Annals of Combinatorics 12 (2008) 123-132
0218-0006/08/010123-10
DOI 10.1007/s00026-008-0341-6

Annals of Combinatorics

Counting Ancestral Reconstructions in a Fixed Phylogeny

Tobias Thierer[1], David Bryant[2, 3], and Mike Steel[4]*

[1]Proteomics Algorithmen und Simulation, ZBIT, Sand 14, 72076 Tübingen, Germany

[2]Department of Mathematics, University of Auckland, Private Bag 92019, Auckland Mail Centre, Auckland 1142, New Zealand

[3]McGill Centre for Bioinformatics, 3775 University Street, Montreal, Quebec, H3A 2B4, Canada

[4]Biomathematics Research Centre, University of Canterbury, Private Bag 4800, Christchurch 8140, New Zealand
M.Steel@math.canterbury.ac.nz

Received July 27, 2005

AMS Subject Classification: 92B05, 92B15, 05C05, 11T06

Abstract. We give formulas for calculating in polynomial time the number of ancestral reconstructions for a tree with binary leaf- and root labels for each number of $0 \to 1$ and $1 \to 0$ arcs. For trees of fixed degree, the corresponding numbers of $0 \to 0$ and $1 \to 1$ arcs can be deduced. We calculate intervals for the relative cost of $0 \to 1$ and $1 \to 0$ transitions over which the same labelings remain the cheapest.

Keywords: generating function, phylogenetic tree, binary character, transition

1. Introduction

The reconstruction of evolutionary trees — theories for the course of evolution and the topology of speciation events — from *character data* on extant species forms an important application of mathematics and computer science in biology (see [5] for an overview).

Information — in our case, binary *character data* coded as 0 and 1 — is typically available only on extant species, which form the leaves of an evolutionary tree. A natural question asks in what ways this leaf-labelling can be extended onto the full tree. The quality of a labelling depends on the number of $0 \to 1$ and $1 \to 0$ transitions that it induces on the tree, as they correspond to mutation events, and the well-known *maximum parsimony* approach aims to find a labelling that minimizes the total number of transitions. Another approach, Dollo Parsimony, allows no more than one $0 \to 1$ transition and is suitable for sufficiently complex characters (e.g., SINEs, [9]) that most

* To whom correspondence should be addressed.

probably arose only once during evolution. This method was first informally suggested by [7], and formally specified by [4].

While maximum parsimony implicitly assumes that $0 \to 1$ and $1 \to 0$ transitions are equally expensive (or implausible), Dollo Parsimony assumes that $0 \to 1$ transitions are sufficiently more expensive as to be prohibited. In this paper, we provide a more general approach that allows for arbitrary relative transition costs. Moreover, these costs need not be known in advance, and our approach yields closed intervals of relative cost and the associated cheapest labelings for each interval.

We also show that for trees of fixed degree, the tree's degree, root label, binary character and two out of the four counts of arcs (one count for each arc type) suffice to calculate the remaining two counts — thus specifically we can infer the number of $0 \to 0$ and $1 \to 1$ arcs from the number of $0 \to 1$ and $1 \to 0$ arcs.

A similar approach that assumed a (not necessarily integer) weighting of $0 \to 1$ and $1 \to 0$ transitions to be known in advance was presented in [8]. Besides calculating the number of labelings possible for each total cost, [8] also provides an algorithm for calculating the average cost that a specific arc contributes, averaged over all the cheapest labelings. See [6] for further applications.

The results from Sections 3 and 4 have already been published in [10].

2. Definitions and Notation

Throughout the paper, $\mathcal{T} = \langle V, E\rangle$ denotes a phylogenetic tree with node set V and a set $E \subseteq V \times V$ of directed arcs, oriented away from the root node $\rho(\mathcal{T}) \in V$. The *outdegree* $\operatorname{outdeg}(v) = |(\{v\} \times V) \cap E|$ of a node v is the number of arcs originating from v. The leaves are the nodes $L(\mathcal{T}) = \{v \in V\colon \operatorname{outdeg}(v) = 0\}$. A binary *character* is a function $c\colon L(\mathcal{T}) \to \{0, 1\}$.

