ALGORITHMS FOR RECONSTRUCTING PHYLOGENETIC TREES FROM DISSIMILARITY MAPS

DAN LEVY, FRANCIS EDWARD SU*, AND RURIKO YOSHIDA

Manuscript, December 15, 2003

Abstract. In this paper we improve on an algorithm by Pachter-Speyer for reconstruction of a phylogenetic tree from its size-m subtree weights. We provide an especially efficient algorithm for reconstruction from 3-weights.

1. Introduction

Let $[n]$ denote the set $\{1, 2, ..., n\}$ and $\binom{[n]}{m}$ denote the set of all m-element subsets of $[n]$. A *m-dissimilarity map* is a function $D : \binom{[n]}{m} \to \mathbf{R}_{\geq 0}$, which may be thought of as a measure of how dissimilar a set of m elements are. Note that the map is non-negative, and we further assume it is symmetric in all the arguments.

In the context of phylogenetic trees, the map $D(i_1, i_2, ..., i_m)$ may measure the weight of a subtree that spans the leaves $i_1, i_2, ..., i_m$.

In this paper, we show how to construct a tree from dissimilarity map information, assuming the map comes from a tree. This follows work of Pachter-Speyer [**?**], who gave theoretical conditions under which a tree may be constructed from knowledge of the dissimilarity map. They show that a tree may be reconstructed from its m-dissimilarities if and only if $n > 2m-2$. Their method proceeds by first finding the splits, then using the fact that the topology can be recovered from the splits, then finding the weights of the edges.

In contrast with their paper, we construct the topology of the tree and its edge weights at the same time.

2. Notation

Instead of writing $D(i_1, i_2, ..., i_m)$, we will often omit the commas and write $D(i_1 i_2 \cdots i_m)$. Also, we will concatenate sets as well as points in this notation, so if R represents a subset of leaves of size $m - 2$, and i, j are other leaves, then we can write $D(ijR)$ for the function D applied to the set $\{i, j\} \cup R$. We write $[ijR]$ for the subtree spanned by the leaves in $\{i, j\} \cup R$. We say $\{i, j\}$ or ij is a *cherry* if there is exactly one intermediate node on the unique path between i and j. We say that ij is a *sub-cherry in the subtree T* if ij is a cherry in the subtree T. When T is understood, we will just say that ij is a sub-cherry. Call the node where ij meets the rest of the tree the *cherry node* of ij.

*Partially supported by NSF Grant DMS-0301129.

The authors thank Bernd Sturmfels and Lior Pachter for encouragement and a seminar that inspired this work.

Four-point condition for m-dissimilarity maps. We now show an analogue of the four-point condition for dissimilarity maps that come from trees.

Theorem 1. *Given a binary tree T, let i, j, k, l be distinct leaves and R a subset of leaves of size $m-2$ not containing i, j, k, l, and D a m-dissimilarity map representing m-subtree weights. Then in the three quantities*

$$\{D(ijR) + D(klR), \quad D(ikR) + D(jlR), \quad D(ilR) + D(jkR)\} \tag{1}$$

the maximum will be achieved at least twice. The maximum will be achieved three times if and only if no pair of leaves in i, j, k, l forms a sub-cherry in $[ijklR]$.

Proof. There are essentially six ways that the subtree $[R]$ could relate to the quartet subtree $[ijkl]$:

(1) The subtree $[R]$ could hang off a leaf of $[ijkl]$.
(2) The subtree $[R]$ could intersect the interior of a leaf of $[ijkl]$.
(3) The subtree $[R]$ could hang off the splitting edge of $[ijkl]$.
(4) The subtree $[R]$ could intersect the interior of the splitting edge of $[ijkl]$.
(5) The subtree $[R]$ could intersect the interiors of the splitting edge and two leaf edges of one sub-cherry of $[ijkl]$.
(6) The subtree $[R]$ could intersect the interiors of all edges of $[ijkl]$.

See Figure **??** for illustrations of these cases.

Only in the final case will all three quantities in (1) be equal. □

As an immediate corollary, we have a criterion that will help us determine whether a pair of leaves i, j form a sub-cherry in a subtree of size $m + 2$.

