ so that $l(u) = l(w) \neq l(v)$ and $u \nsim w$ and $v \prec u$ and $v \prec w$.
- An *inversion* occurs if for some $a, b \in V(H)$ $a \prec b$ and for some $u, v \in V(G)$ so that $l(u) = b$ and $l(v) = a$, $u \prec v$.
- A *negligence* occurs if for some $\langle a, b \rangle \in A(H)$ there is no $\langle u, v \rangle \in A(G)$ with $l(u) = a$ and $l(v) = b$.

Separation and transitivity correspond to Lipscomb's scattering and hierarchical discordance respectively. However, separation is more general than scattering as it applies to all nodes, not just leaves. Specifically, scattering is the non-adjacent, multiple occurrence of the same state in species on a phylogenetic tree. Lipscomb believes separation is independent of the character tree, and therefore only considers separation at the leaves. Figure 3.2 (a) illustrates scattering where two species in state b are not adjacent. However, the occurrence of separation is dependent on the character tree. In Figure 3.2 (b), we show a separation in (G, p, l) which is not a scattering. Thus separation is a more powerful concept than scattering as it can detect potential scattering invisible to the phylogenetic tree itself.

In constructing phylogenetic trees, all five inconsistencies should ideally not occur. It is not difficult to see that conditions (i) and (ii) prevent inversions and additions, respectively. If the other three inconsistencies also did not occur then the tree H would be a rooted-minor of the tree G (Formal definition of rooted-minor as well as all its modifications mentioned below are given in the next section.) We will see however, that deciding whether H is a rooted-minor of G is an **NP**-complete problem.

To allow for some natural inconsistencies, we generalize the notion of rooted-minor.In particular, if H is a *relax-minor* of G, then inversions and negligences are disallowed, and if H is a *pseudo-minor* of G, then inversions and additions are disallowed. In Chapter 4 and Chapter 5, we prove that deciding whether H is a relax-minor of G is an **NP**-complete problem, but deciding whether H is a pseudo-minor of G can be done in polynomial time.

3.1 Minors and their relaxations

Let H and G be two trees. We say that H is a *minor* of G, if H is isomorphic to a tree obtained from G by contracting edges. The corresponding decision problem (to decide whether H is a minor

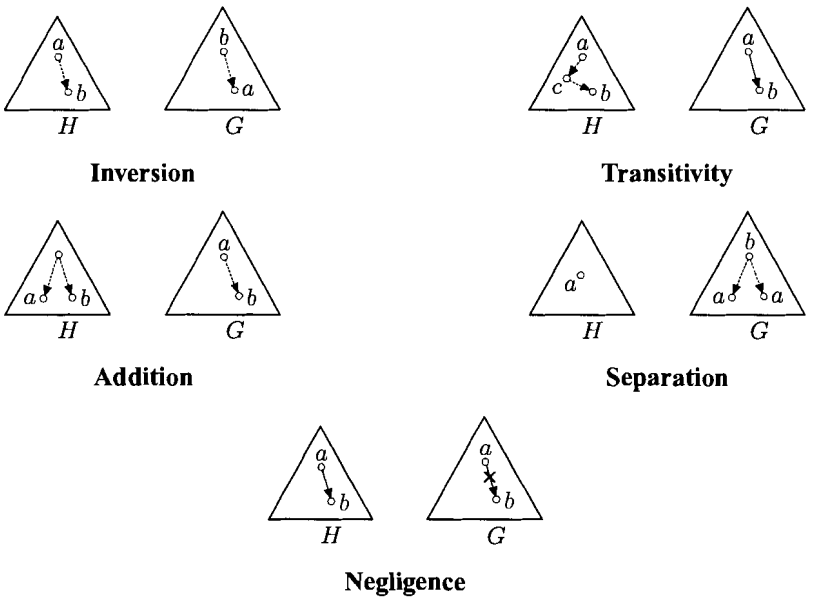


Figure 3.1: Five types of inconsistencies. Solid lines indicate direct derivation, dash lines indicate transitive derivation.

of G) is intractable, see [MR92]. Since both the character tree and the phylogenetic tree are rooted trees and, moreover, the phylogenetic tree has leaves prelabeled, the concept of minor does not quite model our problem. In what follows, we define three modifications of the minor concept. In these modifications we assume trees are rooted and leaves of the host graph are prelabeled.

Definition 4. Given two rooted trees H and G, let $l : V(G) \to V(H)$ be a function. Let $v \in V(H)$, the set of components in the sub-graph B_v^l induced by vertices of G in $l^{-1}(v)$ is called the *bag-set* of v induced by l. Any particular component of B_v^l is referred to as a *bag* of v; see Figure 3.3. The number of components $c(B_v^l)$ of B_v^l is the number of bags of v induced by l.

Definition 5. Given two rooted tree H, (G, p) and a p-constrained functions $l : V(G) \to V(H)$, if for an arc $\langle a, b\rangle \in A(H)$, there exists $\langle u, v\rangle \in A(G)$ such that $l(u) = a, l(v) = b$, we say $\langle a, b\rangle$ is realized by l on $\langle u, v\rangle$. Furthermore let $r(\langle a, b\rangle, l)$ denote the number of arcs in $A(G)$ that realize $\langle a, b\rangle$ by l.

Definition 6. Given two rooted trees H and (G, p), let $M(H, G, p)$ be the set of all p-constrained functions $l : V(G) \to V(H)$ satisfying the following two conditions:

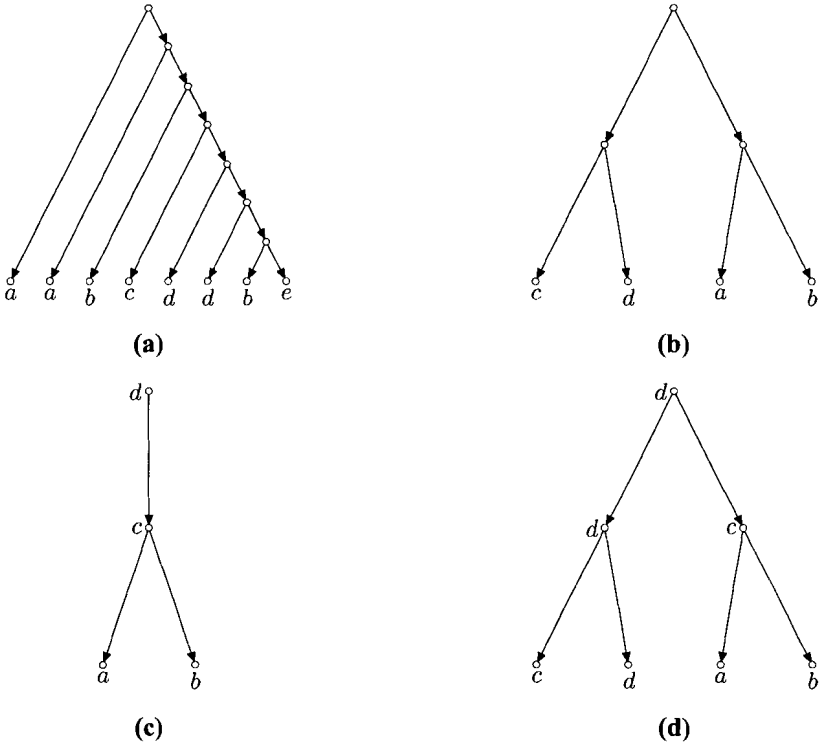


Figure 3.2: Differences between scattering (defined by Lipscomb) and separation. (a) is Lipscomb's example of scattering where two species in state b are not adjacent to each other. With (b) as (G, p) and (c) as H, we show that separation occurs in (c) which is the (G, p, l), although Lipscomb considers (b) without scattering.

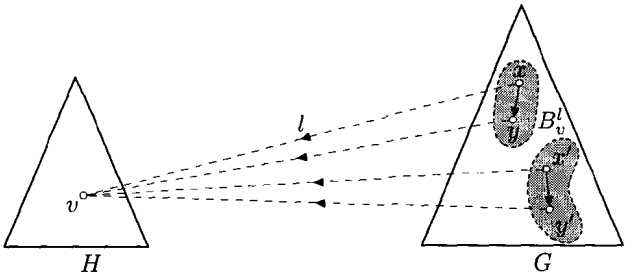


Figure 3.3: Dash lines from vertices $x, y, x', y' \in V(G)$ to vertex $v \in V(H)$ illustrate the function l. The bag-set B_v^l consists of two shadowed areas in G, i.e. $c(B_v^l) = 2$. Every vertex in the bag-set B_v^l has the same image v in H.

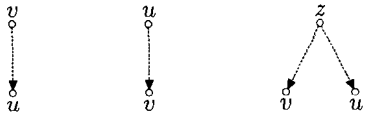


Figure 3.4: Three possible relationships between two different vertices v and u in the same bag.

(1) For each vertex $a \in V(H)$, we have $c(B_a^l) = 1$.

(2) For each arc $\langle a, b\rangle \in A(H), r(\langle a, b\rangle, l) \geq 1$.

We say that H is a *rooted-minor* of (G, p), denoted by $H \leq_{rm} (G, p)$, if $M(H, G, p) \neq \emptyset$.

Figure 3.5 shows an example of $l \in M(H, G, p)$ where $p \neq \varepsilon$.

It is not hard to see that the corresponding decision problem–*Rooted minor problem*–is intractable when $p = \epsilon$ (the details of the proof are in Chapter 4). Conversely, the rooted minor problem is in polynomial time if $p \neq \epsilon$ (the details of the algorithm are in Chapter 5). Now let us consider the small phylogeny problem where H is the character tree and (G, p) is the phylogenetic tree. We will verify that none of the five inconsistencies will occur when $H \leq_{rm} (G, p)$, i.e. $M(H, G, p) \neq \emptyset$.

Lemma 1. *Given two rooted trees H and (G, p). For any $l \in M(H, G, p)$ and for any two different vertices $a, b \in V(H)$ with $a \prec b$, there exists two vertices $u, v \in V(G)$ such that $l(u) = a, l(v) = b$ and there is a directed path of length $\geq d(a, b)$ from u to v.*

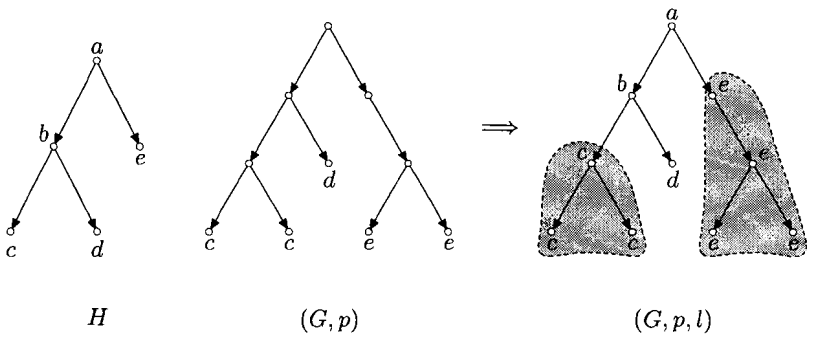


Figure 3.5: An example of $l \in M(H, G, p)$.

Proof. If $\langle a, b\rangle \in A(H)$, this follows from (2) of Definition 6 and we have $d(u, v) = d(a, b)$.

Thus, suppose $\langle a, b\rangle \notin A(H)$. Then there must be a directed path from a to b in H, say $(x_1, x_2, x_3, \ldots, x_n)$ with $x_1 = a$ and $x_n = b$. According to (2) of Definition 6, for every arc $\langle x_i, x_{i+1}\rangle \in A(H)$ $(i = 1, \cdots, n-1)$, there exists an arc $\langle u_i, v_i\rangle \in A(G)$ such that $l(u_i) = x_i$ and $l(v_i) = x_{i+1}$. In order to find a path from u_1 to v_{n-1}, it is enough to find a path from v_i to u_{i+1} for each $i = 1, 2, \cdots, n-2$.

Since $l(v_i) = l(u_{i+1}) = x_{i+1}$, according to (1) of Definition 6, v_i and u_{i+1} belong to the same bag $B^l_{x_{i+1}}$. If $v_i = u_{i+1}$, then there is a trivial path of length 0 joining v_i and u_{i+1}. Otherwise, either $v_i \prec u_{i+1}$ or $u_{i+1} \prec v_i$ or $v_i \nsim u_{i+1}$; see Figure 3.4. Since G is a rooted tree, i.e. every vertex in G has only one parent, and since u_i is the parent of v_i, the only possibility remaining is $v_i \prec u_{i+1}$. It follows that there must be a path from v_i to u_{i+1} in G. Thus, a path from u_1 to v_{n-1} can be constructed by going through the arc $\langle u_i, v_i\rangle$ to v_i, following the path $v_i \rightarrow u_{i+1}$ to u_{i+1} for each $i = 1, 2, \cdots, n-2$, until we reach the arc $\langle u_{n-1}, v_{n-1}\rangle$. See Figure 3.6 for an example of $n = 4$. Furthermore, the length of the path $u_1 \rightarrow v_{n-1}$ is at least $d(a, b)$ which is attained when $v_i = u_{i+1}$ for all $i = 1, 2, \cdots, n-2$.

□

Theorem 1. *Given two rooted trees H and (G, p). If $H \leq_{rm} (G, p)$, then none of the five inconsistencies will occur for any $l \in M(H, G, p)$.*

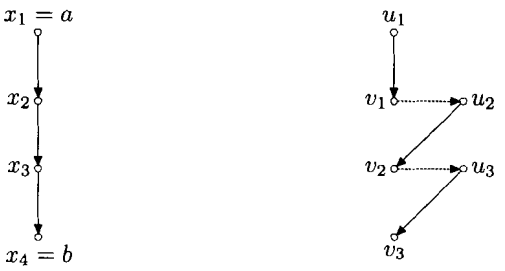


Figure 3.6: An example of $a, b \in V(H)$ with $a \prec b$ has a corresponding path $u_1 \rightarrow v_3$ in G with $l(u_1) = a$ and $l(v_3) = b$.

Proof. Since any $l \in M(H, G, p)$ satisfies (1) and (2) of Definition 6, both separation and negligence cannot occur. Next we prove that the theorem is also true for the remaining three inconsistencies.

- If transitivity occurs, there must exist two vertices $a, b \in V(H)$ so that $a \prec b$, $\langle a, b\rangle \notin A(H)$, and for some $\langle u, v\rangle \in A(G)$, $l(u) = a$ and $l(v) = b$. On the other hand, according to Lemma 1, there exists two vertices u' and v' such that $l(u') = a$, $l(v') = b$, and there is a path $P = u' \rightarrow v'$ of length $\geq d(a, b) > 1$ from u' to v'; see Figure 3.7. Since v and v' are in the same bag B_b^l, if $v \neq v'$, then either $v \prec v'$ or $v' \prec v$ or $v \nsim v'$. Since u is the parent of v, we can exclude the cases $v' \prec v$, and $v \nsim v'$, because they both imply the existence of a parent($\neq u$) of v. Similarly v' has a parent on the path P and hence we cannot have $v \prec v'$ as well. Therefore $v = v'$.

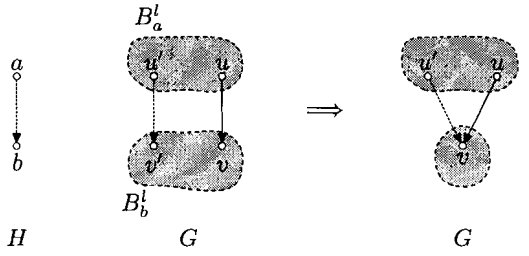


Figure 3.7: Transitivity cannot occur for any $l \in M(H, G, p)$.

Furthermore, u and u' are also in the same bag B_a^l. If $u = u'$, then there exist two paths

from u to v in G, one is the arc $\langle u, v\rangle$, and the other is the path $u' \rightarrow v$ of length > 1. This contradicts the fact that G is a tree. Therefore we suppose $u \neq u'$. If $u \prec u'$, then there are two paths joining u and v; if $u' \prec u$, then there are two paths joining u' and v; and if $u' \nsim u$, then there are two paths joining $\mathrm{LCA}(u, u')$ and v. Therefore transitivity can not occur for any $l \in M(H, G, p)$.

- If addition occurs, there must exist two vertices $a, b \in V(H)$ so that $a \nsim b$, and for some $u, v \in V(G)$ with $l(u) = a$ and $l(v) = b$ either $u \prec v$ or $v \prec u$. Without loss of generality, we assume that $u \prec v$. Let $c = LCA(a, b)$. Applying Lemma 1 to vertices c and a, there exist two vertices w and u' such that $l(w) = c$, $l(u') = a$, and there is a path $w \rightarrow u'$ from w to u'. Similarly applying Lemma 1 to vertices c and b, there exist two vertices w' and v' such that $l(w') = c$, $l(v') = b$, and there is a path $w' \rightarrow v'$ from w' to v'; see Figure 3.8.

  Clearly, u and u' are in the same bag B_a^l. If $u \neq u'$, since u' has a parent on the path $w \rightarrow u'$, we can exclude the cases $u' \prec u$, and $u' \nsim u$. Thus $u \prec u'$ if $u \neq u'$. v and v' are also in the same bag B_b^l. If $v \neq v'$, since v has a parent on the path $u \rightarrow v$ and v' has a parent on the path $w' \rightarrow v'$, any of the three relationships between v and v' is impossible. Therefore $v = v'$. With respect to w and w', which are also in the same bag B_c^l, both $w = w'$ and $w \neq w'$ are possible. If $w = w'$ or $w \prec w'$, then there are two paths joining w and v; if $w' \prec w$, then there are two paths joining w' and v; and if $w \nsim w$, then there are two paths joining v and $\mathrm{LCA}(w, w')$. This contradicts the fact that G is a tree and so addition cannot occur for any $l \in M(H, G, p)$.

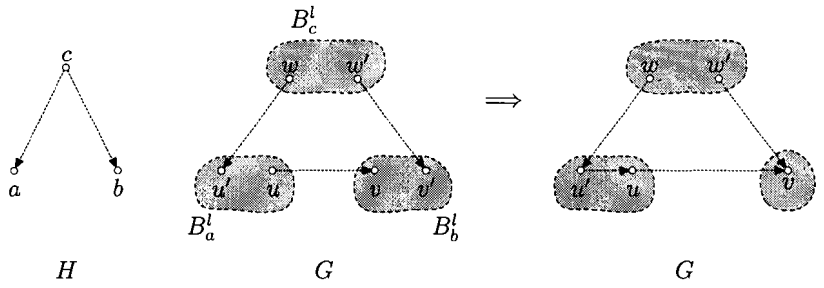


Figure 3.8: Addition cannot occur for any $l \in M(H, G, p)$.

