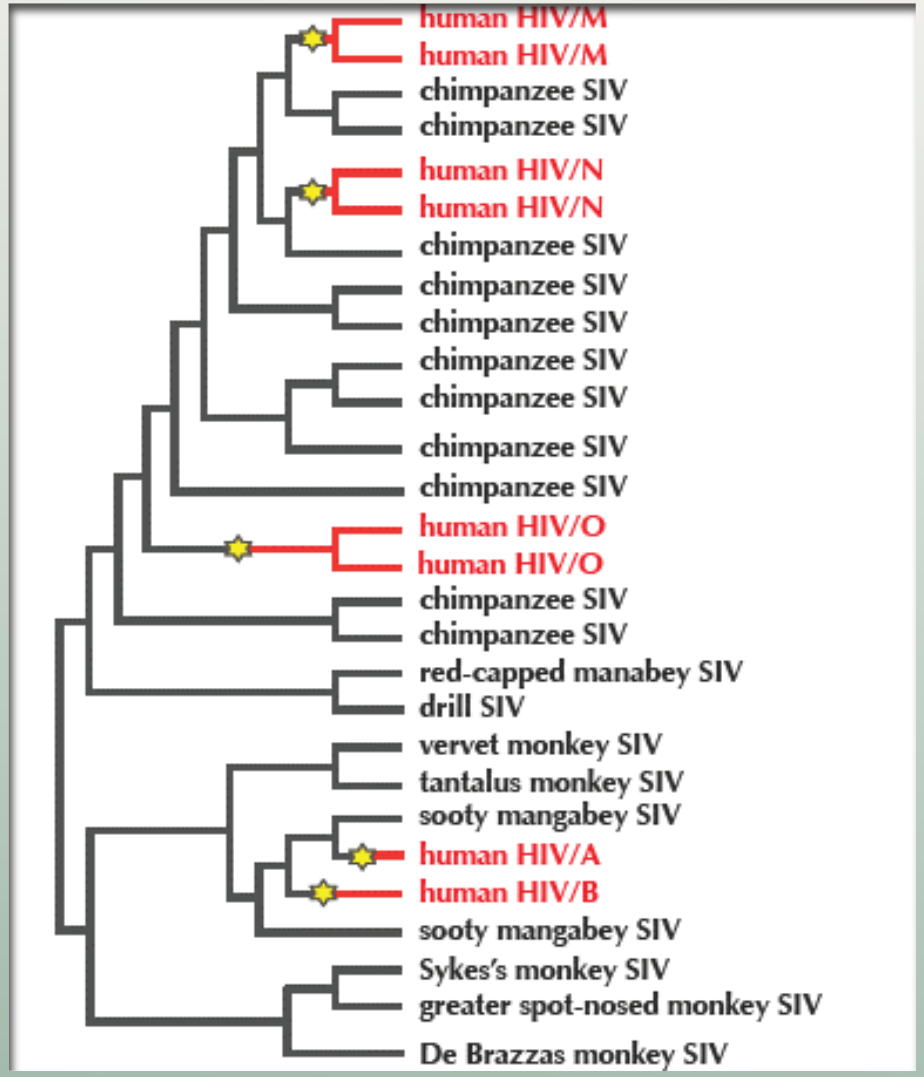
Which Animal Gave Us SARS – Part 2

Evolutionary tree Reconstruction

Selected Topics in Bioinformatics
Lecture 4

AGENDA

- Computing limb lengths
- Trimming the tree
- Attaching a limb



COMPUTING LIMB LENGTH

Definitions revision

- In a tree, nodes with degree larger than 1 are called **internal nodes** while, those with degree 1 are called a **leaves**.
- Given a leaf j, there is only one node connected to j by an edge called the **parent** of j, denoted **PARENT(j).**
- An edge connecting a leaf to its parent is called a **limb**.
- A **rooted tree** is a tree with one node designated as a special node called the **root.**

COMPUTING LIMB LENGTH

Another approach for Neighboring leaves

- The assumption that a minimum element of an additive distance matrix corresponds to neighboring leaves is not necessarily true!
- Thus, we need a new approach to the Distance-Based Phylogeny Problem
- so, we will explore a different recursive algorithm.
  - Rather than looking for a pair of neighbors in TREE (D),
  - Instead, we will reduce the size of the tree by trimming its leaves one at a time.
  - Of course, we don't know TREE (D)
    - So, we must somehow trim leaves in TREE(D) by analyzing the distance matrix.

