Path Costs in Evolutionary Tree Reconstruction

Dorit S. Hochbaum *
Department of Industrial Engineering and Operations Research and
Walter A. Haas School of Business,
University of California, Berkeley
email: dorit@hochbaum.berkeley.edu

Anu Pathria †
Department of Industrial Engineering and Operations Research,
University of California, Berkeley
email: pathria@cimsim.berkeley.edu

February 10, 1997

Abstract

This paper describes a dynamic programming algorithm to solve a family of problems in the reconstruction of evolutionary trees from protein sequence data, that of constructing "minimal" colorings. This dynamic programming formulation can be modified to efficiently enumerate the number of minimal colorings, and thereby be used to calculate the average cost of any given edge, where the average is taken over the entire set of minimal colorings. An extension of our dynamic programming formulation allows for the calculation of average *path* costs amongst all minimal colorings. Our results resolve questions raised in [HP87]; in particular, we develop polynomial time procedures to find the minimum, maximum, and average (expected) cost of an edge, and more generally of a path, for a minimal coloring. Our algorithm is distinguished in its flexibility to address further distribution and statistical questions relating to the minimal colorings.

Furthermore, the more general concept of calculating statistics describing the set of optimal solutions may be of interest in other combinatorial problems.

Keywords: dynamic programming, evolutionary trees, enumeration, minimal colorings.

*Research supported in part by ONR contract N00014-91-J-1241. All future correspondence should be directed to Prof. Hochbaum (Tel: 510-642-4998; Fax: 510-642-1403).

†Author supported in part by an *NSERC '67* scholarship provided by the Natural Sciences and Engineering Research Council of Canada, and in part by ONR contract N00014-91-J-1241.

0 Introduction

The problem of reconstruction of evolutionary, or phylogenetic, trees from protein sequence data has been studied for some time now (see [CSE67, Fit71, HP82, HP87]). The problem is addressed in two different set-ups. In one, the *topology* of the tree is to be determined based on given sequences to be assigned to the leaves [CSE67, HP82]. In the other set-up, the tree topology *and* the sequences are given, and the goal is to infer the sequences for the (internal) nodes of the tree [Fit71, HP87]. We address the latter problem and some generalizations thereof.

An evolutionary tree describes the ancestral relationships between different sets of taxa (biological species), with the external nodes (leaves) representing the extant species. It is thought that protein sequences contain information from which ancestral relationships can be deduced; thus, the evolution of these sequences is used to represent the evolution of the taxa.

Conceptually, the goal is to infer something unknown about the past, based on the current known situation. Consider a phylogenetic tree whose leaves, sometimes referred to as pendant nodes, are each labeled with a sequence of characters corresponding to a protein sequence. The objective is to infer the labelings of the remaining nodes. Corresponding to a labeling of the tree, there is an associated cost. If two endpoints of an edge have different labelings, then there is a positive cost assigned to that edge, where the cost is a measure of dissimilarity between the two sequences. As indicated in [HP87], the choices of labels for different character positions are independent, so that overall cost can be calculated by considering the cost associated with each character position independently. Therefore, it suffices to study the reconstruction problem in which each node is assigned a single character (which we refer to as a color). A unit cost is assigned to an edge if its endpoints have a different color, and zero cost is assigned if the endpoints have the same color.

Essentially, the greater the cost of a potential labeling, the less likely it is to actually be correct. In the widely accepted maximum parsimony model introduced in [CSE67], it is assumed that the only valid labelings are ones in which there are the fewest possible transitions between neighboring sequences. That is, the idea is to infer the labeling, or labelings, on the remaining nodes of the tree by assuming that only labelings of minimum cost (i.e. most likely) may be valid. A tree coloring that minimizes the number of bicolored edges is called a *minimal* coloring.

For a good recent survey of the vast literature on the evolutionary tree reconstruction problem, the reader may wish to consult [S+96].

We consider several problems related to minimal evolutionary colorings.

1. The *Minimal Tree Reconstruction* optimization problem can be described as follows: given a binary tree whose leaves are assigned a particular color from a given set of colors, the problem is to color the remaining (internal) nodes such that the resulting tree has minimum cost. The cost of a tree is equal to the number of edges that have their endpoints assigned different colors; an edge that has its endpoints assigned different colors is referred to as a *bicolored* edge, and has a cost of one unit.

   We provide a dynamic programming procedure to solve a slightly generalized version of this problem in which we, first, allow that any nodes (not just leaf nodes) may be preassigned a color, second, do not restrict the trees to be binary, and third, the cost of bicolored edges can be based on any weights. Furthermore, our procedure can be generalized to handle other questions of interest with regards to minimal colorings, as described next.

2. Another question of interest is that of finding the average edge cost of a given edge e in the set of all minimal colorings, where the average edge cost in a phylogenetic tree is defined as the fraction of minimal colorings in which the given edge is bicolored. Provided that one accepts the assumption of constancy of mutation rate across different branches of an evolutionary tree, this statistic can be used as an indicator of relative time intervals (in terms of time taken for the evolution of taxa from the parent node of the edge to the taxa of the child node). In [HP87], the issue of finding the average edge cost[1] of a given edge, e, in the set of all minimal colorings was addressed but only partially resolved. We provide polynomial time procedures, based on our dynamic programming procedure to solve the *Minimal Tree Reconstruction* problem, to determine precisely the average cost of a given edge e.

3. The more general question of determining the average *path* cost (as well as minimum and maximum cost) for any given path, P, in the set of minimal colorings was raised as an open question. As with the average edge cost, these values are of interest as they can be used as indicators of relative (between the two endpoints of the path) time intervals. In this paper we provide an efficient polynomial time algorithm to answer these questions.

Some early literature addressing the minimal tree reconstruction problem on a given binary tree problem includes work by Hartigan [Har73] describing an algorithm minimizing the number of 'mutations'. The algorithm is effectively a dynamic programming algorithm, although not called that, and there is no complexity analysis. It claims to improve on Fitch's algorithm [Fit71] and constructs a minimum mutation fit for a given *binary* tree. More recently, Swofford and Maddison [SM92] solve similarly the same problem.

Rinsma et al. [RHP90] derived results for other more general problems. They used the approach of generating functions to determine the number of minimal colorings that exist for a given tree, weightings and colors, and the average weight of any edge over all such minimal colorings. Carter et al. [CH+90] count the number of colorations of particular values that can be constructed given a tree with some coloring of leaf vertices. They describe few rather complex generating functions that give the distribution, rather than an algorithm that computes the value for each input, as we do here.

