Lecture 7 – "Algorithmic Approaches" to Phylogeny Estimation

I. Introduction: So, there's this vast tree space out there for phylogenetic studies of more than a few taxa. As I've said, there are phylogenetic approaches that try to search this tree space with varying degrees of rigor, and those that attempt simply to build a tree using some particular algorithm and hope that it's a good tree. (Even for tree-searching approaches, we must start with some tree(s) and this needs to be built from our data.)

These are known as "Algorithmic Approaches" to tree building and may be applied to either character data or a matrix of pairwise distances. One of the most widely used of these methods (neighbor joining) was developed by Saitou & Nei; Nei's justification is as such.

Any estimate of a phylogenetic tree has a large variance. Therefore, any tree that we can demonstrate to be optimal (under some criterion - ML, MP or ME) is not guaranteed to be the true tree; a tree produced by a fast algorithm may be just as close to the true tree. Therefore, it makes little sense to waste time & resources searching for the optimal tree.

This point merits discussion and we'll discuss it more later. For now, I agree that this is true under some limited contexts (e.g., identifying a tree for evaluation of models), but it's probably a bad idea to make it one's default position.

Nevertheless, we should have an understanding of how these methods work, because they can be incorporated into lots of analyses and, as we've seen, they're really important in alignments.

II. UPGMA - Phenetic clustering

Early on, one of the most common algorithms used to generate a tree was Unweighted Pair-Group Method using Arithmetic means. This dates back to Sokal and Michner and the early pheneticists, and as such, **its originally intended purpose was not phylogeny reconstruction, but to visualize overall similarity**. As we saw in our discussion of alignment, UPGMA is still often used in building intermediate guide trees.

Steps:

1) Identify the most similar pair of taxa (i.e., that pair with the smallest $D_{i,j}$).

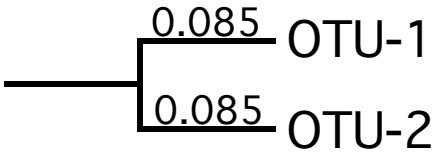
2) Merge them into a composite OTU (called *i/j*), each at the end of a branch ($D_{i,u}$ & $D_{j,u}$) of length $D_{i,j}/2$.

3) Create a new matrix with the distances to composite taxon *i/j* calculated as the average of all pairwise comparisons: $D_{k,i/j} = (n_i D_{k,i} + n_j D_{k,j}) / (n_i + n_j)$, where n_i is the number of taxa in cluster.

4) Iterate until all taxa have been added.

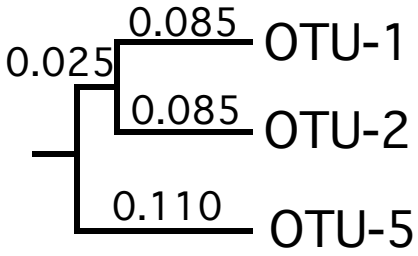
Data are from Olsen et al. (1988).

	OTU-1	OTU-2	OTU-3	OTU-4	OTU-5
OTU-1	-------	**0.17**	0.21	0.31	0.23
OTU-2		-------	0.30	0.34	0.21
OTU-3			-------	0.28	0.39
OTU-4				-------	0.43
OTU-5					------



	OTU-1/2	OTU-3	OTU-4	OTU-5
OTU-1/2	-------	*0.255*	*0.325*	***0.22***
OTU-3		-------	0.28	0.39
OTU-4			-------	0.43
OTU-5				-------

Italics are average distances. Values in plain text are taken from original matrix.



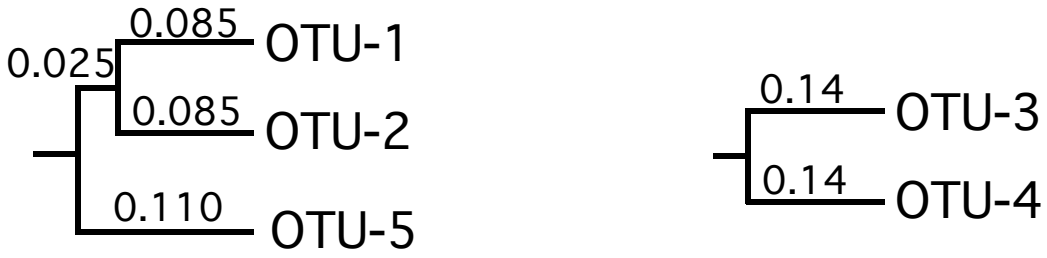
Now in generating the new matrix, we take the unweighted average of the distances (each pairwise distance among members of the composite OTU is weighted equally).