COMPUTING LIMB LENGTH

Another approach for Neighboring leaves

- The first step towards constructing TREE (D), is to compute the lengths of limbs in TREE (D)
- Given a leaf *j* in a tree, we denote the length of the limb connecting *j* with its parent as LIMBLENGHT (*j*).
- Edges that are not limbs must connect two internal nodes and are therefore called **internal edges**.

COMPUTING LIMB LENGTH

Another approach for Neighboring leaves

Limb Length Problem:
Compute the length of a limb in the simple tree fitting an additive distance matrix.

Input: An additive distance matrix D and an integer j.
Output: LIMBLENGTH(j), the length of the limb connecting leaf j to its parent in TREE(D).

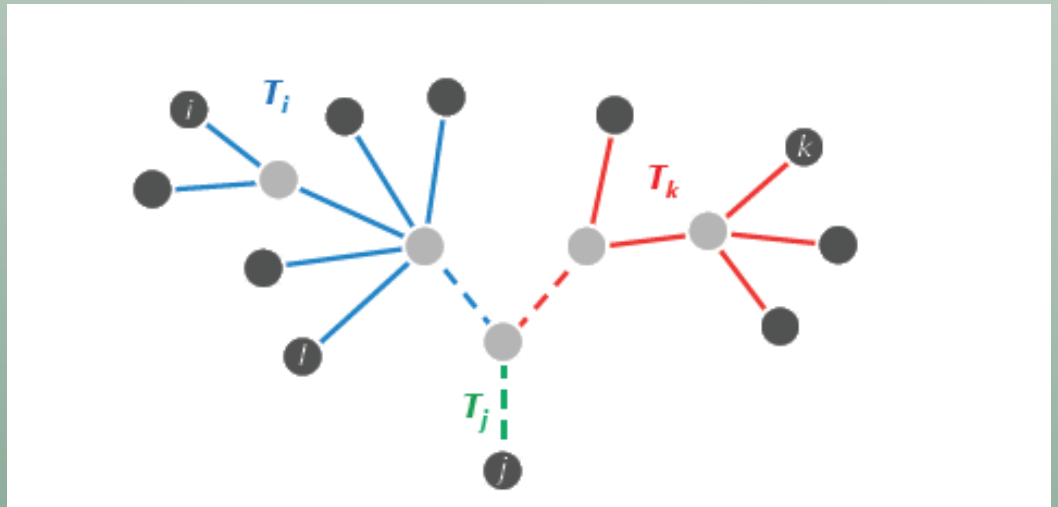
COMPUTING LIMB LENGTH

Limb Length Theorem

- When computing LIMBLENGHT (j) for a given leaf j, and TREE (D) is a simple tree, then PARENT (j) has degree at least 3.
- PARENT (j) as dividing the other nodes of TREE (D) into at least 3 subtrees,

  (i.e. smaller trees that would remain if we remove PARENT (j) with any edges connecting it to other nodes)

  - Because j is a leaf, it must belong to a subtree by itself; we call it T_j.



COMPUTING LIMB LENGTH

Limb Length Theorem

Limb Length Theorem: *Given an additive matrix D and a leaf j,* LIMBLENGTH(*j*) *is equal to the minimum value of* $(D_{i,j} + D_{j,k} - D_{i,k})/2$ *over all leaves i and k.*

i.e., For each leave *j*, we can compute LIMBLENGHT (*j*) by finding the minimum value of $(D_{i,j} + D_{j,k} - D_{i,k})/2$ over all pairs of leaves *i* and *k*.