A *labeling* l is a function $l\colon V \to \{0, 1\}$. The symbol $\mathcal{T}_c$ denotes $\mathcal{T}$ with the constraint that the leaves must be labeled as specified by the character c. Similarly, $\mathcal{T}^r$ demands root label r, and $\mathcal{T}_c^r$ combines both constraints. Thus l labels $\mathcal{T}_c^r$ only if $l_{|L(\mathcal{T})} \equiv c$ and $l(\rho(\mathcal{T})) = r$. Under a labeling l, an arc $u \to v \in E$ is called a *transition* if $l(u) \neq l(v)$, and a *constancy* otherwise. Let l label $\mathcal{T}$ with the following number of arcs of each type.

	transition		constancy	
arc type	$0 \to 1$	$1 \to 0$	$0 \to 0$	$1 \to 1$
# such arcs	i	j	i'	j'

In this case l is said to *induce* (i, j) *transitions and* (i', j') *constancies* (*on* $\mathcal{T}$). Note that the order matters — (j, i) transitions is different from (i, j) transitions. Throughout this paper, the symbols i, j, i', j' denote the respective numbers of transitions and constancies in $\mathcal{T}$ under a specific labeling.

Let $n_{\mathcal{T}_c^r}(i, j)$ denote the number of different labelings of $\mathcal{T}_c^r$ that induce precisely (i, j) transitions. We then call the polynomial

$$p_{\mathcal{T}_c^r}(x, y) := \sum_{i,j} n_{\mathcal{T}_c^r}(i, j) \cdot x^i \cdot y^j$$

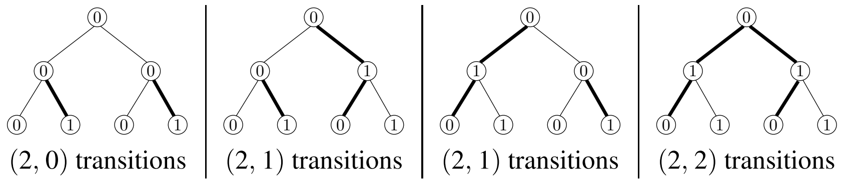


Figure 1: All four possible binary labelings of the two internal nodes (without the root) for a sample tree $\mathcal{T}_c^r$ with an associated leaf-labeling c and label $r = 0$ for the root. Transitions are drawn in bold. The generating function of $\mathcal{T}_c^0$ is $p_{\mathcal{T}_c^0}(x, y) = 1 \cdot x^2y^0 + 2 \cdot x^2y^1 + 1 \cdot x^2y^2$.

the *generating function* of $\mathcal{T}_c^r$. Figure 1 shows an example. We regard $n_{\mathcal{T}_c^r}(i, j)$ as the coefficient table of a two-variate polynomial because this will allow us to view the recursive formula for calculating it as a polynomial addition and multiplication, thus simplifying formulas. Further, if we extend the polynomial to four variables (i.e., count $0 \to 0$ and $1 \to 1$ arcs as well), evaluating it at the relative probabilities of the four arc types yields the likelihood of the tree, i.e., the probability $P(c \,|\, \mathcal{T}^r)$ (see [10]). A similar definition can be made for arbitrary root labels: The polynomial

$$p_{\mathcal{T}_c}(x, y) := p_{\mathcal{T}_c^0}(x, y) + p_{\mathcal{T}_c^1}(x, y) = \sum_{i,j} n_{\mathcal{T}_c}(i, j) \cdot x^i \cdot y^j$$

is called the *generating function* of $\mathcal{T}_c$. It is easy to see that $n_{\mathcal{T}_c}(i, j) = n_{\mathcal{T}_c^0}(i, j) + n_{\mathcal{T}_c^1}(i, j)$ and that, because any binary labeling of $\mathcal{T}_c$ must label the root either 0 or 1, these coefficients $n_{\mathcal{T}_c}(i, j)$ are then the number of different labelings of $\mathcal{T}_c$ that induce precisely (i, j) transitions, with no restriction whether the root label r is 0 or 1.

The polynomials $p_{\mathcal{T}_c}$, $p_{\mathcal{T}_c^0}$, and $p_{\mathcal{T}_c^1}$ are together called the generating functions of $\mathcal{T}_c$. They count only the number of transitions in the labelings of a tree, not the constancies. However, as we will see in Section 4 on page 128, for a tree $\mathcal{T}$ with fixed outdegree for each node, the number of constancies induced by a labeling is completely determined by $\mathcal{T}_c^r$ and the number of transitions.

