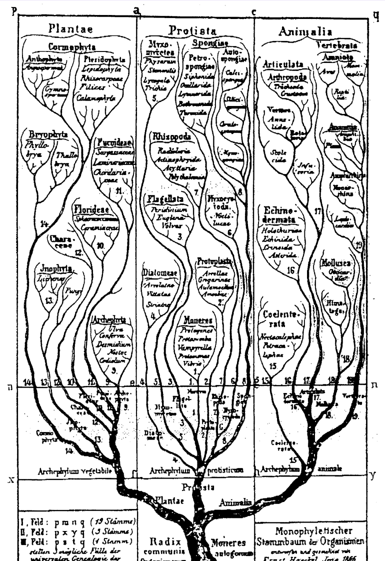
Lecture: Some Topics in Phylogenetics

Lecturer: Dr. Stephan Steigele

Bioinf, University of Leipzig

Leipzig WS07/08

Phylogeny


Plantae
Protista
Animalia
Cormophyta
Spongiae
Vertebrata
Articulata
Arthropoda
Bryophyta
Fucoideae
Rhizopoda
Florideae
Flagellata
Echinodermata
Mollusca
Dialomeae
Protoplasta
Coelenterata
Moneres
Archephylum vegetabile
Archephylum protisticum
Archephylum animale
Radix communis
Monophyletischer Stammbaum der Organismen

Part I

Introduction to Phylogenetics

Main sources for this lecture:

Some parts of this lecture were simply taken from scripts developed at University Tuebingen from **Prof. Daniel Huson** and in part by **Dr. Kay Nieselt** between 2002 and 2005.
More important: All parts of the lecture are highly inspired by the following book:

- **Jospeh Felsenstein, Inferring Phylogenetics, Sinauer Associates, 2004**

Reading of the book is highly recommended ..

Additional Material

More sources for this lecture:

- R. Durbin, S. Eddy, A. Krogh & G. Mitchison, Biological sequence analysis, Cambridge, 1998
- J. Setubal & J. Meidanis, Introduction to computational molecular biology, 1997.
- D.W. Mount. Bioinformatics: Sequences and Genome analysis, 2001.
- D.L. Swofford, G.J. Olsen, P.J.Waddell & D.M. Hillis, Phylogenetic Inference, in: D.M. Hillis (ed.), Molecular Systematics, 2 ed., Sunderland Mass., 1996.

Phylogenetic analysis

Given a collection of extant species. The goal of phylogenetic analysis is to determine and describe the **evolutionary relationship** between the species. In particular, this involves determining the order of speciation events and their approximate timing.
It is generally assumed that **speciation** is a branching process: a population of organisms becomes separated into two sub-populations. Over time, these evolve into separate species that do not cross-breed.

Phylogenetic analysis

Because of this assumption, a **tree** is often used to represent a proposed phylogeny for a set of species, showing how the species evolved from a common ancestor. We will study this in detail.
However, there are a number of situations in which a tree is less appropriate than a phylogenetic **network**. We will study this in a later Chapter.

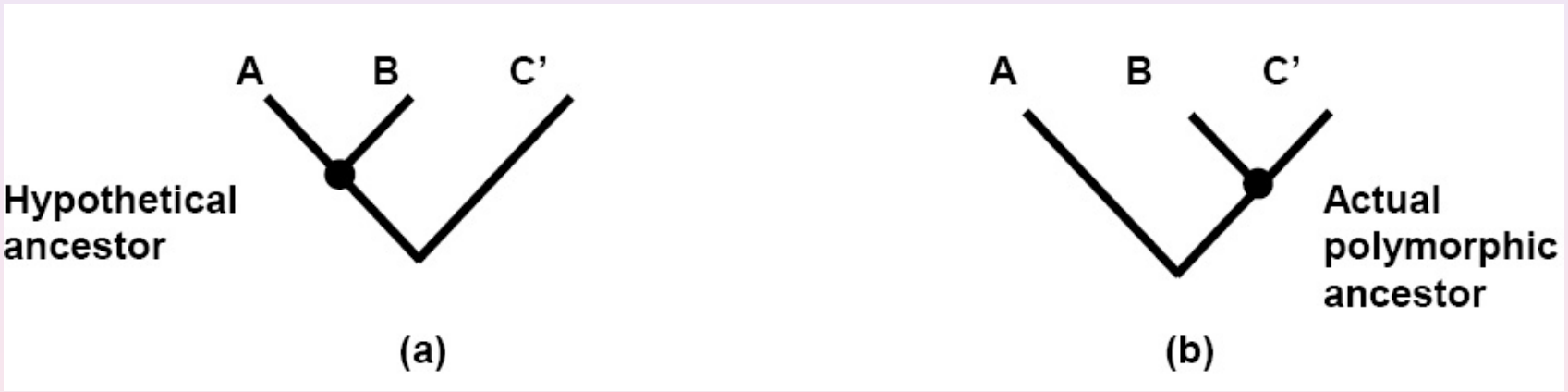
Definitions of Molecular Phylogenetics

- Molecular phylogenetics refers to any method of inferring evolutionary relationships from similarities or differences in molecular structure.
- **However:** Molecular characters suffer from problems that also afflict morphological characters.
- For example, neither molecules nor morphology may be able to resolve the phylogeny of evolution that was both ancient and rapid, as in the Cambrian Explosion.