So here:

$D_{1/2/5,3} = [(0.255 * 2) + 0.39] / 3 = 0.300$

$D_{1/2/5,4} = [(0.325 * 2) + 0.43] / 3 = 0.360$

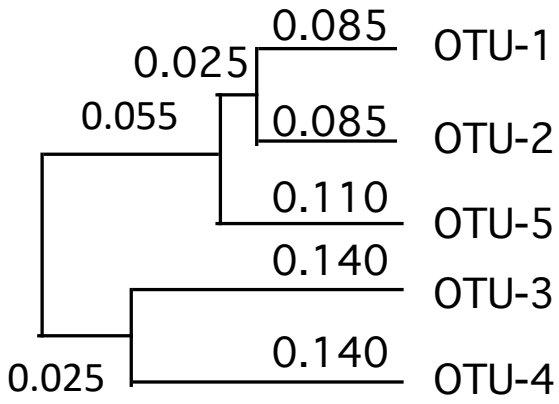
	OTU-1/2/5	OTU-3	OTU-4
OTU-1/2/5	-------	*0.300*	*0.360*
OTU-3		-------	**0.280**
OTU-4			-------



And $D_{1/2/5,3/4}$ = [(0.30 * 3) + (0.36 * 3)] / 6 = 0.330

	OTU-1/2/5	OTU-3/4
OTU-1/2/5	-------	0.330
OTU-3/4		-------

Thus, the distance from the root to any tip is 0.330 / 2 = 0.165.



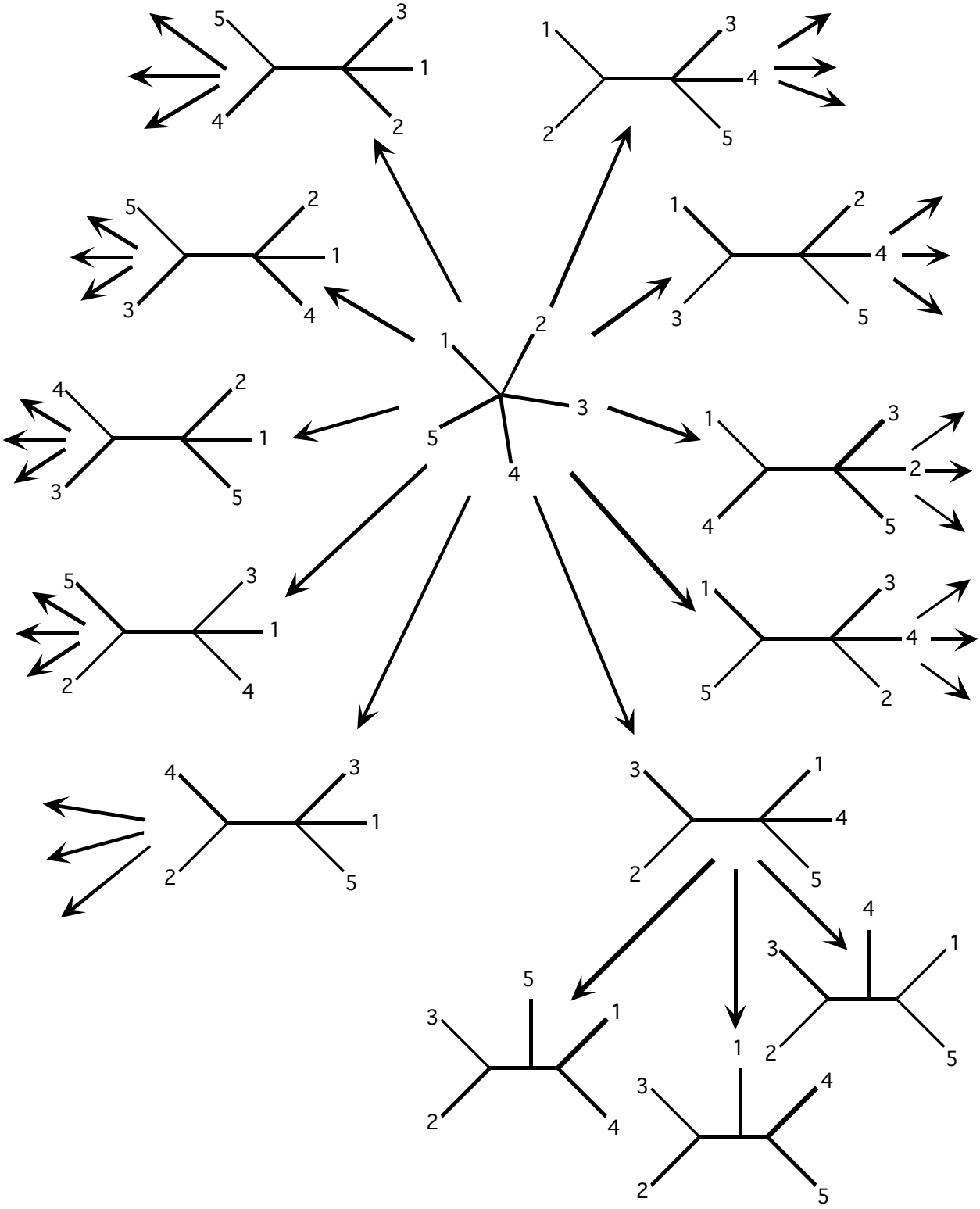
This is an **ultra-metric** tree. The distance to from the root to any tip is identical.

