

# On the Informational Asymmetry between Upper and Lower Bounds for Ultrametric Evolutionary Trees

Ting Chen<sup>1</sup> \* and Ming-Yang Kao<sup>2</sup> \*\*

<sup>1</sup> Department of Genetics, Harvard Medical School  
Boston, MA 02115 USA

tchen@salt2.med.harvard.edu

<sup>2</sup> Department of Computer Science, Yale University  
New Haven, CT 06520-8285 USA

kao-ming-yang@cs.yale.edu

**Abstract.** This paper addresses the informational asymmetry for constructing an ultrametric evolutionary tree from upper and lower bounds on pairwise distances between  $n$  given species. We show that the tallest ultrametric tree exists and can be constructed in  $O(n^2)$  time, while the existence of the shortest ultrametric tree depends on whether the lower bounds are ultrametric. The tallest tree construction algorithm gives a very simple solution to the construction of an ultrametric tree. We also provide an efficient  $O(n^2)$ -time algorithm for checking the uniqueness of an ultrametric tree, and study a query problem for testing whether an ultrametric tree satisfies both upper and lower bounds.

## 1 Introduction

Constructing the evolutionary tree of a species set is a fundamental problem in computational biology. Such trees describe how species are related to one another in terms of common ancestors. A useful computational problem for constructing evolutionary tree is that given an  $n \times n$  distance matrix  $M$  where  $M_{ij}$  is the observed distance between two species  $i$  and  $j$ , find an edge-weighted evolutionary tree in which the *distance*  $d_{ij}$  in the tree between the leaves  $i$  and  $j$ , equals  $M_{ij}$ . Pairwise distance measures carry some uncertainty in practice. Thus, one seeks a tree that is close to the distance matrix, as measured by various choices of optimization objectives [2, 3, 6, 8].

This paper focuses on the class of *ultrametric trees* [5, 7–9]. An ultrametric tree is a rooted tree whose edges are weighted by a non-negative number such that the lengths of all the root-to-leaf paths, measured by summing the weights of the edges, are equal. A distance matrix is *ultrametric* if an ultrametric tree can be constructed from this matrix. Figure 1 shows an example of an ultrametric matrix and an ultrametric tree constructed from this matrix.

---

\* Supported in part by the Lipper Foundation.

\*\* Supported in part by NSF Grant 9531028.

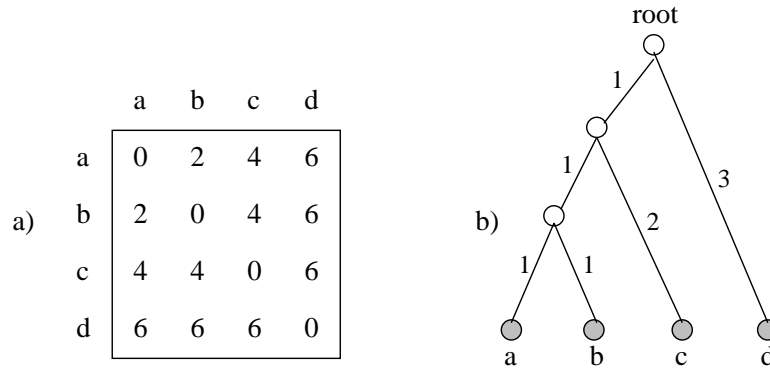


Fig. 1. a. An ultrametric matrix  $M$ . b. An ultrametric tree for  $M$ .

In practice, when distance measures are uncertain, a distance is expressed as an interval, defined by a lower bound and an upper bound for the true distance. From such data, we obtain two distance matrices  $M^\ell$  and  $M^h$ , representing pairwise distance lower and upper bounds. The tree construction problem becomes that of constructing an ultrametric tree whose pairwise distances fit between two bounds, i.e.,  $M_{ij}^\ell \leq d_{ij} \leq M_{ij}^h$ . Farah, Kannan and Warnow [4] gave the first known algorithm for constructing ultrametric trees from the sandwich distances.