3. Calculating the Generating Function

3.1. Mathematical Formulas

In this section, we will give a simple recursion formula that allows us to calculate the generating functions of a tree from those of its subtrees. This leads directly to a simple iterative bottom-up algorithm for calculating the generating function.

Theorem 3.1. *Let $\mathcal{T}_c^r$ be a tree with root $v := \rho(\mathcal{T})$.*

a) *If* $\operatorname{outdeg}(v) = 0$ *(i.e., $\mathcal{T}$ consists only of v), then* $p_{\mathcal{T}_c^r}(x, y) = \begin{cases} 0, & \text{if } r \neq c(v), \\ 1, & \text{if } r = c(v). \end{cases}$

b) *For $d := \mathrm{outdeg}(v) > 0$, let $\mathcal{S}_1, \ldots, \mathcal{S}_d$ denote the subtrees rooted at v's children. $\mathcal{S}_m^{r_m}$ denotes the m-th subtree with $\rho(\mathcal{S}_m^{r_m})$ labeled r_m. Then,*

$$p_{\mathcal{T}_c^0}(x, y) = \prod_{m=1}^{d} \left(p_{\mathcal{S}_m^0}(x, y) + x \cdot p_{\mathcal{S}_m^1}(x, y) \right) \tag{3.1}$$

and, similarly,

$$p_{\mathcal{T}_c^1}(x, y) = \prod_{m=1}^{d} \left(p_{\mathcal{S}_m^1}(x, y) + y \cdot p_{\mathcal{S}_m^0}(x, y) \right). \tag{3.2}$$

Proof. Recall that the exponents (i, j) in $p_{\mathcal{T}_c^r}(x, y) := \sum_{i,j} n_{\mathcal{T}_c^r}(i, j) \cdot x^i y^j$ correspond to the number of transitions in a labeling.

a) $\mathcal{T}_c^r$ consists only of v; the only labeling uses root label $r = c(v)$ and induces no transitions.
b) We will consider only equation (3.1), i.e., we assume that v is labeled $r = 0$; the proof for (3.2) is analogous. We give only a short intuitive argument why (3.1) is correct — see [10] for a more formal and more complete proof.

   Without loss of generality, let $v_m := \rho(\mathcal{S}_m)$ be labeled r_m. Let $\mathcal{G}_m^0$ be the subgraph of $\mathcal{T}_c^0$ consisting of v, $\mathcal{S}_m$ and the arc $v \to v_m$ (Figure 2). The arc $v \to v_m$ is a $0 \to 0$ constancy if $r_m = 0$, and a $0 \to 1$ transition otherwise. Thus compared to $\mathcal{S}_m$, $\mathcal{G}_m^0$ has one more $0 \to 1$ transition if and only if $r_m = 1$. Because the number of $0 \to 1$ transitions is encoded as the power of x in the generating function, this corresponds to multiplying $p_{\mathcal{S}_m^1}(x, y)$ with x. Thus because we can label v_m either 0 or 1,

$$p_{\mathcal{G}_m^0}(x, y) = p_{\mathcal{S}_m^0}(x, y) + x \cdot p_{\mathcal{S}_m^1}(x, y).$$

   It remains to show that

$$p_{\mathcal{T}_c^0}(x, y) = \prod_{m=1}^{d} p_{\mathcal{G}_m^0}(x, y). \tag{3.3}$$

   According to the rules of polynomial multiplication, in equation (3.3), all d-tuples of coefficients $n_{\mathcal{G}_m^0}(i_m, j_m)$ are multiplied and the corresponding powers i_m, j_m of x and y added, thus for each such tuple, the resulting coefficient $n_{\mathcal{T}_c^0}(i_1 + \cdots + i_d, j_1 + \cdots + j_d)$ is increased by $\prod_{m=1}^{d} n_{\mathcal{G}_m^0}(i_m, j_m)$. This yields the correct result because any labeling that has (i_m, j_m) transitions in G_m^0 has a total of $(i_1 + \cdots + i_d, j_1 + \cdots + j_d)$ transitions in $\mathcal{T}_c^0$, and all d-tuples of labelings for different subtrees can be combined independently. ∎

Remark 3.2. The generating function can easily be generalised to semi-labeled trees on which the character labels not only the leaves, but also some internal nodes. To forbid a node from being labeled r, simply force $p_{\mathcal{S}^r}(x, y) := 0$ instead of applying the recursion formula for the subtree $\mathcal{S}$ rooted at that node.