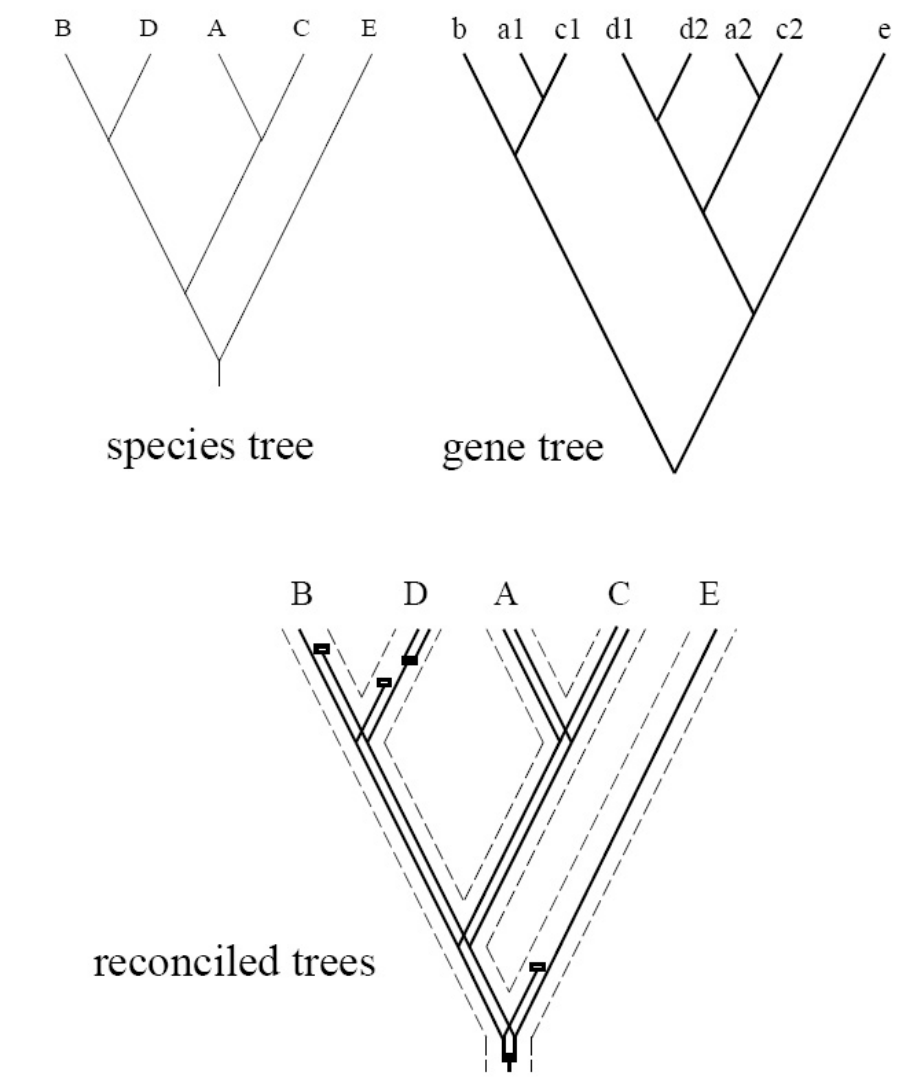
The **Character** of evolution

- One problem shared by molecular and morphological characters is **homoplasy** (nonhomologous characters appearing to be similar in different taxa).
- Another problems shared by molecular and morphological phylogenetics arise from **polymorphism** (homologous characters appearing differently in the same species). Because of polymorphism, the time of divergence may appear to be earlier than it was.
- Polymorphism can also result in the incorrect phylogenetic sequence.
- Similar problems result from different copies of **duplicated genes**

