8.1   Introduction

In the last lecture, we have learnt that **phylogenetic studies the genetic relationship between different species**. The phylogenetic tree reconstructs the evolutionary relations of a set of species.

In addition, we have learned how to reconstruct a phylogenetic tree based on **character data**. In such **character based methods**, we try to minimize the number of mutations.

However, intuitively, species which look similar should be evolutionary more related.This motivates us to define the **distance** between two species to be the number of mutations need to change one species to another.

In this lecture, new methods of phylogenetic tree reconstruction will be presented, namely **distance based methods**. These methods make use of **distance based data**. Given n species, the input is a $n \times n$ distance matrix M where M_{ij} is the mutation distance between species i and species j. Figure 8.1 illustrates an example of such **distance based tree**, and Table 8.1 shows its **distance matrix**.
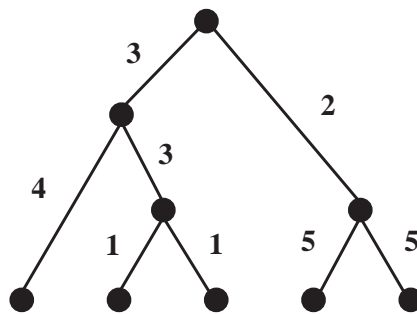

Figure 8.1: A Sample Phylogenetic Tree

M also satisfies centain constrains. We are expecting the method outputs a tree of degree 3 which is consistent with the distance matrix. We will extend the discussion on those constraints in following sections.

M	a	b	c	d	e
a	0	8	8	14	14
b	8	0	2	14	14
c	8	2	0	14	14
d	14	14	14	0	10
e	14	14	14	10	0

Table 8.1: An example distance matrix M: Entry M_{ij} stands for the mutation distance from species i to species j.

8.2 Constraints for the Distance Matrix M

There are three constraints on the distance matrix M:

- **M should be a metric**

- **M is an additive metric**

- **M is ultrametric (optional)**

Let's discuss them one by one.

8.2.1 Metric Space

Definition 8.1 *A distance metric M is said to be a* metric, *if and only if it satisfies:*

Symmetric: $M_{ij} = M_{ji}$ *and* $M_{ii} = 0$

Triangle Inequality: $M_{ij} + M_{jk} \geq M_{ik}$

Look back at the matrix in Table 8.1, it is symmetric and satisfies the triangle inequality. Thus, it is a metric. In the following discussion, we assume that the distances between all species satisfy the metric space.

8.2.2 Additive Metric

Definition 8.2 *Let S be a set of species, and let M be the distance matrix for S. If there exists a tree T where:*

- *Every edge has a positive weight and every leaf is labelled by a disinct species in S*

- *For every $i, j \in S$, $M_{ij} =$ the sum of the edge weights along the path from i to j*

Then, M is an **additive metric**. *T is called an* **additive tree***.*

Table 8.2 shows an example of additive metric and Figure 8.2 is the corresponding additive tree. Note that it is only an unrooted phylogeny, since we do not know the root.

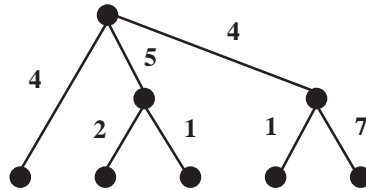| M	a	b	c	d	e
a	0	11	10	9	15
b	11	0	3	12	18
c	10	3	0	11	17
d	9	12	11	0	8
e	15	18	17	8	0

Table 8.2: An Example of Additive Metric

Figure 8.2: The corresponding additive tree of Table 8.2

8.2.2.1 Properties of Additive Metric

P. Buneman proposed a **4-point condition** of additive metric [Buneman. 1].

Theorem 8.1 (Buneman's 4-point Condition) *M is additive if and only if for any four species, we can label them as i, j, l, k such that in S,*
$M_{ik} + M_{jl} = M_{il} + M_{jk} \geq M_{ij} + M_{kl}$

As an example, let us choose a, b, c, d in Figure 8.2 as the four species. Therefore:

$M_{ac} + M_{bd} = 10 + 12 = 22; M_{ab} + M_{cd} = 11 + 11 = 22; M_{ad} + M_{bc} = 9 + 3 = 12.$

So
$$M_{ac} + M_{bd} = M_{ab} + M_{cd} \geq M_{ad} + M_{bc}.$$

Proof: $(\Rightarrow)$

- If M is additive, there exists an additive tree T for S.