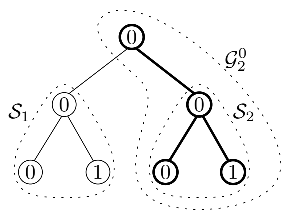


Figure 2: A sample tree $\mathcal{T}_c^r$ with $r = 0$ and the subgraph $\mathcal{G}_2^0$ drawn in bold.

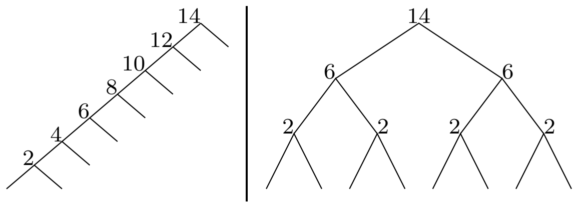


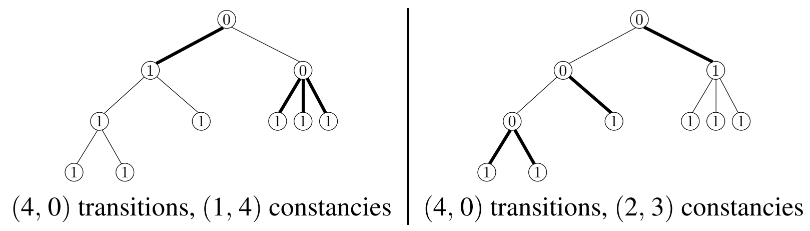
Figure 3: A linear (max. leaf depth $8-1=7$) and a balanced (max. leaf depth $\log_2(8) = 3$) binary tree with 8 leaves each. Each internal node is labeled with the number of arcs in the subtree below it.

3.2. Algorithm

The recursive formula from Theorem 3.1 directly provides a bottom-up algorithm for calculating the generating functions of $\mathcal{T}_c^r$, outlined in pseudo notation in Algorithm GENERATINGFUNCTIONS on page 127.

The runtime (Proposition 3.3) of Algorithm GENERATINGFUNCTIONS depends on the degree of balance in the tree, which can range from a linear to a fully balanced tree (exemplified for binary trees in Figure 3).

```
Algorithm GENERATINGFUNCTIONS
Input: Tree $\mathcal{T} = \langle V, E \rangle$
 1: for all $v \in V$ in a post-order traversal of $\mathcal{T}$ do
 2:     $\mathcal{S}_v :=$ (the subtree rooted at $v$)
 3:     $d := \text{outdeg}(v)$
 4:     $\langle \mathcal{S}_1, \ldots, \mathcal{S}_d \rangle :=$ ($v$'s children)
 5:     if $d = 0$ then {if $v$ is a leaf node}
 6:         $p_{\mathcal{S}_v^{c(v)}}(x, y) := 1$
 7:         $p_{\mathcal{S}_v^{1-c(v)}}(x, y) := 0$
 8:     else {assume $p_{\mathcal{S}_m^r}$ already calculated}
 9:         $p_{\mathcal{S}_v^0}(x, y) := \prod_{m=1}^{d} \left( p_{\mathcal{S}_m^0}(x, y) + x \cdot p_{\mathcal{S}_m^1}(x, y) \right)$
10:         $p_{\mathcal{S}_v^1}(x, y) := \prod_{m=1}^{d} \left( p_{\mathcal{S}_m^1}(x, y) + y \cdot p_{\mathcal{S}_m^0}(x, y) \right)$
11:     end if
12:     $p_{\mathcal{S}_v} := p_{\mathcal{S}_v^0} + p_{\mathcal{S}_v^1}$
13: end for
```



$(4,0)$ transitions, $(1,4)$ constancies | $(4,0)$ transitions, $(2,3)$ constancies

Figure 4: Two labelings of the same $\mathcal{T}_c^r$ with the same number of transitions, but a different number of constancies.