This paper studies the informational asymmetry between lower and upper bounds in the construction of ultrametric trees. Our results are as follows:

- Given an upper bound matrix, the *tallest* ultrametric tree, where the distance of any two leaves reaches the maximum among all satisfied ultrametric trees, can be constructed in  $O(n^2)$  time. This result immediately leads to a new and simpler tree construction algorithm than that of [4].
- Given a lower bound matrix, the *shortest* ultrametric tree, defined similarly to the tallest ultrametric tree, exists if and only if the matrix is ultrametric.
- We provide an  $O(n^2)$ -time algorithm to check the uniqueness of an ultrametric tree satisfying given upper and lower bounds.
- We study a query problem: if a lower bound matrix and an upper bound matrix are given, how fast can we determine whether an ultrametric tree satisfies both constraints? This problem is useful, for example, for developing an interactive software for finding the most suitable tree among many. We present an algorithm to test the satisfaction of the upper bound constraints in  $O(n)$  time after preprocessing the upper bound matrix. A similarly fast algorithm for testing the lower bounds remains open.

## 2 Notation

Let  $M$  represent a distance matrix, and  $M^\ell$  and  $M^h$  represent the lower bound and upper bound matrices for  $M$ . Let  $d_{ij}$  represent the distance between leaf

$i$  and leaf  $j$  in a tree, defined as the length of the path connecting two leaves. Given a tree  $U$ ,  $d_{ij}^U$  represents that distance in  $U$ . If  $U$  is ultrametric, the height of  $U$  is denoted by function  $\mathfrak{h}(U)$ , or  $\mathfrak{h}(r)$  if  $r$  is the root of  $U$ .

An  $n \times n$  distance matrix  $M$  corresponds to an undirected edge-weighted complete graph  $G$  with  $n$  vertices, where a vertex represents a species and the weight  $w(i, j)$  of edge  $(i, j)$  is  $M_{ij}$ .  $G$  is also said to be *the corresponding graph* to  $M$ . For the lower bound matrix  $M^\ell$  and the upper bound matrix  $M^h$ , their corresponding graphs are denoted by  $G^\ell$  and  $G^h$ , and the weight functions are  $w^\ell()$  and  $w^h()$ .

### 3 Constructing the tallest ultrametric tree

This section discusses the problem of finding the tallest ultrametric tree for any upper bound matrix. We give a simple  $O(n^2)$ -time algorithm that takes advantages of the minimum weight spanning tree.

**Fact 1** (see [1, 4, 8]) *A matrix is ultrametric if and only if in the corresponding complete weighted graph, the largest weight edge in any cycle of more than one node is not unique.*

*Proof.* Straightforward.  $\square$

Suppose we have an upper bound matrix  $M^h$  on the pairwise leaf-leaf distances of an ultrametric tree. There are many ultrametric trees satisfying  $M^h$ , but which one is the tallest? The following algorithm gives the answer. Let  $G^h$  be the corresponding graph of  $M^h$ .

#### Algorithm Compute\_Tallest\_Tree

1. Construct the minimum weight spanning tree  $T$  of  $G^h$ .
2. Sort the edges of  $T$  in decreasing order as  $e_1, e_2, \dots, e_{n-1}$ .
3. Return the tree  $U$  constructed by the following procedure:
  - (a) If  $T$  is empty, return a leaf with zero height.
  - (b) Otherwise, remove the first edge  $e_1$  from  $T$ , leaving two subtrees,  $T_1$  and  $T_2$ , each of which maintains its edges in decreasing order.
  - (c) Recursively construct a tree  $U_1$  from  $T_1$ , and a tree  $U_2$  from  $T_2$ .
  - (d) Construct a root  $r$  for  $U$  with height  $\mathfrak{h}(U) = \frac{1}{2}w^h(e_1)$ , and attach  $U_1$  to  $r$  with an edge weighted  $\frac{1}{2}w^h(e_1) - \mathfrak{h}(U_1)$  and  $U_2$  to  $r$  with an edge weighted  $\frac{1}{2}w^h(e_1) - \mathfrak{h}(U_2)$ .
  - (e) Return  $U$ .