Remember that this clustering algorithm was not intended to estimate phylogeny, but to visualize the hierarchical pattern of similarity.

Nevertheless, a UPGMA tree will be a good estimate of the phylogeny if, and only if, the evolution of the group has been rather clock like (rates don't change along branches) and if distances are perfectly additive. Note that this produces rooted trees.

III. **Star Decomposition Methods**

Obviously, Star-decomposition methods start with a star tree, with all taxa emanating from the central node, and the star is usually decomposed following some optimality criterion.



The most common SD method is the Neighbor-Joining algorithm of Saitou and Nei (1987).

This method starts by converting the raw distance matrix to corrected distance matrix. We'll first go through the method, then we'll use the same matrix as we just used for UPGMA.

1) This is accomplished by first estimating the net divergence of each taxon from all others: where n is the number of nodes (equal to the number of taxa on the first pass).

$$r_{i=} \sum_{k=1}^{n} d_{i,k}$$

This essentially provides normalization for unequal rates of evolution.

2) The corrected matrix is given by:

$$M_{i,j} = d_{i,j} - [r_i / (n - 2)] - [r_j / (n - 2)]$$

3) The minimum $M_{i,j}$ (the shortest corrected distance) identifies the two taxa to be united to an ancestral node (u).

This new node u has three branches emanating from it: one to terminal i, one to terminal j, and one to the rest of the tree.

The lengths of these branches are calculated.

4) A new matrix is calculated by replacing taxa i and j with their ancestral node u. The distances from all remaining taxa and node u are calculated as follows:

$$d_{k,u} = (d_{i,k} + d_{j,k} - d_{i,j}) /2$$

5) If more than two nodes remain, reset $n = n - 1$ and return to step 1. If only two nodes remain, $v_{i,j} = d_{i,j}$.

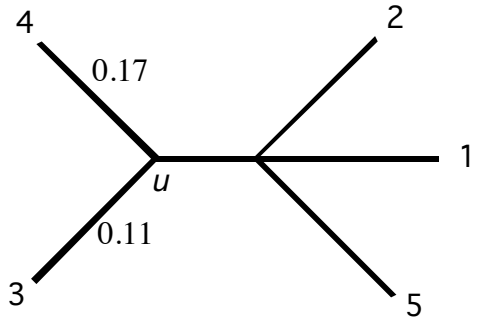
Here's a worked example, using the same data matrix we used in UPGMA demo.

	OTU-1	OTU-2	OTU-3	OTU-4	OTU-5	r_i	$r_i /(n\text{-}2)$
1	-------	0.17	0.21	0.31	0.23	0.92	0.307
2	-0.477	-------	0.30	0.34	0.21	1.02	0.340
3	-0.490	-0.433	-------	0.28	0.39	1.18	0.393
4	-0.450	-0.453	**-0.566**	-------	0.43	1.36	0.453
5	-0.497	-0.550	-0.533	-0.443	------	1.26	0.420

So, the star is decomposed by uniting taxa 3 & 4 and calculating their branch lengths:

$v_{3,u} = d_{3,4} / 2 + (r_3/3 - r_4/3) / 2 = 0.28 / 2 + (0.393 - 0.453) / 2 = 0.11$

$v_{4,u} = 0.28 - 0.11 = 0.17$



Now we set $n = 4$ and build a new matrix by generating distances from node u to OTU's 1, 2, and 5:

$d_{1,u} = (d_{1,3} + d_{1,4} - d_{3,4}) /2 = (0.21 + 0.31 - 0.28) / 2 = 0.12$

$d_{2,u} = (d_{2,3} + d_{2,4} - d_{3,4}) /2 = (0.30 + 0.34 - 0.28) / 2 = 0.18$

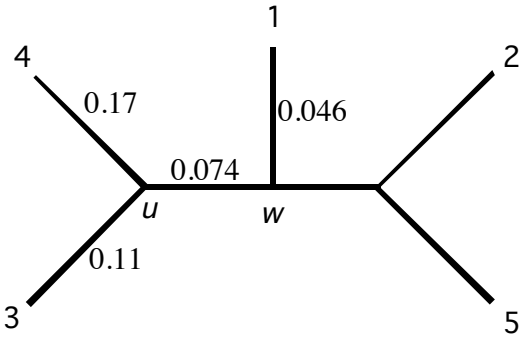
$d_{5,u} = (d_{5,3} + d_{5,4} - d_{3,4}) /2 = (0.39 + 0.43 - 0.28) / 2 = 0.27$

	OTU-1	OTU-2	OTU-5	Node *u*	r_i	$r_i /(n - 2)$
1	-------	0.17	0.23	0.12	0.52	0.260
2	-0.370	-------	0.21	0.18	0.56	0.280
5	-0.385	-0.425	-------	0.27	0.71	0.355
u	**-0.425**	-0.385	-0.370	-------	0.57	0.285

So, the star is decomposed further by uniting node *u* and OTU-1 with an ancestral node *w*, and the two branch lengths are calculated as follows:

$v_{1,w} = d_{1,u} / 2 + (r_1/2 - r_u/2) / 2 = 0.12 / 2 + (0.260 - 0.285) / 2 = 0.046$

$v_{u,w} = 0.12 - 0.046 = 0.074$



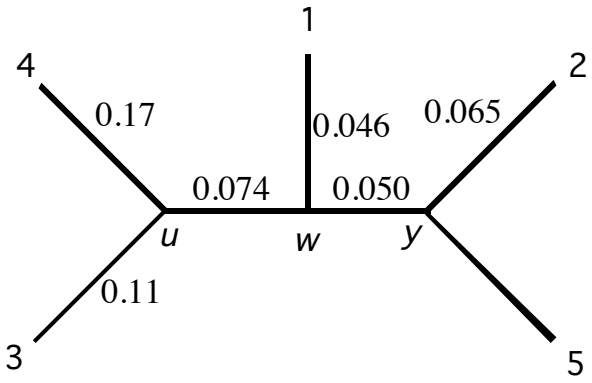
Now we set $n = 3$ and continue with a new matrix by calculating distances from node w to the remaining OTU's 2 and 5.

$d_{2,w} = (d_{1,2} + d_{u,2} - d_{1,u}) /2 = (0.17 + 0.18 - 0.12) / 2 = 0.115$
$d_{5,w} = (d_{1,5} + d_{u,5} - d_{1,u}) /2 = (0.23 + 0.28 - 0.12) / 2 = 0.195$

	OTU-2	OTU-5	Node *w*	r_i	$r_i /(n - 2)$
2	-------	0.21	0.115	0.325	0.325
5	-0.510	-------	0.195	0.395	0.395
w	**-0.520**	-0.510	-------	0.310	0.310

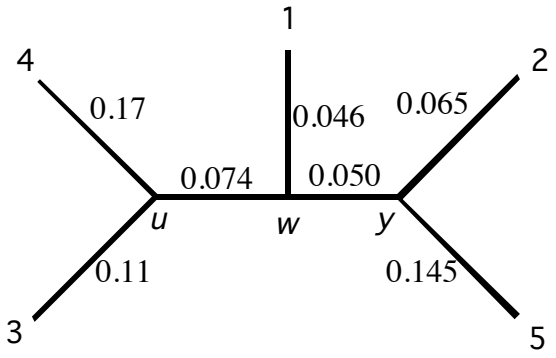
Now, node *y* unites node *w* with OTU-2, and again branch lengths are calculated:

$v_{y,2} = d_{2,w} / 2 + (r_2/1 - r_w/1) / 2 = 0.115 / 2 + (0.325 - 0.310) / 2 = 0.065$
$v_{y,w} = 0.115 - 0.065 = 0.050$



Finally, we calculate the distance from node y to OTU –5:

$d_{5,y} = (d_{w,5} + d_{2,5} - d_{2,w}) /2 = (0.195 + 0.21 - 0.115) / 2 = 0.145 = v_{5,y}$



Note that the number of calculations required decreases as we build the tree.