Proposition 3.3. *Consider Algorithm* GENERATINGFUNCTIONS *on the page before.*

a) *The algorithm correctly calculates the generating functions of all subtrees $\mathcal{S}_v$ of $\mathcal{T}$, and finally of $\mathcal{T}$ itself.*
b) $O\left(\sum_{v\in V} w(v)^2 \cdot \log\left(w(v)\right) \cdot \log\left(\mathrm{outdeg}(v)\right)\right)$ *is an upper bound for the algorithm's runtime, where $w(v) := |E(\mathcal{S}_v)|$ is the number of arcs in the subtree rooted at node v.*
c) *For linear binary trees, the runtime is $O\left(|V|^3 \cdot \log|V|\right)$.*
d) *For balanced binary trees, the runtime is $O\left(|V|^2 \cdot \log|V|\right)$.*
e) *Regardless of the tree topology, for any fixed n the lower $n \times n$ coefficients of $p_{\mathcal{T}_c^r}$, corresponding to the number of labelings with some fixed number of $\leq n$ transitions of each type, can be calculated in $O\left(|V| \cdot n^2 \cdot \log n\right)$.*

Proof. Correctness holds because lines 2–12 directly implement the formulas given in Theorem 3.1 on page 125, and a post-order traversal visits all descendants of a node before visiting the node itself. For the runtime bounds, note that the computationally most expensive step is the multiplication of the d factors in lines 9–10, corresponding to equations (3.1) and (3.2). Using fast Fourier transformation and multiplication within the Fourier domain (see [2] for the basic idea, and [1] for an advanced algorithm), this product can be calculated in $O\left(w(v)^2 \cdot \log(w(v)) \cdot \log(d)\right)$ time because the resulting polynomials have no more than $w(v) \times w(v)$ coefficients. The stated runtime for b) follows from summing over all $v \in V$; c) through e) are special cases (note the degree of the intermediate polynomials). ∎

4. Only Two Degrees of Freedom for Trees of Fixed Outdegree

Somewhat surprisingly, if $\mathcal{T}_c^r$ has some fixed outdegree d (i.e., each internal node of $\mathcal{T}$ has exactly d children), then two of the four arc counts i, j, i', j' of a labeling (along with d, r, and c) suffice to calculate the other two. Thus if two labelings of $\mathcal{T}_c^r$ are equivalent in two of their arc counts (e.g., have the same number of transitions of both types), they are also equivalent in the other two. If they differ in any arc counts, they must differ in at least three of them. Theorem 4.1 and Table 1 give the closed formulas.

Note that this does not hold for trees of non-fixed degree: For such trees it is possible for two labelings to differ in just two out of the four arc counts (Figure 4).

Theorem 4.1. *Let $\mathcal{T}_c^r$ be a tree of fixed outdegree d leaf-labeled by a binary character c and with root label $r \in \{0, 1\}$. Let $A := |c^{-1}(0)| + r - 1$ and $B := |c^{-1}(1)| - r$, and, without loss of generality, let l label $\mathcal{T}_c^r$ with (i, j) transitions and (i', j') constancies. Then the relations between the values i, j, i', j', d, A and B that are given in Table 1 hold.*

Proof. The proof is somewhat long and tedious. It is based on showing that there are only two degrees of freedom for the four arc counts of labelings of $\mathcal{T}_c^r$ of fixed degrees. The key to proving this is that besides $i + j + i' + j' = |E|$, the term $(d+1) \cdot (i - j) - (d-1) \cdot (i' - j')$ is also constant over all labelings of $\mathcal{T}_c^r$. Its value is $d \cdot (B - A)$. Further, one can prove by induction on the tree size that $i + j' + r = \frac{j+j'}{d} + c^{-1}(1) = |l^{-1}(1)|$ and $i' + j + (1 - r) = \frac{i+i'}{d} + c^{-1}(0) = |l^{-1}(0)|$. See [10] for the detailed proofs. ∎