- Consider the subtree for the 4 species i, j, k, l. the subtree is shown in Figure 8.3. Note that x and y may be identical (when $M_{xy} = 0$).
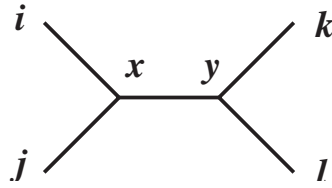
Figure 8.3: Buneman's 4-Point Condition

$$M_{ik} + M_{jl}$$
$$= (M_{ix} + M_{xy} + M_{yk}) + (M_{jx} + M_{xy} + M_{yl})$$
$$= M_{ix} + M_{jx} + M_{yk} + M_{yl} + 2M_{xy}$$

$$M_{jk} + M_{il}$$
$$= (M_{jx} + M_{xy} + M_{ik}) + (M_{ix} + M_{xy} + M_{yl})$$
$$= M_{ix} + M_{jx} + M_{yk} + M_{yl} + 2M_{xy}$$

$$M_{ij} + M_{kl}$$
$$= M_{ix} + M_{xj} + M_{ky} + M_{yl}$$

So it can be easily verified that: $M_{ik} + M_{jl} = M_{il} + M_{jk} \geq M_{ij} + M_{kl}$.
($\Leftarrow$) Will not present here. ∎

8.2.2.2 Additive Check

Based on the 4-point condition, we can check whether a matrix M is additive or not.

8.2.2.3 Validity of the Additive Metric

Let's have a look of the validity of additive metric. Recall that the evolution distance captures the actual number of mutations between a pair of species.

Therefore, if we know the correct tree for a set of species and the exact number of mutations for each edge, the distance (number of mutations) between two species i and j should be the sum of the edge weights along the path from i to j. On this perspective, additive metric seems reasonable.

8.2.2.4 About Hamming Distance

We may ask "is hamming distance additive?" In other word, for any two species i and j, we define M_{ij} to be the hamming distance between species i and j. (E.g. Let's say species i is (A,C,G,C,T) and species j is (C,C,A,C,T). The hamming distance $h_{ij} = 2$.)

Unfortunately, the answer is "NO"! Hamming distance fails to capture the "multiple" mutations on the same site. It has no additive property.

For example, at the first position of species i and j, we can not tell whether the mutation is $A \Rightarrow C$, or $A \Rightarrow G \Rightarrow C$.

The solution is to use the corrected distance transformation. For CF tree model as explained in the last lecture, we compute the corrected distance as:

$$M_{ij} = -\log(1 - 2h_{ij}/m)/2$$

As the number of characters increase, M converges to an additive metric.

8.2.3 Ultrametric

Definition 8.3 *Let M be an additive metric. If there exists a tree such that*

1. *The distance between any two species i and j equals the sum of the edge weights along the path from i to j.*

2. *A root of the tree can be identified such that the distance to all leaves from the root is the same, that is, the length is a fixed value.*

Then M is known as an **ultrametric** *and the tree mentioned is called an* **ultrametric tree**.

Note that any subtree of an ultrametric tree is also an ultrametric tree. An example of an ultrametric matrix and it's corresponding ultrametric tree are given in Figure 8.1 and Table 8.1, respectively.

8.2.3.1 Properties of Ultrametric

Since ultrametic is an additive metric, it satisfies 4-point condition. In addition, it satisfies **3-point condition**.

Theorem 8.2 *M is ultrametric if and only if for any three species in S, we can label them i, j, k such that $M_{ik} = M_{jk} \geq M_{ij}$*

Proof: ($\Rightarrow$)Given any 3 species, they must be arranged in the Ultrametric tree in the manner shown in Figure 8.4.

By Property 2 of an additive tree:

$$
\begin{aligned}
M_{ij} &= M_{iy} + M_{jy} \\
M_{ik} &= M_{iy} + M_{yx} + M_{kx} \\
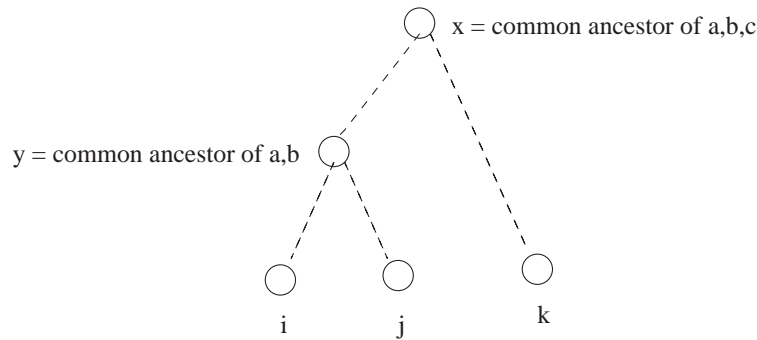M_{jk} &= M_{jy} + M_{yx} + M_{kx}
\end{aligned}
$$

Figure 8.4: Ultrametric Tree

From the above formulas,and by Property 3 of an Ultrametric tree. There is

$$M_{ik} = M_{jk} = 2 * (M_{iy} + M_{yx}) > 2M_{iy} = M_{iy} + M_{jy} = M_{ij}$$

proven!
($\Leftarrow$)Exercise.   ■

8.2.3.2   Ultrametric Check

Based on the 3-point condition, we can check whether a matrix M is ultramatric or not.