Hanazawa et al. [HMN95] solve the problem of minimal coloring for a given tree, and produce such coloring in linear time. They further obtain all colors for internal nodes that are compatible with some minimal coloring in $O(n^2)$ time, where n the number of nodes in the tree. Their approach relies on the linear time median finding algorithm of [BFPRT72] as an essential building block for their algorithm. Note that this algorithm, while having excellent theoretical performance, is rather involved and difficult to program.

Our approach addresses all these questions using a uniform dynamic programming technique. Our algorithm adds to the existing capabilities: it can address reconstruction and minimal colorings given colorings on internal vertices (rather than only leaves); the measure of weights on the edges is arbitrary and can thus accomodate the character stepmatrices of the generalized parsimony criterion ([SM92]), and be used relative to *any* known parsimony criteria; the calculation of average cost, over all minimal colorations, for a *set* of edges on a path is addressed here for the first time; and, the algorithm works for non-binary trees as well.

[1] In [HP87], the term "edge *length*" is used for what we refer to as "edge cost". We will reserve the term "length" to denote the number of edges in a path.

The paper is organized as follows. First, in Section 1, we present a simple dynamic programming procedure that finds a minimal coloring (and its cost). We also develop a similar procedure that calculates the number of minimal colorings. In Section 2 we show how these results allow us to precisely calculate the average cost of a given edge in the set of minimal colorings. We also generalize the dynamic programming procedures so that enough information regarding the set of minimal colorings is preserved to solve the more general problems of calculating the average, minimum, and maximum cost of a given path in the set of minimal colorings. Section 3 discusses some generalizations to the tree reconstruction problem. The final section gives a brief summary of the results presented in this paper; it is also noted that, in addition to the application of our techniques to the evolutionary tree problem itself, we believe that the concept of calculating statistics for the set of optimal solutions may be of interest in other combinatorial optimization problems, and that problems solvable via dynamic programming procedures are particularly amenable to polynomial time algorithms for calculating simple statistics describing the set of optimal solutions.

1 Evolutionary Tree Reconstruction from Sequence Data

We will consider a slightly more general version of the tree reconstruction problem in which we, first, allow that any nodes (not just leaf nodes) may be preassigned a color and, second, do not restrict the trees to be binary. The introduction of general weights as costs is excluded from the description in order to simplify somewhat the exposition, but it can be included with a minor variation. Suppose we have a tree T on n nodes, and a set of colors $\mathcal{C}$. A subset of nodes in the tree are each assigned a prespecified color. The optimization problem is to assign the remaining nodes a color from $\mathcal{C}$ such that the number of bicolored edges in the tree is minimized. Formally:

Problem Name: <u>*Minimal Tree Reconstruction*</u>

Instance: *Given a tree $T = (V, E)$, a set of colors $\mathcal{C}$, and a subset $V' \subseteq V$ of the nodes of T. For each $v \in V'$, the color of v, denoted $color(v)$, is specified from amongst the colors in $\mathcal{C}$.*

Optimization Problem: *Assign colors from $\mathcal{C}$ to the nodes $V \setminus V'$ so as to minimize the number of bicolored edges; that is, color the remaining nodes so as to minimize the number of edges whose endpoints are assigned different colors.*

The generalization of allowing more than just leaf nodes to be preassigned a color can be useful, depending upon the approach adopted, when calculating the expected edge cost of a minimal coloring, which is a topic of Section 2. In this section, we lay the groundwork by considering the problems of finding the cost of a minimal coloring and enumerating the number of minimal colorings. Our algorithm for computing the minimal coloring cost has some resemblance to that presented in [Fit71], and achieves the same running time; however, it can be modified to also answer the questions raised in [HP87], as outlined in the Introduction. We also note that our algorithms can be easily modified to solve the more general tree reconstruction problem in which different bicolorations have different associated costs.

1.1 Dynamic Programming Formulation

Our dynamic programming approach to the minimal tree reconstruction problem is one that is commonly used in computer science and operations research for solving combinatorial optimization problems on trees (see [Baa88]).

Select an arbitrary node, r, and "root" the tree at r. We define some notation for the tree T, with respect to the choice of r as the *root*. The *parent* of a node $x \neq r$ is the node adjacent to x that is on the unique path from x to r. If y is the parent of x, then x is called a *child* of y; $\Gamma(x)$ denotes the set of all children of x. T_x refers to the subtree of T that has root x and contains all nodes that have x on their unique path in T to r. Those nodes x for which T_x consists only of the node x itself, are referred to as *leaf* or *pendant* nodes. See Figure 1 for an example to make these ideas more concrete.

Let $M_c(x)$ be the cost of a minimal coloring for the subtree T_x, given that x is assigned the color c. An optimal coloring of T_x will be made up of optimal colorings of T_y's, where $y \in \Gamma(x)$. Thus, the values $M_c(x)$ (from which the cost M^* of a minimal coloring can be found) can be calculated from the leaves up to the root via dynamic programming by using the relationships given in Figure 2. A formal proof of correctness is given in Theorem 1.1.