Theorem 2. *Given a binary tree T, let R be a subset of leaves of size $m - 2$, and i, j, k, l leaves not in R, and D representing m-subtree weights. Then*

$$D(ijR) + D(klR) < D(ikR) + D(jlR) = D(ilR) + D(jkR) \tag{2}$$

if and only at least one of ij or kl is a sub-cherry in $[ijklR]$.

Proof. Consider the six figures in Figure **??** and note that except for the final case, either ij or kl is a sub-cherry of $[ijklR]$. □

Define the difference:

$$d(ij, kl, R) = \frac{1}{2}(D(ikR) + D(jlR) + D(ilR) + D(jkR)) - (D(ijR) + D(klR)). \tag{3}$$

Note that order of the arguments of d matters. One can check that $d(ij, kl, R)$ is symmetric in the first two leaves, the next two leaves, and $d(ij, kl, R) = d(kl, ij, R)$. Also,

Theorem 3. *For any dissimilarity map D and d defined as in (3), the following identity holds:*

$$d(ij, kl, R) + d(ik, jl, R) + d(il, jk, R) = 0. \tag{4}$$

Proof. It is easy to check that the sum of these quantities is 0 by inspecting the right-hand sides:

$$\begin{aligned}
d(ij, kl, R) &= \frac{1}{2}(D(ikR) + D(jlR) + D(ilR) + D(jkR)) - (D(ijR) + D(klR)).\\
d(ik, jl, R) &= \frac{1}{2}(D(ijR) + D(klR) + D(ilR) + D(jkR)) - (D(ikR) + D(jlR)).\\
d(il, jk, R) &= \frac{1}{2}(D(ikR) + D(jlR) + D(ijR) + D(klR)) - (D(ilR) + D(jkR)).
\end{aligned}$$

□

Thus the above identity holds even if the data for the dissimilarity map D does not come from a tree. If the data for D does arise from a tree, then in any identity of the form (4), either all three terms will be zero, or exactly one will be positive and the other two will be negative and equal.

For convenience, we rephrase Theorem 2 in terms of our function d:

Theorem 4. *Let i, j, k, l be leaves of a tree and R a subtree of size $m-2$ that does not contain i, j, k, l. Then $d(ij, kl, R) > 0$ if and only if at least one of ij or kl is a sub-cherry in $[ijklR]$.*

We can use this to locate sub-cherries with respect to R, i.e., test if ij is a sub-cherry in $[ijR]$. First, find some pair kl that is *not* a sub-cherry in $[klR]$ by finding some $d(kl, pq, R) <= 0$. Then we can use $d(ij, kl, R)$ to determine if ij is a sub-cherry in $[ijklR]$, and hence in $[ijR]$.

What is more, the quantity $d(ij, kl, R)$ actually measures something!

Theorem 5. *In a binary tree, if $d(ij, kl, R) > 0$ and if kl is not a sub-cherry in $[klR]$, then $d(ij, kl, R)$ measures the distance between $[klR]$ and the cherry node of ij in $[ijklR]$.*

Proof. This follows from considering the cases in Figure **??**. Case 6 cannot occur because $d(ij, kl, R) > 0$. Cases 3 and 4 cannot occur because kl is a sub-cherry in $[klR]$. In all the other cases, one can check that $d(ij, kl, R)$ measures the distance between $[klR]$ and the cherry node of ij in $[ijklR]$. □

3. Reconstructing trees from 3-dissimilarity maps

We consider the case of reconstructing trees from 3-weights. (In this section, the letter m will refer to a leaf.)

Note that a 5-leaf tree always consists of two cherries and a *middle leaf* along the path between them.