8.2.3.3   Constant Molecular Clock

The Constant Molecular Clock is an assumption stating that all species evolved at a constant rate from a common ancestor. This is the assumption behind an Ultrametric tree, which is reflected by the property that the distance from the root to all leaves are the same. However it must be noted that the assumption of a constant molecular clock is not always correct.

8.3   Computational Problems

Let M be a distance matrix for a set of species S. There are three computational problems on the distance matrix M:

- **Given M is ultrametric, we want to reconstruct the corresponding ultrametric tree T in polynomial time**

- **Given M is additive, we want to find an polynomial time algorithm to recover the corresponding additive tree T**

- **Given M is not exactly additive, we want to find the nearest additive tree T**

We will discuss them one by one.

8.3.1 Ultrametric Tree Reconstruction from Matrix

Given an ultrametric matrix M of a set of species, we want to reconstruct the phylogenetic tree T for S.

There are two methods for constructing the Ultrametric tree from a given Ultrametric matrix for n species. The first method is the UPGMA (Unweighted Pair Group Method with Arithmetic Mean) and the second is the WPGMA (Weighted Pair Group Method with Arithmetic Mean).

8.3.1.1 UPGMA (Unweighted Pair Group Method with Arithmetic Mean)

This method works by building the phylogenic tree bottom up from its leaves (the given set of species). It is basically a clustering algorithm with each species forms a cluster first, then 2 smaller clusters of nodes are grouped together (forms a bigger node represented by the root of the subtree) recursively until there is only one phylogenic tree which contains all the species.

Consider an ultrametric tree T. If a subset of species S form a subtree of T, we call it a **cluster**. Thus, every species forms a cluster by itself.

For a node u, define **height(u)** be the path length from u to any of its descendent leaf (Since T is ultrametric, every path should have the same length!). Let i and j be the descendent leaves of u in two different subtrees. To ensure that the distance from the root to both i and j are the same, height(u) = $M_{ij}/2$.

For any two clusters C1 and C2 of T, Define

$$dist(C_1, C_2) = \frac{\sum_{i \in C_1, j \in C_2} M_{ij}}{|C_1| \times |C_2|}$$

Note that $dist(C_1, C_2) = M_{ij}$ for all $i \in C_1$ and $j \in C_2$.

For example, let u be the lowest common ancestor of i and j. dist(C1, C2) = 2 height(u)! As illustrated in Figure 8.5. And for any node C_x whose ancestor is not u, it is obvious that

$$dist(C_1 \cup C_2, C_x) = \frac{dist(C_1, C_x) + dist(C_2, C_x)}{2}$$

According to these analysis, we have

Lemma 8.1 *For a set C of clusters, let C_i, C_j be two clusters in C, and let C_k be a tree formed by joining C_i and C_j with a root. C_k is a cluster (subtree) of the ultrametric tree T.*
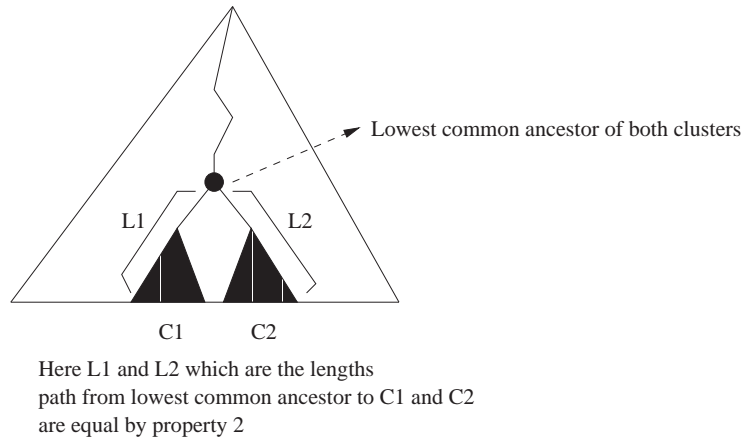
And here comes the UPGMA algorithm

Here L1 and L2 which are the lengths
path from lowest common ancestor to C1 and C2
are equal by property 2

Figure 8.5: Ultrametric Tree

Algorithm 8.1

Input: An $n \times n$ Ultrametric distance matrix M for n species

Output: An Ultrametric tree T for M

Initialization: Define C to be the set of clusters

1. *Let $C = \{\{c_1\}, \{c_2\}, \ldots, \{c_n\}\}$. Here each cluster is a singleton cluster that contains only 1 species*

2. *For all $c_i, c_j \in C$, initialize $dist(\{c_i\}, \{c_j\}) = M_{ij}$*

3. *Iteration($n - 1$ times):*

   (a) *Determine cluster C_i, $C_j \in C$ such that $dist(C_i, C_j)$ is minimum*

   (b) *Define a new cluster $C_k = C_i \cup C_j$*

   (c) *$C = C - \{C_i, C_j\} \cup \{C_k\}$*

   (d) *Define a new node c_k and let c_k be the parent of c_i and c_j. Also, define $height(c_k) = dist(c_i, c_j)/2$*