1.2 Enumerating the Set of Minimal Colorings

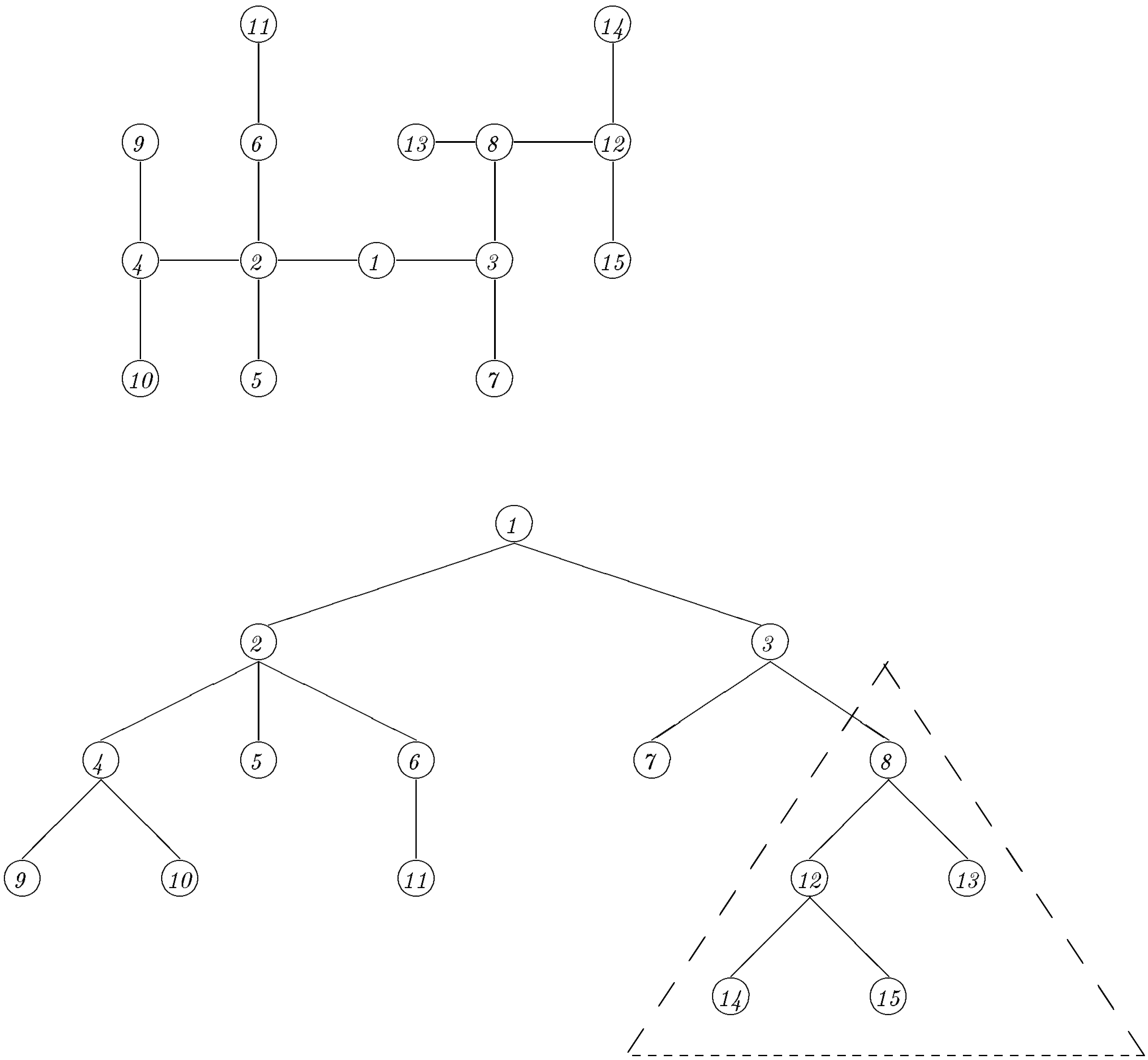
We now show how we can incorporate the calculation of the *number* of minimal colorings into the general dynamic programming scheme just outlined. The basic idea is that a minimal coloring for T_x, conditioned on the fact that $color(x) = c$, is built up from minimal colorings for the subtrees rooted at the children of x. By noting those child colorings in the recurrence that lead to minimal colorings for T_x, the dynamic programming procedure that we have presented for calculating the cost of a minimal coloring tells us which colorings for the children of x lead to minimal colorings for T_x. Suppose we know which colorings of the subtrees of T_x lead to minimal colorings for T_x, which we do via the recurrence for $M_c(x)$; furthermore, suppose we knew how many such colorings of each of the children of x there were. Because the colorings of the children of x are independent, the number of optimal colorings for T_x is the product of the number of such optimal colorings of the children of x. Thus, knowing the number of minimal colorings for the children of x in conjunction with which child colorings lead to minimal colorings for T_x, we can enumerate the number of minimal colorings for T_x.

Let $N_c(x)$ be the *number* of minimal colorings for the subtree T_x, given that x is assigned the color c. A dynamic programming procedure for calculating N^*, the total number of minimal colorings, can be based on the equations presented in Figure 3. Again, a formal proof of correctness is given in Theorem 1.1.

Clearly, the dynamic programming schemes for calculating the cost and enumerating the number of minimal colorings can be combined to run in conjunction; indeed, the enumeration procedure uses results from the cost evaluating procedure as it proceeds.

1.3 Proof of Correctness and Running Time

The calculations of the quantities of interest in the dynamic programming procedures should be carried out in the tree from the pendant nodes up to the root. The correctness of the cost and



The bottom diagram illustrates the rooting of the tree depicted in the top diagram, with node 1 selected as the root. The *leaf* nodes of the tree are $\{5, 7, 9, 10, 11, 13, 14, 15\}$. To clarify some of the other notation: $\Gamma(2) = \{4, 5, 6\}$ are the *children* of node *2*; the *parent* of node *8* is node *3*; T_8 is shown in the dashed triangle.

Figure 1: Example of "rooting" a tree.

Boundary Condition: For x a leaf node ($x \neq r$),

$$M_c(x) = \begin{cases} \infty, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ 0, & \text{otherwise.} \end{cases}$$

Recurrence: For x an internal node,

$$M_c(x) = \begin{cases} \infty, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ \sum_{y \in \Gamma(x)} \min\{M_c(y), M_{\bar{c}}(y)+1\}, & \text{otherwise.} \end{cases}$$

where $M_{\bar{c}}(y) = \min_{k \in \mathcal{C}, k \neq c} \{M_k(y)\}$.

If we let $M(y) = \min_{k \in \mathcal{C}} \{M_k(y)\}$, and observe that $\min\{M_c(y), M_{\bar{c}}(y)+1\} = \min\{M_c(y), M(y)+1\}$, then we get the following more computationally efficient relationship:

$$M_c(x) = \begin{cases} \infty, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ \sum_{y \in \Gamma(x)} \min\{M_c(y), M(y)+1\}, & \text{otherwise} \end{cases}$$

Answer: $M^* = M(r) = \min_{c \in \mathcal{C}} M_c(r)$, where r is the root of the tree T.

Figure 2: Calculating the cost of a minimal coloring.

Boundary Condition: For x a leaf node ($x \neq r$),

$$N_c(x) = \begin{cases} 0, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ 1, & \text{otherwise} \end{cases}$$

Recurrence: For x an internal node,

$$N_c(x) = \begin{cases} 0, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ \prod_{y \in \Gamma(x)} \tilde{N}_c(y), & \text{otherwise.} \end{cases}$$

where $\tilde{N}_c(y)$ is the number of different colorings of T_y that lead to an optimal coloring of T_x, given that x is assigned the color c, where x is the parent of y.