TRIMMING THE TREE

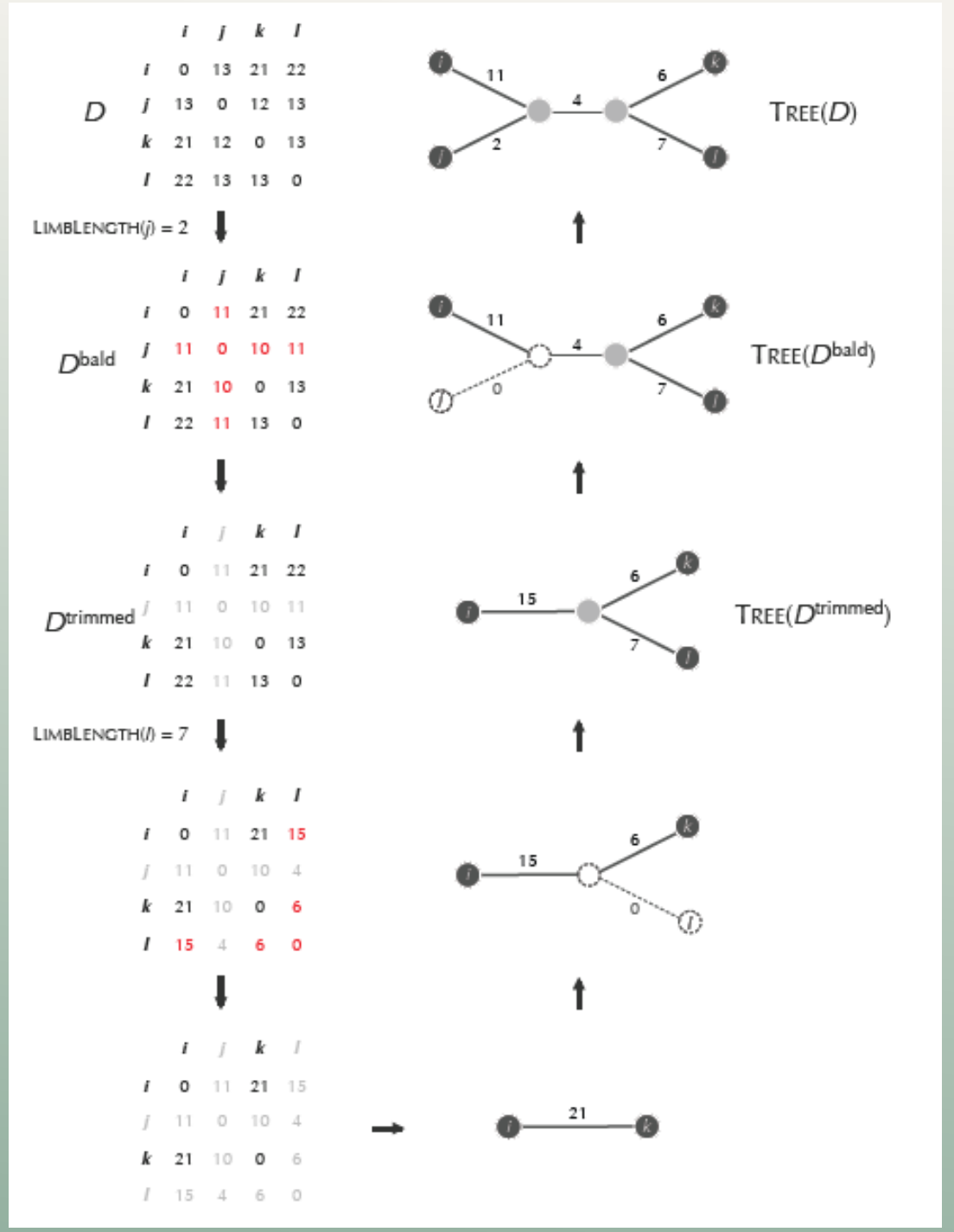
A New Algorithm for reconstructing the TREE

- Since we now know how to find the length of any limb in TREE(D), we can construct TREE (D) recursively using a new algorithm

- pick an arbitrary leaf j, compute LIMBLENGTH(j), and construct the distance matrix D^{trimmed};
- solve the Distance-Based Phylogeny Problem for D^{trimmed};
- identify the point in TREE(D^{trimmed}) where leaf j should be attached in TREE(D);
- add a limb of length LIMBLENGTH(j) growing from this attachment point in TREE(D^{trimmed}) to form TREE(D).