   (e) *For all $C_x \in C-\{C_k\}$, define $dist(C_x, C_k) = dist(C_k, C_x) = (dist(C_i, C_x)+ dist(C_j, C_x))/2$*

   (Note that after each iteration, 1 new cluster is created and 2 removed, therefore after $n - 1$ iterations, the algorithm would be able to recreate the phylogenic tree)

Here again referring to the Figure 8.6, we know that $dist(b, d) = dist(c, d)$ therefore average out by dividing by 2.
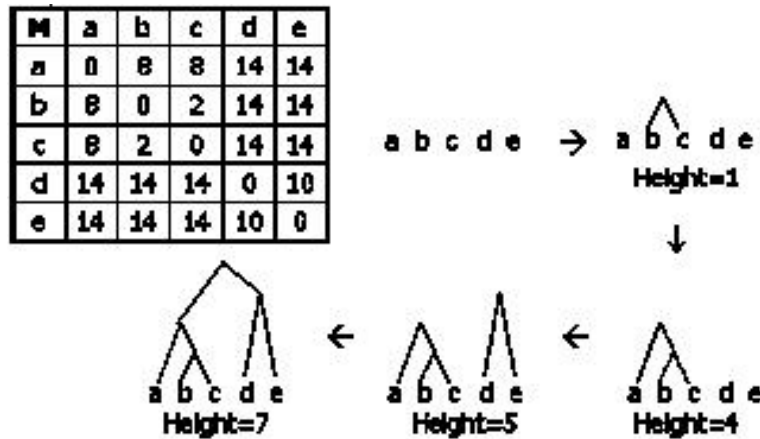
Figure 8.6: Ultrametric Tree

The algorithm takes n iterations as mentioned above, and in each iteration the dominating operation is to find the pair of cluster with the minimum distance i.e step 1 in the iteration which is bounded by n^2 operations in the 1st iteration. Subsequently by maintaining a pointer to the minimum pair, step 1 needs only n operations as we just have to compute the pairwise distance between the new cluster and the rest of the clusters, while updating the minimum pair if a new minimum is found.

Therefore the final time complexity is $O(n^2)$.

The space usage is bounded by the distance matrix used, thus it is $O(n^2)$.

8.3.1.2 WPGMA (Weighted Pair Group Method with Arithmetic Mean)

Basically WPGMA works in the same as UPGMA with the only difference in the distance function used in the last step. Instead of finding the average, we find the weighted average, which ensures that **each species contributes equally to the final result**.

In WPGMA, the following function is used to calculate the distance from the newly created cluster to all other clusters.

$$dist(C_{(i \cup j)}, C_x) = dist(C_k, C_x) = \frac{n_i * dist(C_i, C_x) + n_j * dist(C_j, C_x)}{n_i + n_j}$$

where C_i and C_j are clusters to be merged, and C_k is the cluster such that $C_k = C_i \cup C_j$. C_x is cluster in C but not in C_k. n_i and n_j are the respective size of C_i and C_j.

Below lemma shows that this distance function is a consistent definition with the original cluster distance definition. And it is the proof that according to the above equation, each species contributes equally to the final result.

Lemma 8.2

$$dist(C_i, C_j) = \frac{1}{n_i \times n_j} \sum_{c_i \in C_i} \sum_{c_j \in C_j} dist(c_i, c_j)$$

Proof: This can be shown by mathematical induction. Let

$$dist(C_{i \cup j}, C_x) = \frac{n_i dist(C_i, C_x) + n_j dist(C_j, C_x)}{n_i + n_j}$$

Then by definition, we have

$$
\begin{aligned}
dist(C_{i \cup j}, C_x) &= \frac{\frac{n_i}{n_i n_x} \sum_{c_i \in C_i} \sum_{c_x \in C_x} dist(c_i, c_x) + \frac{n_j}{n_j n_x} \sum_{c_j \in C_j} \sum_{c_x \in C_x} dist(c_j, c_x)}{n_i + n_j} \\
&= \frac{\frac{1}{n_x}(\sum_{c_i \in C_i} \sum_{c_x \in C_x} dist(c_i, c_x) + \sum_{c_j \in C_j} \sum_{c_x \in C_x} dist(c_j, c_x))}{n_i + n_j} \\
&= \frac{\frac{1}{n_x} \sum_{c_k \in C_i \cup C_j} \sum_{c_x \in C_x} dist(c_k, c_x)}{n_i + n_j}
\end{aligned}
$$

Let $C_i \cup C_j = C_k$, which means $n_i + n_j = n_k$

$$dist(C_k, C_x) = \frac{\sum_{c_k \in C_k} \sum_{c_x \in C_x} dist(c_k, c_x)}{n_k n_x}$$

proven!   ∎

According to this lemma, we can easily conclude that according to the distance function used by WPGMA,**each species contributes equally to the final result**.