Then, by noting which colorings of y lead to minimal colorings for T_x, given that x is assigned the color c, we find that

$$\tilde{N}_c(y) = \begin{cases} N_c(y), & \text{if } M_c(y) = M(y) \\ N(y), & M_c(y) > M(y)+1 \\ N(y) + N_c(y), & \text{otherwise (i.e. if } M_c(y) = M(y)+1) \end{cases}$$

where $N(y)$ is the sum of those $N_k(y)$ for which $M_k(y) = M(y)$.

Answer:

$$N^* = N(r) = \sum_{c \in \mathcal{C} \,|\, M_c(r) = M^*} N_c(r) \,,$$

where r is the root of the tree T.

Figure 3: Calculating the number of minimal colorings.

enumeration procedures for the calculation of M^* and N^*, respectively, can be proved by establishing the validity of the recurrences given for $N_c(x)$ and $M_c(x)$ in the previous subsections. We can also show that the running time of each procedure is $O(n|\mathcal{C}|)$.

Theorem 1.1 *M^* and N^* can be found in $O(n|\mathcal{C}|)$ time.*

Proof: *Correctness:* We need to establish the validity of the expressions involving $M_c(x)$ and $N_c(x)$. As the boundary expressions can easily be seen to be correct, we consider the main recurrences.

Consider first the calculation of $M_c(x)$, for a given color $c \in \mathcal{C}$ and a non-leaf (internal) node x. For a particular $y \in \Gamma(x)$, consider the minimum number of bicolored edges in the subtree of T_x made up of T_y and the edge $\{x, y\}$, under the assumption that y is assigned the color k: if $k \neq c$, this number is $M_k(y) + 1$; if $k = c$, this number is $M_c(y)$. Thus, the minimum number of bicolored edges in this subtree of T_x, given that x is assigned color c, is given by $\min\{M_c(y), M_{\bar{c}}(y) + 1\}$, where $M_{\bar{c}}(y) = \min_{k\in\mathcal{C},k\neq c}\{M_k(y)\}$. Now, by conditioning on whether or not there is a minimal coloring for T_y such that y is assigned the color c (that is, whether or not $M_c(y) = M(y)$), it is an easy matter to see that $\min\{M_c(y), M_{\bar{c}}(y) + 1\} = \min\{M_c(y), M(y) + 1\}$. The expression for $M_c(x)$ follows by summing over $y \in \Gamma(x)$.

Next we turn to the calculation of $N_c(x)$, for a given color $c \in \mathcal{C}$ and a non-leaf node x. The number of optimal colorings of T_x, given that x is colored c, is given as the product (by independence) of the number of colorings $\tilde{N}_c(y)$ of each T_y (where $y \in \Gamma(x)$) that induce such an optimal coloring of T_x. Thus, we need to find the number of such colorings for each $y \in \Gamma(x)$. By considering three (mutually exclusive, and exhaustive) cases, and taking the product over $y \in \Gamma(x)$, we establish the validity of the recurrence for $N_c(x)$:

(i) if $M_c(y) = M(y)$, then y must be colored c and the number of optimal colorings of T_y is $N_c(y)$;

(ii) if $M_c(y) > M(y) + 1$, then all optimal colorings of T_y can be included (assigning the color c to y will not be lead to an optimal coloring), the number of which is $N(y)$;

(iii) if $M_c(y) = M(y) + 1$, then y can either be colored c or with any color that induces an optimal coloring of T_y, and there are $N(y) + N_c(y)$ such colorings.

Complexity: Assuming that the $M(y)$ values are available, it can be observed that for a given $c \in \mathcal{C}$, calculating the $M_c(x)$ values for all nodes requires a fixed number of operations per edge; since the number of edges in a tree is $O(n)$ and there are a total of $|\mathcal{C}|$ colors, the running time for calculating all $M_c(x)$ values is $O(n|\mathcal{C}|)$. Now, the calculation of $M(y)$ requires $O(|\mathcal{C}|)$ operations for each node y, for a total running time of $O(n|\mathcal{C}|)$ also. Note that $|\mathcal{C}| < n$, for otherwise the problem is trivial, so that this running time is no worse than quadratic in n.

Assuming that the $N(y)$ are available, it can be observed that for a given $c \in \mathcal{C}$, calculating the $N_c(x)$ values for all nodes requires a fixed number of operations per edge; since the number of edges in a tree is $O(n)$ and there are a total of $|\mathcal{C}|$ colors, the running time for calculating all $N_c(x)$ values is $O(n|\mathcal{C}|)$. Now, the calculation of $N(y)$ requires $O(|\mathcal{C}|)$ operations for each node y, for a total running time of $O(n|\mathcal{C}|)$ also. ∎

2 Calculating the Average Path Cost Amongst all Minimal Trees

In [HP87], the problem of calculating the average edge cost, for some edge e, amongst all minimal colorings was raised. There, the issue of whether the average edge cost was 0, 1, or somewhere in between, was resolved; the issue of calculating the average edge cost *exactly* was not. In addition, the more general question of finding the average, minimum, and maximum *path* cost, for some path P, amongst all minimal colorings was raised as an interesting open problem. We show how to efficiently calculate these quantities of interest in this section.

2.1 Paths of Length 1 (Calculating Average Edge Costs)

We can make use of the dynamic programming enumeration procedure presented in the previous section to calculate *exactly* the average, or expected, cost of a particular edge e for a tree taken from the set of all minimal colorings. We suggest a number of approaches. Basically, the idea of each approach is to count the number of minimal colorings for which e is not bicolored (that is, those minimal colorings in which the cost of the edge e is 0). The average cost of e immediately follows by dividing the number of of minimal colorings for which e *is* bicolored by the overall number of minimal colorings.

1. First of all, one can enumerate the number (and cost) of minimal colorings in which e is not bicolored by preassigning both endpoints of e the same color, for each color in $\mathcal{C}$ in turn. Then, because the number (and cost) of minimal colorings without any such additional restrictions on e can be calculated, the fraction of minimal colorings in which the edge e is bicolored is easily found. This procedure has the disadvantage of calling the dynamic programming procedures of the previous section $O(|\mathcal{C}|)$ times, for a total running time of $O(n|\mathcal{C}|^2)$, which is still no worse than cubic in n.