- If inversion occurs, there must exist two vertices $a, b \in V(H)$ so that $a \prec b$, and for some $u, v \in V(G)$ with $l(u) = a$ and $l(v) = b$, and there is a path $v \rightarrow u$. On the other hand, by

Lemma 1, there exist two vertices u' and v' such that $l(u') = a$, $l(v') = b$, and there is a path $u' \rightarrow v'$ from u' to v'; see Figure 3.9. Since v and v' are in the same bag B_b^l, if $v \neq v'$, then $v' \prec v$ because v' has a parent on the path $u' \rightarrow v'$. u and u' are also in the same bag B_a^l, if $u \neq u'$, then $u \prec u'$ because u has a parent on the path $v \rightarrow u$. However, irrespective of whether $u = u'$ or $u \prec u'$, we can always find a loop going through u, u', v' and v. Therefore inversion cannot occur for any $l \in M(H, G, p)$.

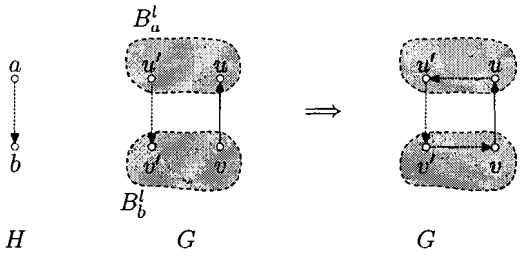


Figure 3.9: Inversion cannot occur for any $l \in M(H, G, p)$.

Thus we exclude the possibility for any five inconsistencies to occur for any $l \in M(H, G, p)$. □

However in practice, some of the five inconsistencies may happen which means $H \not\leq_{rm} (G, p)$. Hence we consider relaxations of rooted-minors in the following way.

Definition 7. Given two rooted trees H and (G, p), let $R(H, G, p)$ be the set of all p-constrained functions $l : V(G) \rightarrow V(H)$ satisfying the following two conditions:

(1) For each arc $\langle a, b\rangle \in A(H)$, $r(\langle a, b\rangle, l) \geq 1$.

(2) If for some $u, v \in V(G)$ $u \prec v$, then $l(v) \not\prec l(u)$ in H.

We say that H is a *relax-minor* of (G, p) if $R(H, G, p) \neq \emptyset$.

See Figure 3.10 for a specific example of relax-minor where $\left|l^{-1}(a)\right| = 2, \left|l^{-1}(b)\right| = 2$.

Note 1. Given two rooted trees H and (G, p), any $l \in R(H, G, p)$ is surjective and maps the root of G to the root of H, i.e. $l(g) = h$. (g and h are used to denote the roots of G and H respectively hereafter.) Furthermore, for every $v \in V(H)$ $c(B_v^l) \geq 1$. If there exists an $l \in R(H, G, p)$ such that $c(B_v^l) = 1$ for every $v \in V(H)$, then $H \leq_{rm} G$.

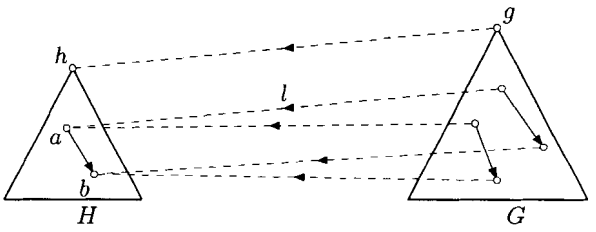


Figure 3.10: Relax-minor

In terms of the inconsistency between H and (G,p), it is not difficult to see that (1) and (2) of Definition 7 prevent negligence and inversion. However, transitivity, addition, and separation may occur. Figure 3.11 shows an example of an H which is a relax-minor of (G,p), but there are two separations (because $c(B_b^l) = 2$ and $c(B_d^l) = 2$), a transitivity on the arc $\langle v_5, v_2\rangle$ (because $l(v_5) \prec l(v_2)$ and $\langle l(v_5), l(v_2)\rangle \notin A(H)$), and an addition on the arc $\langle v_6, v_3\rangle \in A(G)$ (because $v_6 \prec v_3$ and $l(v_5) \nsim l(v_2)$).

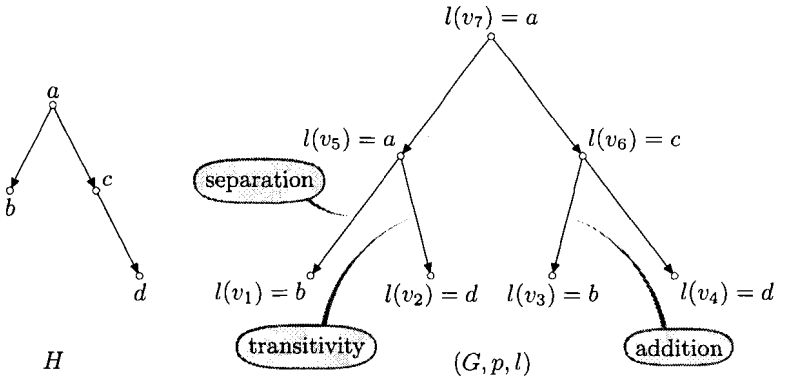


Figure 3.11: H is a relax-minor of (G,p) in which addition, transitivity and separation occur.

However, the problem of deciding whether H is a relax-minor of G remains intractable in both cases, $p = \epsilon$ and $p \neq \epsilon$; see Chapter 4.

Therefore we will consider another relaxation of the concept of rooted-minor, called pseudo-minor.

Definition 8. Let H and G be directed trees. A function $l : V(G) \to V(H)$ is *smooth* if for every

arc $\langle u, v\rangle \in A(G)$, there is a directed path from $l(u)$ to $l(v)$ in H; see Figure 3.12. Note that a single vertex is considered as a directed path of length 0.

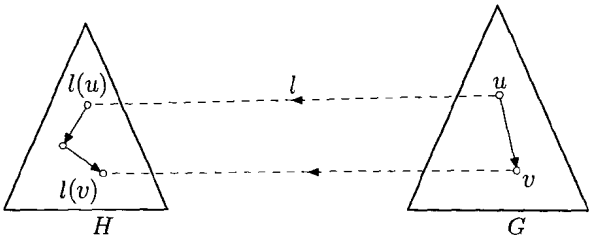


Figure 3.12: Smooth function

Definition 9. Given two rooted trees H and (G,p), let $Q(H,G,p)$ be the set of all smooth p-constrained functions $l : V(G) \rightarrow V(H)$. We say that H is a *pseudo-minor* of G if $Q(H,G,p) \neq \emptyset$.

Since l is a smooth function, (2) of Definition 7 is satisfied; hence pseudo-minor extends the notion of relax-minor. Note that rooted-minor implies both relax-minor and pseudo-minor, but relax-minor and pseudo-minor are incomparable; see Figure 3.13, 3.14 and 3.15 for examples.

Similarly, it is easy to see that the concept of smooth function prevents inversion and addition when H is a pseudo-minor of (G,p), but transitivity, separation, and negligence may occur. Figure 3.16 shows an example of an H which is a pseudo-minor of (G,p), but there is a separation (because $c(B_d^l) = 2$), a transitivity on the arc $\langle v_5, v_3\rangle \in A(G)$ (because $l(v_5) \prec l(v_3)$ and $\langle l(v_5), l(v_3)\rangle \notin A(H)$), and a negligence on the arc $\langle a, b\rangle \in A(H)$ (because there is no arc $\langle u, v\rangle in A(G)$ such that $l(u) = a$ and $l(v) = b$).

If $p = \varepsilon$, then it is not difficult to see that any H is a pseudo-minor of any G (just map every vertex of G to the root of H). If $p \neq \varepsilon$, then the problem of deciding whether H is a pseudo-minor of G can be solved in polynomial time. In our approach to small phylogeny problem, we define two natural metrics and will search for feasible labellings that minimize these two metrics.

The following table summarizes the occurrence of the five inconsistencies in rooted-minor, relax-minor, and pseudo-minor respectively.

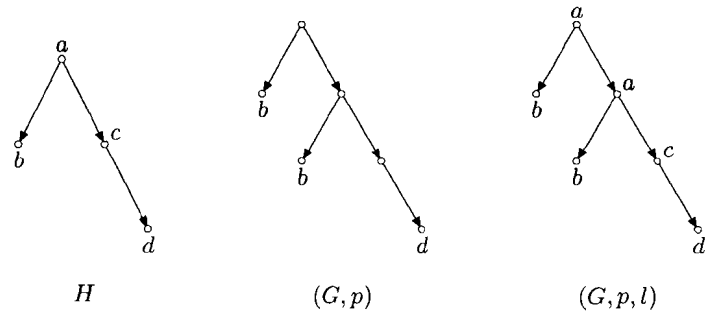


Figure 3.13: H is a relax-minor of (G,p), but not rooted-minor since the bag number of $b \in v(H)$ must > 1.

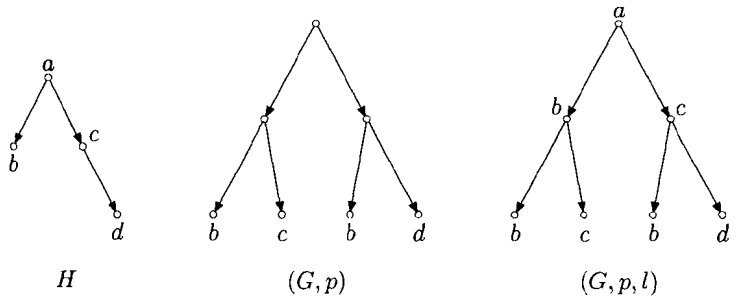


Figure 3.14: H is a relax-minor of (G,p), but not pseudo-minor since $c \nsim b$ in H.

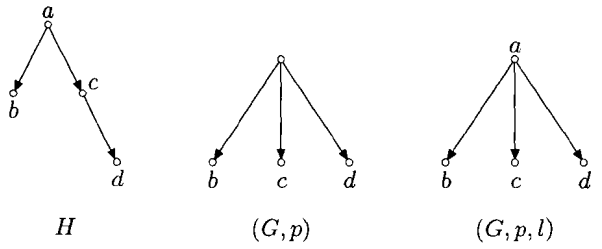


Figure 3.15: H is a pseudo-minor of (G,p), but neither rooted-minor nor relax-minor since it's impossible for any labeling function l to have both $r(\langle a,c\rangle,l) = 1$ and $r(\langle c,d\rangle,l) = 1$.

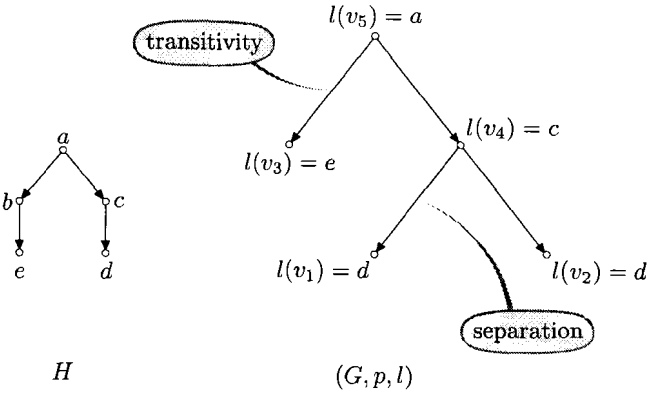


Figure 3.16: H is a pseudo-minor of (G, p) in which transitivity, separation and negligence occur.

	rooted-minor	*relax-minor*	*pseudo-minor*
inversion	N	N	N
addition	N	Y	N
transitivity	N	Y	Y
separation	N	Y	Y
negligence	N	N	Y

Table 3.1: Properties of *rooted-minor*, *relax-minor* and *pseudo-minor*

3.2 Two metrics for parsimony with general character evolution

The two standard parsimony criteria for measuring the quality of l correspond to the unweighted and the weighted cost. The unweighted parsimony assumes a constant cost for every state change, while the weighted parsimony treats the different state changes differently by taking the cost of each state change into consideration. In our approach, we define two metrics to reflect these two criteria, bag cost for unweighted and arc cost for weighted.

Definition 10. Given two rooted trees H and G together with a labeling $l : V(G) \to V(H)$, the *bag cost* of l is $\mathrm{bcost}(H, G, l) := \sum_{v \in V(H)} c(B_v^l)$.

Definition 11. Given two rooted trees H and G together with a labeling $l : V(G) \to V(H)$, the *arc cost* of l is $\mathrm{acost}(H, G, l) := \sum_{\langle u,v \rangle \in A(G)} d(l(u), l(v))$.

It is not hard to see that bag cost expresses the number of state changes, i.e. the number of state changes is the bag cost minus 1, and the arc cost weights each state change by the distance between the two states. Furthermore, bag cost and arc cost are also measures of scattering and hierarchical discordance, since a bag cost bigger than $|V(H)|$ implies the occurrence of scattering, and an arc cost bigger than $|A(H)|$ indicates the occurrence of hierarchical discordance. Therefore, we will measure the quality of relax-minor and pseudo-minor mappings in terms of their bag cost and arc cost, respectively. For this purpose, we define the three problems. Given a character tree H and a phylogenetic tree (G, p).

Problem 1. *Minimum relax-minor bag cost.* Find a labeling $l \in R(H, G, p)$ such that the bag cost of l is $\mathrm{rbcost}(H, G, p) := \min\{\mathrm{bcost}(H, G, l') \mid l' \in R(H, G, p)\}$.

Problem 2. *Minimum pseudo-minor bag cost.* Find a labeling $l \in Q(H, G, p)$ such that the bag cost of l is $\mathrm{qbcost}(H, G, p) := \min\{\mathrm{bcost}(H, G, l') \mid l' \in Q(H, G, p)\}$.

Problem 3. *Minimum pseudo-minor arc cost.* Find a labeling $l \in Q(H, G, p)$ such that the arc cost of l is $\mathrm{qacost}(H, G, p) := \min\{\mathrm{acost}(H, G, l') \mid l' \in Q(H, G, p)\}$.

Note that for the relax-minor mapping the distance function d might be unbounded and therefore the arc cost measure makes sense only for pseudo-minor mappings.

In the remainder of this thesis, we study the complexity of these three problems. We show that Problem 1 is **NP**-complete, and both Problem 2 and Problem 3 can be solved in polynomial time.

Chapter 4

Complexity Results

Theorem 2. *[MR92] Given two unrooted tree H and G. It is* **NP***-complete to decide whether H is minor of G . We call this the unrooted tree minor problem*

Theorem 3. *Given two rooted trees H and (G,p). It is* **NP***-complete to decide whether H is a rooted-minor of (G,p) when $p = \varepsilon$. We call this the rooted tree minor problem.*

Proof. We show that the unrooted tree minor problem can be reduced to the rooted tree minor problem.

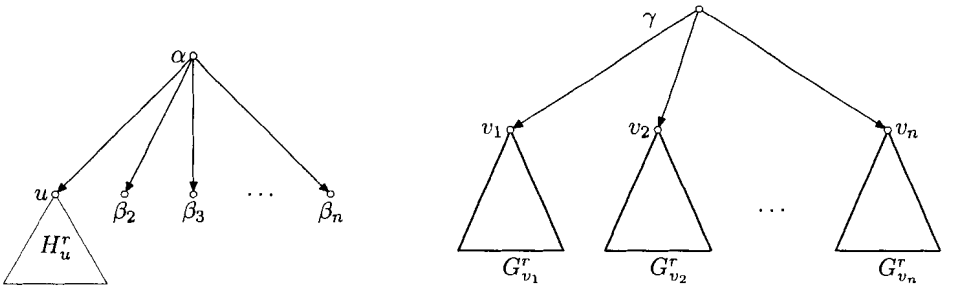
Let H and G be an instance of unrooted tree minor problem. We construct new rooted trees H^r and G^r as follows:

Let H^r_u be a rooted tree obtained from H by choosing an arbitraty vertex $u \in V(H)$ to be the root of H^r_u. For each vertex v_i of G ($i = 1, 2, \cdots, |V(G)|$), let $G^r_{v_i}$ be a rooted tree obtained from G by choosing v_i to be the root of $G^r_{v_i}$.

Define H^r to be a rooted tree with root α (α differs from any vertex of H or G) where α has children $u, \beta_2, \cdots, \beta_n$, and H^r_u is the sub-tree rooted at u; see Figure 4.1 (a). Define G^r to be a rooted tree with root γ (γ differs from any vertex of H or G) where γ has children $v_1, v_2, \cdots, v_n$ ($n = |V(G)|$), and $G^r_{v_i}$ is the sub-tree rooted at v_i for $i = 1, \cdots, n$; see Figure 4.1 (b).

Claim 1. *H is a minor of G iff H^r is a rooted-minor of G^r.*

Proof. If H is a minor of G, suppose $v_i \in V(G)$ ($1 \leq i \leq n$) is in the bag-set of $u \in V(H)$, then H^r_u is a rooted-minor of $G^r_{v_i}$. Without loss of generality, assume $i = 1$. Now let the bag-set of α be $\{\gamma\}$., and the bag-set of $\beta_2, \cdots, \beta_n$ be $\{G^r_{v_2}\}, \cdots, \{G^r_{v_n}\}$ respectively. Thus, just defined mapping shows that H^r is a rooted-minor of G^r.



(a) H^r **(b)** G^r

Figure 4.1: Construct rooted trees H^r and G^r from unrooted trees H and G.

On the other hand, if H^r is a rooted-minor of G^r, consider the bag-set B_1 of α and the bag-set B_2 of u. Because α and γ are degree n vertices, B_2 must be contained in $G^r_{v_i}$ for some i ($1 \leq i \leq n$). Then H^r_u is a rooted-minor of $G^r_{v_i}$ and it follows that H is a minor of G. □

Now, Claim 1 together with Theorem 2 proves the **NP**-completeness of the rooted tree minor problem. □

Theorem 4. *It is* **NP***-hard to solve Problem 1 when* $p = \varepsilon$.