This algorithm constructs an ultrametric tree in  $O(n^2)$  time:

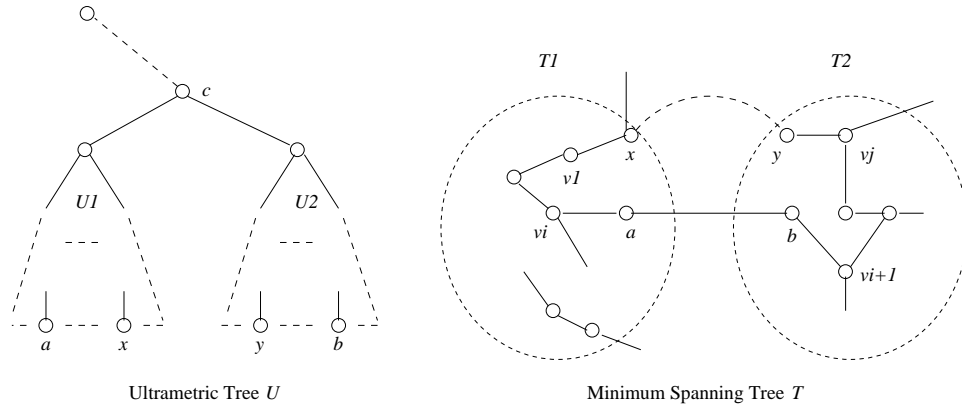
**Lemma 1.** *Algorithm Compute\_Tallest\_Tree runs  $O(n^2)$  time and returns an ultrametric satisfying the given upper bound matrix.*

*Proof.* The algorithm takes  $O(n^2)$  time to build the minimum weight spanning tree  $T$ , and  $O(n \log n)$  time to sort the edges. Maintaining the edges in the subtrees of  $T$  in decreasing order (at Step 3b) takes  $O(n\alpha(n, n))$  time by using the disjoint-set forest data structure, where  $\alpha()$  is the inverse of Ackermann's function, and  $\alpha(n, n) \leq 4$  in any conceivable application. Here we implicitly use the fact that any connected subtree of a minimum spanning tree is still a minimum spanning tree. Therefore, the total time complexity is  $O(n^2)$ . At Step 3d, if both  $U_1$  and  $U_2$  are ultrametric,  $U$  is ultrametric because the distance from root  $r$  to any leaf is  $\frac{1}{2}w^h(e_1)$ . Step 3a returns a tree that has only one leaf and is ultrametric, so by recursion  $U$  is ultrametric.  $U$  also satisfies the upper bound matrix: for any two leaves  $x$  of  $U_1$  and  $y$  of  $U_2$ , the weight of the edge  $w^h(x, y)$  in graph  $G^h$  is at least  $w^h(e_1)$ , which equals  $d_{xy}^U$ ; otherwise, we could replace  $e_1$  by  $(x, y)$  in  $T$ .  $\square$

**Theorem 2.** *The tree  $U$  constructed by Algorithm Compute\_Tallest\_Tree is the tallest ultrametric tree.*

*Proof.* Let  $x$  and  $y$  be any two leaves of  $U$ , and  $c$  be their lowest common ancestor. Let  $(a, b)$  be the edge in the minimum spanning tree  $T$  whose removal generates  $c$  at Step 3d of Algorithm Compute\_Tallest\_Tree. Then,

$$d_{xy}^U = d_{ab}^U = M_{ab}^h.$$



**Fig. 2.** Constructing an ultrametric tree using a minimum spanning tree

Let  $U_1$  and  $U_2$  be two immediate subtrees of  $c$ , which are constructed from two subtrees  $T_1$  and  $T_2$  of  $T$ , respectively. Assume that leaf  $a$  and leaf  $x$  are under  $U_1$ , and leaf  $b$  and leaf  $y$  are under  $U_2$ , as shown in Figure 2. Note that  $T_1$  is a connected component and thus contains a path from  $x$  to  $a$ , and similarly  $T_2$  contains a path from  $b$  to  $y$ . Combining them with the edge  $(a, b)$ , we obtain