8.3.1.3   Final Notes

When the distance matrix M is slightly deviate from ultrametric, both UPGMA and WPGMA still can recover the topology.

Also note that the generalization of ultrametric trees are additive trees. Recall that in an ultrametric tree, the number of mutations was assumed to be proportional to the temporal distance of a node to the ancestor and it was also assumed that the mutations took place with the same rate in all paths. Thus an ultrametric tree is assigned a root and the distance from the root to every leave is constant. But it's a fact that the evolutionary clock is running differently for different species and even for different regions e.g., in a protein sequence.

8.3.2   Additive tree reconstruction

The additive tree constructed from an additive metric works by successive insertion. There is exactly one tree topology that allows for realization of an additive

metric. Suppose M is an additive metric, the construction algorithm provided by Waterman et al, can reconstructs the additive tree T in $O(n^2)$ time.

For simplicity, in the following discussing, we assume that the constructed additive tree is an unrooted degree-3 tree.

8.3.2.1   Two Species

For 2 species i and j, the additive tree is just an edge with weight M_{ij}.

8.3.2.2   Three Species

Definition 8.4 *We call d_{xy} be the length of the path from x to y.*

3-star method!

For 3 species i, j and k, the additive tree must be three leaves attaching to a common center c (see Figure 8.7).
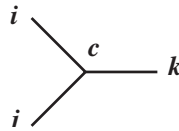

Figure 8.7: Three Species

Theorem 8.3 *Let M be an additive matrix, for 3 species i, j and k, the corresponding additive tree contains one internal node c such that:*

$$d_{ic} = \frac{M_{ij} + M_{ik} - M_{jk}}{2} \tag{8.1}$$

$$d_{jc} = \frac{M_{ij} + M_{jk} - M_{ik}}{2} \tag{8.2}$$

$$d_{kc} = \frac{M_{ik} + M_{jk} - M_{ij}}{2} \tag{8.3}$$

Proof:

$$M_{ik} = d_{ic} + d_{ck} \tag{8.4}$$

$$M_{jk} = d_{jc} + d_{ck} \tag{8.5}$$

$$M_{ij} = d_{ic} + d_{cj} \tag{8.6}$$

∎

By solving the three equations, the lemma follows.

Lemma 8.3 *Since the distance from internal node c to i, j and k is decided, the additive tree for three species is unique.*

8.3.2.3 Four Species

Given four species h, i, j, k, we can reconstruct the additive tree for the four species contains 2 internal nodes (see Figure 8.8). Here we describe a method to identify the two internal nodes.
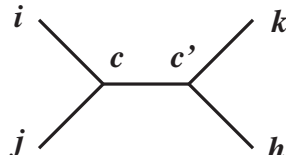

Figure 8.8: Four Species with two internal nodes

1. Get the additive tree of i, j, k by using the 3-star method, then we identify the location of the internal node c.

2. To include h into the tree, we need to introduce one more internal node c'. c' will split either (i, c), (j, c) or (k, c).

3. To check whether h splits edge (k, c), we apply 3-star method for species i, k and h to create a new internal node c'. If $d_{kc'} < d_{kc}$, then split (k, c) to insert an edge from c' to h.

4. Otherwise, use the same approach in step 2 to check whether c' split edge (i, c) or (j, c)

   Note that since c' can only split exactly one edge, the additive tree for four species is unique.

8.3.2.4 k Species

Given k species, we can reconstruct the additive tree as follows:

- Assuming that we have already gotten the additive tree T' for $k-1$ species, we are going to insert the species k, which can split only one edge of T'.

- For every edge of T', we check whether k splits the edge using 3-star method. Because species k can only split exactly one edge of T', the additive tree for k species is unique.

Note that the check time required is $O(k-1)$, and the tree is also unique.

8.3.2.5   Time Complexity

The time complexity of each 3-star check is $O(1)$. To insert a species to a tree with k leaves, we need to check the $O(k)$ edges by the 3-star method using $O(k)$ time. Therefore, if there are n species, the time complexity to construct an additive tree is $O(1 + 2 + \ldots + n) = O(n^2)$.

8.3.2.6   Uniqueness

Be aware that the additive tree for matric M is **unique** by using 3-star method.

8.3.2.7   Examples

M is an additive metric in Table 8.3:

M	a	b	c	d	e
a	0	11	10	9	15
b	11	0	3	12	18
c	10	3	0	11	17
d	9	12	11	0	8
e	15	18	17	8	0

Table 8.3: An Example Matrix M

Step 1 Choose a pair of species, say, a and b, which results in the first path in the tree.

Step 2 Choose a third species, say, c, and establish the linear equations with 3-star method to let the species branch off the path.

Step 3 To add the forth species d to the additive tree, choose a pair of leaves in the tree constructed so far and compute the point a newly chosen species is inserted at.

If the new path branches off an existing branch in the tree, then do the insertion step once more in the branching path.

If the new path branches off an edge in the tree, then this insertion is finished.