Proof. The proof is based on a reduction from the rooted tree minor problem. Suppose there is a polynomial-time algorithm A that can find l such that $bcost(H, G, l) = rbcost(H, G, p)$. We can decide whether $H \leq_{rm} G$ simply by comparing $\mathrm{bcost}(H, G, l)$ with $|V(H)|$.

If they are equal, according to (1) of Definition 7, we have $c(B^l_a) = 1$ for every $a \in V(H)$. This implies $l \in M(H, G, p)$, and therefore $H \leq_{rm} G$.

If $\mathrm{bcost}(H, G, l) > |V(H)|$, then H cannot be a rooted-minor of G, since $H \leq_{rm} G$ contradicts the assumption that $\mathrm{bcost}(H, G, l) > |V(H)|$. Thus, $H \leq_{rm} G$ if and only if $\mathrm{bcost}(H, G, l) = |V(H)|$. According to Theorem 3, it is **NP**-complete to decide whether $H \leq_{rm} G$ when $p = \varepsilon$, so it must be **NP**-hard to find a labeling $l \in R(H, G, p)$ such that $\mathrm{bcost}(H, G, l) = \mathrm{rbcost}(H, G, p)$ when $p = \varepsilon$. □

Theorem 5. *It is* **NP***-hard to solve Problem 1 when* $p \neq \varepsilon$.

Proof. The proof is also based on the reduction from the rooted tree minor problem. However since an input to Problem 1 has $p \neq \varepsilon$ in this theorem, an input to rooted tree minor problem cannot be directly used as an input to Problem 1 as we did in the proof of Theorem 4. Therefore we need the

following input construction process. Assume that two rooted trees Y and (Z, q) (with $q = \varepsilon$) are given as an input to the rooted tree minor problem. We construct the corresponding input, H and (G, p), to the Problem 1 as follows.

Initially, we set G to be a copy of Z, i.e. $G := Z$, and then for each leaf $v_i \in L(Z)$, we add a pair of two new vertices v_i' and v_i'' to G with v_i' joined to v_i and v_i'' joined to the root g. Thus, $|V(G)| = |V(Z)| + 2 \cdot |L(Z)|$. Similarly we set H to be a copy of Y, i.e. $H := G$, and then for each leaf $v_i \in L(Z)$, we add a new vertex w_i to H with w_i joined to the root h. Thus, $|V(H)| = |V(Y)| + |L(Z)|$. Finally, we define the prelabels p of leaves in G as $p(v_i') := p(v_i'') := w_i$.

So far, we defined H and (G, p) from Y and Z; see Figure 4.2.

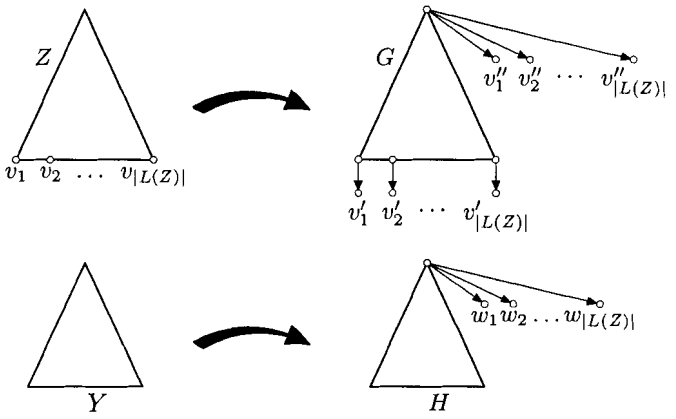


Figure 4.2: Construction H and G from Y and Z

Next, we will prove the following two claims dealing with properties of H and G.

Claim 2. *For every* $l \in R(H, G, p)$, $\mathrm{bcost}(H, G, l) \geq |V(Y)| + 2 \cdot |L(Z)|$

Proof. Since $p(v_i') := p(v_i'') := w_i$, and since v_i', v_i'' are separated by the root g which has to be labeled as the root of H according to Note 1, it is not hard to see that for any feasible l, $c(B^l_{w_i}) \geq 2$ for each w_i, $i = 1, \cdots, |L(Z)|$. Moreover, $c(B^l_a) \geq 1$ for any other vertex $a \in V(H)$ according to (1) of Definition 7. The number of such vertices is $|V(Y)|$.

Therefore we have that $\mathrm{bcost}(H, G, l) \geq |V(Y)| + 2 \cdot |L(Z)|$. □

Claim 3. *There exists a labeling* $l \in R(H, G, p)$ *such that* $\mathrm{bcost}(H, G, l) = |V(Y)| + 2|L(Z)|$ *if and only if* $Y \leq_{rm} Z$.

Proof. We first suppose $Y \leq_{rm} Z$. Then there exists a labeling $l' : V(Z) \rightarrow V(Y)$ satisfying the following three properties:

1. $\forall a \in V(Y), c(B_a^{l'}) = 1$,
2. For each arc $(a, b\rangle \in A(H), r((a, b\rangle, l) \geq 1$.
3. If there is a path from u to v in Z, then there is no path from $l'(v)$ to $l'(u)$ in Y.

To finish the proof in this direction, we construct a labeling $l : V(G) \rightarrow V(H) \in R(H, G, p)$ such that $\mathrm{bcost}(H, G, l) = |V(Y)| + 2\,|L(Z)|$. We define l as follows.

For each vertex $v \in V(G)$ that is also originally in Z, let $l(v) := l'(v)$, and for each pair of v_i' and v_i'' $(i = 1, 2, \ldots, |L(Z)|)$, let $l(v_i') := l(v_i'') := w_i$.

Therefore, $c(B_a^l) = 1$, for each vertex $a \in V(H)$ that is also originally in Y, because of the above property 1; and $c(B_a^l) = 2$, for each vertex $a \in V(H)$ that is not originally in Y (these vertices are essentially all the w_i).

Moreover, by the above property 2 and 3, we also have $l \in R(H, G, p)$. So we have $\mathrm{bcost}(H, G, l) = |V(Y)| + 2\,|L(Z)|$, as required.

We second suppose that for some $l \in R(H, G)$, $\mathrm{bcost}(H, G, l) = |V(Y)| + 2\,|L(Z)|$. Based on the proof of Claim 2, for any such l, $c(B_a^l) \geq 1$, for each vertex $a \in V(H)$ that is also originally in Y, and the total number of such vertices is $|V(Y)|$; $c(B_a^l) \geq 2$, for each vertex $a \in V(H)$ that is not originally in Y, and the total number of such vertices is $|L(Z)|$. Therefore for our labeling l, $c(B_a^l) = 1$, for each vertex $a \in V(H)$ that is also originally in Y; $c(B_a^l) = 2$, for each vertex $a \in V(H)$ that is not originally in Y.

Hence, we conclude that there is exactly one bag for each vertex $a \in V(H)$ that is also originally in Y, and therefore $Y \leq_{rm} Z$. Indeed, we can define $l' : V(Z) \rightarrow V(Y)$ as $l'(v) := l(v)$ for each vertex $a \in V(Y)$, and such labeling l' satisfies the above three properties. The claim is proved. □

Finally with respect to Claim 3, we can prove the **NP**-hardness of Problem 1 by contradiction. Given Y and (Z, q) (with $q = \varepsilon$) as an input of the rooted tree minor problem, we construct the corresponding input H and (G, p) of Problem 1 using the steps listed in Theorem 5. Suppose there exists a polynomial algorithm A that can find a labeling $l \in R(H, G, p)$ with $\mathrm{bcost}(H, G, l) = \mathrm{rbcost}(H, G, p)$. Then we can decide if $Y \leq_{rm} Z$ simply by comparing $\mathrm{bcost}(H, G, l)$ with $|V(Y)| + 2\,|L(Z)|$. If they are equal, then $Y \leq_{rm} Z$, otherwise Y cannot be the rooted-minor of Z. However according to Theorem 3, it is *NP-complete* to decide whether $Y \leq_{rm} Z$, so such an algorithm A cannot exist.

□

Chapter 5

Algorithms

5.1 Minimum pseudo-minor arc cost

The following is the algorithm for Problem 3.

Input: A character tree H and a phylogenetic tree (G, p).

Output: A labeling $l \in Q(H, G, p)$ such that $\text{acost}(H, G, l) = \text{qacost}(H, G, p)$.

Description: As an initialization, the algorithm assigns $l(u) := p(u)$ for every $u \in L(G)$. Then it traverses the tree G in *post-order* to assign $l(u) := \text{LCA}(\{l(v) | \langle u, v\rangle \in A(G)\})$ for every internal vertex $u \in V(G)$. Note that $l(u)$ is well defined, since all children of u already have their l value defined before u is visited. Computation of LCA in the tree H can be done in constant time after a preprocessing on H which also takes linear time, see [HT84]. The preprocessing algorithm can easily be modified so that even computation of $d(a, b)$ takes constant time for any given $a, b \in V(H)$ with $a \prec b$. This tree preprocessing is performed on H in Line 1 of Algorithm 5.1.

The cost of each arc $\langle u, v\rangle$ is calculated as the distance between $l(u)$ and $l(v)$. Finally the total arc cost of l is the sum of the cost of every arc in G.

Pseudo code:

Algorithm 5.1 Minimum-Pseudo-Minor-Arc-Cost(H,G,p)

```
1  preprocess H (build auxiliary trees) to speed up the computation of LCA ;
   ▷ initialization
2  set e := 0;
3  for each vertex u ∈ V(G) do
4      if u is a leaf then
5          set l(u) := p(u);
```

6 **else**
7 set $l(u) := null$;
▷ *assigning labels and calculating arc cost*
8 traverse the tree G in *post-order*, **for** each internal vertex $u \in V(G)$, **do**
9 set $l(u) := \mathrm{LCA}(l(v)|\langle u, v\rangle \in A(G))$;
10 **for** each child v of u
11 set $e := e + d(l(u), l(v))$;
▷ *output result*
12 output labeling function l and cost e;

Complexity: The preprocessing on H takes $O(|V(H)|)$ time. After the tree preprocessing, it is possible to determine both LCA and the distances of any pair of nodes in constant time. Computing the LCA of δ nodes can be reduced to successive $O(\log_2 \delta)$ iterations of the LCA for two nodes. Therefore, for each internal vertex $u \in V(G)$, it takes $\log_2 \delta$ time to compute $l(u)$. The total running time for the labeling is $O(|V(G)| \cdot \log_2 \delta + |V(H)|)$. In particular if G is a binary tree, then the time complexity is $O(|V(G)| + |V(H)|)$.

5.1.1 Proof of correctness

Lemma 2. *For any labeling $l' \in Q(H, G, p)$, and any internal vertex $u \in V(G)$, either $l'(u) = \mathrm{LCA}(p(v)|v \in L(G_u))$ or $l'(u) \prec \mathrm{LCA}(p(v)|v \in L(G_u))$.*

Proof. Let $X := \{p(v)|v \in L(G_u)\}$ and $q := \mathrm{LCA}(X)$. We will prove the lemma by excluding the cases of $q \prec l'(u)$ and $q \nsim l'(u)$.

Suppose $q \prec l'(u)$. Since q is the *least common ancestor* of X, there must exist an $s \in X$ such that either $s \prec l'(u)$ or $s \nsim l'(u)$, otherwise q would not be the least common ancestor of X. Let v be the leaf with $p(v) = s$. Either $s \prec l'(u)$ or $s \nsim l'(u)$, there is at least one arc $\langle x, y\rangle$ on the path $u \rightarrow v$ such that $d(l'(x), l'(y)) = \infty$; see Figure 5.1 (a). It contradicts the definition of pseudo-minor, therefore $l'(u) \notin Q(H, G, p)$.

Suppose $q \nsim l'(u)$. $\forall s \in X$, if $s \prec l'(u)$, the $q \prec l'(u)$ since either $q \prec s$ or $q = s$, it contradicts the assumption that $q \nsim l'(u)$; on the contrary, if $l'(u) \prec s$, then there are two paths from z to s where $z = \mathrm{LCA}(q, l'(u))$; one is through $l'(u)$, and the other is through q since $l(u) \prec s$. This contradicts the fact that H is a tree; see Figure 5.1 (b). Therefore the only possibility is that $s \nsim l'(u)\ \forall s \in X$. Again it causes at least one arc $\langle x, y\rangle$ on the path $u \rightarrow v$ ($p(v) = s$) such that $d(l'(x), l'(y)) = \infty$, so $l'(u) \notin Q(H, G, p)$ if $q \nsim l'(u)$.

Thus any other labeling $l' \in Q(H, G, p)$ must satisfy either $l'(u) = q$ or $l'(u) \prec q$ for any

Figure 5.1: (a): the arc cost of the path $u \rightarrow v$ in G_u is infinite; (b): There exist two paths from $Z = \mathrm{LCA}(l(u), l'(u))$ to s.

internal vertex $u \in V(G)$. □

Since for the labeling l returned by Algorithm 5.1, $l(u) = \mathrm{LCA}(\{l(v)|\langle u, v\rangle \in A(G)\}) = \mathrm{LCA}(p(v)|v \in L(G_u))$, we have the following corollary.

Corollary 1. *For any other labeling $l' \in Q(H, G, p)$, and any internal vertex $u \in V(G)$, $l'(u) = l(u)$ or $l'(u) \prec l(u)$ where l is the labeling returned by Algorithm 5.1.*

Theorem 6. *Algorithm 5.1 returns a unique labeling $l \in Q(H, G, p)$ such that* $\mathrm{acost}(H, G, l) = \mathrm{qacost}(H, G, p)$.

Proof. There is a directed path from $l(u)$ to $l(v)$ in H for each arc $\langle u, v\rangle \in A(G)$ because $l(u)$ is the least common ancestor of the labels of its children, i.e. either $l(u) \prec l(v)$ or $l(u) = l(v)$. In other words, l is a smooth function. Moreover, l is obviously *p-constrained* according to the initialization of the algorithm. Therefore $l \in Q(H, G, p)$.

We prove $\mathrm{acost}(H, G, l) = \mathrm{qacost}(H, G, p)$ and the uniqueness of l by induction on the height t of tree G.

In case $t = 1$, the theorem is obvious, because there is only one vertex, say u, and for any other labeling $l' \in Q(H, G, p)$, $l'(u) = l(u) = p(u)$. The arc cost of such labeling is 0. Suppose that the theorem is true for all trees of height $t < n$. We prove it is also true for trees of height $t = n$.

Consider a rooted tree G with height $t = n$. Let $v_1, v_2, .., v_k$ be the children of the root g of G. We decompose G into k sub-tree; see Figure 5.2. Let G_i, $i = 1, 2, \ldots, k$, be the sub-tree of G rooted at the vertex v_i. Let p_i be the prelabeling of leaves of G_i that is compatible with the prelabeling p of G, i.e. for all $i = 1, 2, \ldots, k$, we set $p_i(v) = p(v)$ for all $v \in L(G_i)$.

Now we run Algorithm 5.1 on (H, G, p). Let l and c be the labeling function and arc cost returned by the algorithm. Next we run Algorithm 5.1 on (H, G_i, p_i). Let c_i be the returned arc

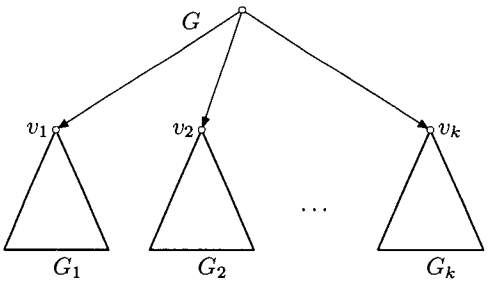


Figure 5.2: Decompose tree G into k sub-tree $G_1, G_2, \cdots, G_k$.

cost. Note that every internal vertex $v \in V(G_i)$ will be assigned the same label as the label it gets when running Algorithm 5.1 on (H, G, p) because the children vertices of v are not changed and so does the *least common ancestor* of their labels.

It is easy to observe that $c = \sum_i (c_i + d(l(g), l(v_i)))$. By induction we have $c_i = \text{qacost}(H, G_i, p_i)$. We will prove that c is the minimum cost by proving that any other labeling $l' \in Q(H, G, p)$ will produce more cost.

Let $c' = \text{acost}(H, G, l')$, $c'_i = \text{acost}(H, G_i, l'|V(G_i))$. It is also true that $c' = \sum_i (c'_i + d(l'(g), l'(v_i))$.

Now for each $i = 1, 2, \ldots, k$, let G_i^w be the tree obtained from G_i by joining a new leaf w_i to the root v_i of G_i. Let p_i^w denote the leaf labeling of G_i^w, where $p_i^w(v) := p(v)$ for all leaves $v \in G_i$ except that $p_i^w(w_i) := l'(v_i)$. In Figure 5.3, G_i^w on the right side has the labeling compatible to l' while the left one has the labeling returned by Algorithm 5.1.

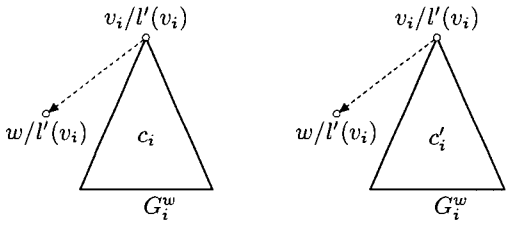


Figure 5.3: G_i^w

Note that every internal vertex $u \in V(G_i^w)$ except v_i will get the same label by running Algorithm 5.1 on (H, G_i^w, p_i^w) and (H, G_i, p_i) because the children vertices of u are not changed and so

Figure 5.4: (a) shows $d(l'(v_i), l(v_i)) = d(l'(g), l(v_i)) - d(l'(g), l'(v_i))$ and (b) shows $d(l'(g), l(v_i)) \geq d(l(g), l(v_i))$

does the least common ancestor of their labels. As for v_i, its label is $l'(v_i)$ in G_i^w while it is $l(v_i)$ in G_i. It is because G_i^w has an extra leaf w_i with $p_i^w(w_i) = l'(v_i)$, and either $l'(v_i) \prec l(v_i)$ or $l'(v_i) = l(v_i)$ by Corollary 1.

Let c_i^w be the arc cost returned by Algorithm 5.1 on input (H, G_i^w, p_i^w). If v_i is a leaf, then $c_i^w = d(l'(v_i), l'(v_i)) = 0 = c_i$. Otherwise $c_i^w \geq c_i + d(l'(v_i), l(v_i))$ since v_i has at least one child in G_i, say x, and $d(l'(v_i), l(x)) = d(l'(v_i), l(v_i)) + d(l(v_i), l(x))$.