a path  $P$  from  $x$  to  $y$ , *i.e.*,

$$P = x - v_1 - \dots - v_i - a - b - v_{i+1} - \dots - v_j - y.$$

Note that  $(a, b)$  is selected in the algorithm because it has a larger weight than any other edge in  $T_1 \cup T_2$ . Thus,

$$\max_{(w,z) \in P} M_{wz}^h = M_{ab}^h.$$

Let  $V$  be any other satisfied ultrametric tree. By Fact 1,  $d_{xy}^V$  should not be the sole largest distance in the cycle  $P \cup (x, y)$ . Thus,

$$d_{xy}^V \leq \max_{(w,z) \in P} d_{wz}^V \leq \max_{(w,z) \in P} M_{wz}^h.$$

Combining all the inequalities,  $d_{xy}^V \leq d_{xy}^U$  and  $U$  is the tallest ultrametric tree.  $\square$

Theorem 2 immediately leads to the following theorem:

**Theorem 3.** *Algorithm Compute\_Tallest\_Tree constructs an ultrametric tree satisfying both lower bound and upper bound constraints, if one exists.*

*Proof.* Algorithm Compute\_Tallest\_Tree constructs the tallest ultrametric tree  $U$  for upper bounds. If lower bounds are also given, either  $U$  satisfies the lower bounds, or no tree satisfies them.  $\square$

## 4 Asymmetry between upper and lower bounds

We have shown how to construct the tallest ultrametric tree in Section 3. However, the shortest ultrametric tree may not exist. The following lemma explains the asymmetry between upper and lower bounds.

**Lemma 2.** *There exists a lower bound matrix  $L$  such that for any ultrametric tree  $V$  that satisfies  $L$ , there is an upper bound matrix  $H$  which  $V$  cannot satisfy but some ultrametric tree  $U$  satisfies both  $L$  and  $H$ .*

*Proof.* Consider a lower bound matrix  $L$  having three elements  $a, b$  and  $c$ , whose distances are  $x, y$  and  $z$ , satisfying  $x > y$  and  $x > z$ . Let  $d_m$  be the maximum distance in  $L$ . We construct two upper bound matrices  $H_1$  and  $H_2$ , where every distance equals  $d_m$  except the three distances shown below.

$L$	$b$	$c$	$H_1$	$b$	$c$	$H_2$	$b$	$c$
$a$	$x$	$y$	$a$	$x$	$x$	$a$	$x$	$y$
$b$	$z$		$b$	$z$		$b$	$x$	

Every element in  $H_1$  is at least as large as the corresponding element in  $L$ , and  $H_1$  is ultrametric by Fact 1. Thus, there are ultrametric trees satisfying  $L$  and  $H_1$ . Let  $V$  be such a tree. So  $d_{ab}^V$  and  $d_{bc}^V$  are fixed:  $d_{ab}^V = x$  and  $d_{bc}^V = z$ . By Fact 1,  $d_{ac}^V = x$ , which is larger than the corresponding element  $y$  in  $H_2$ .  $V$  does not satisfy  $H_2$ . Similarly,  $H_2$  is ultrametric and any ultrametric tree satisfying  $L$  and  $H_2$  does not satisfy  $H_1$ . Therefore, no ultrametric tree satisfies both  $H_1$  and  $H_2$  at the same time.  $\square$

By Lemma 2, a lower bound matrix does not have a shortest ultrametric tree if there exists a three-element cycle whose largest value is unique. On the other hand, if the largest value of any of its three-element cycle is non-unique, then,