There are a couple important modifications to the neighbor-joining algorithm, BIONJ (incorporates variances and covariances of $d_{i,j}$) and weighbor, that use different weighting factors either in Step 3 or in Steps 2 and 3.

Neighbor joining is often interpreted as an approximation to the ME solution.

To me, this is not phenetic. It is not attempting to cluster based on overall similarity; it attempts to parse that into primitive similarity and derived similarity (through the normalization). It also explicitly treats the internal nodes as ancestral nodes.

In different implementations of NJ, ties are broken differently. Originally, they were broken in and arbitrary by constant method. Newer implementations break ties randomly.

Also note that the UPGMA tree and the NJ tree for this matrix are different.

IV. Stepwise Addition Methods

Other algorithms to build trees take the opposite approach. They add taxa sequentially rather than decomposing a star. Some older distance methods took this approach (e.g., the distance Wagner approach), but these are almost never used anymore (I published one in my first paper ever) and have been supplanted by NJ.

Nevertheless, Stepwise Addition is incredibly important in modern phylogenetics because of its role in searching through tree space under optimality based searching methods.

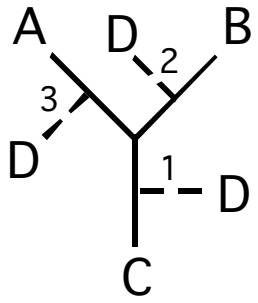
The basic idea is to start with three taxon tree, then compute its score under some criterion (usually ML or MP), and add a fourth in the manner that incurs the smallest decrease in optimality (e.g., smallest increase in length).

Rather than provide equations as I did for the previous two methods, I'll use a parsimony example that's easy to optimize (from Felsenstein, pp. 2 & 53).

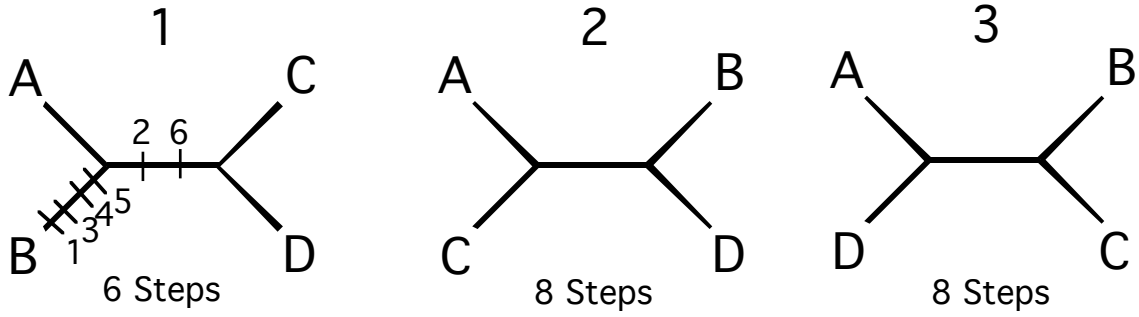
	Characters					
	1	**2**	**3**	**4**	**5**	**6**
A	1	0	0	1	1	0
B	0	0	1	0	0	0
C	1	1	0	1	1	1
D	1	1	0	1	1	1
E	0	0	1	1	1	0

First, we pick three taxa. We'll use the order the taxa appear in the matrix as our input order. This matters because the method is starting-point dependent.

So, for A, B, & C there is only a single unrooted tree, shown by the solid lines.



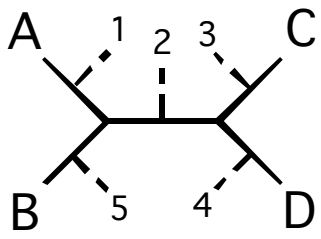
There are 3 possible places to add taxon D, indicated by numbers 1 – 3. The optimality score (tree length, in this example, using either the Fitch or Sankoff algorithm) is computed for each of these:



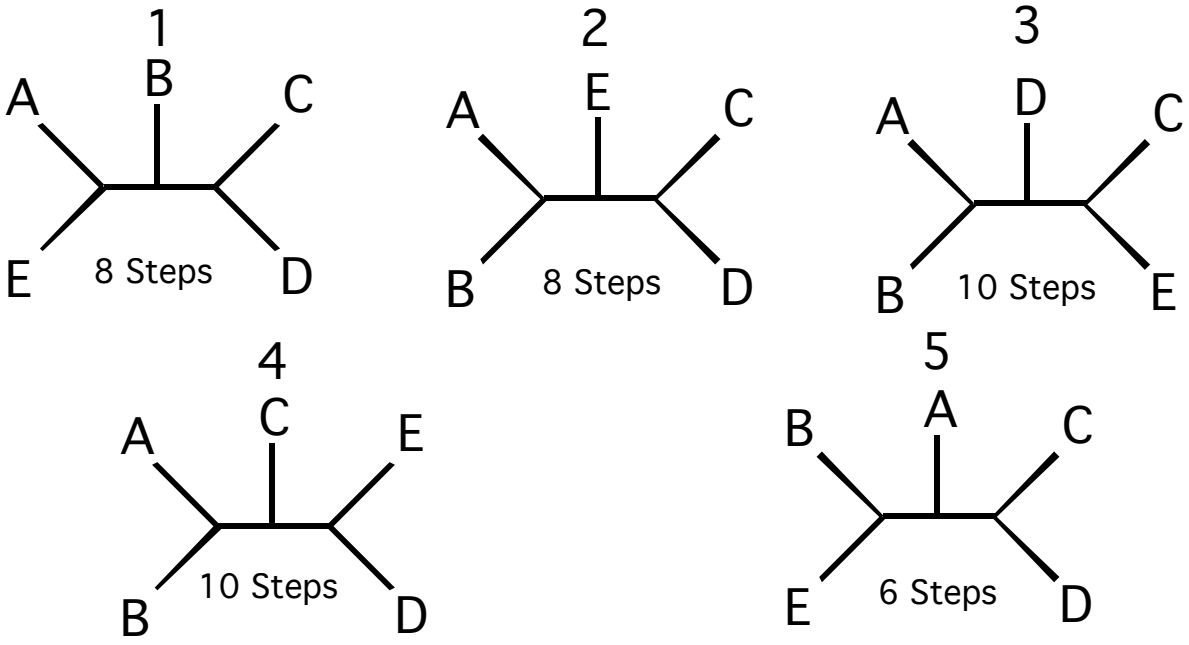
Option 1 is chosen because it has the shortest in length (i.e., is optimal).

Taxon E is then added to tree 1 and there are 5 possible placements for a new taxon (because there are 5 branches in the tree at this point).

These are labeled 1 – 5, and again, are indicated by the dashed lines:



Again, characters are optimized on each of these 5 possibilities:



Here, we've used stepwise addition build a tree(s) using parsimony as an optimality criterion. We could have used any criterion wish (so long as the data are amenable).

Now, if we had a 6th taxon to add, we would consider all 7 possible placements.

Thus, unlike in star decomposition, the number of calculations increases as we build a tree via stepwise addition.

Addition sequence is critical. The method is starting-point dependent, so different addition sequences can produce different stepwise addition trees.

Options:

As is – In the order that the taxa appear in the matrix

Shortest – One may check all triplets, chose that which has the shortest tree and, at each step, assess all candidates, and choose that which adds least length.

Random – One may use random addition sequences and replicate a number of times. For clean data, there should be only small differences in the stepwise addition tree, but for most real data there will be multiple stepwise addition trees. We'll make use of this in searching tree space.

V. Quartet Methods

We shouldn't leave this topic without at introducing a class of methods that operates on quartets of taxa.

An important (historically) quartets methods is called Quartet Puzzling (Strimmer and von Heasler, 1996: Mol. Biol. Evol., 13: 964-969.), and it's meant to provide an approximation to an ML tree.

This method breaks a data set into all possible quartets of taxa.

For our example with 5 taxa, there are 5 quartets: $\binom{5}{4} = \frac{5!}{(5-4)!(4!)}$

{1,2,3,4} {1,2,3,5} {1,2,4,5} {1,3,4,5} {2,3,4,5}

Each of the three resolutions for all quartets is evaluated and the ML resolution is stored.

((1,2),3,4) ((1,2),3,5)) ((1,2),4,5)) ((1,3),4,5)) ((2,3),4,5))

((1,3),2,4) ((1,3),2,5)) ((1,4),2,5)) ((1,4),3,5)) ((2,4),3,5))

((1,4),2,3) ((1,5),2,3)) ((1,5),2,4)) ((1,5),3,4)) ((2,5),3,4))

Then the tree is assembled that maximizes the compatibility among the ML resolutions of all possible quartets:

((1,3), 5, (2,4))

The method is becoming popular as an ML approximation for large data sets and has become a focus of species-tree estimation (e.g., SVD Quartets).