Step 4 We add the last species e to the additive tree with the same method in step 3.

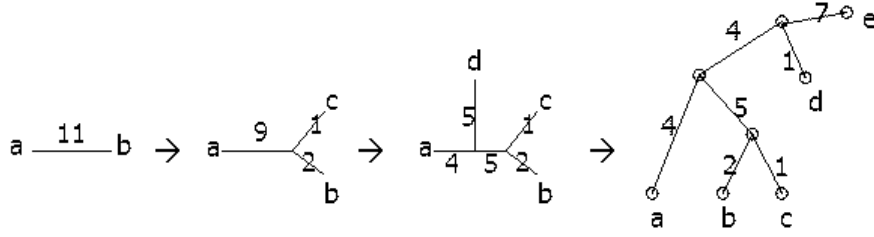The process is illustrated in Figure 8.9

Figure 8.9: Example of additive tree reconstruction

8.3.2.8   Relationship Between Additive tree and Ultrametric Tree

An additive tree is constructed based on an additive distance matrix. And an ultrametric tree is constructed based on an ultrametric distance matrix. Note that any ultrametric tree is an additive tree since it satisfied 4-point condition of additive tree. But the converse is not true.

Since the ultrameric tree has an extra property, it is easier to construct the ultrametric tree. So, one way to ease the construction of an additive tree is to transform an additive distance matrix M to an ultrametric distance matrix E, and build up the tree based on E. One method to transform M to E is ***Farris Transform***:

- Let M be an additive distance matrix for a set of species $S = \{1, 2, \cdots, n\}$.

- Let T be the corresponding additive tree for M and r denotes an internal node of T.

- According to the Ferris Transformation, M is transformed to an matrix E, defined as

$$\forall \quad 1 < i, j < n, \quad E_{ij} = \frac{M_{ij} - M_{ir} - M_{jr}}{2} + \overline{M_r} \tag{8.7}$$

where

$$\overline{M_r} = \frac{1}{n} \sum_{i=1}^{n} M_{ir}. \tag{8.8}$$

Theorem 8.4 *Based on the above transformation, E is ultrametric.*

Proof: Let T be the additive tree for M. Assume that T is rooted at some internal node r. Also assume that $lca(i, j)$ denotes the least common ancestor of species i and j. So, the leaves i and j have the least common ancestor $lca(i, j)$ (see Figure 8.10).

Now, we know that M_{ij} is the sum of the weights of the edges along the path from i to j, where each edge on the path is visited once only. This path contains the node $lca(i, j)$ of i and j. So, M_{ij} is the sum of $M_{i,lca(i,j)}$ and $M_{j,lca(i,j)}$. i.e.,

$$M_{ij} = M_{i,lca(i,j)} + M_{j,lca(i,j)}$$

We have , $M_{i,lca(i,j)} =$ distance from i to root r - distance from $lca(i,j)$ to r, and $M_{j,lca(i,j)} =$ distance from j to the root r - distance from $lca(i,j)$ to r. Hence,

$$M_{i,lca(i,j)} = M_{ir} - M_{r,lca(i,j)} \qquad (8.9)$$

$$M_{j,lca(i,j)} = M_{jr} - M_{r,lca(i,j)} \qquad (8.10)$$

$$i.e., \quad M_{ij} = M_{ir} - M_{r,lca(i,j)} + M_{jr} - M_{r,lca(i,j)} \qquad (8.11)$$

$$= M_{ir} + M_{jr} - 2.M_{r,lca(i,j)} \qquad (8.12)$$

$$i.e., \frac{M_{ij} - M_{ir} - M_{jr}}{2} = -M_{r,lca(i,j)}. \qquad (8.13)$$

$$By \ equation \ (8.7), \quad E_{ij} = \overline{M_r} - M_{r,lca(i,j)} \qquad (8.14)$$

Now, Equation (8.14) implies that, for any two species (i.e., any two leaves) i and j, E_{ij} is fixed quantity $\overline{M_r} - M_{j,lca(i,j)}$ which is constant if the internal node r is pre-defined (or fixed). Here we have already chosen r as the root. So, Equation (8.14) implies that the transformation is well defined. Now replace j by r in Equation (8.14). Then as r is the root, $lca(i,j)$ is nothing but r and $M_{rr} = 0$. Hence we have,

$$i.e., \quad E_{ir} = \overline{M_r} - M_{rr} = \overline{M_r} \qquad (8.15)$$

Equation (8.15) implies that the distance from the root to any of the leaves (i.e., species) is constant, namely the quantity $\overline{M_r}$. This shows that E is ultrametric.

∎

8.3.3 Nearly Additive Tree Reconstruction

In Section 8.3.2.8, we have already discussed the method of converting an additive metric M to an ultrametric E. The ultrametric tree can be constructed based on the transformed metric.