**Lemma 3.** *If any three element cycle in a matrix has a non-unique largest value, so does any cycle in the matrix.*

*Proof.* Let  $M$  be a matrix where the largest value in any three-element cycle is not unique. Assume there exists a cycle  $C = (v_1, v_2, \dots, v_k, v_1)$ , in which  $k > 3$  and  $M_{v_1v_2}$  is the unique largest value. We decompose  $C$  into three-element cycles:  $(v_1, v_2, v_3, v_1)$ ,  $(v_1, v_3, v_4, v_1)$ , ..., and  $(v_1, v_{k-1}, v_k, v_1)$ . Since the largest value in any of them is not unique, the first cycle has  $M_{v_1v_2} = M_{v_1v_3}$  because  $M_{v_1v_2} > M_{v_2v_3}$  by the assumption. Similarly,  $M_{v_1v_i} = M_{v_1v_{i+1}}$  in the  $i$ th cycle, for  $i = 2, \dots, k-1$ . Combining these results,  $M_{v_1v_2} = M_{v_1v_k}$ , which contradicts our assumption. Thus the largest value in any cycle of  $M$  is not unique.  $\square$

**Theorem 4.** *A lower bound matrix has a shortest ultrametric tree if and only if it is ultrametric.*

*Proof.* If a lower bound matrix is ultrametric, by definition, an ultrametric tree can be constructed from this matrix. This tree is the shortest. If a lower bound matrix has a shortest ultrametric tree, by the proof in Lemma 2, the largest value in any three-element cycle is not unique. So does any cycle in the matrix, by Lemma 3. Finally, by Fact 1 this matrix is ultrametric.  $\square$

## 5 Uniqueness of ultrametric tree

Given  $M^\ell$  and  $M^h$ , is the ultrametric tree built by Algorithm `Compute_Tallest_Tree` topologically unique? Knowing the uniqueness is of vital importance because it indicates the significance of the tree.

We first define what kind of trees have the same topological structure. An edge-weighted tree is *compact* if it has no zero-weight edge. A tree can be *compactified* by merging any two internal nodes into a single node if they are connected by a zero weight edge. A *compact* ultrametric tree is one that is compact. Any ultrametric tree can be converted into a compact ultrametric tree by merging, but the resulting tree may not be binary. Assume all the trees have the same set of labeled leaves. An internal node can be represented by a *leaf set*, consisting of all the descendent leaves of the node. A tree can be represented as the *superset* of leaf sets, where every element in the superset corresponds to an internal node,

and vice versa. Two trees are *equivalent* or have the same *topological structure* if their representing supersets are the same, in other words, if there is a one-to-one mapping between all nodes that preserve the parent-child relation. Two compact ultrametric trees may have the same topological structure even though their edge weights are completely different. In discussing topological structures, the edge weights are ignored, because equivalent evolutionary trees give the same evolutionary process, and the difference in distances are usually caused by measuring errors.

Given  $M^\ell$  and  $M^h$ , we construct an ultrametric tree from  $M^h$  by Algorithm `Compute_Tallest_Tree`, and then convert it into a compact tree  $U$ . We can check the uniqueness of  $U$  by the following lemma:

**Lemma 4.** *For given  $M^\ell$  and  $M^h$ , the compact ultrametric tree  $U$ , constructed from Algorithm `Compute_Tallest_Tree`, is unique if and only if for every internal node  $u$ , any two children  $u_i$  and  $u_j$  of  $u$  satisfy that  $\max M_{xy}^\ell = \hbar(u)$ , for any leaf  $x$  under  $u_i$  and any leaf  $y$  under  $u_j$ .*

*Proof.* Suppose that  $M^\ell$  and  $M^h$  define a unique compact ultrametric tree. Since  $U$  is the tallest compact ultrametric tree for  $M^h$ , it satisfies  $M^\ell$  and is unique. Assume the lemma does not hold, then there exist two different children  $u_i$  and  $u_j$  of  $u$ , such that  $\max M_{xy}^\ell < \hbar(u)$ , for any two leaves  $x$  under  $u_i$  and  $y$  under  $u_j$ . Let  $d_{u_i, u_j} = \max M_{xy}^\ell$  and  $d_m = \max(d_{u_i, u_j}, \hbar(u_i), \hbar(u_j))$ . We can construct a different tree from  $U$  by deleting two children  $u_i$  and  $u_j$  from  $u$ , and replacing by a child  $u'$  who has two children  $u_i$  and  $u_j$ , and  $\hbar(u') = \frac{1}{2}(\hbar(u) + d_m)$ . This tree is a compact ultrametric tree because  $\hbar(u) > \hbar(u') > d_m$ , and it is topologically different from  $U$ , contradicting the uniqueness of  $U$ .