Example: Polymorphism



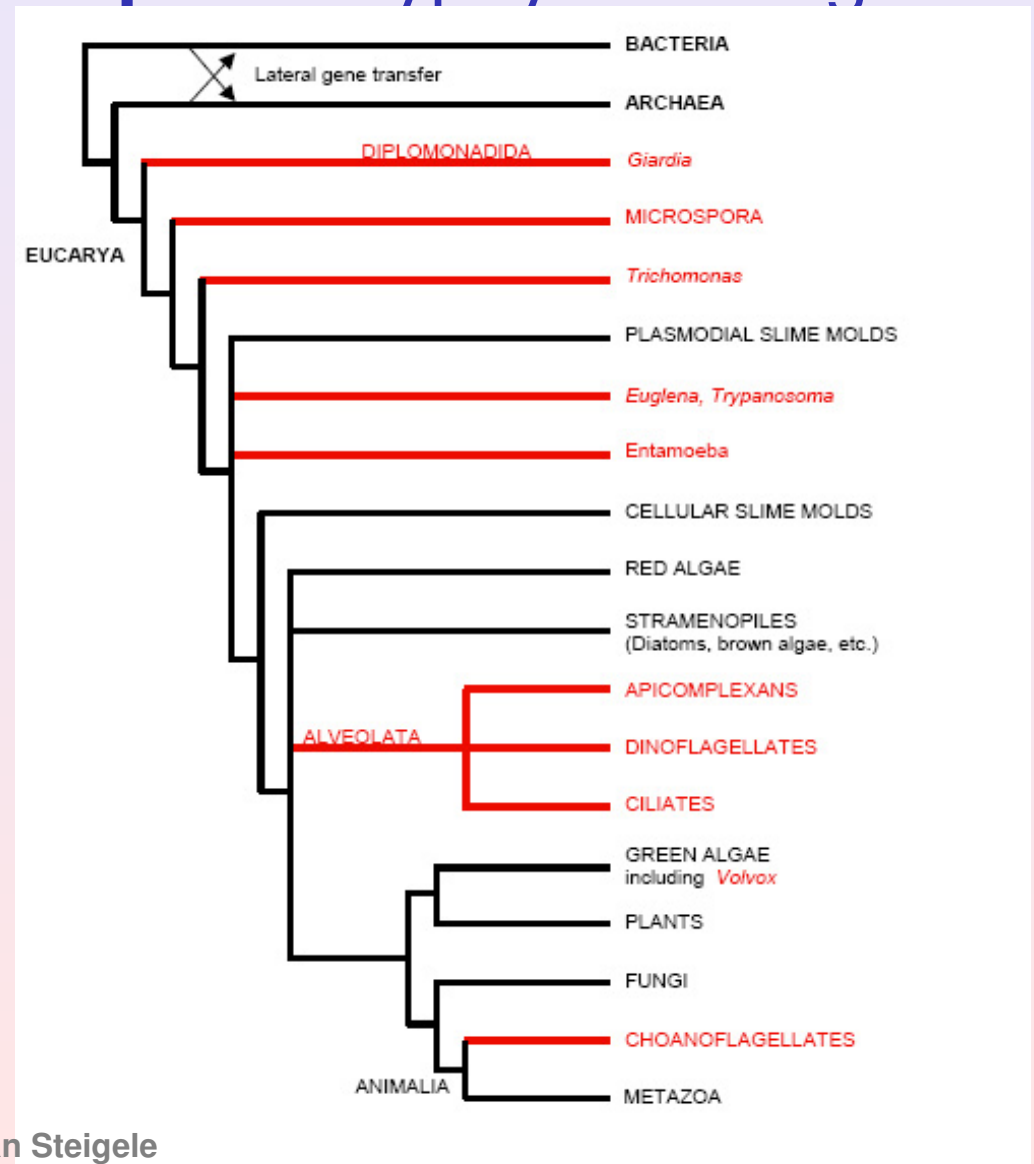
Example: Gene Tree Vs Species Tree



Some Problems and new insights

- Another problem with molecular phylogenetics is **long-branch attraction**: the tendency of fast-evolving molecules to appear more closely related than they actually are.
- Molecular phylogenetics has gained wide acceptance in spite of these and other problems because it provides a large amount of evidence that is independent of morphology, as well as other advantages.
- Several kinds of **experiments support the validity of molecular phylogenetics**
- Because of long-branch attraction, differences in sequence alignment, limitations in the size of study groups, and different methods of tree reconstruction, conflicting molecular phylogenies have been proposed.

Example: Polyphyletic Origin of Protozoans



Molecular Characters

- Molecular characters can be of two types: discrete (**qualitative**) differences in molecular sequence and continuous (**quantitative**) distance between molecules.
- The first step in molecular phylogenetics is to select a suitable molecule that is **homologous** in all the taxa to be included in the phylogeny.
- Many molecular characters are much less susceptible to homoplasy and longbranch attraction than are nucleic-acid sequences.
- These characters include amino-acid sequences, the positions of short and long interspersed elements, and Hox genes.