Here the problem is when the matrix M is not additive. In these cases, approximate algorithms are used to build a nearly additive tree. Three basic approximation methods are discussed to construct nearly additive tree:

- Least Squares Method

- Neighbor-Joining Method

- L_∞-metric
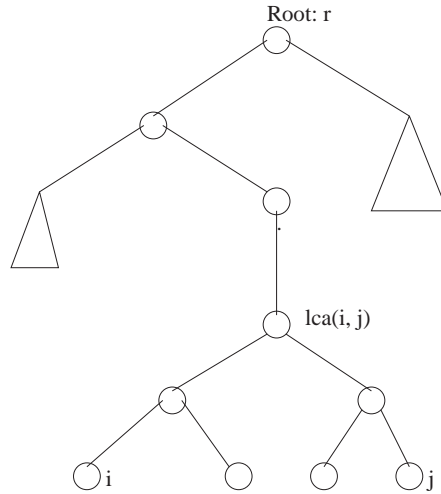
Figure 8.10: Least Common Ancestor lca(i,j) of species i and j

Among these three methods, Neighbor-Joining Method is the most poplar one.

8.3.3.1 Least Squares Method

The principle of *Least Squares* is a general method for estimating unknown parameters values so that error is minimized. It has widely applied in various fields like statistical inferrence, finding linear regression, solving linear system of equations when the exact solution does not exist. This method is also used for estimating nearest additive metric D for a given non-additive metric.

Suppose a metric M is given for a set S of n species. Let M be a non-additive metric and D be an additive metric. Let us consider a tree T which is constructed based on the additive metric D. The differenc between M and D can be estimated based on the sum of the squares of error (due to D) is $SSQ(T)$, where,

$$SSQ(T) = \sum_{i=1}^{n} \sum_{j \neq i} (D_{ij} - M_{ij})^2$$

In a more general setup, a set of weights w_{ij} is chosen for each pairs of species i and j, and define SSQ(T) as:

$$SSQ(T) = \sum_{i=1}^{n} \sum_{j \neq i} w_{ij} (D_{ij} - M_{ij})^2$$

If there exists an additive metric D^* such that $SSQ(T)$ is minimum among all choices of D, i.e., if T minimizes the sum of squares of errors $SSQ(T)$, then D^* is called a least square additive estimate for the non additive metric M. So,

for finding such estimation, we need to search all possible additive tree based on all possible additive metric D and choose one which give the smallest value of $SSQ(T)$. No polynimial time algorithm exits for solving such problem. Basically this is an NP-hard problem[2]. That's why people use other approximation algorithm, e.g., Neighbor-joining method.

8.3.3.2   Neighbor Joining Method

Since finding the least square tree (discussed above) of a non-additive metric is NP-hard, people are interested to find an approximation to the least square tree. *Neighbor-joining (NJ) method*[Saitou. 3, Studier. 4] attempts to approximate the least square tree for a given distance matrix M. It is a hierarchical method of constructing a larger cluster (say C) by joining two nearest neighbors(say A and B) so that the distances of these two clusters from the remaining clusters are as far as possible. Initially this method considers a collection of clusters of all the species, where each species corresponds to a cluster. In each iteration, two clusters are merged to form a larger cluster and at the end we are left with two cluster to merge to form the root of the tree. Figure 8.11 shows a few steps of NJ-algorithm. The NJ-method can be formalized as follows:

Input: An $n \times n$ distance matrix M of n species.

Output: An approximate least square tree for M.

Initialization: 1. Let $Z = \{\{1\}, \{2\}, \cdots, \{n\}\}$ be the set of initial clusters.

2. For all $\{i\}, \{j\} \in Z$, set $dist(\{i\}, \{j\}) = M_{ij}$.

Iterative Steps: Repeat the following steps $n-1$ times:

1. For every cluster $A \in Z$,
   compute $u_A = \frac{1}{n-2}\sum_{D \in Z} dist(D, A)$.

2. Find two clusters $A, B \in Z$ such that $dist(A, B) - u_A - u_B$ is minimun.

3. Make A and B as two subtree of a new internal node r. Denote this bigger cluster by C.

4. Compute the branch length:

$$d_{Ar} = \tfrac{1}{2}dist(A, B) + \tfrac{1}{2}(u_A - u_B) \text{ and}$$
$$d_{Ar} = \tfrac{1}{2}dist(A, B) + \tfrac{1}{2}(u_B - u_A).$$