Conversely, if any two children  $u_i$  and  $u_j$  of  $u$  satisfy that  $\max M_{xy}^\ell = \hbar(u)$  for any leaf  $x$  under  $u_i$  and any leaf  $y$  under  $u_j$ , we state that  $U$  is unique. Assume there exists another topologically different tree  $V$ . Let  $d_{xy}^U$  be the minimum value among those who satisfy  $d_{xy}^V < d_{xy}^U$  ( $U$  is the tallest).  $d_{xy}^U$  must exist, otherwise  $V$  equals  $U$  because all pairwise distances of  $V$  are equal to those of  $U$ . Let  $u$  be the least common ancestor of  $x$  and  $y$  in  $U$ ,  $u_x$  be the child of  $u$  that contains  $x$  as a descendant, and  $u_y$  be the child of  $u$  that contains  $y$  as a descendant. Let  $S_x$  be the set of leaves under  $u_x$  and  $S_y$  be the set of leaves under  $u_y$ .

For any  $w \in S_x$ ,  $d_{xw}^V \leq d_{xw}^U = h^U(u_x) < d_{xy}^U$ , and for any  $z \in S_y$ ,  $d_{yz}^V \leq d_{yz}^U = h^U(u_y) < d_{xy}^U$ . By Lemma 3 and Fact 1,  $d_{xz}^V \leq \max(d_{xy}^V, d_{yz}^V) < d_{xy}^U$ . Thus, in  $V$ , the distance between  $x$  and any other leaf in  $S_x \cup S_y$  is less than  $d_{xy}^U$ . So is the distance of any pair of leaves in  $S_x \cup S_y$ . However, it contradicts the condition in the lemma that there exist a pair of leaves  $w \in S_x$  and  $z \in S_y$  with  $M_{wz}^\ell = h^U(u) = d_{xy}^U$ . Hence,  $V$  can not exist, and  $U$  is unique.  $\square$

**Theorem 5.** *Given  $M^\ell$  and  $M^h$  as input, we can determine the uniqueness of ultrametric trees in  $O(n^2)$  time.*

*Proof.* We can construct a compact ultrametric tree  $U$  in  $O(n^2)$  time by Algorithm `Compute_Tallest_Tree`, and then check the conditions of every internal

node in a child-parent topological order in  $O(n^2)$  time by Lemma 4, because every pair of leaves is visited exactly once.  $\square$

## 6 Query of ultrametric trees

Assuming that we have obtained new ways to estimate the evolutionary distances between species (by some new evidence), and there are many previously computed ultrametric trees (by other evidences), finding the suitable trees among them may suggest new relations between these evidences. This section studies this query problem: given  $M^\ell$  and  $M^h$ , and an ultrametric tree  $V$ , does  $V$  satisfy  $M^\ell$  and  $M^h$ ? A naive algorithm runs in  $O(n^3)$  time by checking  $O(n^2)$  pairs of leaves in  $V$  and calculating the distance of each pair in  $O(n)$  time. We can improve  $O(n^3)$  to  $O(n^2)$  by pre-calculating the *lowest common ancestor* in linear time, and thus finding the distance of each pair of leaves in constant time. If preprocessing is permitted, is there a faster algorithm than  $O(n^2)$ ?