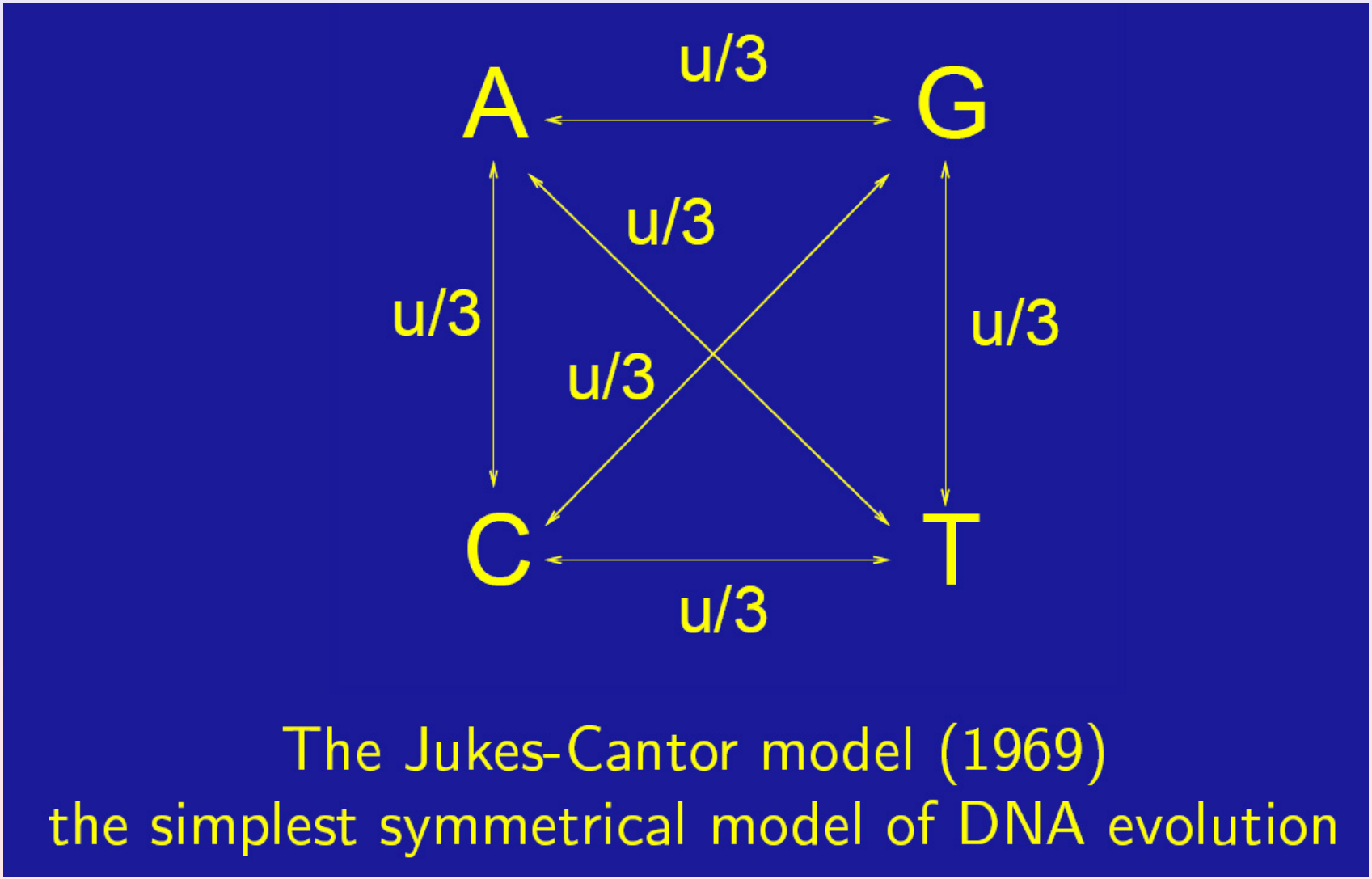
Molecular Characters

- Elongation factors, actin, and tubulins are among the widely used proteins.
- The positions of short and long interspersed elements (SINEs and LINEs) are another increasingly common source of discrete characters.
- Hox genes have also been used to infer phylogenetic relationships.
- The most commonly used molecular data for higher taxonomic levels are base sequences from genes that encode ribosomal RNA, especially 18S rDNA.

Modeling Evolution

- **Assumptions** may be needed about the **probabilities** of different molecular changes.
- Molecular relationships are represented as trees constructed of branches with nodes at both ends of each branch.
- Inferring (reconstructing) a phylogeny consists of creating or selecting **one tree out of perhaps millions** of possible ones.