We need the following useful fact:

Theorem 6. *In a binary tree, if $d(ij, kl, m) > 0$, then we have a split $(i, j; k, l)$ and $d(ij, kl, m)$ measures the distance between the cherry nodes of ij and kl in the subtree $[ijklm]$, regardless of the choice of m.*

Proof. The proof is similar to that of Theorem 5. Case 6 cannot occur because $d(ij, kl, R) > 0$. Cases 2, 4, 5. cannot occur because R is size 1. kl is a sub-cherry in $[klR]$. In all the other cases, one can check that $d(ij, kl, R)$ measures the distance between $[klR]$ and the cherry node of ij in $[ijklR]$. □

We can use this idea to reconstruct the tree from 3-weights. Here's the algorithm.

Algorithm 7. (Binary Tree Reconstruction from Three Weights)

(0) Input: n leaves ($n > 4$), all 3-weights of a tree T.
Output: a rooted tree T, as a linked list of nodes, with associated variables at each node that record important information (like parent, children, etc.).

(1) Choose any five leaves i, j, k, l, m. Consider the fifteen quantities:

$$\begin{array}{lll} d(ij,kl,m), & d(ik,jl,m), & d(il,kj,m), \\ d(mj,kl,i), & d(mk,jl,i), & d(ml,kj,i), \\ d(im,kl,j), & d(ik,ml,j), & d(il,km,j), \\ d(ij,ml,k), & d(im,jl,k), & d(il,mj,k), \\ d(ij,km,l), & d(ik,jm,l), & d(im,kj,l). \end{array}$$

(All others possibilities are equal to one of these by the symmetries of the function d.) For any dissimilarity map D, the sum of the numbers in each row will add to 1, by (4). Also, it will be the case that, for any i, j, k, l, m, $d(ij,kl,m) = \frac{1}{2}[d(ij,km,l) + d(ij,ml,k)] + \frac{1}{2}[d(mj,kl,i) + d(im,kl,j)]$. If the data for D comes from a tree, then exactly one quantity in each row will be positive (the other two will be negative and equal); moreover, when these 5 positive quantities are ordered by magnitude, they will satisfy $x_1 > x_2 = x_3 > x_4 = x_5$, where $x_1 = x_3 + x_5$.

Suppose the largest positive quantity is achieved by $d(ij,kl,m)$. Then we know that tree $[ijklm]$ must look like:

FIGURE 1

The picture shows that ij and kl are cherries in $[ijklm]$, with cherry nodes a and b, respectively, and and m is incident to the path between a and b at r The edge $[a, r]$ has weight $d(ij,ml,k) = d(ij,km,l)$ and the edge $[r, b]$ has weight $d(im,kl,j) = d(mj,kl,i)$. (If the data is imperfect, then use the average of $d(ij,ml,k)$ and $d(ij,km,l)$ for the weight of $[a, r]$, and the average of $d(im,kl,j)$ and $d(mj,kl,i)$ for the weight of $[r, b]$. From the observation above, the sum of these averages will be $d(ij,kl,m)$ as desired.)

We may figure out the weights of the remaining leaf edges using this Lemma.

Lemma 8. *If e_i is any leaf edge, the weight of e_i can be computed by choosing leaves j, k, l such that in the subtree $[ijkl]$, ij is a sub-cherry and the edge e_i is precisely the path in $[ijkl]$ from i to the cherry node of ij. Let m be any other leaf. Then the weight of e_i is given by*

$$E(i, j, klm) = \frac{1}{3}(D(ijk) + D(ijl) + D(kli) + D(klj) - d(ij,kl,m)) - D(jkl).$$

Proof. The main idea here is that $E(i, j, klm) = D_4(ijkl) - D(jkl)$, where D_4 is the 4-weight. In any subtree $[ijklm]$, we can determine the 4-weight from the 3-weights and d by this formula:

$$D_4(ijkl) = \frac{1}{3}(D(ijk) + D(ijl) + D(kli) + D(klj) - d(ij,kl,m)).$$

Note that the choice of the leaf m does not matter, because of Lemma 6. □

(2) Now, think of r as the root of this tree and direct all the edges away from r. We can represent this tree by a data structure in which nodes point to other nodes. To each node x (except r) are associated the following variables: $P(x), L(x), R(x), W(x), g_x, p_x, q_x$, which are, respectively, the parent of x, the left child of x, the right child of x, the weight of the edge that points to x, and the name of the descendant of x that follows

the left leaves at every stage; the final two variables are the names of two leaves that lie on two different branches (other than this one) that emanate from the parent of x.