2. A second approach that avoids so many calls to the dynamic programming procedures is to modify the recurrence relation in the procedures so as to force the edge e to not be bicolored. This has the disadvantage of having to modify the procedures; it would seem, from a programming point of view, preferable to use the existing procedures and only modify the input.

3. A third approach, one that seems the most attractive, is to *contract* the edge e so that the node in e that is farther from the root gets collapsed into the node in e that is closer to the root. For example, contracting the edge $\{3, 8\}$ in the example in Figure 1 leads to the tree depicted in Figure 4.

   The idea behind contracting an edge is to force each endpoint of the edge to have the *same* color. Then the procedures of the previous section can be called (one time each) on this modified tree and the results used to calculate the quantity of interest, namely the fraction of minimal colorings in which the contracted edge is bicolored. When contracting the edge e, if both endpoints are preassigned a color then the answer is clearly 0 if the colors are different, and 1 if they are the same; if only one endpoint is preassigned a color then the node formed by contracting e gets assigned that color; otherwise, the node

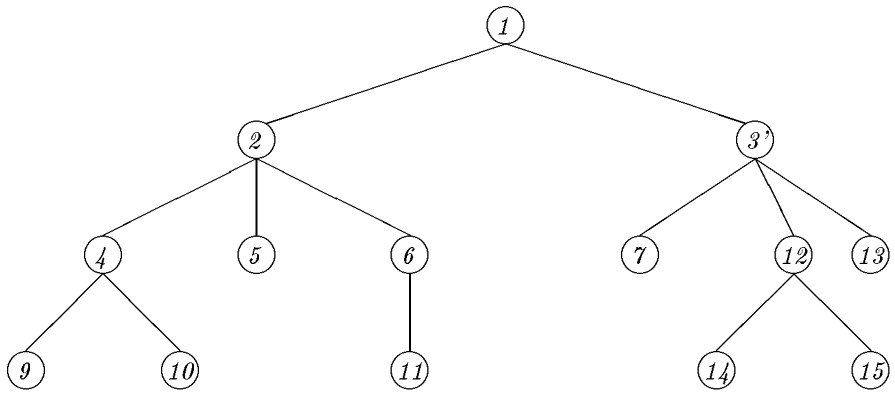


Figure 4: Example of edge contraction.

formed has no color assigned. In the procedure below, it is assumed that we are not in the first easy case (in which both endpoints are preassigned a color):

Algorithm *Average-Edge-Cost*
Input: An instance of the reconstruction problem on tree T, and an edge $e \in E(T)$.
Calculate C and N, the cost and number of minimal colorings for T.
Let T' be the tree formed by contracting e in T.
Calculate C' and N', the cost and number of minimal colorings for T.
if $C' > C$ **then** output 1; **else** output $\frac{N-N'}{N}$

We shift now to the next subsection in which we address the more general problem of average costs for paths of arbitrary length. Of course, the results of the next section can be used to provide an answer to the average edge cost problem since an edge is simply a path of length 1; it should be realized, however, that if the goal is only to calculate average edge costs then the techniques alluded to above may be preferable, for reasons of simplicity, to using the more general procedures that we present next for calculating average path costs.

2.2 Paths of Arbitrary Length

For a given path P we wish to find the average cost of the path in the set of minimal colorings. In other words, we wish to calculate the expected number of bicolored edges on the path P in a minimal coloring. As with edge lengths, path lengths can be used as indicators of relative time intervals. We need to maintain enough information about the trees in the solution set of minimal colorings so that we know how a minimal coloring "typically" behaves on the path of interest. In particular, we generalize the dynamic programming procedures of the previous section so that the calculations condition on the number of bicolored edges in the path P. In

Boundary Condition: For x a leaf node ($x \neq r$),

$$M_c^p(x) = \begin{cases} \infty, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ \infty, & \text{if } p \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

Recurrence: For x an internal node that has no child in P,

$$M_c^p(x) = \begin{cases} \infty, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ \infty, & \text{if } p \neq 0 \\ \sum_{y \in \Gamma(x)} \min\{M_c^0(y), M^0(y)+1\}, & \text{otherwise} \end{cases}$$

where $M^p(y) = \min_{k \in \mathcal{C}} M_k^p(y)$.

For x an internal node that has a child w in P, and $p \in \{0, 1, \ldots, |P|\}$,

$$M_c^p(x) = \begin{cases} \infty, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ \sum_{y \in \Gamma(x), y \neq w} \min\{M_c^0(y), M^0(y)+1\} \\ \quad + \min\{M_c^p(w), M_{\bar{c}}^{p-1}(w)+1\}, & \text{otherwise} \end{cases}$$

where $M_{\bar{c}}^p(w) = \min_{k \in \mathcal{C}, k \neq c} \{M_k^p(w)\}$.

To improve the computational efficiency of the scheme, observe that if we set $M^p(w)$ and $\hat{M}^p(w)$ to be the first and second smallest values, respectively, from the list of $M_k^p(w)$, $\forall k \in \mathcal{C}$, then we get the following relationship:

$$M_{\bar{c}}^{p-1}(w) = \begin{cases} \hat{M}^{p-1}(w), & \text{if } M_c^{p-1}(w) = M^{p-1}(w) \\ M^{p-1}(w), & \text{otherwise} \end{cases}$$

Answer: $M^p(r) = \min_{k \in \mathcal{C}} \{M_k^p(r)\}$, where r is the root of the tree T.

Figure 5: Calculating the cost of a minimal coloring, conditioned on the number of bicolored edges on the path P.

addition to finding the average path cost, P_{ave}, we will be able to easily calculate P_{min} and P_{max}, the minimum and maximum path costs, respectively, of P in the set of minimal colorings. We describe these procedures next.