On the other hand, l' remains a feasible labeling of G_i^w, i.e. $l' \in Q(H, G_i^w, p_i^w)$, with the edge cost c_i', since the cost on the arc $\langle v_i, w \rangle$ is 0.See the right G_i^w of Figure 5.3.

Since G_i^w is the tree with the height less than n, according to induction, we have

$$\begin{aligned} c_i' > c_i^w \geq c_i + d(l'(v_i), l(v_i)) \Rightarrow \\ c_i' - c_i > d(l'(v_i), l(v_i)). \end{aligned} \tag{5.1}$$

Besides, we have $l'(v_i) \prec l(v_i)$ or $l'(v_i) = l(v_i)$, $l'(g) \prec l(g)$ or $l'(g) = l(g)$ by Corollary 1. In addition, g is the root implies that $l'(g) \prec l'(v_i)$. Therefore it is also true that

$$d(l'(v_i), l(v_i)) = d(l'(g), l(v_i)) - d(l'(g), l'(v_i)), \tag{5.2}$$

$$d(l'(g), l(v_i)) \geq d(l(g), l(v_i)), \tag{5.3}$$

(See Figure 5.4).

Substituting (5.2) and (5.3) into (5.1) gives the following:

$$\begin{aligned} c'_i - c_i > d(l'(g), l(v_i)) - d(l'(g), l'(v_i)) \Rightarrow \\ c'_i + d(l'(g), l'(v_i)) > c_i + d(l'(g), l(v_i)) \Rightarrow \\ c'_i + d(l'(g), l'(v_i)) > c_i + d(l(g), l(v_i)) \Rightarrow \\ c' = \sum_i c'_i + d(l'(g), l'(v_i)) > \sum_i c_i + d(l(g), l(v_i)) = c. \end{aligned}$$

Hence, $c = \sum_i (c_i + d(l(g), l(v_i)))$ is the *minimum arc cost* compared to any other labeling $l' \in Q(H, G, p)$.

□

Theorem 7. *For any character tree H and phylogeny tree (G, p), Sankoff's algorithm outputs the same labeling as Algorithm 5.1 does.*

Proof. As an input to Sankoff's algorithm, the cost matrix M can be generated from H by letting $M_{ij} := d(i, j)$ for $i, j \in V(H)$ according to Definition 2. Let l' be the labeling produced by Sankoff's algorithm given (G, p) and M as input. Since for each arc $\langle u, v \rangle \in V(G)$, $d(l'(u), l'(v)) \neq \infty$ which means there is a path from $l'(u)$ to $l'(v)$ in H, otherwise $d(l'(u), l'(v)) = \infty$ and then $\text{acost}(H, G, l') = \infty$. Therefore $l' \in Q(H, G, p)$. According to Theorem 6, $l' = l$. □

However, Algorithm 5.1 has better performance, since it runs in linear time $O(|V(G)| \cdot \log_2 \delta + |V(H)|)$, compared to Sankoff's algorithm which runs in time $O(|V(G)| \cdot \delta \cdot |V(H)|)$.

5.1.2 Example

Given the character tree H and phylogenetic tree (G, p) as presented in Figure 5.5, Algorithm 5.1 outputs the labeling as showed in Figure 5.6. The labels of the leaves v_1,v_2,v_3,v_4 and v_5 remains the same as their predefined values. The algorithm visited the internal vertices in the order of v_6, v_7, v_8 and v_9. It sets $l(v_6) := \text{LCA}(l(v_1), l(v_2)) = \text{LCA}(c, d) = b$, $l(v_7) := \text{LCA}(l(v_4), l(v_5)) = \text{LCA}(c, e) = a$, $l(v_8) := \text{LCA}(l(v_6), l(v_3)) = \text{LCA}(b, e) = a$ and $l(v_9) := \text{LCA}(l(v_7), l(v_8)) = \text{LCA}(a, a) = a$. The minimum arc cost of (G, p) is seven.

5.2 Minimum pseudo-minor bag cost

The following is the algorithm for Problem 2.

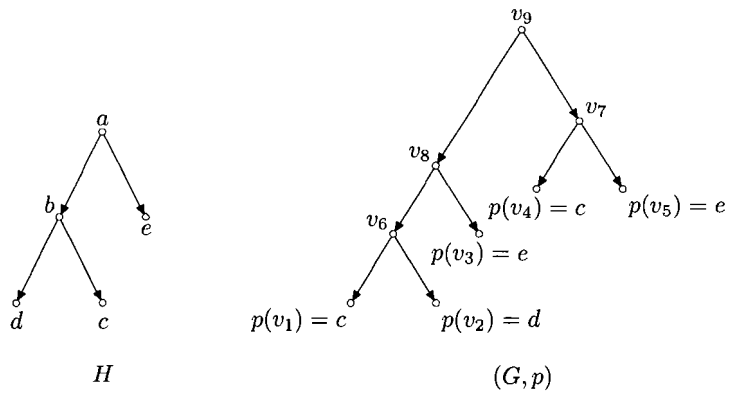


Figure 5.5: An input of H and (G, p) to Algorithm 5.1

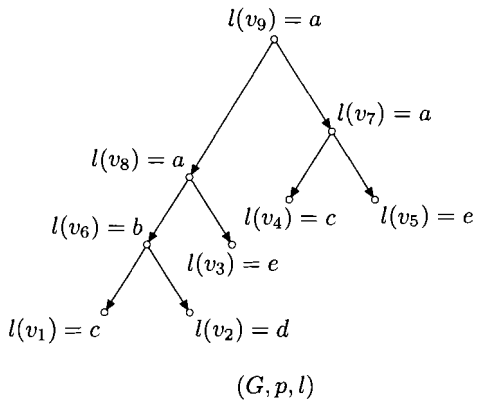


Figure 5.6: (G, p, l) where l is output by Algorithm 5.1 given H and (G, p) in Figure 5.5

Input: A character tree H and a phylogenetic tree (G, p).

Output: A labeling $l \in Q(H, G, p)$ such that $\text{bcost}(H, G; l) = \text{qbcost}(H, G, p)$.

Description: As an initialization, for every $u \in L(G)$, let $l(u) := p(u)$; For each vertex $u \in V(G)$, we will set $x(u) := 1$ if $l(u)$ appears as a leaf label in G_u, and $x(u) := 0$ otherwise. Initially, $x(u) := 1$ if u is a leaf, and $x(u) := 0$ otherwise. The algorithm then works in two stages. In the first stage, the tree G is traversed in *post-order*: For each internal vertex $u \in V(G)$, let $l(u) := \text{LCA}(\{l(v) | \langle u, v\rangle \in A(G)\})$. (This again requires a linear time preprocessing on H as described in previous algorithm.) If there exists some child v of u such that $l(u) = l(v)$ and $x(v) = 1$, then $x(u) := 1$, else we do not update the value of $x(u)$.

In the second stage, the tree G is traversed in *pre-order* to update the value of l for some internal vertices $u \in V(G)$ (except the root g) as follows: If $x(u) = 0$, then $l(u) := l(u')$ where u' is the parent of u.

Finally, the number of bags, which initially is set to $|V(G)|$, is calculated by subtracting the total number of arcs $\langle u, v\rangle \in A(G)$ with $l(u) = l(v)$.

Pseudo code:

Algorithm 5.2 Minimum-Pseudo-Minor-Bag-Cost(H,G,p)

```
1  preprocess H (build auxiliary trees) to speed up the computation of LCA ;
   ▷ initialization
2  for each vertex u ∈ V(G) do
3      if u is a leaf then
4          set l(u) := p(u);
5          set x(u) := 1;
6      else
7          set l(u) := null;
8          set x(u) := 0;
9  Set b := |V(G)|;
   ▷ first stage
10 traverse the tree in post-order, for each internal vertex u ∈ V(G) do
11     set l(u) := LCA(l(v)|⟨u, v⟩ ∈ A(G));
12     for each child v of u
13         if l(u) = l(v) and x(v) = 1 then do
14             set x(u) := 1;
       ▷ second stage
15 traverse the tree in pre-order, for the internal vertex u ∈ V(G) do
16     if u ≠ g and x(u) = 0 then do
```

17 set $l(u) = l(u')$ where u' is the parent of u;
18 **for** each arc $e = \langle u, v \rangle$ in G
19 **if** $l(u) = l(v)$ **then do**
20 set $b := b - 1$;
▷ *output result*
21 output labeling function l and bag cost b

Complexity: It takes $O(|V(H)|)$ to preprocess H. For each node $u \in V(G)$, in first stage, it takes $O(\log_2 \delta)$ time to compute $l(u)$ and $x(u)$; in second stage, it takes $O(1)$ time to update $l(u)$. Therefore the labeling takes $O(|V(G)| * \log_2 \delta + |V(H)|)$ steps. In particular if G is a binary tree, then the time complexity is $O(|V(G)| + |V(H)|)$.

5.2.1 Proof of correctness

Theorem 8. *Algorithm 5.2 returns a labeling* $l \in Q(H, G, p)$ *such that* $\mathrm{bcost}(H, G, l) = \mathrm{qbcost}(H, G, p)$.

Proof. l is obviously *p-constrained* according to the initialization of the algorithm. Moreover, for every arc $\langle u, v \rangle \in A(G)$, there is a directed path from $l(u)$ to $l(v)$ in H since $l(u)$ is either the least common ancestor or common ancestor of the labels of its children, i.e. $l(u) \prec l(v)$, so l is a smooth function. Therefore $l \in Q(H, G, p)$.

We prove $\mathrm{bcost}(H, G; l) = \mathrm{qbcost}(H, G, p)$ by induction on the number of vertices of G. Thus let $n = |V(G)|$.

In case $n = 1$, it is obviously true, $\mathrm{bcost}(H, G, l) = 1$. Suppose the theorem is true when $|V(G)| < n$. We want to prove that it is also true when $|V(G)| = n$. We will distinguish two cases:

- Consider the case when there exists an arc $\langle v_1, v_2 \rangle$ in G such that Algorithm 5.2 gives different labels to v_1 and v_2, i.e. $l(v_1) \neq l(v_2)$ and v_2 is not a leaf. In this case $l(v_2) = \mathrm{LCA}(\{p(w) | w \in L(G_{v_2})\})$ and $l(v_2) \in \{p(w) | w \in L(G_{v_2})\}$ since otherwise $l(v_2) = l(v_1)$. Let G_1 and G_2 be two components of $G - e$ so that v_1 is in G_1 and v_2 is in G_2. In what follows, we will run the Algorithm 5.2 on inputs G_1 and G_2 together with H, respectively. In order to do this, we need to perform the following: We add a new leaf v_2' into G_1 and join it to the vertex v_1, i.e. v_1 will be its parent. Let p_1,and p_2 denote the prelabeling functions of leaves of G_1 and G_2 respectively, defined as follows. For G_2, each leaf keeps the same label as in G, i.e. $p_2(v) = p(v)$ for all $v \in L(G_2)$. Likewise each leaf except v_2' of G_1 has the same label as in G, i.e.$p_1(v) = p(v)$ for all $v \in L(G_1)$ except v_2', let $p_1(v_2') := l(v_2)$; see Figure 5.7.

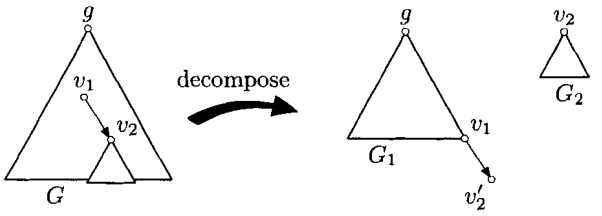


Figure 5.7: Construct G_1 and G_2 from G

Note that each internal vertex in G_1 will be assigned the same label by running Algorithm 5.2 on input (H, G_1, p_1) and on input (H, G, p) since $p_1(v_2') = l(v_2)$. The same holds for each internal vertex in G_2.

Let M_1 be the bag cost returned by Algorithm 5.2 on input (H, G_1, p_1), M_2 be the bag cost returned by Algorithm 5.2 on input (H, G_2, p_2), and M be the bag cost returned by Algorithm 5.2 on input (H, G, p), respectively. It is not hard to see that $M = M_1 + M_2 - 1$.

Now consider any other labeling $l' \in Q(H, G, p)$. Let l_1' and l_2' be the restrictions of l' to sub-tree G_1 and G_2, respectively. Let p_1', and p_2' denote the prelabeling functions of leaves of G_1 and G_2, respectively, so that for all vertices $v \in L(G_2)$ $p_2'(v) = p_2(v)$, and for all vertices $v \in L(G_1)$ $p_1'(v) = p_1(v)$ except $p_1'(v_2') = l'(v_2)$. According to the proof of Lemma 2, either $l'(v_2) = l(v_2)$ or $l'(v_2) \prec l(v_2)$ since $l(v_2) = \text{LCA}(p(w) | w \in L(G_{v_2}))$.

Let $N_1 = \text{bcost}(H, G_1, l_1')$, $N_2 = \text{bcost}(H, G_2, l_2')$, and $N = \text{bcost}(H, G, l')$, then $N = N_1 + N_2 - 1$. We will prove that $N_1 + N_2 \geq M_1 + M_2$ in either case when $l'(v_2) = l(v_2)$ or $l'(v_2) \prec l(v_2)$.

By induction, $N_2 \geq M_2$ since $p_2' = p_2$. However N_1 and M_1 cannot be compared directly since p_1 and p_1' may have different value for the vertex v_2'. Let M_1' be the bag cost returned by running Algorithm 5.2 on input (H, G_1', p_1'), $N_1 \geq M_1'$ by induction. We will compare N_1 and M_1 through M_1'; see Figure 5.8.

When $l'(v_2) = l(v_2)$, we have $M_1 = M_1'$ since p_1 and p_1' have the exact same leaf labels. Besides, $N_1 \geq M_1'$ and $N_2 \geq M_2$ by induction. Therefore we have $N_1 + N_2 \geq M_1 + M_2$.

Now suppose $l'(v_2) \prec l(v_2)$. We first claim that $N_1 \geq M_1' \geq M_1 - 1$. This follows from the fact that the labeling l_1'' returned by Algorithm 5.2 on input (H, G_1, p_1') is a feasible internal labeling of (G_1, p_1), i.e. $l_1'' \in Q(H, G_1, p_1)$. It is because the leaf labels defined by p_1' and p_1

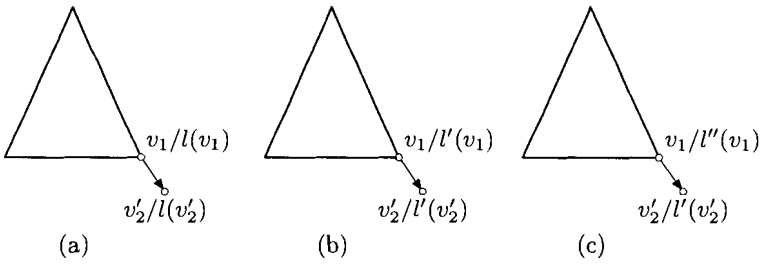


Figure 5.8: Comparing N_1 and M_1 using M_1'. (a) and (b) shows (G_1, p_1, l) with the bag cost M_1 and (G_1, p_1', l') with the bag cost N_1 respectively, and (c) shows (G_1, p_1') as an input to Algorithm 5.2 with the potienal bag cost M_1'.

are the same except for v_2', and $p_1'(v_2') = l'(v_2) \prec p_1(v_2') = l(v_2)$. The total bag number of such labeling on (G_1, p_1) is at most $M_1' + 1$ which attains when $l''(v_1) = l'(v_2') \neq l(v_2')$, i.e. $\mathrm{bcost}(H, G_1, l_1'') \leq M_1' + 1$. Moreover, by induction, we have $\mathrm{bcost}(H, G_1, l_1'') \geq M_1$, so $M_1' + 1 \geq M_1$. Since $N_1 \geq M_1'$, $N_1 \geq M_1' \geq M_1 - 1$, i.e. $N_1 \geq M_1 - 1$.

We second claim that $N_2 \geq M_2 + 1$. Indeed, we can construct a internal labeling $l_2'' \in Q(H, G_2, p_2)$ from l_2'. The only change we need to make is to replace the internal labels b such that $b \prec l(v_2)$ with $l(v_2)$. Since all the leaf labels in G_2 are either $l(v_2)$ or the descendants of $l(v_2)$, the resulting labeling $l_2'' \in Q(H, G_2, p_2)$. The number of bags l_2'' produces is at most $N_2 - 1$, i.e. $\mathrm{bcost}(H, G_2, l_2'') \leq N_2 - 1$. Moreover, by induction, we have $\mathrm{bcost}(H, G_2, l_2'') \geq M_2$, therefore $N_2 - 1 \geq M_2$, i.e. $N_2 \geq M_2 + 1$.

Therefore, with $N_1 \geq M_1 - 1$ and $N_2 \geq M_2 + 1$, we have $N_1 + N_2 \geq M_1 + M_2$. Since $N = N_1 + N_2 - 1$ and $M = M_1 + M_2 - 1$, so $N \geq M$ which means the bag cost of l is the minimum.

- Now suppose every arc $\langle v_1, v_2 \rangle$ in G with v_2 is not a leaf satisfies $l(v_1) = l(v_2)$, in other words, all the internal vertices of G get the same label by Algorithm 5.2. For any two leaves v_i and v_j with the same parent v_k (See Figure 5.9), there are three different cases for their labels.

  - First, if $l(v_i) = l(v_j)$, then Algorithm 5.2 will assign $l(v_k) := l(v_i)$, so these tree vertices will be in the same bag.
  - Second, if $l(v_i) \prec (v_j)$, then Algorithm 5.2 will assign $l(v_k) := l(v_i)$. Therefore these

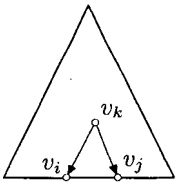


Figure 5.9: Three different cases for the labels of v_i, v_j and v_k

three vertices will contribute two bags. However, with two leaves with different labels, any other labeling must contribute by at least two bags. The case when $l(v_j) \prec (v_i)$ is similar.

- Third, if $l(v_i) \not\sim l(v_j)$, then Algorithm 5.2 will assign $l(v_k) := \text{LCA}(l(v_i), l(v_j))$, so these three vertices will contribute three bags. Since $l(v_i), l(v_j)$ are incomparable, any labeling $l' \in Q(H, G, p)$ must produce three bags on these three vertices otherwise either $d(l'(v_k), l'(v_i))$ or $d(l'(v_k), l'(v_j))$ will be infinite.