The node r will be represented in 3 different ways depending on which tree it roots—call them r_1, r_2, r_3. Here's how we initialize them:

Let $P(r_1) = a, L(r_1) = b, R(r_1) = null, W(r_1) = null, g_{r_1} = k, p_{r_1} = i, q_{r_1} = j$.

Let $P(r_2) = b, L(r_2) = a, R(r_2) = null, W(r_2) = null, g_{r_2} = i, p_{r_2} = j, q_{r_2} = k$.

Let $P(r_3) = a, L(r_3) = m, R(r_3) = null, W(r_3) = null, g_{r_3} = k, p_{r_3} = i, q_{r_3} = j$.

For the nodes of our initial 5-leaf tree:

Let $P(m) = r_3, L(m) = null, R(m) = null, W(m) = E(m, l, ijk), g_m = m, p_m = i, q_m = k$.

Let $P(a) = r_2, L(a) = i, R(a) = j, W(a) = (d(ij, ml, k) + d(ij, km, l))/2, g_a = i, p_a = l, q_a = m$.

Let $P(b) = r_1, L(b) = k, R(b) = l, W(b) = (d(im, kl, j) + d(mj, kl, i))/2, g_b = k, p_b = m, q_b = i$.

Let $P(i) = a, L(i) = null, R(i) = null, W(i) = E(i, j, klm), g_i = i, p_i = p_a, q_i = j$.

Let $P(j) = a, L(j) = null, R(j) = null, W(j) = E(j, i, klm), g_j = j, p_j = p_a, q_j = i$.

Let $P(k) = b, L(k) = null, R(k) = null, W(k) = E(k, l, ijm), g_k = k, p_k = p_b, q_k = l$.

Let $P(l) = b, L(l) = null, R(l) = null, W(l) = E(l, k, ijm), g_l = l, p_l = p_b, q_l = k$.

This constructs an initial 5-leaf tree on leaves i, j, k, l, m, with root at node r. We still have to figure out where all the other leaves go.

(3) We now define a subroutine $Sub(x, z, t)$ which will take t and determine on which edge below x it lies in the direction of z, where z is a child of x.
   (a) Input: x, z, t
   (b) Output: Places t on an edge below x and modifies data structures appropriately.
   (c) Let $test = d(p_z q_z, tg_z, q_x)$. (For imperfect data, also let $test2 = d(p_z q_z, tg_{R(z)}, q_x)$.)
   If $test > 0$ then:
      (i) If $test < W(z)$ (for imperfect data, check if $W(z) >> test \cong test2$),
      then insert t in the edge from x to z: create node t and fill in the data for t later.
      Create new node y: let $P(y) = x, L(y) = z, R(y) = t, W(y) = test, g_y = g_z, p_y = p_x, q_y = q_z$.
      Check whichever of x's children was z; replace it with y.
      Then fill in the data for t: let $P(t) = y, L(t) = null, R(t) = null, W(t) = E(t, g_z, q_x p_z q_z), g_t = t, p_t = p_x, q_t = g_z$.
      Then re-assign: $P(z) = y, W(z) = W(z) - test, q_z = t$.
      (ii) If $test = W(z)$ (for imperfect data, check if $test2 >> test \cong W(z)$),
      then perform $Sub(z, R(z), t)$.
      (iii) If $test > W(z)$ (for imperfect data, check if $test >> test2 \cong W(z)$),
      then perform $Sub(z, L(z), t)$.
      (iv) Return
   (d) If $test <= 0$ then there's an error, this part of the subroutine should never be reached!
   (e) Return (this should never be reached either).

(4) Now we begin to add leaves to our 5-leaf tree. For any t not yet in the current tree, we test in which of three directions from r it lies.

Let $test = d(it, km, j)$. Let $test2 = d(im, kt, l)$.