Root the tree T at *one of the endpoints* of P, say r. Let $M_c^p(x)$ be the optimal minimal coloring value for the subtree T_x, given that x is assigned the color c *and* that there are exactly p bicolored edges in the portion of P that is contained in T_x. The dynamic programming procedure for calculating $M_c^p(x)$ is given in Figure 5. Let $N_c^p(x)$ be the *number* of minimal colorings for the subtree T_x, given that x is assigned the color c and that there are exactly p bicolored edges in the portion of path P in T_x; let $N_{\bar{c}}^p(x)$ be the number of minimal colorings for the subtree T_x, given that x is *not* assigned the color c and that there are exactly p bicolored edges in the portion of path P in T_x; Now, we can calculate $N_c^p(x)$ in conjunction with $M_c^p(x)$ in a manner analogous to the calculation of $N_c(x)$ based upon the dynamic programming scheme for $M_c(x)$, as described in Figure 6. Conceptually, the calculation of $N_c^p(x)$ is nearly identical to that of $N_c(x)$, though the details are somewhat more messy and we need to consider two cases in the main recurrence, depending on whether or not x has a child w in P.

The formal proof of correctness of these procedures is presented in Theorem 2.1.

Boundary Condition: For x a leaf node ($x \neq r$),

$$N_c^p(x) = \begin{cases} 0, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ 0, & \text{if } p \neq 0 \\ 1, & \text{otherwise.} \end{cases}$$

Recurrence: For x an internal node that has no child in P,

$$N_c^p(x) = \begin{cases} 0, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ 0, & \text{if } p \neq 0 \\ \prod_{y \in \Gamma(x)} \tilde{N}_c(y), & \text{otherwise} \end{cases}$$

where, as before, $\tilde{N}_c(y)$ is the number of different colorings of T_y that lead to an optimal coloring of T_x, given that x is assigned the color c, where x is the parent of y.

For x an internal node that has a child w in P, and $p \in \{0, 1, \ldots, |P|\}$,

$$N_c^p(x) = \begin{cases} 0, & \text{if } color(x) \text{ preassigned, and } color(x) \neq c \\ \prod_{y \in \Gamma(x), y \neq w} \tilde{N}_c(y) \\ \times\ (\delta_c^p(w) N_c^p(w) + \delta_{\bar{c}}^{p-1}(w) N_{\bar{c}}^{p-1}(w)), & \text{otherwise} \end{cases}$$

where

$$\delta_c^p(w) = \begin{cases} 1, & \text{if } M_c^p(w) = \min\{M_c^p(w), M_{\bar{c}}^{p-1}(w) + 1\}, \\ 0, & \text{otherwise} \end{cases}$$

and

$$\delta_{\bar{c}}^{p-1}(w) = \begin{cases} 1, & \text{if } M_{\bar{c}}^{p-1}(w) + 1 = \min\{M_c^p(w), M_{\bar{c}}^{p-1}(w) + 1\}, \\ 0, & \text{otherwise.} \end{cases}$$

Now, to improve the computational efficiency for calculating $N_{\bar{c}}^{p-1}(w)$, as c varies over $\mathcal{C}$, observe that

$$N_{\bar{c}}^{p-1}(w) = \begin{cases} \sum_{k \in \mathcal{C} | M_k^{p-1}(w) = M^{p-1}(w)} N_k^{p-1}(w), & \text{if } M_c^{p-1}(w) > M^{p-1}(w) \\ \left[\sum_{k \in \mathcal{C} | M_k^{p-1}(w) = M^{p-1}(w)} N_k^{p-1}(w)\right] - N_c^{p-1}(w), & \text{if } M_c^{p-1}(w) = M^{p-1}(w) = \hat{M}^{p-1}(w) \\ \sum_{k \in \mathcal{C} | M_k^{p-1}(w) = \hat{M}^{p-1}(w)} N_k^{p-1}(w), & \text{otherwise.} \end{cases}$$

And, like before, we note that

$$\tilde{N}_c(y) = \begin{cases} N_c^0(y), & \text{if } M_c^0(y) = M^0(y) \\ N^0(y), & M_c^0(y) > M^0(y) + 1 \\ N^0(y) + N_c^0(y), & \text{otherwise (i.e. if } M_c^0(y) = M^0(y) + 1) \end{cases}$$

where $N^0(y)$ is the sum of those $N_k^0(y)$ for which $M_k^0(y) = M^0(y)$.

Answer:

$$N^p(r) \;=\; \sum_{k \in \mathcal{C}} N_k^p(r) \;,$$

where r is the root of the tree T.

Figure 6: Calculating the number of minimal colorings, conditioned on the number of bicolored edges on the path P.

Let

$$N_p = N^p(r) = \sum_{k \in \mathcal{C} \mid M_k^p(r) = M^*} N_k^p(r),\ p \in \{0, 1, \ldots, |P|\}.$$

N_p is the number of minimal colorings in which the path P has exactly p bicolored edges.

Thus, we can easily calculate the expected, minimum, and maximum cost of the path P in a minimal coloring as,

$$\begin{aligned} P_{ave} &= \frac{\sum_p pN_p}{N^*}, \\ P_{min} &= \min\{p \mid N_p > 0\}, \qquad \text{and} \\ P_{max} &= \max\{p \mid N_p > 0\}. \end{aligned}$$

Notice that, using the values N_p, we can also calculate other statistical measures describing the cost of P in a minimal coloring such as the range, variance, or higher order moments.

2.3 Proof of Correctness and Running Time

We next establish the correctness and complexity of the dynamic programming schemes just presented.

Theorem 2.1 *M^* and N^* can be found in $O((n + |P|^2)|\mathcal{C}|)$ time.*

Proof: *Correctness:* We need to establish the validity of the expressions involving $M_c^p(x)$ and $N_c^p(x)$. As the boundary expressions can easily be seen to be correct, we consider the main recurrences. The correctness of the recurrences for internal nodes x without a child in P follows (with the superscript $p = 0$) as in the proof of Theorem 1.1. Note that if x has a child in P, then $x \in P$ as the tree is rooted at an endpoint of P.

Consider first the calculation of $M_c(x)$, for a given color $c \in \mathcal{C}$ and an internal node x with a child w in P. For a particular $y \in \Gamma(x)$, $y \neq w$, the minimum number of bicolored edges in the subtree of T_x made up of T_y and the edge $\{x, y\}$ is, as in Theorem 1.1, $\min\{M_c^0(y), M^0(y) + 1\}$. For $w \in \Gamma(x)$ such that $w \in P$, the minimum number of bicolored edges in the subtree of T_x made up of T_w and the edge $\{x, w\}$ is given by considering whether or not w is assigned the color c, namely $\min\{M_c^p(w), M_{\bar{c}}^{p-1}(w) + 1\}$, where $M_{\bar{c}}^{p-1}(w) = \min_{k \in \mathcal{C}, k \neq c}\{M_k^{p-1}(w)\}$. By conditioning on whether or not $M_c^{p-1}(w) = M^{p-1}(w)$, it is easy to see that $M_{\bar{c}}^{p-1}(w)$ can take on one of two values: $\hat{M}^{p-1}(w)$ if $M_c^{p-1}(w) = M^{p-1}(w)$; or, $M^{p-1}(w)$ otherwise (i.e. if $M_c^{p-1}(w) > M^{p-1}(w)$). This is computationally advantageous when considering evaluating $M_{\bar{c}}^{p-1}(w)$ for all $c \in \mathcal{C}$. Now, the expression for $M_c(x)$ follows by summing over $y \in \Gamma(x)$.