We conclude that l given by Algorithm 5.2 gives smallest possible bag cost.

□

5.2.2 Example

Given the character tree H and phylogenetic tree (G, p) as presented in Figure 5.10, Algorithm 5.2 outputs the labeling as showed in Figure 5.11. The tree on the left side shows the intermediate result after the first stage of the algorithm where values of l and x for each internal vertex in G are set. The tree on the right side shows the final labeling l after the second stage. The labels of the leaves v_1,v_2,v_3,v_4 and v_5 remain the same as their predefined labels. In the first stage, for vertex v_6, knowing that $l(v_6) = \text{LCA}(l(v_1), l(v_2)) = \text{LCA}(b, d) = b$ appears as a leaf label in G_{v_6}, $x(v_6)$ is assigned 1. All the other internal vertices have 0 as x value. In the second stage, for v_8 and v_7, since $x(v_8) = 0$ and $x(v_7) = 0$, so the algorithm updates $l(v_8) := l(v_9)$ and $l(v_7) := l(v_9)$; for v_6, since $x(v_6) = 1$, $l(v_6)$ is not updated. The minimum bag cost of the given (G, p) is six.

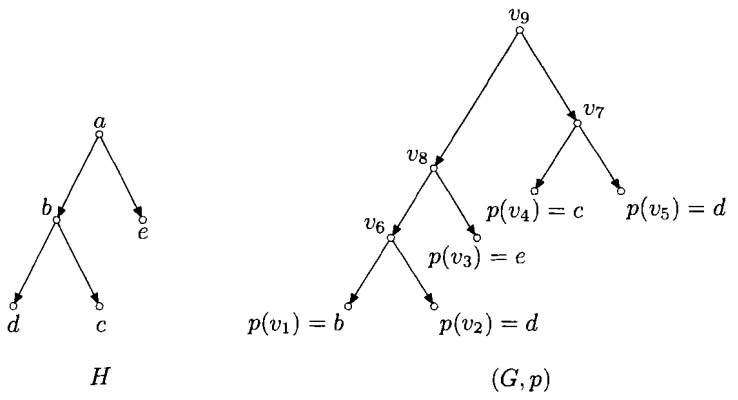


Figure 5.10: An input of H and (G, p) to Algorithm 5.2

5.3 The decision problem of $H \leq_{rm} (G, p)$ when $p \neq \epsilon$

Definition 12. A path $P = u \rightarrow v$ in a rooted tree G is called a *single branch path* if every inner vertex of the path has only one child, and u, v either both have more than one child or otherwise u must be the root of G with only one child, and v must be a leaf. Moreover for any labeling function l, P has a corresponding path $P' = l(u) \rightarrow l(v)$ in H.

For example, an arc with two end vertices both having more than one child is a single branch path of length 1. Figure 5.12 shows a more general example of a single branch path $v_k \rightarrow v_1$ in G of length > 1 having a correponding path $l(v_k) \rightarrow l(v_1)$ in H.

5.3.1 The algorithm

The following is the algorithm for the decision problem of rooted minor when $p \neq \varepsilon$.

Input: Two rooted trees H and (G, p).

Output: YES/NO to if $H \leq_{rm} (G, p)$.

Description: The algorithm works by trying to build a labeling $l \in M(H, G, p)$ and meanwhile keeps track of $r(\langle a, b\rangle, l)$ for each $\langle a, b\rangle \in A(H)$. If such l is found to be impossible to exist at any point, the algorithm outputs NO. There are five steps in total.

In the first step, the algorithm checks whether there exists a vertex $a \in L(H)$ such that for every $u \in L(G)$ $p(u) \neq a$. If such a is found, then the algorithm outputs NO.

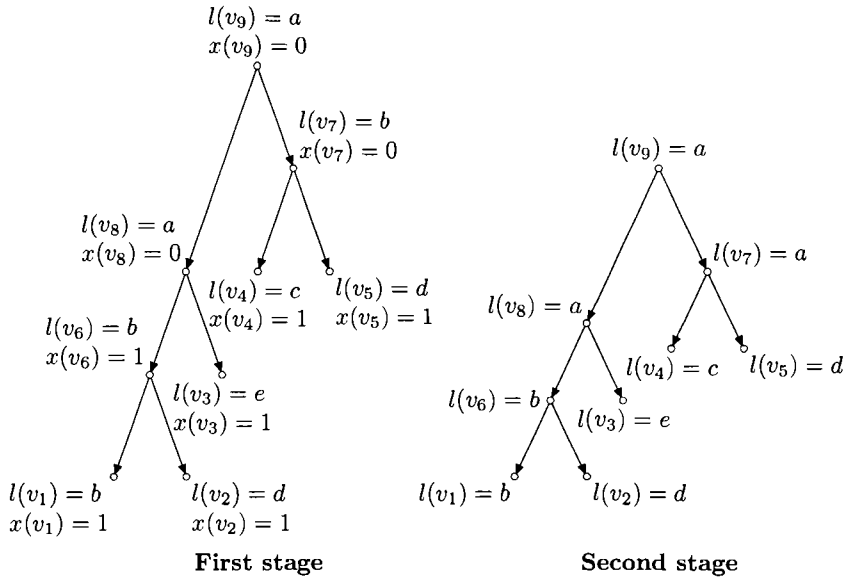


Figure 5.11: The example (G, p, l) where l is output by Algorithm 5.2 given H and (G, p) in Figure 5.10.

The second step is to initialize a labeling l such that $l(u) := p(u)$ for every $u \in L(G)$. We also initialize $r(\langle a, b\rangle, l) := 0$ for each $\langle a, b\rangle \in A(H)$

In the third step, G is traversed G in *post-order* to do the following. For every internal vertex $u \in V(G)$, we assign $l(u) := \text{LCA}(\{l(v)|\langle u, v\rangle \in A(G)\})$. (This again requires a linear time preprocessing on H as described in previous algorithm.) Moreover, if u has more than one child, then for each child v of u such that $l(u) \neq l(v)$ and v is either a leaf or a internal vertex with more than one child, we check whether $\langle l(u), l(v)\rangle \in A(H)$. If the answer is no, then the algorithm outputs NO, else $r(\langle l(u), l(v)\rangle, l)$ is increased by one.

In the fourth step, the (G, p, l) is examined for some special cases. We look for the longest single branch path P in (G, p, l) of length at least two, say $P = (v_k, ..., v_2, v_1)$ (Note that $l(v_i) = l(v_1)$ for $i = 2, 3, \cdots, k-1$ on the path P.). If P satisfies one of the following three conditions (See Theorem 9 for justification),

1. $l(v_k) \neq l(v_1)$ and $\langle l(v_k), l(v_1)\rangle \in A(H)$.

2. $v_k \neq g$ and $l(v_1) = l(v_k)$.

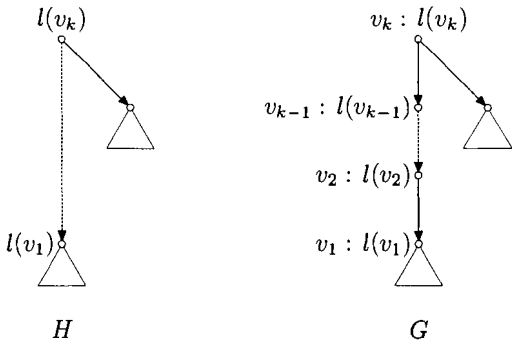


Figure 5.12: A single branch path of length > 1 in G and its corresponding path in H.

3. $v_k = g$ and $l(v_1) = l(v_k) = h$.

we just mark P as processed and continue to process the next unprocessed longest single branch path, since the labels of v_i ($i = 2, 3, \cdots, k-1$) cannot be updated any more. Otherwise, if none of the above three conditions is satisfied, let $P' = (w_m), ..., w_1)$ be the corresponding path of P in H. m is the length of P' where $w_1 := l(v_1)$ and w_m depends on v_k. If $v_k \neq g$ then $w_m := l(v_k)$ else $w_m := h$ according to Note 1. By comparing P and P', the algorithm will decide if it is possible to realize every arc of P' on P. If P' is not a single branch path, then the algorithm outputs NO since it is impossible to realize P' on P. If P' is longer than P then the algorithm outputs NO as well for the same reason. Otherwise, P and P' are both single branch paths, and P is either as long as P' or longer than P'. Therefore it is possible to realize every arc of P' on P. We accomplish this by updating the labels of v_i ($i = 2$ to k) as follows: $l(v_k) := w_m$,$l(v_{k-1}) := w_{m-1}$,$l(v_{k-2}) := w_{m-2}, \cdots, l(v_{k-m+1}) := w_1$. If P is longer than P', then the labels of vertices $v_1, \cdots, v_{k-m}$ on P remain unchanged. At mean time, we keep updating $r(\langle a, b\rangle, l)$ for every arc $\langle a, b\rangle$ of P'. Finally, we mark P as processed and continue to find the next unprocessed longest single branch path until all such paths are processed.

At the end, if there exists any $\langle a, b\rangle \in A(H)$ such that $r(\langle a, b\rangle, l) > 1$, then the algorithm outputs NO, otherwise outputs Yes, i.e. $H \leq_{rm} (G, p)$.

Pseudo code:

Algorithm 5.3 Rooted-Minor-Decision(H,G,p)

1 preprocess H (build auxiliary trees) to speed up the computation of LCA ;

▷ *Step 1: Checking if there is any leaf of H not appearing as a leaf label in G*

2 **for** each leaf $v \in L(H)$ **do**
3 set $q(v) := 0$
4 **for** each leaf $u \in L(G)$ **do**
5 set $q(p(u)) := q(p(u)) + 1$
6 **for** each leaf $v \in L(H)$ **do**
7 **if** $q(v) = 0$ **then do**
8 output NO
▷ *Step 2: Initializing the labeling l and r*
9 **for** each internal vertex $u \in V(G)$ **do**
10 set $l(u) := null$
11 **for** each leaf $u \in L(G)$ **do**
12 set $l(u) := p(u)$
13 **for** each edge $\langle a, b\rangle \in A(H)$ **do**
14 set $r(\langle a, b\rangle, l) := 0$
▷ *Step 3: Starting to assign labels to internal vertices*
15 traverse the tree G in *post-order*, **for** every internal vertex $u \in V(G)$ **do**
16 set $l(u) := \mathrm{LCA}(l(v) | \langle u, v\rangle \in A(G))$;
17 **if** u has more than one child **then do**
18 **for** each child v of u such that $l(u) \neq l(v)$ and v is either a leaf or has more than one child **do**
19 **if** $\langle l(u), l(v)\rangle \notin A(H)$ **then do**
20 output NO
21 **else**
22 set $r(\langle l(u), l(v)\rangle, l) := r(\langle l(u), l(v)\rangle, l) + 1$
▷ *Step 4: Checking single branch paths*
23 find the longest unprocessed single branch path $P = (v_k, ..., v_2, v_1)$ of length at least two in G
24 **if** P does not exist **then do**
25 goto line 43
26 **if** $\langle l(v_k), l(v_1)\rangle \in A(H)$ or
$v_k \neq g$ and $l(v_1) = l(v_k)$ or
$v_k = g$ and $l(v_1) = l(v_k) = h$ **then do**
27 mark P as processed and goto line 23
28 **else**
29 **if** $v_k = g$ **then do**
30 set $P' := (w_m, ..., w_1)$ as a path in H where $w_m = h$ and $w_1 = l(v_1)$
31 **else do**
32 set $P' := (w_m, ..., w_1)$ as a path in H where $w_m = l(v_k)$ and $w_1 = l(v_1)$

33 **for** each vertex a on the path P' in H **do**
34 **if** a has more than one child **then do**
35 output NO
36 **if** P' is longer than P **then do**
37 output NO
38 **for** $i = 1$ to m **do**
39 set $l(v_{k-i+1}) := w_{m-i+1}$
40 **for** $i = m$ to 2 **do**
41 set $r(\langle w_i, w_{i-1}\rangle, l) := r(\langle w_i, w_{i-1}\rangle, l) + 1$
42 mark P as processed and goto line 23
▷ *Step 5: Checking if there is any arc in H being realized more than once by l*
43 **for** each edge $\langle a, b\rangle \in A(H)$ **do**
44 **if** $r(\langle a, b\rangle) > 1$ **then do**
45 output NO
▷ *Coming to a conclusion that* $H \leq_{rm} (G, p)$
46 output YES

5.3.2 Proof of correctness

Lemma 3. *Given two rooted tree H and (G,p), if $H \leq_{rm} (G,p)$, then for any $l' \in M(H,G,p)$, and for any vertex $w \in V(G)$ with at least two children,*

$l'(w) = \mathrm{LCA}(p(v)|v \in L(G_w))$.

Proof. Let $q = \mathrm{LCA}(p(v)|v \in L(G_w))$. According to Lemma 2, either $l'(w) = q$ or $l'(w) \prec q$. Therefore we only need to prove that $l'(w) \prec q$ is not true. It can proved by the induction on the height t of G_w.

When $t = 2$, let $v_1, v_2, \cdots, v_k$ denote the set of k children of w. There are two cases: $q = p(v_1) = p(v_2) = \cdots = p(v_k)$ or $q \prec p(v_i)$ for some v_i $(1 \leq i \leq k)$. In first case, if $l'(w) \prec q$, then $c(B_q^{l'}) > 1$ since $l'(v_i) = p(v_i)$ for $i = 1, 2, \cdots, k$, therefore separation occurs. In second case, if $l'(w) \prec q$, then transitivity happens since $l'(w) \prec q$ and $q \prec p(v_i)$ imply $l'(w) \prec p(v_i)$. Thus $l'(w) = q$ in both cases.

Suppose the lemma is true when $t < n$. We prove it is also true when $t = n$. Again let $v_1, v_2, \cdots, v_k$ denote k children of w, respectively. Let z_i be the first descendant of v_i which has more than one child or is a leaf; see Figure 5.13. In particular, if v_i has more than one child, then $z_i := v_i$. Thus each z_i $(i = 1, 2, \cdots k)$ is either a internal vertex with more than one child or a leaf. If z_i has more than one child, by induction, we have $l'(z_i) = \mathrm{LCA}(p(v)|v \in L(G_{z_i}))$. It implies

that either $q = l'(z_i)$ or $q \prec l'(z_i)$. Otherwise z_i is a leaf, and then $l'(z_i) = p(z_i)$ which also implies either $q = l'(z_i)$ or $q \prec l'(z_i)$.

Now assuming $l'(w) \prec q$, there must exist a vertex a_i on the path $w \rightarrow z_i$ such that $l'(a_i) = q$ no matter $q = l'(z_i)$ or $q \prec l'(z_i)$ since $l' \in M(H, G, p)$. In particular, $a_i = z_i$ if $l'(z_i) = q$. However, we have $c(B_q^{l'}) > 1$, therefore separation occurs. Thus $l'(w) = q$.

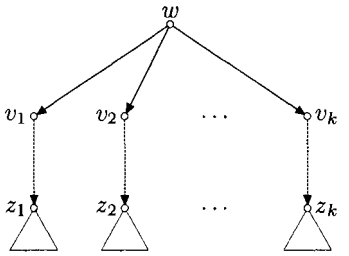


Figure 5.13: $z_1, z_2, \cdots, z_k$ in G_w.

□

Theorem 9. *Algorithm 5.3 outputs YES if and only if* $H \leq_{rm} (G, p)$.

Proof. Let l be the labeling build by Algorithm 5.3.

We first prove that if Algorithm 5.3 outputs YES, then $H \leq_{rm} (G, p)$, i.e. $l \in M(H, G, p)$.

If Algorithm 5.3 outputs YES, then l must satisfy two attributes. First, for each arc $\langle u, v\rangle \in A(G)$, $\langle l(u), l(v)\rangle \in A(H)$. This is guaranteed by the first, third, fourth steps of Algorithm 5.3. Second, for every arc $\langle a, b\rangle \in A(H)$, $r(\langle a, b\rangle, l) = 1$ which is guaranteed by fifth step. It is obvious that the second attribute of l implies (2) of Definition 6. Moreover (1) of Definition 6 is implied by both attributes. Therefore $l \in M(H, G, p)$.

Next we prove that if Algorithm 5.3 outputs NO, then $M(H, G, p) = \emptyset$, i.e. no other labeling function exists to satisfy (1) and (2) of Definition 6. We distinguish five situations under which Algorithm 5.3 outputs NO. Please note that before line 37, for every internal vertex $v \in V(G)$, $l(v) = \mathrm{LCA}(p(v)|v \in L(G_w))$. This fact will be used frequently in the following proof.

- The algorithm outputs NO in Line 8 because there exists a vertex $b \in L(H)$ such that $p(u) \neq b$ for every $u \in L(G)$. Let a be the parent of b in H. For any other labeling $l' \in M(H, G, p)$, there must exist an arc $\langle u, v\rangle \in A(G)$ such that $l'(u) = a, l'(v) = b$. Since v is not a leaf in G, and b is a leaf in H, and $l'(w) \neq b$ for every vertex w in $L(G_v)$, addition must occur in l'.

It contradicts the assumption that $l' \in M(H, G, p)$. Therefore if the algorithm outputs NO at the line 8, $M(H, G, p) = \emptyset$.

- The algorithm outputs NO in Line 20 because there exists an internal vertex u such that u has more than one child and for some child v (either a leaf or a internal vertex with more than one child) of u, $(l(u), l(v))\rangle \notin A(H)$. For any other labeling $l' \in M(H, G, p)$, since u has more than one child, $l'(u) = l(u)$ according to Lemma 3. As to v, if it is a leaf, we have $l'(v) = l(v) = p(v)$; otherwise it is a internal vertex with more than one child, and we also have $l'(v) = l(v)$ according to Lemma 3. However $l'(u) = l(u)$ together with $l'(v) = l(v)$ imply that $(l'(u), l'(v))\rangle \notin A(H)$ which contradicts the assumption that $l' \in M(H, G, p)$. Therefore no such l' can possibly exist, i.e. if the algorithm outputs NO in Line 20, then $M(H, G, p) = \emptyset$.