If $test > 0$, then perform $Sub(r_2, a, t)$.
If $test2 > 0$, then perform $Sub(r_1, b, t)$.
Otherwise (since both $test1, test2 < 0$) perform $Sub(r_3, m, t)$.
Repeat for each t.

(5) The above process will construct 3 trees with with roots r_1, r_2, r_3. Identifying these three roots, and paste just the left branches of each tree under these roots together will yield the complete tree under the root r.

When all is said and done, this algorithm should be very fast. In fact, the time complexity of this algorithm is $O(n^2)$.

Theorem 9. *The time complexity of Algorithm 7 is $O(n^2)$, where n is the number of leaves of the tree.*

Proof. First we want to show that the number of interior nodes of the tree is $n-2$. Let T be the tree and let $L(T)$ be the set of leaves of T. At Step (2) we have 3 interior nodes and 5 leaves, namely i, j, k, l, m. For each $t \in L(T)\backslash\{i, j, k, l, m\}$, we add one interior node if we find a place for a leaf t. There are $n-5$ many $t \in L(T)\backslash\{i, j, k, l, m\}$. Thus we have $(n-5)+3 = n-2$ many interior nodes. So, we have $2n-2$ nodes in total. Notice that Step (1) and Step (2) take only the time complexity $O(1)$. Also one notices that at Step (3), computing each edge weight takes $O(1)$. Since T is a binary tree with the root which has three children, at Step (4) we call Step (3) recursively to find a place to insert t at most $2n-2$ times for each t. So, inserting t into the tree T takes time complexity $O(n)$. Since there are $n-5$ many $t \in L(T)\backslash\{i, j, k, l, m\}$, we have the total time complexity to reconstruct a tree T from the data $(2n-2) * (n-5) = O(n^2)$. □

It avoids computing splits, Buneman indices, using Splitstree, and then constructing weights, as the Pachter-Speyer paper would have done.

We might even get some data, and check how good this works, comparing reconstruction from 2-weights, to deriving 3-weights from 2-weights and then reconstruct from 3-weights, as Pachter had suggested.

Example 10. Suppose we have $n = 6$ and a binary tree T such as in Figure 2.
Step 1: First we choose leaves 1, 2, 3, 4, 5. Then we have 15 values:

$$d(23, 45, 1),\ d(35, 24, 1),\ d(34, 25, 1),$$
$$d(13, 45, 2),\ d(15, 34, 2),\ d(14, 35, 2),$$
$$d(12, 45, 3),\ d(15, 24, 3),\ d(14, 25, 3),$$
$$d(12, 35, 4),\ d(13, 25, 4),\ d(15, 23, 4),$$
$$d(12, 34, 5),\ d(13, 24, 5),\ d(14, 23, 5).$$

Then we notice that $d(12, 45, 3) = 4$, $d(12, 34, 5) = d(12, 34, 5) = 3$ and $d(13, 45, 2) = d(23, 45, 1) = 1$ and other values are negative. So we know that $(1, 2; 4, 5)$ splits, and we know that the distance between cherry node of 12 and the cherry node of 45 is 4, the distance between cherry node of 13 and the cherry node of 45 is 1, and the distance between cherry node of 12 and the cherry node of 34 is 3. Let $i = 1$, $j = 2$, $m = 3$, $k = 4$, $l = 5$.

Step 2: Initialize: Let $P(r_1) = a, L(r_1) = b, R(r_1) = null, W(r_1) = null, g_{r_1} = 4, p_{r_1} = 1, q_{r_1} = 2$.