Jukes-Cantor Model

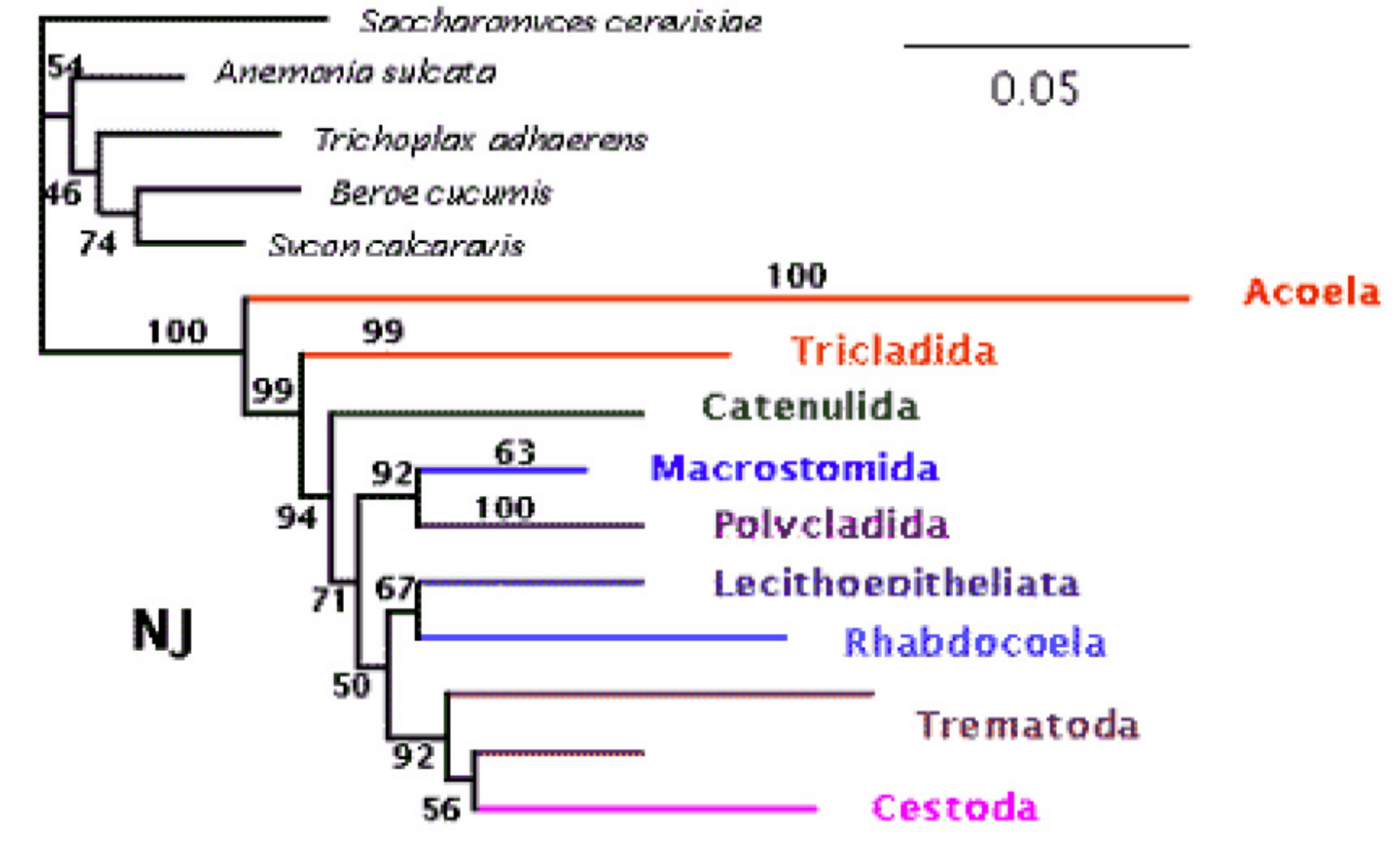


The Jukes-Cantor model (1969)
the simplest symmetrical model of DNA evolution

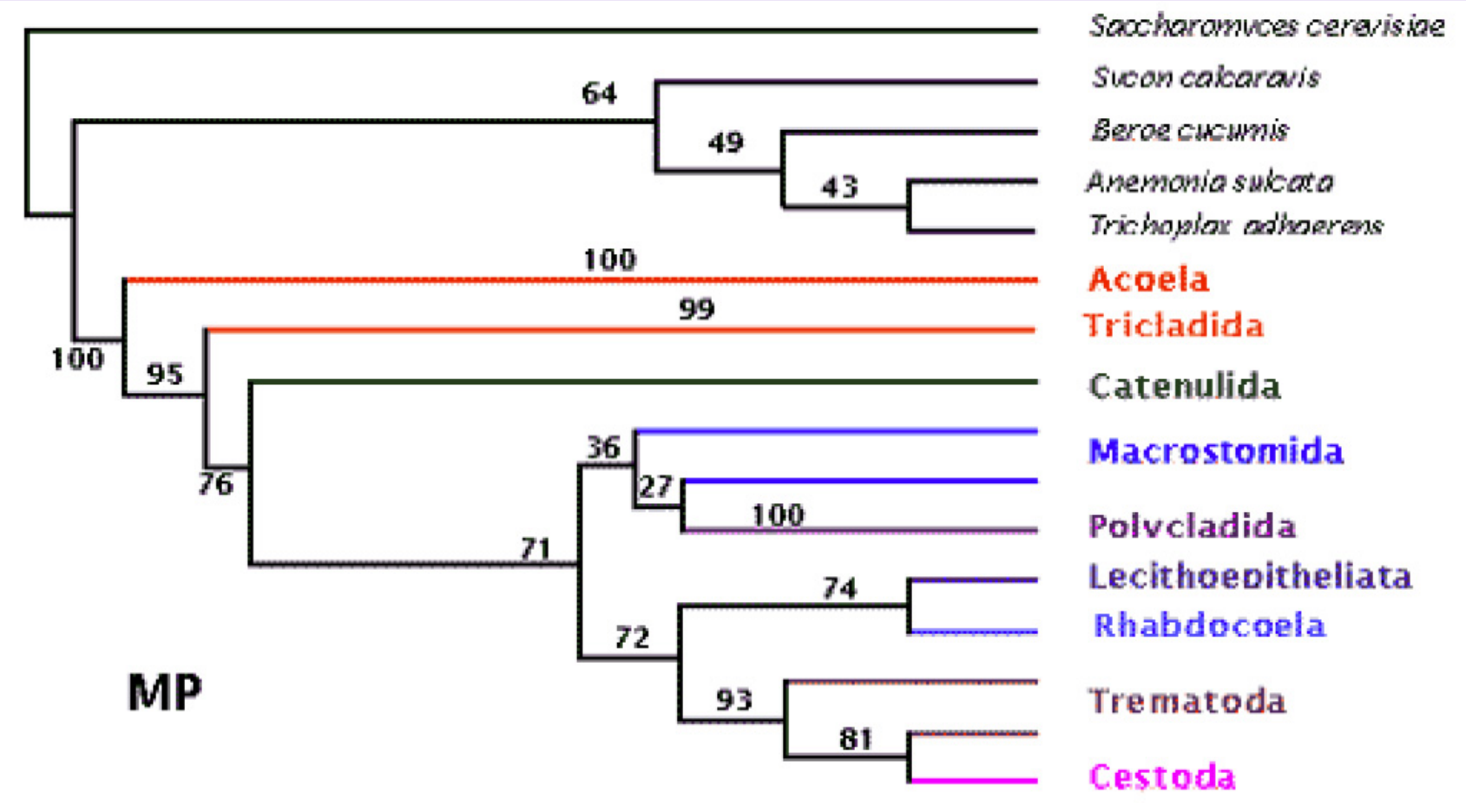
Many Ways to do it !!

- The neighbor-joining method (**distance based**) is an algorithm that generates one tree with the shortest total branch length.
- The maximum parsimony method (**character based**) selects the cladogram with the minimum number of changes in character state.
- The maximum likelihood method (**character based**) begins with an explicit model of evolution and possible trees, then it attempts to find the tree that is most likely with the given data.
- With more than a few taxa, any method requires a computer.
- Phylogenies reconstructed by different methods are generally **similar to each other**