given?					
i	j	i'	j'	formula for sought value #1	formula for sought value #2
✔	✔	-	-	$i' = A \cdot \frac{d}{d-1} + \frac{i-d\cdot j}{d-1}$	$j' = B \cdot \frac{d}{d-1} + \frac{j-d\cdot i}{d-1}$
✔	-	✔	-	$j = A + \frac{i+(1-d)\cdot i'}{d}$	$j' = \frac{A+d\cdot B}{d-1} - \frac{(d+1)\cdot i+i'}{d}$
✔	-	-	✔	$j = -d \cdot B + d \cdot i + (d-1) \cdot j'$	$i' = \frac{d\cdot A+d^2\cdot B}{d-1} - (d+1) \cdot i - d \cdot j'$
-	✔	✔	-	$i = -d \cdot A + (d-1) \cdot i' + d \cdot j$	$j' = \frac{d^2\cdot A+d\cdot B}{d-1} - d \cdot i' - (d+1) \cdot j$
-	✔	-	✔	$i = B + \frac{j+(1-d)\cdot j'}{d}$	$i' = \frac{B+d\cdot A}{d-1} - \frac{(d+1)\cdot j+j'}{d}$
-	-	✔	✔	$i = \frac{d\cdot A+d^2\cdot B}{d^2-1} - \frac{i'+d\cdot j'}{1+d}$	$j = \frac{d^2\cdot A+d\cdot B}{d^2-1} - \frac{j'+d\cdot i'}{1+d}$

Table 1: Formulas to calculate the remaining two values if a tree $\mathcal{T}_c^r$ with root label $r \in \{0, 1\}$, fixed outdegree d, binary character c and two out of the four arc counts i, j, i', j' (number of transitions and constancies, respectively) in a labeling are given. A, B are defined as $A := \left|c^{-1}(0)\right| + r - 1$ and $B := \left|c^{-1}(1)\right| - r$.

5. Identifying the Cheapest Labelings

We are interested in the cheapest labelings because they are the most likely, i.e., the biologically most plausible: If the transition weights are chosen as the negative logarithms of the transition probabilities, then the cheapest labeling l also maximises $P(l|\mathcal{T}_c^r)$.

Of course the cheapest labelings are not necessarily those with the most likely transition counts. For example, if $n_{\mathcal{T}_c^r}(5, 5) = 1$ and $n_{\mathcal{T}_c^r}(5, 6) = 10\,000$, then even though the one labeling with $(5, 5)$ transitions is more likely than each of the others, overall it is still more likely that the correct labeling had $(5, 6)$ rather than $(5, 5)$ transitions — because there are 10 000 times more ways the former could occur.

Nevertheless it is helpful to describe the set of the cheapest labelings, and the purpose of this section is to provide such a characterization (Theorem 5.1 below). We begin with some further notation. We will denote the cost of a $0 \to 1$ transition as $w \in [0, 1]$ and the cost of a $1 \to 0$ transition as $1 - w$ (i.e., we scale the costs to a sum of 1, which does not influence those whose labelings are the cheapest). Then the total cost of a labeling with $(i,\, j)$ transitions is:

$$\mathrm{cost}_w(i,\, j) := i \cdot w + j \cdot (1 - w). \tag{5.1}$$

Let $\mathcal{P}$ be the set of possible transition counts,

$$\mathcal{P} := \left\{(i,\, j)\colon n_{\mathcal{T}_c^r}(i,\, j) > 0\right\} \subseteq \mathbb{N}^2.$$

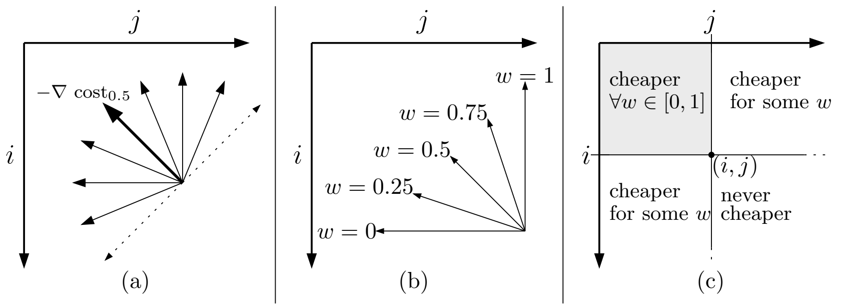


Figure 5: (a) An example for $w = 0.5$ showing the vector $-\nabla\mathrm{cost}_{0.5}(i,\, j)$ and a set of directions in which $\mathrm{cost}_{0.5}(i,\, j)$ decreases. Along the border of the half-plane (dotted arcs), $\mathrm{cost}_{0.5}(i,\, j)$ is constant.
(b) The vector $-\nabla\mathrm{cost}_w(i,\, j) = (-w, w-1)$ indicating the half-plane in which $\mathrm{cost}_w(i,\, j)$ decreases, for some example values of $w \in [0,\, 1]$.
(c) An example $(i,\, j)$ and the sets of numbers $(i_1,\, j_1)$ of transitions that are cheaper for all (shaded area), none (lower right) or some choices of $w \in [0,\, 1]$.