TRIMMING THE TREE

- First compute LIMBLENGHT (j) = 2,
- Subtract 2 from the non-diagonal elements in row j and column j of D to obtain D^{bald} (updated values are shown in red).
- Removing this row and column yields a 3 × 3 distance matrix $D^{trimmed}$.
- We find that LIMBLENGHT (l) = 7 in $D^{trimmed}$
- Subtract 7 from the non-diagonal elements in row l and column l.
- Graying out this row and column yields a 2 × 2 distance matrix.
- On the right side, we fit this 2 × 2 distance matrix to a tree consisting of a single edge.
- By finding the attachment points of removed limbs (shown on the left), we reconstruct TREE ($D^{trimmed}$), TREE (D^{bald}), and then TREE (D).



ATTACHING A LIMB

How to find the attachment point of leaf *j* in TREE ($D^{trimmed}$) ?

- TREE (D^{bald}) is the same as TREE (D) except that LIMBLENGHT (*j*) = 0.
- From the Limb Length Theorem, there must be leaves *i* and *k* in TREE (D^{bald}) such that:

$$\frac{D_{i,j}^{bald} + D_{j,k}^{bald} - D_{i,k}^{bald}}{2} = 0,$$

which implies that

$$D_{i,k}^{bald} = D_{i,j}^{bald} + D_{j,k}^{bald}.$$

- Thus, the attachment point for leaf *j* must be located at distance $D^{bald}_{i,j}$ from leaf *i* on the path connecting *i* and *k* in the trimmed tree.
- This attachment point may occur at an existing node, in which case we connect *j* to this node.
- On the other hand, the attachment point for *j* may occur along an edge, in which case we place a new node at the attachment point and connect *j* to it.

ATTACHING A LIMB

Distance-based phylogeny construction Algorithm

```
ADDITIVEPHYLOGENY(D, n)
    if n = 2
        return the tree consisting of a single edge of length D_{1,2}
    limbLength ← LIMB(D, n)
    for j ← 1 to n − 1
        D_{j,n} ← D_{j,n} − limbLength
        D_{n,j} ← D_{j,n}
    (i, n, k) ← three leaves such that D_{i,k} = D_{i,n} + D_{n,k}
    x ← D_{i,n}
    remove row n and column n from D
    T ← ADDITIVEPHYLOGENY(D, n − 1)
    v ← the (potentially new) node in T at distance x from i on the path between i and k
    add leaf n back to T by creating a limb (v, n) of length limbLength
    return T
```

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

New Goal

- Given a non-additive $n \times n$ distance matrix D, then
  - instead of looking for a weighted tree T whose distances between leaves approximate the entries in D.
  - We want look for T that minimize the sum of squared errors DISCREPANCY (T, D), which is given by the formula

$$\text{DISCREPANCY}(T, D) = \sum_{1 \leq i < j \leq n} (d_{i,j}(T) - D_{i,j})^2 .$$

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

Least Squares Distance-Based Phylogeny Problem

Least Squares Distance-Based Phylogeny Problem:
Given a distance matrix, find the tree that minimizes the sum of squared errors.

Input: An $n \times n$ distance matrix D.
Output: A weighted tree T minimizing DISCREPANCY(T, D) over all weighted trees with n leaves.

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

Challenges for this problem

- For a specific tree T, it is easy to find edge weights in T minimizing DISCREPANCY (T, D).
- But minimizing the sum of squared errors for a specific tree does not imply that we can efficiently solve the Least Squares Distance-Based Phylogeny Problem, since the number of different trees grows very quickly as the number of leaves in the tree increases.
- In fact, the Least Squares Distance-Based Phylogeny is an NP-Complete problem
  - So, we look for a heuristics for constructing trees from non-additive matrices that solve this problem ***approximately***.

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

Rooted/Unrooted Binary Trees

- Biologists assume that every internal node in an evolutionary tree corresponds to a species that underwent a speciation event:
  - Splitting one ancestral species into two descendants.
- So, we define an **unrooted binary tree** as a tree where every node has degree equal to either 1 or 3.
- A **rooted binary tree** is an unrooted binary tree that has a root (of degree 2) placed on one of its edges.
  - we replace an edge (v , w) with a root and draw edges connecting the root to each of v and w

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

Ultrametric Evolutionary Trees

- We assign an **age** to every node v in a rooted binary tree (denoted $\text{AGE}(v)$), where all the leaves of the tree have age 0
  - because they correspond to present-day species.
- We define the weight of an edge (v, w) in the tree as the difference $\text{AGE}(v) - \text{AGE}(w)$.
- The length of a path between the root and any node would be equal to the difference between their ages.
- A tree, in which the distance from the root to any leaf is the same, is called **ultrametric.**

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

Ultrametric Evolutionary Trees

- Our aim is to derive an ultrametric tree that explains a distance matrix (even if it is only approximately).
- **UPGMA** (Unweighted Pair Group Method with Arithmetic Mean) is a simple clustering heuristic construct an ultrametric evolutionary tree.