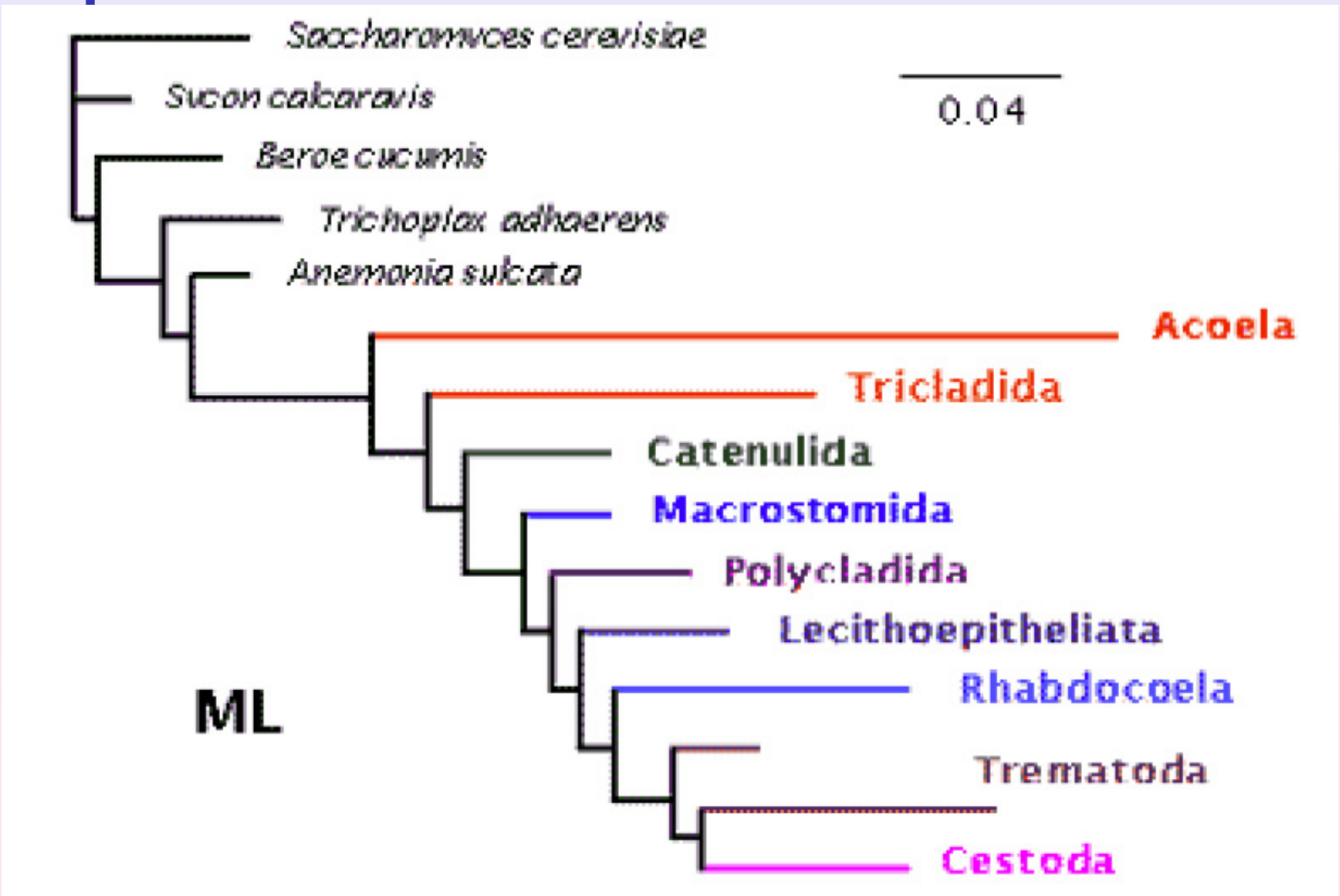
Example: Neighbor-Joining Tree



Example: Maximum Parsimony Tree



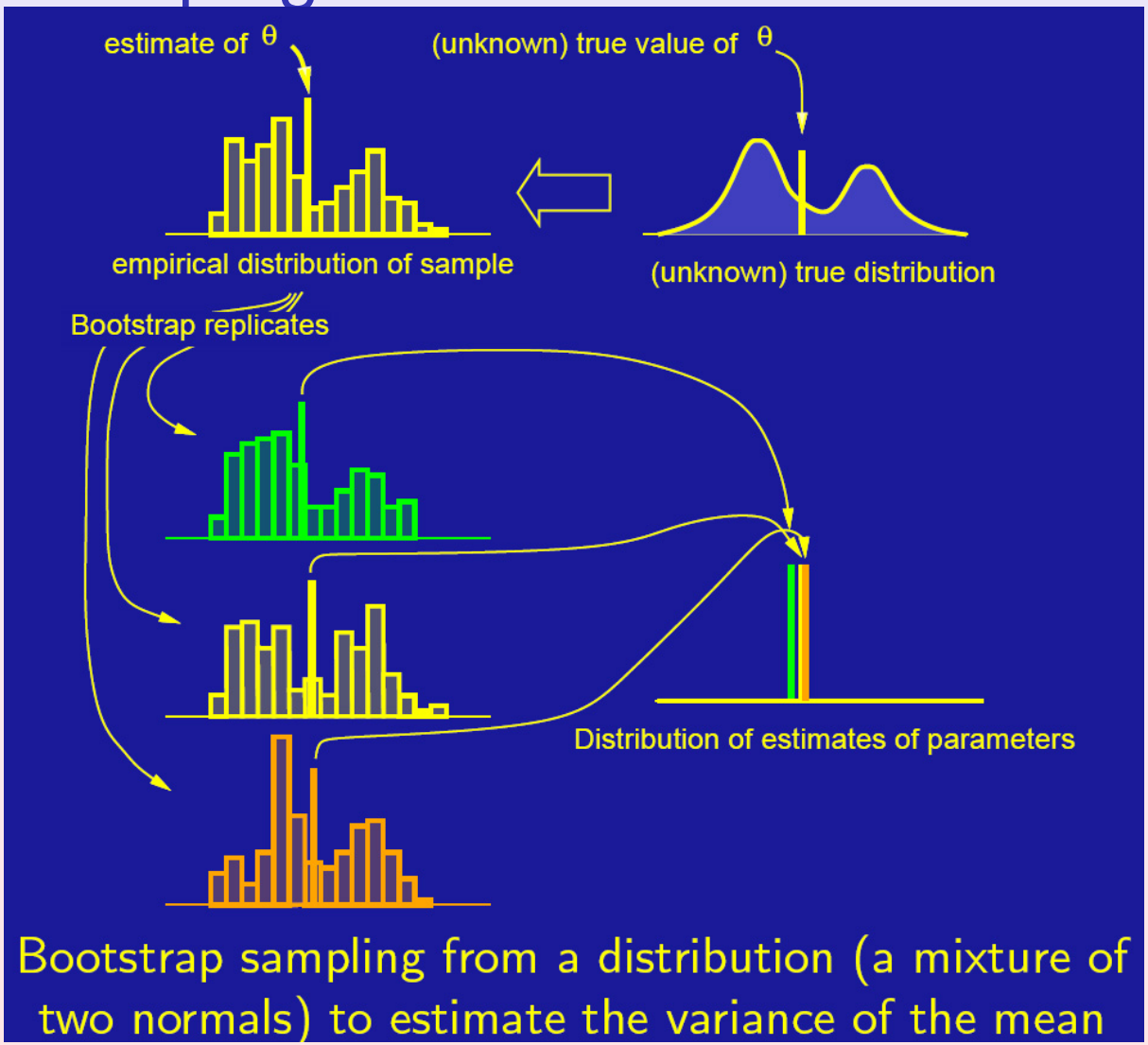
Example: Maximum Likelihood Tree



Getting Confidence

- Confidence in an internal branch can be tested by **bootstrapping**
- A branch with low bootstrap support may be collapsed.
- Molecular and morphological data can be combined to create a total-evidence tree.

Bootstraping



Bootstrap sampling from a distribution (a mixture of two normals) to estimate the variance of the mean

Collapsed Tree