Next we turn to the calculation of $N_c^p(x)$, for a given color $c \in \mathcal{C}$ and an internal node x with a child in P. The number of optimal colorings of T_x, given that x is colored c, is given as the product (by independence) of the number of colorings of each T_y (where $y \in \Gamma(x)$) that induce an optimal coloring of T_x. The expression giving this number $\tilde{N}_c(y)$ for $y \in \Gamma(x)$, $y \neq w$, follows as in Theorem 1.1. For $w \in \Gamma(x)$ such that $w \in P$, the number of such colorings is given as the contribution conditioned on w being assigned color c (in which case the number is $N_c^p(w)$ if $M_c^p(w) = \min\{M_c^p(w), M_{\bar{c}}^{p-1}(w)+1\}$ and 0 otherwise) plus the contribution conditioned on w being assigned a color other than c (in which case the number is $N_{\bar{c}}^{p-1}(w)$ if $M_{\bar{c}}^{p-1}(w)+1 = \min\{M_c^p(w), M_{\bar{c}}^{p-1}(w)+1\}$ and 0 otherwise, where $N_{\bar{c}}^{p-1}(w) = \min_{k\in\mathcal{C},k\neq c}\{N_k^{p-1}(w)\}$). Now, for computational efficiency, we note that $N_{\bar{c}}^{p-1}(w)$ is the sum of $N_k^{p-1}(w)$ (that is, the number of optimal colorings of T_w with exactly $p-1$ bicolored edges of P, conditioned on the color assigned to w) for certain $k \in \mathcal{C}$, and that this sum can take on at most three different values (as given in Figure 6) as c varies, depending upon whether:

(i) if $M_c^{p-1}(w) > M^{p-1}(w)$, in which case c is not in an optimal coloring;

(ii) $M_c^{p-1}(w) = M^{p-1}(w) = \hat{M}^{p-1}(w)$, in which case the number of optimal colorings with c assigned to w must be removed from the sum;

(iii) otherwise (i.e. $M_c^{p-1}(w) = M^{p-1}(w) < \hat{M}^{p-1}(w)$), in which case all optimal colorings assign color c to w so that we must look at all colorings that give the second fewest number of bicolorations.

Once again, the expression for $N_c(x)$ follows by taking the product over $y \in \Gamma(x)$.

Complexity: As before, the work to evaluate the functions for x a node that *does not* have a child in P is $O(n|\mathcal{C}|)$.

So, consider a node x that does have a child w in P. It is easy to see, using the computationally efficient expressions derived above, that a constant amount of work suffices for each particular $M_c^p(x)$ and $N_c^p(x)$ calculation. Because there are $O(|P|)$ such nodes x, and each such node requires the calculation of $O(|P||\mathcal{C}|)$ corresponding values for $M_c^p(x)$ and $N_c^p(x)$ as p and c vary, the total work required for all such nodes x is $O(|P|(|P||\mathcal{C}|))$. ■

Observe that for $|P| \in O(\sqrt{n})$ the complexity of calculating expected path costs is the same as that presented for calculating average edge costs, namely $O(n|\mathcal{C}|)$. It may be interesting to note that the average over all binary trees of the length of a path between two labelled leaves is $O(\sqrt{n})$ (Theorem 3(a) in [SP93]).

3 Generalizations of the Tree Reconstruction Problem

In this section we briefly discuss some generalizations to the tree reconstruction problem. First, we consider the same reconstruction optimization criteria, that of minimizing the number of bicolored edges, but consider instances where the problem is defined on general graphs rather than only trees. Second, we suggest variations that generalize the original optimization objective and that can be handled, when the input instances are restricted to trees, by dynamic programming procedures similar to the ones already presented in this paper.

3.1 The Reconstruction Problem on General Graphs

The generalization of the reconstruction problem from trees to general graphs is an interesting, and well-studied, combinatorial problem. If we consider the problem, on a general graph, of preassigning colors to some of the nodes with the goal of coloring the remaining nodes so as to minimize the number of bicolored edges, then we have the *K-cut* optimization problem with specified nodes ($K = |\mathcal{C}|$). In the *K-cut* problem, we wish to partition the nodes of the graph into K sets (each set corresponds to a color in the reconstruction problem) such that the number of edges with endpoints in different sets is minimized; in addition, certain nodes are prespecified to be in particular subsets (this corresponds to prespecifying colors to certain nodes).

When $K = 2$ this is the well-known *minimum-cut* problem, which can be efficiently solved by maximum network flow techniques, as well as by other methods (see [HO92, NI92]). For $K \geq 3$, however, this problem is known to be NP-hard for general graphs even when there are no prespecified vertices, although polynomial algorithms have been developed for this problem when K is assumed to be fixed (see [GH94]). If we allow for prespecified vertices, then the problem is NP-hard even for fixed K (see [DJP+94]). As we have seen in this paper, however, the *K-cut* problem with specified nodes can be solved for *any* K in polynomial time (polynomial also in K) on trees (recall that our tree reconstruction procedure applies for trees in which *any* nodes, not just leaves, are preassigned a color).

It may be interesting to see if we can find average edge costs in the set of minimal graphs (i.e. minimal cuts) for classes of graphs, beyond trees, that have polynomial algorithms for solving *K-cut* (for example, *K-cut* with prespecified vertices is solvable in polynomial time for planar graphs, when K is fixed [DJP+94]).

3.2 Generalizing the Reconstruction Problem on Trees

We now consider generalizing the reconstruction objective. Because the problem as stated is NP-hard on general graphs, we will restrict instances for which the generalized problem is to be solved to trees.