- The algorithm outputs NO in Line 35 because $P = (v_k, ..., v_2, v_1)$ in G is a single branch path, but its corresponding path $P' = (w_m, ..., w_2, w_1)$ in H is not. Since v_1 is either a leaf or a vertex with more than one child, $l'(v_1) = l(v_1)$ for any labeling $l' \in M(H, G, p)$. If v_k is also a vertex with more than one child, then $l'(v_k) = l(v_k)$ as well, otherwise v_k must be a root with only one child which means $l'(v_k) = h$. Thus l and l' correspond P in G to the same path P' in H according to Line 30 and Line 32 of the algorithm. However when P' is not a single branch path, either negligence or separation or transitivity will happen no matter how the labels of $v_k, v_{k-1}, \cdots, v_2$ are updated. Figure 5.14 shows two examples.

  In the left example, a single branch path $P = (v_3, v_2, v_1)$ in G with both end points having more than one child has a corresponding path $P' = (a, b, c)$ in H by l. P' is not a single branch. Since $l'(v_1) = l(v_1)$ and $l'(v_3) = l(v_3)$ for any $l' \in M(H, G, p)$, so the corresponding path of P induced by l' is the same as P'. However we argue that such l' cannot exist no matter how v_2 is labeled. In particular, suppose $l'(v_2) = b$, although $\langle a, b\rangle \in A(H)$ and $\langle b, c\rangle \in A(H)$ are realized, it is impossible to realized $\langle b, d\rangle \in A(H)$ on the path P. Even if $\langle b, d\rangle$ is realized in some other part of G, then we have $c(B_b^{l'}) > 1$. Thus either $r(\langle b, d\rangle, l') = 0$ or $c(B_b^{l'}) > 1$ when $l'\langle v_2) = b$. Suppose $l'(v_2) = c$, then transitivity happens obviously.

  The right example shows another specific example where a single branch path $P = (v_3, v_2, v_1)$ with v_3 being the root in G with only one child. Any $l' \in M(H, G, p)$ corresponds P in G to the path $P' = (a, b, c)$ in H since $l'(v_3) = h$ and $l'(v_1) = c$. P' is not a single branch path. Similarly, such l' cannot exist no matter how v_2 is labeled. In particular, if $l'(v_2) = b$, then

$\langle h, b\rangle \in A(H)$ and $\langle b, c\rangle \in A(H)$ are realized, but it is impossible to realized $\langle b, d\rangle \in A(H)$ on the path P.

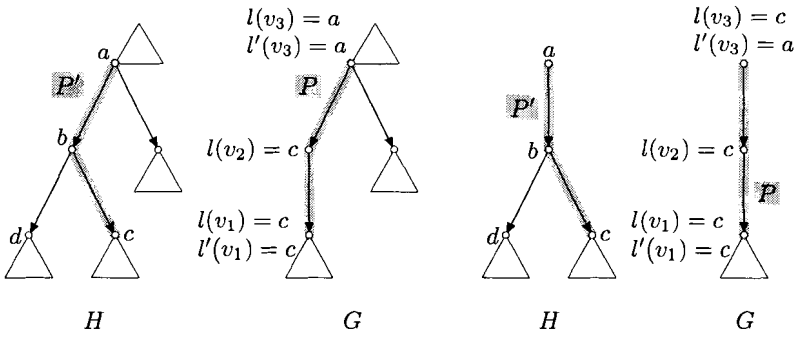


Figure 5.14: Two examples of a path P in G is a single branch path, but its corresponding path P' in H is not. The left example shows a P beginning with a vertex with two children, while the right example shows a P beginning with the root of G ($v_3 = g$) which has only one child, and therefore the corresponding P' begins at the root of H (a=h) although $l(v_3) = c$ after third step of Algorithm 5.3.

Therefore if the algorithm outputs NO in Line 35, any labeling $l' \in M(H, G, p)$ cannot possibly exist, i.e. $M(H, G, p) = \emptyset$.

Note that, so far it is easy to see why the algorithm does nothing except marking P as processed when P satisfies any of the following three conditions in Line 26:

- $l(v_k) \neq l(v_1)$ and $\langle l(v_k), l(v_1)\rangle \in A(H)$. Since $l(v_k) \neq l(v_1)$, v_k must be a vertex with more than one child. According to Lemma 3, $l'(v_k) = l(v_k)$ for any $l' \in M(H, G, p)$. Similarly, either v_1 is a leaf or v_1 is a vertex with more than one child, so $l'(v_1) = l(v_1)$. Because $\langle l(v_k), l(v_1)\rangle \in A(H)$, no change should be made to the labels of the internal vertices on P.
- $v_k \neq g$ and $l(v_1) = l(v_k)$. Since v_k is not the root of G, v_k must be the vertex with more than one child. Therefore for any $l' \in M(H, G, p)$, we have $l'(v_k) = l(v_k)$, and also $l'(v_1) = l(v_1)$. Again no change should be made to the labels of the internal vertices on P.
- $v_k = g$ and $l(v_1) = l(v_k) = h$. It is clear that $l'(v_1) = l(v_1)$ for any $l' \in M(H, G, p)$. Since v_k is the root of G, $l'(v_k) = h$, i.e. $l'(v_k) = l(v_k)$, therefore again no change need

to be made in this case.

- The algorithm outputs NO in Line 37 because P' is longer than P. Similarly it is true that for any labeling $l' \in M(H, G, p)$, l and l' designate the same corresponding path P' in H of P in G. However such l' cannot exist since either negligence or separation or transitivity must occur when P' is longer than P. Figure 5.15 shows two examples. The left example shows that l corresponds the path $P = (v_3, v_2, v_1)$ in G to the path $P' = (a, b, c, d)$ in H. They are both single branch paths, but P' is longer than P. For any $l' \in M(H, G, p)$, $l'(v_3) = l(v_3)$ and $l'(v_1) = l(v_1)$ since v_3 and v_1 are both vertices with more than one child. Therefore the corresponding path of P induced by l' is the same as P'. However such l' cannot exist since transitivity always occurs no matter $l'(v_2) = b$ or $l'(v_2) = c$. The right example shows a similar example where v_3 is the root g with one child and $P' = (a, b, c, d)$ where $a = h$. Again transitivity always occurs no matter $l'(v_2) = b$ or $l'(v_2) = c$. Therefore the algorithm outputs NO in Line 37, no $l' \in M(H, G, p)$ can possibly exist, i.e. $M(H, G, p) = \emptyset$.

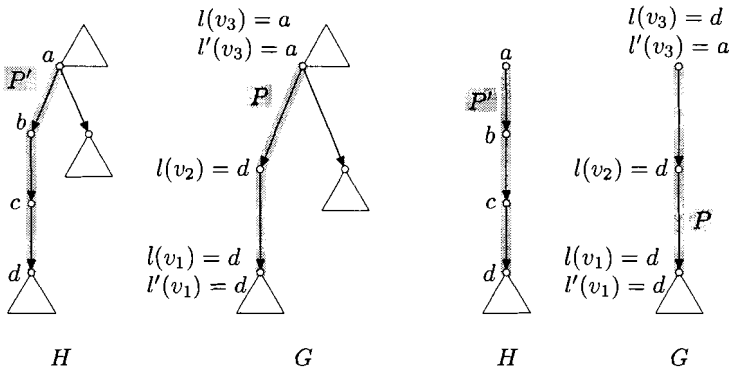


Figure 5.15: Two examples of P' in H is a longer single branch path than P in G is. The first example shows a P beginning with a vertex with more than one child, while the second example shows a P beginning with the root of G ($v_3 = g$) which has only one child, and therefore the corresponding P' begins at the root of H ($a = h$) although $l(v_3) = d$ after third step of Algorithm 5.3.

- The algorithm outputs NO in Line 45 because there is at least one arc $\langle a, b \rangle \in A(H)$ being realized more than once. Let $\langle u_1, v_1 \rangle$ and $\langle u_2, v_2 \rangle$ be the two arcs in G such than $l(u_1) = l(u_2) = a$ and $l(v_1) = l(v_2) = b$.

  Let P_1 be the longest single branch path containing the arc $\langle u_1, v_1 \rangle$. We claim that for any

$l' \in M(H, G, p)$, there must be some arc $\langle x, y\rangle$ on P_1 such that $l'(x) = a$ and $l'(y) = b$. We differentiate four cases.

First, if both u_1 and v_1 have more than one child or u_1 has more than one child and v_1 is a leaf, then $P_1 = \langle u_1, v_1\rangle$. In this case, for any $l' \in M(H, G, p)$, $l'(u_1) = l(u_1)$ and $l'(v_1) = l(v_1)$. Thus the claim is true.

Second, if u_1 has more than one child and v_1 is an internal vertex with one child, then $P_1 = u_1 \rightarrow z_1$ where z_1 is the first descendant of v_1 with more than one child or a leaf. Let $P'_1 = a \rightarrow l(z_1)$ in H be the corresponding path of P. Recall that we update $l(v_1)$ from the value $l(z_1)$ to b in Line 39 because $\langle a, b\rangle$ is an arc on P'. Therefore the claim must be true since $l'(u_1) = l(u_1)$ and $l'(z_1) = l(z_1)$, otherwise transitivity will happen.

Third, if u_1 is an internal vertex with one child and v_1 either has more than one child or is a leaf, then $P_1 = z_1 \rightarrow v_1$ where z_1 is the first ancestor of u_1 with more than one child or the root g with one child. Again we update $l(u_1)$ from the value $l(v_1)$ to a in Line 39 is because $\langle a, b\rangle$ is an arc on the corresponding path $P'_1 = l(z_1) \rightarrow b$ in H. Therefore the claim must be true since $l'(v_1) = l(v_1)$ and $l'(z_1) = l(z_1)$, otherwise transitivity will happen.

Finally when it comes to the case that u_1 and v_1 are both the internal vertices with one child, then $P_1 = z_1 \rightarrow z_2$ where z_1 is the first ancestor of u_1 with more than one child or the root g with one child, and z_2 is the first descendant of v_1 with more than one child or a leaf. Again we update $l(v_1)$ from the value $l(z_2)$ to b, $l(u_1)$ from the value $l(z_2)$ to a in Line 39 is because $\langle a, b\rangle$ is an arc on the corresponding path $P'_1 = l(z_1) \rightarrow l(z_2)$ in H. Therefore the claim must true since $l'(z_1) = l(z_1)$ and $l'(z_2) = l(z_2)$, otherwise transitivity will happen.

Likewise let P_2 be the longest single branch path containing the arc $\langle u_2, v_2\rangle$. We can claim that for any $l' \in M(H, G, p)$, there must be some arc $\langle x, y\rangle$ on P_2 such that $l'(x) = a$ and $l'(y) = b$. Thus any l' will have $r(a, b, l') = 2$ if $r(a, b, l) = 2$, so such l' can not exist.

In conclusion, we prove that $M(H, G, p) = \emptyset$ whenever Algorithm 5.3 outputs NO. The proves of both directions are completed. □

5.3.3 Examples

Example 1: Algorithm 5.3 outputs NO in Line 8.

Given the character tree H and phylogenetic tree (G, p) as presented in Figure 5.16, Algorithm 5.3 outputs NO in Line 8 since it finds $e \in L(H)$ does not appear as a leaf label in (G, p).

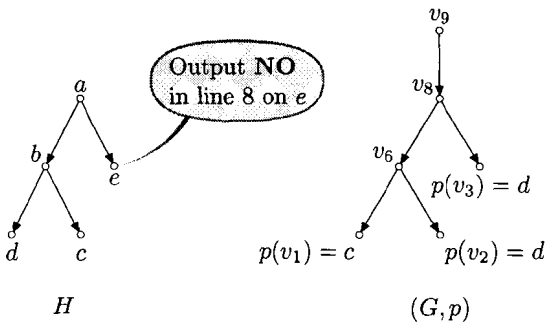


Figure 5.16: The input H and (G,p) to Algorithm 5.3 in example 1.

Example 2: Algorithm 5.3 output NO in Line 20.

Given the character tree H and phylogenetic tree (G,p) as presented in Figure 5.17, Algorithm 5.3 assigns the labels to vertices the same way algorithm 5.1 does except that it checks whether $\langle l(u), l(v)\rangle \notin A(H)$ for each internal vertices with more than one child. In this example, v_6 past the check since both $\langle l(v_6), l(v_1)\rangle \in A(H)$ and $\langle l(v_6), l(v_2)\rangle \in A(H)$. However the algorithm outputs NO after v_7 is checked, because $\langle l(v_7), l(v_4)\rangle = \langle a, c\rangle \notin A(H)$.

Example 3: Algorithm 5.3 outputs NO in Line 35.

Given the character tree H and phylogenetic tree (G,p) as presented in Figure 5.18, Algorithm 5.3 outputs NO in Line 35 while checking the path $P = v_9 \rightarrow v_7$ of length 2. Since v_9 is the root of G and $l(v_9)$ is not the root of H, the corresponding path P' in H is $a \rightarrow b$. Obviously P' is not a single branch path, although $\langle h, e\rangle$ and $\langle e, b\rangle$ can both be realized by updating $l(v_9) := a$ and $l(v_8) := e$, there is no way to realize $\langle e, f\rangle$.

Example 4: Algorithm 5.3 output NO in Line 37.

Given the character tree H and phylogenetic tree (G,p) as presented in Figure 5.19, Algorithm 5.3 outputs NO in Line 37 while checking the path $P = v_8 \rightarrow v_7$ which is also an arc. Since v_8 is the root of G and $l(v_8)$ is not the root of H, the corresponding path P' in H is $a \rightarrow b$. Obviously P' is longer than P, so there is no way to realize both $\langle h, e\rangle$ and $\langle e, b\rangle$ on the path P with the length of 1.

Example 5: Algorithm 5.3 output YES.

Given the character tree H and phylogenetic tree (G,p) as presented in Figure 5.20, Algorithm 5.3 outputs YES at the end.

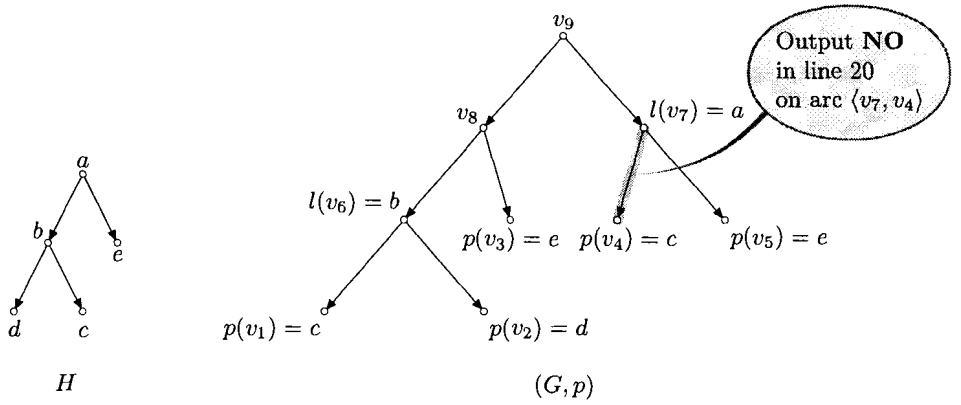


Figure 5.17: The input H and (G, p) to Algorithm 5.3 in example 2.

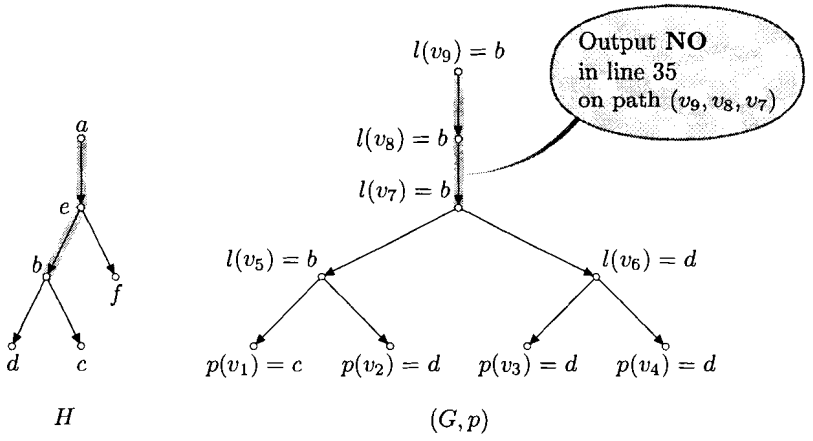


Figure 5.18: The input H and (G, p) to Algorithm 5.3 in example 3.

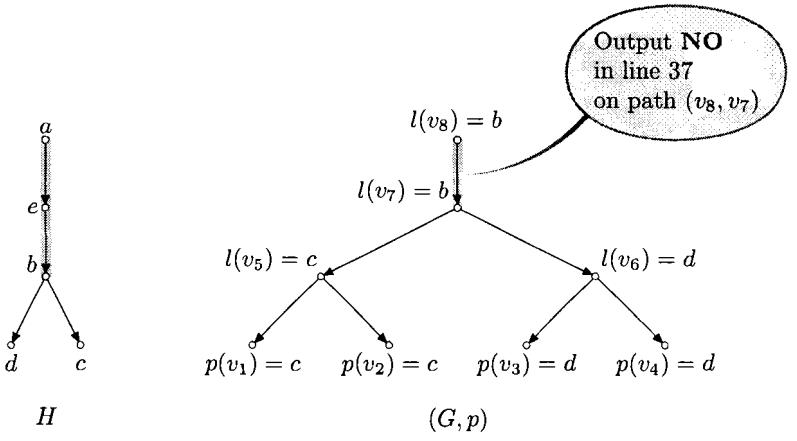


Figure 5.19: The input of H and (G, p) to Algorithm 5.3 in example 4.

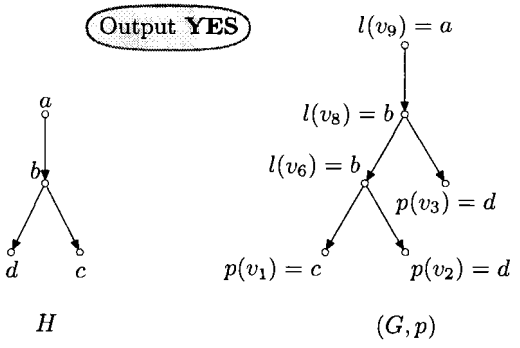


Figure 5.20: The input of H and (G, p) to Algorithm 5.3 in example 5.