**Lemma 5.** *We can preprocess the upper bound matrix  $M^h$  in  $O(n^2)$  time, so that for any given ultrametric tree  $V$ , whether  $V$  satisfies  $M^h$  can be determined in  $O(n)$  time.*

*Proof.* Assume we have built the tallest ultrametric tree  $U$  by Algorithm Compute\_Tallest\_Tree, and have calculated the lowest common ancestor (lca) function for  $V$  in linear time. We define a recursive function  $f$  to map a node in  $U$  to a node in  $V$ :

$$f(u) = \begin{cases} v & \text{if } u \text{ is a leaf in } U, v \text{ is a leaf in } V, u = v \\ \text{lca}(f(u_l), f(u_r)) & \text{if } u \text{ has children } u_l \text{ and } u_r \end{cases}$$

For each internal node  $u$ ,  $f$  returns an internal node  $v$  under which its leaves form a superset of the leaves under  $u$ . By lca function,  $v$  is the lowest node whose leaves form the minimum superset.

We sort all the leaves and internal nodes of  $U$  into a topological order  $u_1, u_2, \dots$ , where a node appears before its parent, in  $O(n)$  time by a depth-first search. For each  $u_i$ , if  $h^V(f(u_i)) \leq h^U(u_i)$ ,  $V$  satisfies  $M^h$ . We next show this in two steps. First, because  $u_1, u_2, \dots$  follow the topological order (child to parent), we can calculate  $v_1 = f(u_1), v_2 = f(u_2), \dots$  in this order in  $O(n)$  time. Second, if  $u$  has two children  $u_1$  and  $u_2$  and  $v = f(u)$ ,  $h^V(v) \leq h^U(u)$  indicates that for any pair of leaves  $w$  under  $u_1$  and  $z$  under  $u_2$ ,  $d_{f(w)f(z)}^V \leq 2h^V(v) \leq 2h^U(u) = d_{wz}^U \leq M_{wz}^h$ . Visiting every node in  $U$  and checking the corresponding inequality is equivalent to comparing the distance of every pair of leaves in  $U$  with that in  $V$ . If all of them hold,  $V$  satisfies  $M^h$ . Otherwise, if  $h^V(v) > h^U(u)$ , there must exist a leaf  $w$  under  $u_1$  and  $z$  under  $u_2$  such that  $d_{f(w)f(z)}^V > d_{wz}^U$ , violating the assumption that  $U$  is the tallest (by construction) in Theorem 2.

The preprocessing takes  $O(n^2)$  time and  $O(n)$  space to construct  $U$  and sort a topological order. To answer a query, it takes  $O(n)$  time and  $O(n)$  space to calculate function  $f$ , and the same for visiting  $O(n)$  nodes of  $U$  and evaluating  $O(n)$  inequalities.  $\square$

However, for lower bound matrices, whether there is an algorithm better than  $O(n^2)$  time remains an open question. Unlike upper bound matrices having the tallest ultrametric tree, lower bound matrices may have multiple minimal trees as shown in Section 4. This asymmetry prevents the linear time checking algorithm in Lemma 5 from being applicable to the lower bounds.

**Open Problem** *By preprocessing, is there an algorithm that can test whether a tree satisfies a lower bound matrix faster than  $O(n^2)$  time?*

## 7 Acknowledgments

We thank Raphael Ryger for useful comments.

## References

1. J.-P. Barthélémy and A. Guénoche. *Trees and proximity representations*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, New York, NY, 1991.
2. V. Berry and O. Gascuel. Inferring evolutionary trees with strong combinatorial evidence. In T. Jiang and D. T. Lee, editors, *Lecture Notes in Computer Science 1276: Proceedings of the 3rd Annual International Computing and Combinatorics Conference*, pages 111–123. Springer-Verlag, New York, NY, 1997.
3. J. C. Culbertson and P. Rudnicki. A fast algorithm for constructing trees from distance matrices. *Information Processing Letters*, 30(4):215–220, 1989.
4. M. Farach, S. Kannan, and T. Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13(1/2):155–179, 1995.
5. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, 1997.
6. J. J. Hein. An optimal algorithm to reconstruct trees from additive distance data. *Bulletin of Mathematical Biology*, 51:597–603, 1989.
7. D. M. Hillis, C. Moritz, and B. K. Mable, editors. *Molecular Systematics*. Sinauer Associates, Sunderland, Ma, 2nd edition, 1996.
8. J. C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, MA, 1997.
9. M. S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman & Hall, New York, NY, 1995.