5. Update the set of cluster $Z = Z \cup \{C\} - \{A, B\}$.

6. Update the pairwise distances of the clusters in Z: compute $dist(D, C)$ for all $D \in Z - \{C\}$,

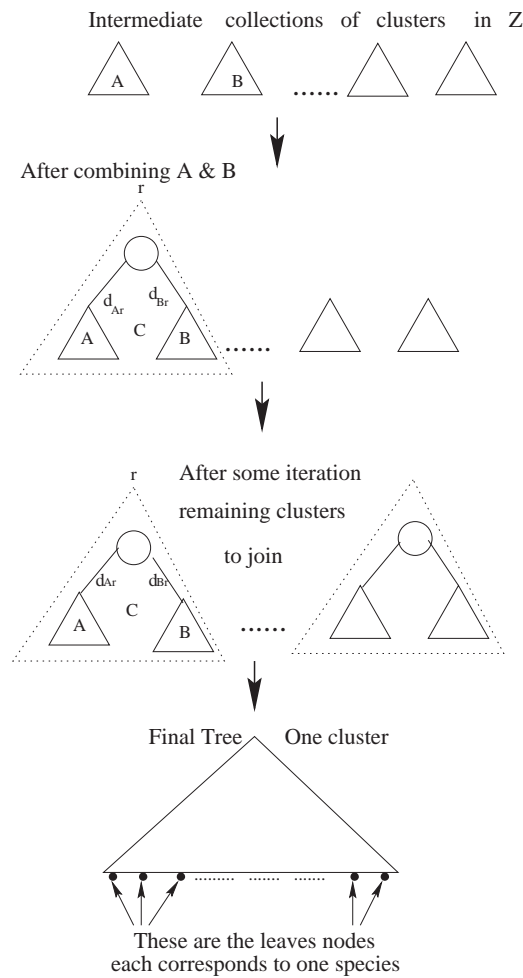$$dist(D, C) = dist(C, D) = \tfrac{1}{2}(dist(A, D) + dist(B, D) - dist(A, B))$$

Figure 8.11: Neighbor Joining Method

Time Complexity: Since there are n species, i.e., n^2 entries, initialization requires $O(n^2)$ time. It seems that Step 1 requires $O(n^2)$ time. But this can be done in linear time. In Step 1, instead of computing the values u_D directly, at every iteration, one can update u_D for any $D \in Z$. Say, for $D \in Z$, new $u_D = \frac{1}{|Z|-2}((|Z| - 1).u_D - dist(D, A) - dist(D, B) + dist(D, C))$, where $|Z|$ is the current size of Z, i.e., number of clusters in Z after merging A and B. This can be performed in constant time. Computation of u_C takes at most $O(n)$ time. So, Step 1 requires $O(n)$ time. Step 2 requires $O(n)$ time; Steps 3 and 4 require

constant time. Since C is the new clusters added to Z, in Step 6, updating (of at most n) pairwise distances among the clusters requires $O(n)$ time. So, each iteration requires $O(n)$ time. There are $O(n)$ iterations. Hence the total time complexity is $O(n^2)$.

8.3.3.3 L_∞-Metric Method

The L_∞-norm for two distance matrices M and E is defined as:

$$L_\infty(M, E) = MAX_{i,j}|M_{ij} - E_{ij}|$$

According to the method of L_∞, given a non-additive matrix M, we have to find an additive matrix E such that $L_\infty(M, E)$ is minimized. i.e., we need to consider all possible additive matrix E and compute $L_\infty(M, E)$, and choose an E which gives the minimum value. This is an NP-hard problem. But Agarwala et.al.[Agarwala. 5] have proposed an 3-approximation algorithm with respect to L_∞-metric which runs in polynomial time.

References

[1] BUNEMAN, P. (1974), A Note on the Metric Properties of Trees, Journal of Combinatorial Theory (B), 17, 48-50.

[2] Farach, M., Kannan S. and T. Warnow(1996). A robust Model for Finding Optimal Evolutionary Trees, *Algorithmica, special issue on Computational Biology*, vol. 13, No. 1, pp. 155-179.

[3] Saitou, N. and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:405-425, 1987.

[4] Studier, J.A. and K.L. Keppler. A note on the neighbor-joining algorithm of Saitou and Nei. *Molecular Biology and Evolution*, 5:729-731, 1988

[5] Agarwala, R. and D. Fernandez-Baca(1993). A Polynomial-Time Algorithm for the Perfect Phylogeny when the Number of Character States is Fixed. *SIAM Journal on Computing*, vol. 23, no. 6, pp. 1216-1224, Dec. 1994.

[6] Phylogenetics related resources on the Internet

- List of phylogenetic resources
  $http://www.ucmp.berkeley.edu/subway/phylogen.html$
- List of phylogeny software packages available on the web
  $http://evolution.genetics.washington.edu/phylip/software.html$

- Detailed information on phylogentic algorithms
  *http : //www.icp.ucl.ac.be/ opperd/private/phylogeny.html#anchor*2824426
  UPGMA
  http : //www.icp.ucl.ac.be/ opperd/private/upgma.html
  Neighbor-joining
  http : //www.icp.ucl.ac.be/ opperd/private/neighbor.html
  Parsimony
  http : //www.icp.ucl.ac.be/ opperd/private/parsimony.html
- Database of phylogenetic knowledge
  http : //herbaria.harvard.edu/treebase/
- Additional figures, scribes and information
  http : //www.cs.washington.edu/education/courses/590bi/98wi/