Chapter 6

Applications and Experiments

The possible applications of our approach include hypothesis testing of character evolution when a phylogenetic tree is given, and the phylogeny inference when certain character trees are available. In particular, we performed two experiments for hypothesis testing, and one experiment for alternative phylogenetic trees evaluation.

6.1 Testing hypotheses of character evolution

To investigate the evolution of a character, the character should be mapped onto a phylogenetic tree that is constructed independently from other characters. This approach maybe very useful in scenarios where hypothesis of character evolution must be validated.

The first experiment follows from Lipscomb's idea of testing transformation series when there are multiple trees [Lip92]. She believes that if two or more transformation series are proposed for a multistate character and these alternative character state trees result in different cladograms, all transformations should be tested with the congruence criterion (scattering and hierarchical discordance) on all of the trees. She gives an example of such case in Figure 13 of [Lip92] where three transformation series of a character are proposed and two phylogenetic trees obtained from these character transforamtion series. There are seven species involved. The character states of these species are showed in Table 6.1.

In her test, transformation series 1 (Figure 6.1 H_1) is found to be congruent with both phylogenetic trees (Figure 6.1, (G_1, p_1) and (G_2, p_2)), transformation series 2 (Figure 6.1 H_2) is congruent with only one phylogenetic tree, and transformation series 3 (Figure 6.1 H_3) conflicts with both phylogenetic trees. H_3 is therefore eliminated since it is not congruent with both phylogenetic trees.

In our approach, bag cost and arc cost are two metrics to detect separation and transitivity respectively, and furthermore separation and transitivity correspond to Lipscomb's concepts of scattering and hierarchical discordance. In order to check whether our metrics will eliminate the same transformation series, we ran Algorithm 5.1 and Algorithm 5.2 on two phylogenetic trees and three character trees, see Table 6.2 for the output arc costs and bag costs. Under our metrics, the minimum pseudo-minor bag cost of each pair of character tree and phylogenetic tree is the same, but they do have different minimum pseudo-minor arc cost. H_1 has the least arc cost 5, which equals to $|A(H)|$, on both (G_1, p_1) and (G_2, p_2). It is consistent with Lipscomb's analysis that no hierarchical discordance detected in H_1 and either G_1 or G_2. H_3 requires the highest arc cost 9 on both (G_1, p_1) and (G_2, p_2), in other words, the highest hierarchical discordance. Thus, H_3 will also be eliminated by our metrics detecting the same worst tree as Lipscomb.

Species	OUT	A	B	C	D	E	F
State	a	f	f	e	b	c	d

Table 6.1: The character states of seven species.

H	(G_1, p_1)		(G_2, p_2)	
	acost	bcost	acost	bcost
H_1	5	6	5	6
H_2	5	6	6	6
H_3	9	6	9	6

Table 6.2: acost and bcost of two phylogenetic trees (G_1, p_1) and (G_2, p_2) on H_1, H_2 and H_3.

The second experiment was run on the data set from [vT65] and [MKG96] which is a continuation work of [vT65]. The problem of whether behavioral characters could be used to provide accurate estimates of phylogenies was investigated in [MKG96]. Their research was conducted on a behavioral data set for the pelecaniforms based upon van Tets's classic comparative study of these species's social behavior [vT65]. They show that it is possible to investigate the homology of the behavioral characters by mapping the distribution of the characters to best-estimate phylogenetic trees. Once specific example they give is to test the hypotheses about the derivation of male advertising displays proposed by van Tets ([vT65]) based on a best-estimate phylogenetic trees of

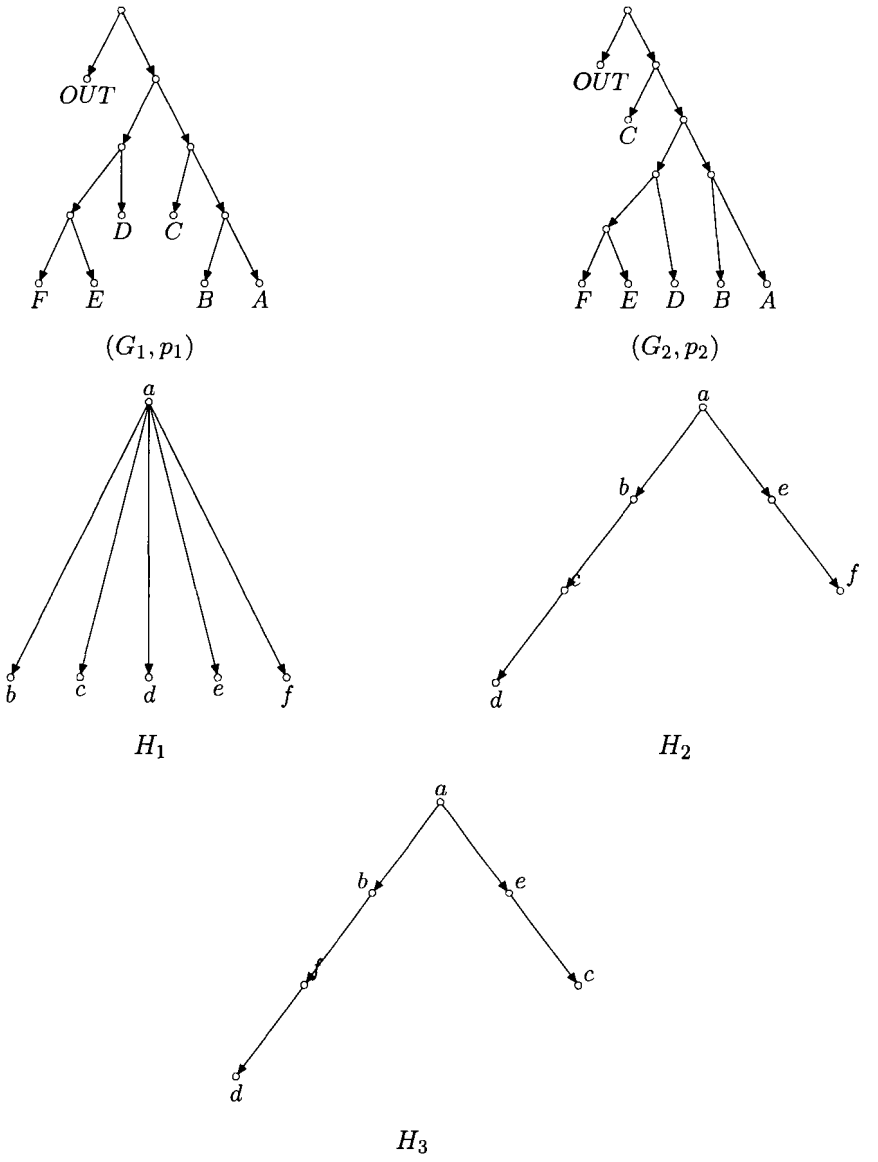


Figure 6.1: Testing character trees when there are multiple phylogenetic trees. (G_1, p_1) and (G_2, p_2) are two phylogenetic trees. The capital letter beside each leaf stands for the species it represents. The character state of each leaf species is shown in Table 6.1. H_1,H_2, and H_3 are three character trees for the same character. Both character trees and phylogenetic trees in this example are taken from [Lip92].

pelecaniforms. In our experiment, we further conducted the same hypothesis test using our metrics. The goodness of the hypothesis is measured in terms of both minimum pseudo-minor arc cost and minimum pseudo-minor bag cost. In addition to three existing hypothesis from [vT65] and [MKG96] ([MKG96], Figure 6), we took another two alternative character trees into consideration.

The best estimate behavioral tree ([MKG96],Figure 3) is used as (G, p); see Figure 6.2, it is a combination of three phylogenetic trees constructed independently by Cracraft (1985), Sibley & Anhlquist (1990) and Siegel-Causey (1988) based on the morphological and genetic data. The character being tested is pre-take-off behavior with 7 states:

(0) general intentional movement

(1) pre-take-off display of the gannets

(2) sky-pointing display of the boobies

(3) slow rate wing-waving display of the great cormorant

(4) rapid flutter wing-waving of pelagic shag

(5) throwback of European shag and

(6) wing-waving display of the darter.

The state of species is shown in the phylogenetic tree (Figure 6.2) beside each leaf. Five alternative character trees are listed in Figure 6.3. We ran Algorithm 5.1 and Algorithm 5.2 on (G, p) and each of the five character trees. The results are listed in Table 6.4.

Character tree (b) and (c) have the best score under our metric. However, it has been justified in [MKG96] that (b) and (c) are not plausible in a biological sense. Thus they agree with van Tets on (a). However, in our experiment, two additional alternative trees (d) and (e) are found to have better score than (a). They are actually much closer to the phylogenetic tree than (a) according to the criteria used by Michevich [Mic82]. One difference between (a) and (d) is that state 0 evolves to 1, and 1 to 2 in (a), but 0 evolves into both 1 and 2 in (b). It is implied in (a) that species in state 1 is closer to species in state 0 than species in state 2. This is not true since species in state 1 and 2 , for example S.sula and M.serrator, are equally close to species in state 0 in the phylogenetic tree; see Figure 6.2. In contrast, (d) reflects the fact implied by the phylogenetic tree that species in state 1 and 2 are equally close to species in state 0. The other difference between (a) and (d) is that 3 evolves into both 4 and 5 in a, but it is the state 4 that evolves into both 3 and 5 in d. Obviously having 4 evolves into 3 and 5 gives most parsimonious result.

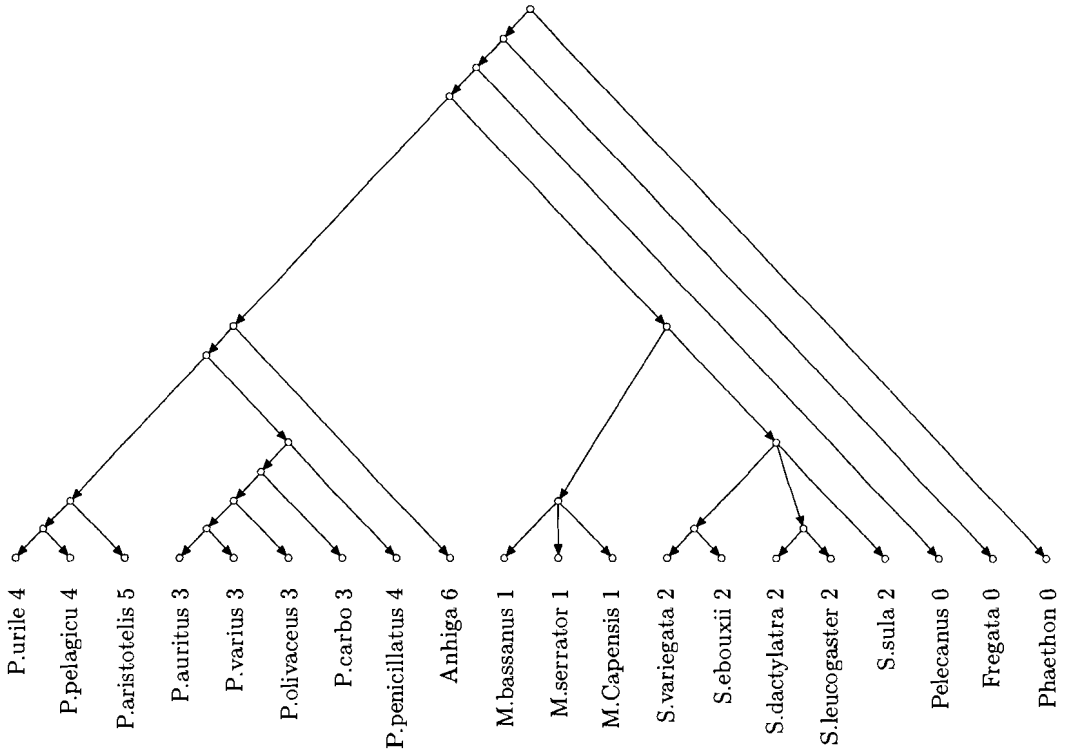


Figure 6.2: The phylogenetic tree of Pelecaniform. The number beside each leaf represents the state of the behavior character that the leaf species is in. Taken from [MKG96].

6.2 Phylogenetic tree inference

Contrasting to testing hypotheses of character evolution, another application of our approach is to evaluate the alternative phylogenetic trees constructed independently with different method or data sets, assuming a character state tree is known for some character that species share.

We investigate the phylogenetic trees for a family of twelve anura species. Three phylogenetic trees constructed independently with different data sources are considered. First tree, (G_1, p_1), is published by Kluge in [KF69]; see Figure 6.4. Second tree, (G_2, p_2), is published by Ford and Cannatella in [FC93], and acknowledged in the Tree of Life project [Tre03]; see Figure 6.5. Both trees are inferred from morphological characters. However, third tree, (G_3, p_3), is inferred from

H	acost	bcost
(a)	8	8
(b)	6	7
(c)	6	7
(d)	7	7
(e)	7	7

Table 6.3: acost and bcost of the phylogenetic tree (Figure 6.2) on five character trees (Figure 6.3).

molecular data [JMHM95]; see Figure 6.6. G_1, G_2 and G_3 mainly differ in the position of Pelobatidae on the tree. G_1 suggests that the common ancestor of Pelobatidae and Pipidae/Rhinophrynidae is also an ancestor of Bufonidae/Atelopodidae/Leptodaectylidae/Hylidae and

Ranidae/Rhacophoridae/Microhylidae. However G_2 implies the opposite point of view. G_3 suggests that the common ancestor of Pelobatidae and Pipidae/Rhinophrynidae is also the ancestor of Ascaphidae/Discoglossidae, while G_1 and G_2 both suggests that the common ancestor of Ascaphidae and Discoglossidae is the ancestor of Pelobatidae and Pipidae/Rhinophrynidae.

The goodness of these three trees are evaluated based on three multistate characters that anura share. The three characters are ribs with four states, vertebral ossification with three states, and pectoral girdle with three states, respectively. The character trees are taken from [Ing67] and [KF69]; see Figure 6.7. We run Algorithm 5.1 and Algorithm 5.2 on each pair of phylogenetic tree and character trees. Table 6.5 shows the running results in terms of minimum pseudo-minor arc cost and minimum pseudo-minor bag cost. (G_1, p_1) has the best score over three phylogenetic trees. It is not surprising since the characters we use in our test are also the part of data Kluge applied to construct (G_1, p_1). (G_3, p_3) was constructed solely based on the DNA and RNA sequences, therefore it does not reflect the morphological characters well and has the highest total arc cost and bag cost.

Species	Characters 1	2	3
Ascaphidae	G	B	C
Discoglossidae	G	B	c'
Pipidae	g	b	c'
Rhinophrynidae	G	b*	C
Pelobatidae	G	b'	c'
Bufonidae	G	b'	c
Atelopodidae	g	b'	c
Leptodactylidae	G	b'	c
Hylidae	G	b'	c
Ranidae	g'	b'	c
Rhacophoridae	g'	b'	c
Microhylidae	g'	b'	c

Table 6.4: Families and character state matrix.

	(G_1, p_1)		(G_2, p_2)		(G_3, p_3)	
	acost	bcost	acost	bcost	acost	bcost
H_1	3	4	3	4	4	5
H_2	3	4	4	5	4	5
H_3	4	5	4	5	4	5
Total:	10	13	11	14	12	15

Table 6.5: acost and bcost of three phylogenetic trees of anura (G_1, p_1), $(G_2, p_2)$and (G_2, p_2) on H_1, H_2 and H_3.

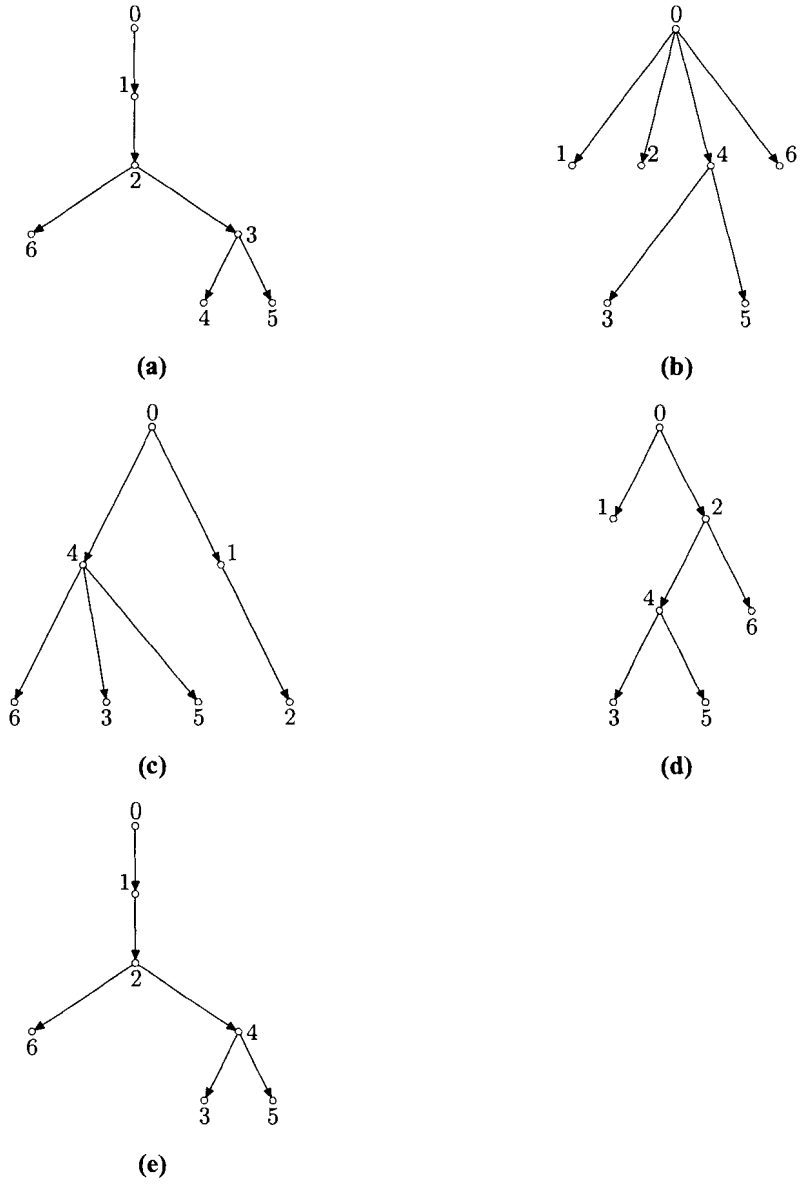


Figure 6.3: The character trees of pre-take-off behavior. (a) is van Tets' hypothesis, (b) and (c) are the two alternative hypotheses listed in [MKG96]. (d) and (e) are our hypothesis which turn out to have better scores than (a).