FIGURE 2. Example 10

Let $P(r_2) = b, L(r_2) = a, R(r_2) = null, W(r_2) = null, g_{r_2} = 1, p_{r_2} = 2, q_{r_2} = 4$.
Let $P(r_3) = a, L(r_3) = m, R(r_3) = null, W(r_3) = null, g_{r_3} = 4, p_{r_3} = 1, q_{r_3} = 2$.
Let $P(3) = r_3, L(3) = null, R(3) = null, W(3) = E(3, 5, 124) = 1, g_3 = 3, p_3 = 1, q_3 = 4$.
Let $P(a) = r_2, L(a) = i, R(a) = j, W(a) = (d(12, 35, 4) + d(12, 34, 5))/2 = 3, g_a = 1, p_a = 5, q_a = 3$.
Let $P(b) = r_1, L(b) = k, R(b) = l, W(b) = (d(13, 45, 2) + d(23, 45, 1))/2 = 1, g_b = 4, p_b = 3, q_b = 1$.
Let $P(1) = a, L(1) = null, R(1) = null, W(1) = E(1, 2, 453) = 4, g_1 = 1, p_1 = p_a = 5, q_1 = 2$.
Let $P(2) = a, L(2) = null, R(2) = null, W(2) = E(2, 1, 453) = 2, g_2 = 2, p_2 = p_a = 5, q_2 = 1$.
Let $P(4) = b, L(4) = null, R(4) = null, W(4) = E(5, 4, 123) = 1, g_4 = 4, p_4 = p_b = 3, q_4 = 5$.
Let $P(5) = b, L(5) = null, R(5) = null, W(5) = E(5, 4, 123) = 4, g_5 = 5, p_5 = p_b = 3, q_5 = 4$.

Step 4: For $t = 6$, we have $test = d(16, 43, 2) < 0$ and $test2 = d(13, 46, 5) = 1 > 0$. So we proceed $Sub(r_1, b, 6)$.

$Sub(r_1, b, 6)$: $x = r_1$, $z = b$, $t = 6$. So we have $test = d(31, 64, 2) = 1$ and $W(b) = 1$. This means that since $test = W(b)$, we proceed $Sub(b, R(b), 6) = Sub(b, 5, 6)$.

$Sub(b, 5, 6)$: $x = b$, $z = 5$, $t = 6$. So we have $test = d(34, 65, 1) = 1$ and $W(5) = 4$. Since $test < W(5)$, we intert an interior node y and the node $t = 6$. Set $P(y) = b, L(y) =$

$5, R(y) = 6, W(y) = test = 1, g_y = g_5, p_y = p_b, q_y = q_5$ and set $P(6) = y, L(6) = null, R(6) = null, W(6) = E(6, g_5, q_b p_5 q_5) = E(6, 5, 134) = 2, g_6 = 6, p_6 = p_b = 3, q_6 = g_5 = 5$.

Then set $P(5) = y, W(5) = W(5) - test = 3, q_5 = 6$. Since there is no anymore leaf in $L(T)\backslash\{1, 2, 3, 4, 5\}$, we are done.

□

4. Reconstructing trees from m-dissimilarity maps

In this section, we sketch how to use higher-order map information to reconstruct trees.

We wish to find i, j, k, l, R so that ij is a sub-cherry in $[ijklR]$ but kl is not.

So, we can search randomly for an i, j, k, l, R for which $d(ij, kl, R) > 0$. (Can we do this in a better way to help in the later steps?)

We can then determine which pair is a sub-cherry by taking any $t \in R$, and letting $R' = R - t + j$. If $d(it, kl, R') > 0$, then we claim kl is a sub-cherry in $[ijklR] = [itklR']$. This is because either kl or it is a sub-cherry, but if it is a sub-cherry then ij is not a sub-cherry so that kl is a sub-cherry. If the condition does not hold, then automatically ij is a sub-cherry.

Now that we have found a sub-cherry (wlog suppose it is ij), then we should check if kl is a sub-cherry. If it is not, we are done. If it is, then take any $t \in R$, then kt is not a sub-cherry.

4.1. **Constructing the Tree.** We now assume that ij is a sub-cherry but kl is not.

Given $t \notin [ijklR]$ we wish to figure out where it goes, i.e., where the path from t to $[ijklR]$ first touches $[ijklR]$; we say that t is *incident* to $[ijklR]$ at that point.

Note that there is an internal node c whose three edges are incident to i, to j, and to $[klR]$. Call these edges $[ic], [jc], [xc]$.

We give a criterion for determining where t is incident to $[ijklR]$.