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

UPGMA

- Given an $n \times n$ matrix D, UPGMA first forms n trivial clusters, each containing a single leaf.
- The algorithm then finds a pair of "closest" clusters.
- To calculate closest clusters, UPGMA defines the distance between clusters *C1* and *C2* as the average pairwise distance between elements of *C1* and *C2*.

$$D_{C_1,C_2} = \frac{\sum_{i \in C_1} \sum_{j \in C_2} D_{i,j}}{|C_1| \cdot |C_2|} .$$

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

UPGMA

- $|C|$ denotes the number of leaves in cluster C.
- Once a pair of closest clusters $C1$ and $C2$ are identified, merges them into a cluster C with $|C1|+|C2|$ elements and then creates a node for C, which it connects to each of $C1$ and $C2$ by a directed edges.
- The age of C is set to be $D_{C1,C2}/2$.
- UPGMA then iterates this process of merging the two closest clusters until only a single cluster remains, which corresponds to the root.

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

UPGMA Algorithm

UPGMA(D, n)
- *Clusters* $\leftarrow$ n single-element clusters labeled $1, \ldots, n$
- construct a graph T with n isolated nodes labeled by single elements $1, \ldots, n$
- **for** every node v in T
  - $\text{AGE}(v) \leftarrow 0$
- **while** there is more than one cluster
  - find the two closest clusters C_i and C_j (break ties arbitrarily)
  - merge C_i and C_j into a new cluster C_{new} with $|C_i| + |C_j|$ elements
  - add a new node labeled by cluster C_{new} to T
  - connect node C_{new} to C_i and C_j by directed edges
  - $\text{AGE}(C) \leftarrow D_{C_i,C_j}/2$
  - remove the rows and columns of D corresponding to C_i and C_j
  - remove C_i and C_j from *Clusters*
  - add a row/column to D for C_{new} by computing $D(C_{new}, C)$ for each C in *Clusters*
  - add C_{new} to *Clusters*
- *root* $\leftarrow$ the node in T corresponding to the remaining cluster
- **for** each edge (v, w) in T
  - length of $(v, w) \leftarrow \text{AGE}(v) - \text{AGE}(w)$
- **return** T

USING LEAST SQUARE TO CONSTRUCT PHYLOGENY

Example:

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

⬇

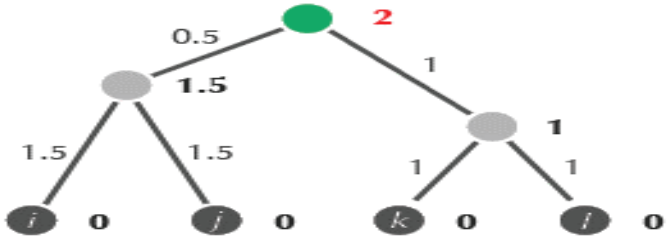
	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	**2**
l	3	5	**2**	0

⬇

	i	*j*	{*k*, *l*}
i	0	**3**	3.5
j	**3**	0	4.5
{*k*, *l*}	3.5	4.5	0

⬇

	{*i*, *j*}	{*k*, *l*}
{*i*, *j*}	0	**4**
{*k*, *l*}	**4**	0



Disadvantages of Using the Neighbors

- The first step that UPGMA is to merge the two leaves i and j with minimum distance $D_{i,j}$ into a single cluster.
- And we have already seen that the smallest element in the distance matrix does not necessarily correspond to a pair of neighboring leaves!
- This is a concern, since if UPGMA generates incorrect trees from additive matrices, then it is not an ideal heuristic for evolutionary tree construction from non-additive matrices.
- ***Can we find an algorithm that always identifies neighboring leaves in an additive distance matrix but also performs well on a non-additive distance matrix?***