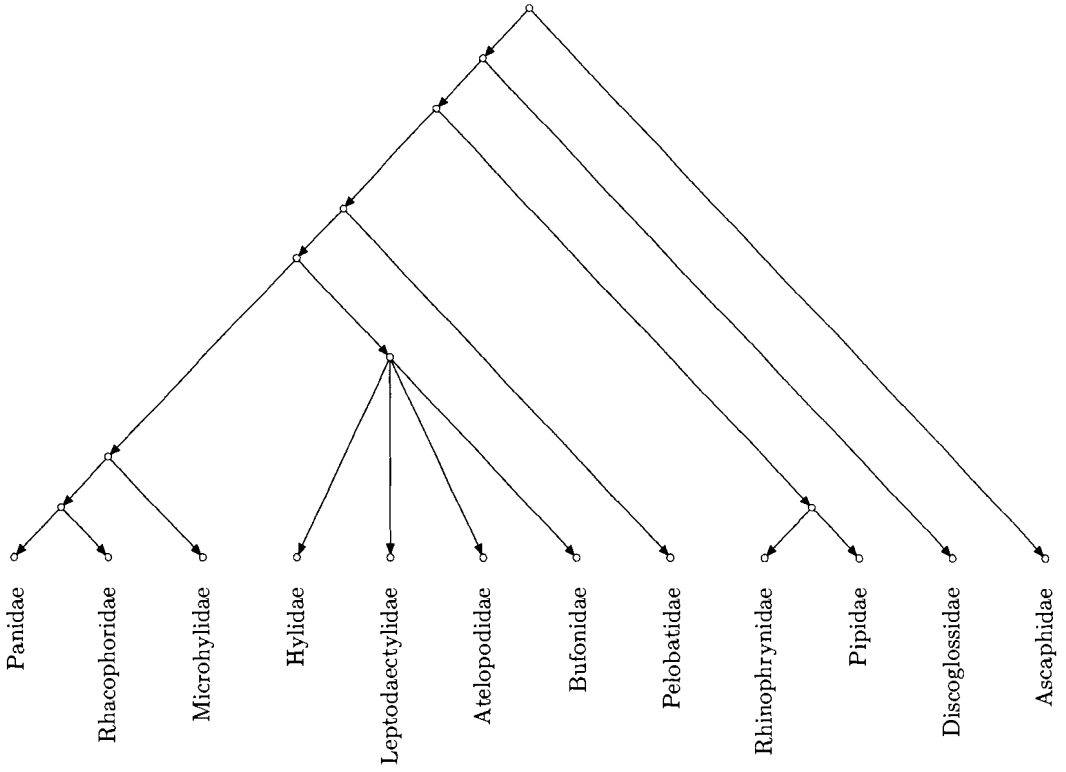


Figure 6.4: (G_1, p_1): the phylogenetic tree of Anura inferred from morphological data [KF69].

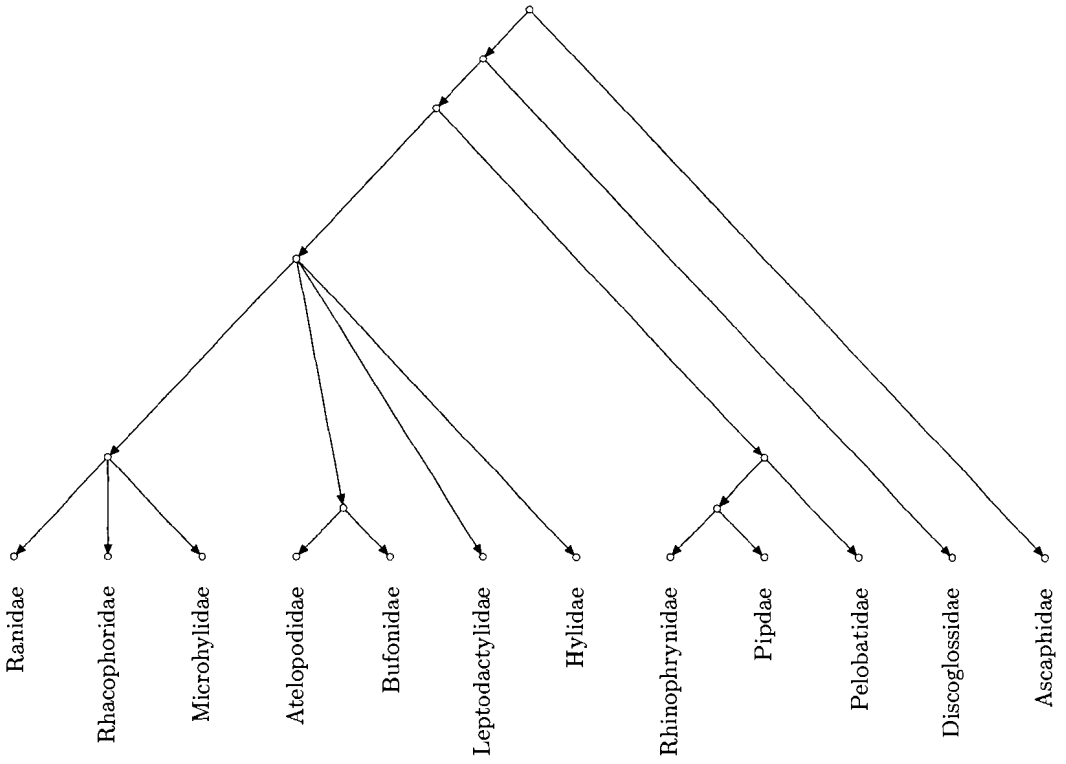


Figure 6.5: (G_2, p_2): the phylogenetic tree of Anura in Tree of Life.

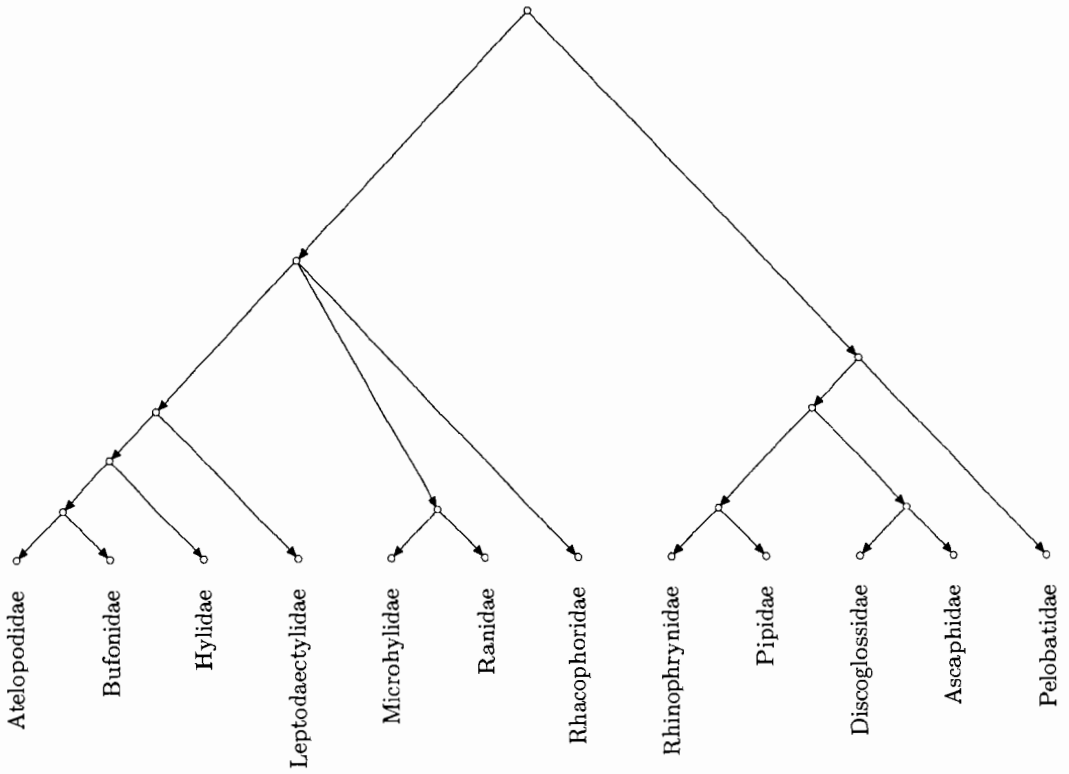


Figure 6.6: (G_3, p_3): the phylogenetic tree of Anura inferred from molecular data [JMHM95].

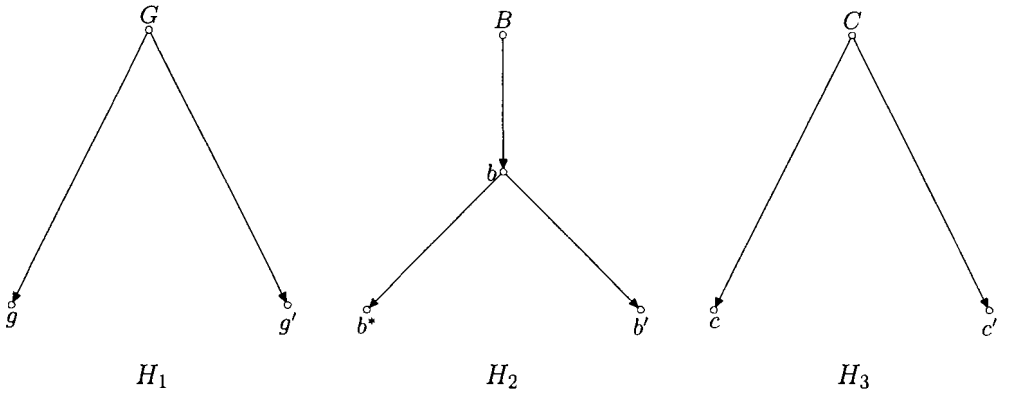


Figure 6.7: Three character trees. H_1 is for the character of Pectoral girdle with three states as well. They are arciferal - G, transitional - g, and firmisternal - g'. H_2 is for the character of ribs with four states. They are free in both subadults and adults - B, free in subadults, fused in adults - b, fused in both subadults and adults - b', and lost in both subadults and adults - b*. H_3 is for the character of vertebral ossification with three states. They are ectochordal - C, stegochordal - c', and holochorda - c. Taken from [Ing67] and [KF69].

Chapter 7

Conclusions and Open Problems

In this thesis, we have considered the problem of testing alternative transformation series of a multistate character and the issue of constructing phylogenetic trees incorporating character evolution. Since there are many problems induced by coding a multistate character to binary character such as loss of logical dependency between states and the requirement of extra big space, we propose a new approach to work with multistate characters which does not require recoding, and therefore is free of the disadvantages of binary coding. In particular, we extend the small phylogeny problem by assuming a character state tree of a particular character is also given. Five inconsistencies defined between character tree and phylogenetic tree are used as the optimization criteria. When none of the five inconsistencies is allowed, we have shown the problem is essentially rooted tree minor which is **NP**-hard.

In practice inconsistencies do occur. To handle those that arise most often, we introduced two relaxations of rooted minors. Relax minors allow addition and separation while pseudo minors allow transitivity and separation. Extending from Lipscomb's definition of non congruences between characters, we introduced two new metrics, *bag cost* and *arc cost* as the target scoring functions of the problem. From the structure of relax-minor and pseudo-minor and these two score functions, we defined three versions of the extended small phylogeny problem. The minimum relax-minor bag cost was shown to be a **NP**-hard, but both minimum pseudo-minor bag cost and minimum pseudo-minor arc cost have linear time solutions. Since bag cost and arc cost are consistent with Lipscomb's criterion of non congruences between characters, our linear time solutions can also solve her problem of testing transformation series. Based on the experiment conducted with her data, we found that our methods detected the same worst case hypotheses. Another way to utilize our algorithms is to evaluate alternative phylogenetic trees of a same set of species using available

character trees.

Our algorithms are all based on modeling character evolution by rooted trees. A more realistic scenario is to allow a representation by a general Hasse diagram. Two open problems are to characterize the complexity of these problems and to find approximation algorithms for those that are **NP**-hard.

Bibliography

[Ben01] M. J. Benton. Finding the tree of life: matching phylogenetic trees to the fossil record through the 20th century. *Proceedings of the Royal Society of London (B)*, 268:2123–2130, 2001.

[BFC00] M. A. Bender and M. Farach-Colton. The lca problem revisited. *Latin American Theoretical INformatics*, 2000.

[Bro81] D. R. Brooks. Hennig's parasitological method: A proposed solution. *Systematic Zoology*, 30:229–249, 1981.

[CS65] J. H. Camin and R. R. Sokal. A method for deducing branching sequences in phylogeny. *Evolution*, 19:311–326, 1965.

[Doy92] J. J. Doyle. Gene trees and species trees: Molecular systematics as one-character taxonomy. *Systematic Botany*, 17:144–163, 1992.

[DS86] W. I. E. DAY and D. SANKOFF. Computational complexity of inferring phylogenies by compatibility. *Systematic Zoology*, 35:224–229, 1986.

[EOWF91] D. R. Brooks E. O. Wiley, D. Siegel-Causey and V. A. Funk. *The Complete Cladist, A primer of Phylogenetic Procedures*, volume 4: Tree Building and Optimization. The University of Kansas, Museum of Natural History, 1991.

[Far70] J. S. Farris. Methods for computing wagner trees. *Systematic Zoology*, 19:83–92, 1970.

[Far77] J. S. Farris. Phylogenetic analysis under dollo's law. *Systematic Zoology*, 26:77–88, 1977.

[Far82] J. S. Farris. Outgroups and parsimony. *Zoology*, 31:314–320, 1982.

[FC93] L. S. Ford and D. C. Cannatella. The major clades of frogs. *Herpetological Monography*, 7:94–117, 1993.

[Fel81] J. Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[Fit71] W. M. Fitch. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.

[FL82] L. R. Foulds and R. L.Graham. The steiner problem in phylogeny is **np**-complete. *Advances In Applied mathematics*, 3:43–49, 1982.

[FM67] W. M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155:279–284, 1967.

[Hen66] W. Hennig. *Phylogenetic Systematics*. University of Illinois Press, 1966.

[HT84] D. Harel and R. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13:338–355, 1984.

[Ing67] R. F. Inger. The development of a phylogeny of frogs. *Evolution*, 21:369–384, 1967.

[JAHS97] C. E. Hughes J. A. Hawkins and R. W. Scotland. Primary homology assessment, characters and character states. *Cladistics*, 13:275–283, 1997.

[JMHM95] S. B. Hedges J. M. Hay, I. Ruvinsky and L. R. Maxson. Phylogenetic relationships of amphibian families inferred from dna sequences of mitochondrial 12s and 16s ribosomal rna genes. *Molecular Biology of Evolution*, 12:928–937, 1995.

[KF69] A. K. Kluge and J. S. Farris. Quantitative phyletics and the evolution of anurans. *Systematic Zooloy*, 18:1–32, 1969.

[Lip92] D. L. Lipscomb. Parsimony, homology and the analysis of multistate characters. *Cladistics*, 8:45–65, 1992.

[Mad93] W. P. Maddison. Missing data versus missing characters in phylogenetic analysis. *Systematic Biology*, 42:576–581, 1993.

[Mic82] M. F. Mickevich. Transformation series analysis. *Systematic Zoology*, 31:461–478, 1982.

[MJBH00] M. A. Willis M. J. Benton and R. Hitchin. Quality of the fossil record through time. *Nature*, 403:534–537, 2000.

[MKG96] H. G. Spencer M. Kennedy and R. D. Gray. Hop, step and gape: do the social displays of the pelecaniformes reflect phylogeny? *Animal Behaviour*, 51:273–291, 1996.

[ML91] M. F. Mickevich and D. L. Lipscomb. Parsimony and the choice between different transformations for the same character set. *Cladistics*, 7:111–139, 1991.

[(mo98] A. Smith (moderator). Is the fossil record adequate. *Nature on-line debates*, 1998.

[MR92] J. Matousek and R.Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Journal of Algorithms*, 108:343–364, 1992.

[MSSdJ01] O. Madsen M. J. Stanhope M. S. Springer, E. C. Teeling and W. W. de Jong. Integrated fossil and molecular data reconstruct bat echolocation. *Proc. Natl. Acad. Sci. USA*, 98:6241–6246, 2001.

[MW90] M. F. Mickevich and S. Weller. Evolutionary character analysis: Tracing character change on a cladogram. *Cladistics*, 6:137–170, 1990.

[OD87] R. T. O'Grady and G.B. Deets. Coding mulitistate characters, with special reference to the use of parasites as characters of their hosts. *Systematic Zoology*, 36:268–279, 1987.

[PM90] M. Pogue and M. F. Michevich. Character definitons and character state delineations: the bete noire of phylogenetics. *Cladistics*, 6:365–369, 1990.

[RS86] N. Robertson and P. D. Seymour. Graph minors ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[San75] D. D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28:35–42, 1975.

[SC83] D. Sankoff and R. Cedergren. *Simultaneous comparisons of three or more sequences related by a tree, in D. Sankoff and J. Kruskal (eds), Time Warp, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison.* Addison Wesley, Reading Mass, 1983.

[SI89] N. Saitou and T. Imanishi. Relative efficiencies of the fitch-margoliash, maximum parsimony, maximum likelihood, minimum-evolution, and neighbor-joining methods of phylogenetic tree construction in obtaining the correct tree. *Journal of Molecular Evolution*, 6:514525, 1989.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[SO90] D. L. Swofford and G. J. Olsen. Phylogeny reconstruction. *Molecular Systematics*, 1990.

[Sob88] E. Sober. *Reconstructing the past: parsimony, evolution, and inference.* MIT Press, Cambridge, MA, 1988.

[Tre03] Tree of life web project. *http://tolweb.org/*, April, 2003.

[vT65] G. F. van Tets. A comparative study of some social communication patterns in the pelecaniformes. *Ornithology Monograph*, 2:1–88, 1965.

[YTM94] N. Takezaki Y. Tateno and M.Nei. Relative efficiencies of the maximum-likelihood, neighbor-joining and maximum-parsimony methods when substitution rate varies with site. *Journal of Molecular Evolution*, 11:261–277, 1994.