Because the gradient (direction of steepest ascent) of cost_w,

$$\nabla\mathrm{cost}_w := \nabla\mathrm{cost}_w(i,\, j) = \left(\frac{d}{di}\mathrm{cost}_w(i,\, j),\, \frac{d}{dj}\mathrm{cost}_w(i,\, j)\right) = (w,\, 1-w) \tag{5.2}$$

is independent of $(i,\, j)$, $\mathrm{cost}_w(i,\, j)$ becomes minimal for those $(i,\, j) \in \mathcal{P}$ which are the furthest in the direction of the negative gradient, $-\nabla\mathrm{cost}_w = (-w,\, w-1)$. A point $(i_1,\, j_1)$ is further in the direction $-\nabla\mathrm{cost}_w$ than $(i,\, j)$ if the scalar product $-\nabla\mathrm{cost}_w \cdot \binom{i_1 - i}{j_1 - j}$ is strictly positive. Figure 5 illustrates the direction of $-\nabla\mathrm{cost}_w$ as a function of w, in which directions $\mathrm{cost}_w(i,\, j)$ decreases. Because $-\nabla\mathrm{cost}_w$ always points in the direction of decreasing transition counts (to the left and up in Figure 5), then only the $(i,\, j)$ that are on the "upper left" part of $\mathcal{P}$ can correspond to the cheapest labelings. More formally, $(i,\, j) \in \mathcal{P}$ corresponds to a cheapest labeling for some choice of w if and only if $[0,\, i[\times[0,\, j[$ does not intersect the convex hull $\mathcal{H}$ of $\mathcal{P}$, i.e., if and only if

$$(i,\, j) \in \mathcal{P}_{min} := \mathcal{P} \cap \mathcal{H}_{min}, \tag{5.3}$$

where $\mathcal{H}_{min}$ is the "upper left" part of $\mathcal{H}$:

$$\mathcal{H}_{min} := \left\{(i,\, j) \in \mathcal{H}\colon [0,\, i[\times[0,\, j[\, \cap \mathcal{H} = \emptyset\right\}. \tag{5.4}$$

We can then calculate intervals in which the same labelings are the cheapest:

Theorem 5.1. *Let $(i_1, j_1), (i_2, j_2), \dots$ be in clockwise order of $\mathcal{P}_{min}$ as defined in equations* (5.3), (5.4)*. Set $w_0 := 0$, $w_{|\mathcal{P}_{min}|} := 1$ and*

$$w_k := \frac{j_k - j_{k+1}}{j_k - j_{k+1} + i_{k+1} - i_k},$$

for $k \in \{1, \dots, |\mathcal{P}_{min}| - 1\}$. Then (i_k, j_k) corresponds to a cheapest labeling, i.e.,

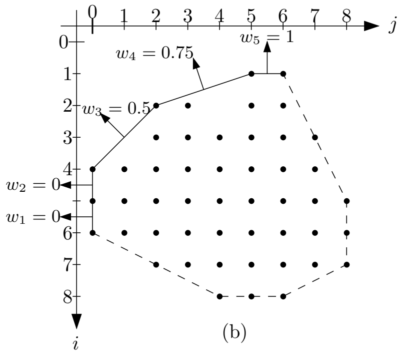
$$(i_k, j_k) \in \arg \min_{(i,j) \in \mathcal{P}_{min}} \operatorname{cost}_w(i, j)$$

if and only if $w \in [w_{k-1}, w_k]$.