We have already allowed for the problem generalization in which any nodes can have preassigned colors; other generalizations of the problem can also be handled:

1. As far as the problem of determining the edge costs of phylogenetic trees is concerned, it may be interesting to vary the costs associated with different bicolorations.

2. We can also consider the generalization where costs (not necessarily equal) are associated with using particular colors.

Generalization 1 enables the handling of generalized parsimony criteria. The dynamic programming tree reconstruction procedure can be easily modified to handle these problems.

4 Summary

In this paper we have considered the problem of reconstructing minimal colorings from sequence data. Our algorithms provide polynomial solutions to several problems of interest raised in [HP87]. In particular, we have addressed the problem of calculating the average edge cost of a given edge in the set of all minimal colorings. Furthermore, we have provided an efficient solution to the more general problem of calculating the average, maximum, and minimum costs of a given path amongst all elements in the set of minimal colorings. These results improve the results in [HP87] that, in polynomial time, could establish the cost of a minimal coloring and establish whether the average cost of given edge e in the set of minimal colorings was 0, 1, or somewhere in between.

We may consider additional inference problems for evolutionary trees. First of all, we may consider the problem of finding the average cost of any specified group of edges in the set of minimal colorings, not simply edges on a specified path. Secondly, perhaps we would like to attach a probability measure to each possible labeling, rather than assuming those of minimum cost are equally and exclusively likely to occur, and then calculate average edge or path costs.

The basic idea that we have employed for calculating statistics about the set of minimal colorings is to exploit our ability to incorporate into the dynamic procedure that calculates the *cost* of a minimal coloring the calculation of the *number* of such minimal colorings. Additionally, we can calculate the number of minimal colorings conditioned on the number of bicolored edges in a specified path.

The idea that dynamic programming procedures on trees can be designed to enumerate (or otherwise quantify) the optimal structures may be interesting in other applications, with different underlying combinatorial problems. Consider some examples. First, one can imagine wishing to infer the likelihood that a certain link was used in a network in getting from point A to point B. Under the assumption that all shortest paths are equally likely to have been used, being able to count the number of optimal paths can be used to calculate the quantity of interest. Second, [Lan94] has described an example from the field of ecology in which food webs are modeled on directed acyclic graphs. Nodes represent species and a node has arcs to all nodes representing one of its prey. The average path length (amongst all paths, not just "optimal" ones) between a given species' node and all nodes of "bottom" species (those without any prey) is sometimes referred to as the *trophic level* of the species and is used as a measure of the complexity of the species in question. A third example in which the enumeration of optimal structures is useful is related to measuring network reliability (see [NSI91]).

Dynamic programming procedures seem to be particularly amenable to modifications that allow for the enumeration, and more detailed statistical description, of the optimal solution set. For more difficult problems, just being able to answer coarser questions similar to *"is the fraction of minimal colorings in which the edge e is bicolored 0, 1, or somewhere in between?"* may be challenging.

Acknowledgement

We wish to thank the anonymous referees and Todd Wareham for their detailed comments and pointers to numerous useful additional references.

References

[Baa88] S. Baase. *Computer Algorithms: Introduction to Design and Analysis.* Addison-Wesley, Reading, Mass., 1988.

[BFPRT72] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest and R.E. Tarjan. Time Bounds for Selection. *Computer Systems Science.* 7, 448–461, 1972.

[CH+90] M. Carter, M.D. Hendy, D. Penny, L.A. Szekely and N.C. Wormald. On the distribution of lengths of evolutionary trees. *SIAM J. on Discrete Math*, 3:1, 38–47, 1990.

[CSE67] L. L. Cavalli-Sforza and A. W. F. Edwards. Phylogenetic Analysis: Models and Estimation Procedures. *Evolution*, 21:550–570, 1967.

[DJP+94] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23(4):864–894, August 1994.

[Fit71] W. M. Fitch. Towards Defining the Course of Evolution. *Syst. Zool.*, 20:406–416, 1971.

[GH94] O. Goldschmidt and D. S. Hochbaum. A Polynomial Algorithm for the κ-Cut Problem for fixed κ. *Mathematics of Operations Research*, 19(1):24–37, February 1994.

[HMN95] M. Hanazawa, H. Narushima, and M. Minaka. Generating most parsimonious reconstructions on a tree: a generalization of the Farris-Swofford-Maddison method. *Discrete Applied Math*, 56, 245–265, 1995.

[Har73] A. Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29, 53–65, 1973.

[HO92] L. Hao and J. B. Orlin. A Faster Algorithm for finding the Minimum Cut in a Graph. In *Proceedings of the Third Annual ACM-SIAM Symposium*, pages 165–174, January 1992.

[HP82] M. D. Hendy and D. Penny. Branch and Bound Algorithms to Determine Minimal Evolutionary Trees. *Mathematical Biosciences*, 59:277–290, 1982.

[HP87] M. D. Hendy and D. Penny. Edge Lengths of Trees from Sequence Data. *Mathematical Biosciences*, 83:157–165, 1987.

[Lan94] D. Landy, 1994. Private Communication.

[NI92] H. Nagamochi and T. Ibaraki. Computing Edge-Connectivity in Multigraphs and Capacitated Graphs. *SIAM Journal on Discrete Mathematics*, 5(1):54–66, February 1992.

[NSI91] H. Nagamochi, Z. Sun, and T. Ibaraki. Counting the Number of Minimum Cuts in Undirected Multigraphs. *IEEE Transactions on Reliability*, 40(610–614), December 1991.

[RHP90] I. Rinsma, M. D. Hendy and D. Penny. Minimally colored trees. *Mathematical Bioscience*. 98, 201–210, 1990.

[SP93] M.A. Steel and D. Penny. Distribution of tree comparison metrics - some new results. *Systematic Biology* 42:2, 126–141, 1993.

[SM92] D.L. Swofford and W.P. Maddison. Parsimony character-state reconstructions and evolutionary inference. In R.L Mayden (ed.) *Systematic histrorical ecolofy, and North Amrican freshwater fishes*. Sinauer Associates, Sunderland, MA, 411–501, 1992.

[S+96] D.L. Swofford G.J. Olsen, P.J. Waddell and D.M. Hillis. Phylogeny Reconstruction. In D.M. Hillis, C. Moritz, and B. K. Mable (eds.) *Molecular Systematics* (Second Edition), Sinauer Associates, Sunderland, MA. pages 407–514, 1996.