There are four cases:

(1) If $d(it, kl, R) < d(ij, kl, R)$, then t is incident to $[ic]$ at distance $d(ij, kl, R) - d(it, kl, R)$ from c.
(2) If $d(it, kl, R) > d(ij, kl, R)$, then t is incident to $[xc]$ at distance $d(it, kl, R) - d(ij, kl, R)$ from c.
(3) If $d(it, kl, R) = d(ij, kl, R)$, then t is incident to $[jc]$ at distance $d(ij, kl, R) - d(jt, kl, R)$ from c.
(4) If $d(it, kl, R) = 0$ then t is incident to the subtree $[klR]$.

4.2. **Resolving bubbles.** We perform the above step for each t not in $[ijklR]$. This clusters all leafs into sub-tree bubbles incident to $[ijklR]$ at various places. If any bubbles are size 1, we are done with them. Bubbles not of size one can be resolved by repeating this procedure. In particular, note that any two elements in the same bubble, say t_1 and t_2 are a sub-cherry in the tree spanned by $[t_1 t_2 klR]$ so each element in the bubble may be resolved recursively.

4.3. **Reconstruction Algorithm.** Step 1: Choose a set R' of m+2 leaves and find ij a sub-cherry in $[R']$. If the data is a tree, such a sub-cherry must always exist. Let $R = R' - ij$.

Step 2: For each $t \notin R'$, check if t is a sub-cherry in $[Rit]$. If it is, resolve its location as discussed in the previous section. If it is not a sub-cherry, set it aside. Denote the set of resolved leaves (including i and j) by A and the set of unresolved leaves (including R) by B.

Step 3: If $|A| \geq m$ we may choose a subset R of A of such that $|R| = m$ and any two elements $s, t \in B$ will be a sub-cherry in the tree $[tsR]$. Hence, relative to this new R, we may resolve every element in B by repeating step 2 with this new R.

Step 4: If $|A| < m$ then $|B| \geq m$. We vary R' of size m over B and compute the ij cherry distance to R'. If it is greater than the ij cherry distance to our previous R, return to step 2.

We may now assume that $|A| < m$ and for all $R' \subseteq B$, the ij cherry distance is the same. From this, we may conclude that we have a situation as in Figure [?] and that $|B_1| < m$ and $|B_2| < m$.

Step 5: Let $|A| = a < m$. Choose $b = m + 2 - a$ leaves from B and call the set of leaves R'. Note that $m - a \geq 1 \Rightarrow b \geq 3$. Hence, either B_1 or B_2 has at least two leaves, so there must be a sub-cherry kl in $[R'A]$. In particular, the sub-cherry is either in B_1 or B_2, so without loss of generality, we assume $kl \in B_1$.

Step 6: Place all elements of $B - R'$ that form sub-cherries with k or l (as we did for i and j in step 2).

Step 7: As in step 4, vary the subset R' over the leaves we have yet to resolve. If the kl cherry distance increases, return to step 6.

When we reach the point that the kl cherry distance is the same for any choice of R', we claim that $R' \subseteq B_2$. If not, then $|B_2| \leq m - a - 1$ and we already have that $|B_1| \leq m - 1$ and so

$$2m - 1 \leq |A| + |B_1| + |B_2| \leq a + (m - 1) + (m - a - 1) \leq 2m - 2$$

a contradiction.

But, since $R' \subseteq B_2$ implies that we have placed every leaf in A and B_1. In particular, we have placed at least m leaves since $|A| + |B_1| \geq m$.

Step 8: Choose $R \subseteq A \cup B_1$ of size m, then every pair of elements not yet accounted for, say s, t will be a sub-cherry in $[stR]$.

Step 9: In this way we have placed all our leaves and found the edge weights associated to each internal edge. We may find the leaf edges, then, in the same manner described in Pachter-Speyer.

Acknowledgments.

References

[1]

Department of Mathematics, University of California, Berkeley, CA
E-mail address: levyd@math.berkeley.edu

Department of Mathematics, Harvey Mudd College, Claremont, CA 91711
E-mail address: su@math.hmc.edu

Department of Mathematics, University of California, Davis, CA
E-mail address: ruriko@math.ucdavis.edu