Proof. When increasing w from $w_0 = 0$ to $w_{|\mathcal{P}_{min}|} = 1$, the vector $-\nabla \operatorname{cost}_w$ rotates clockwise from left to up (Figure 5 (b)). A labeling with (i_k, j_k) transitions becomes equally expensive as one with (i_{k+1}, j_{k+1}) when $-\nabla \operatorname{cost}_w$ is orthogonal to the line between them, i.e., when $\nabla \operatorname{cost}_w \cdot \binom{i_{k+1} - i_k}{j_{k+1} - j_k} = 0$. This is the case for $w = w_k$ (see equation (5.2)). ∎

$i \backslash j$	0	1	2	3	4	5	6	7	8
0									
1						**1**	**1**		
2			**1**	1		2	2		
3			2	2	2	4	5	3	
4	**1**	3	5	5	6	6	10	8	
5	**1**	2	4	9	11	11	16	11	1
6	**1**	3	7	10	9	14	18	10	2
7			2	5	8	12	10	4	1
8					1	2	1		

(a)



(b)

k	1	2	3	4	5	6
(i_k, j_k)	$(6, 0)$	$(5, 0)$	$(4, 0)$	$(2, 2)$	$(1, 5)$	$(1, 6)$
$[w_{k-1}, w_k]$	$[0, 0]$	$[0, 0]$	$[0, 0.5]$	$[0.5, 0.75]$	$[0.75, 1]$	$[1, 1]$

(c)

Figure 6: (a) Coefficient table $n_{\mathcal{T}_c^0}$ for a sample tree $\mathcal{T}_c^0$ obtained from gene order data (see [3, 10]). The six entries in $\mathcal{P}_{min}$ are bold.
(b) The set $\mathcal{P}$ of coordinates for which the coefficient table has nonzero entries, and its convex hull $\mathcal{H}$. $\mathcal{H}_{min}$ is drawn in a solid line. The arrows indicate the normal vectors $-\nabla \operatorname{cost}_{w_k}$ for the edge $\left[\binom{i_k}{j_k}, \binom{i_{k+1}}{j_{k+1}}\right] \subset \mathcal{H}_{min}$.
(c) The coordinates (i_k, j_k) as they appear in $\mathcal{H}_{min}$ in clockwise order, and the interval $[w_{k-1}, w_k]$ for which (i_k, j_k) corresponds to a cheapest labeling.

If (i_k, j_k) is not a corner of $\mathcal{H}$, then $(i_{k-1}, j_{k-1}) = (i_k, j_k)$. Then, (i, j) corresponds to the cheapest labelings only for exactly one choice of w (Figure 6).

Remark 5.2. The labelings corresponding to coefficients in the generating function can be reconstructed (see [10]). The approach can also be extended to multi-state characters.

Acknowledgments. T. Thierer was supported by the New Zealand Institute of Mathematics and its Applications and by the *Allan Wilson Centre for Molecular Ecology and Evolution.* M. Steel also thanks the New Zealand Marsden Fund.

References

1. V.S. Alagar and D.K. Probst, A fast, low-space algorithm for multiplying dense multivariate polynomials, ACM Trans. Math. Software **13** (1) (1987) 35–57.
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, 2nd Ed., MIT Press, Cambridge, 2001.
3. M.E. Cosner, R.K. Jansen, B.M.E. Moret, L.A. Raubeson, L.S. Wang, T. Warnow, and S.K. Wyman, An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae, In: Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families, D. Sankoff and J. Nadeau, Eds., Kluwer Academic Publishers, Dordrecht (2000) pp. 99–121.
4. J.S. Farris, Phylogenetic analysis under Dollo's law, Syst. Zool. **26** (1) (1977) 77–88.
5. J. Felsenstein, Inferring Phylogenies, Sinauer Associates Inc., Massachusetts, 2004.
6. D.S. Hochbaum and A. Pathria, Path costs in evolutionary tree reconstruction, J. Comput. Biol. **4** (2) (1997) 163–176.
7. W.J. Le Quesne, The uniquely evolved character concept and its cladistic application, Syst. Zool. **23** (1974) 513–517.
8. I. Rinsma, M. Hendy, and D. Penny, Minimally colored trees. Math. Biosci. **98** (1990) 201–210.
9. A.M. Shedlock and N. Okada, SINE insertions: powerful tools for molecular systematics, Bioessays **22** (2000) 148–160.
10. T. Thierer, Generalised and directed characters in phylogenetics, Master's thesis, University of Canterbury, 2004, `http://www.tobias-thierer.de/mscthesis/`.