A **consensus tree** can be created by collapsing branches that are not supported in all trees created by different methods of analysis. A consensus tree can also be produced by comparing molecular and morphological trees.

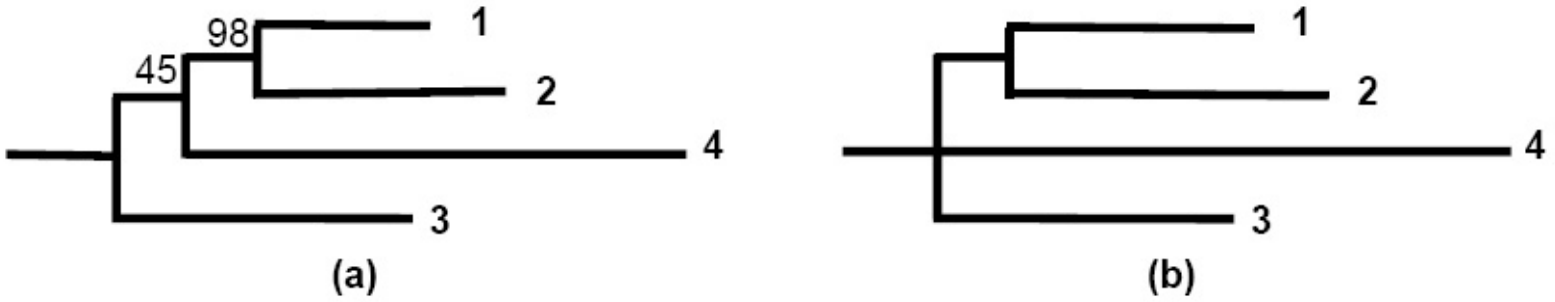


Figure 13. Collapsing branches that are poorly supported. (a) The original tree with one branch having a bootstrap value of only 45%. (b) After collapsing the poorly supported branch there is an unresolved trichotomy for branches (1 + 2), 3, and 4.

Consistency

- As techniques have improved and more molecules from more species have been sequenced, many of the past conflicts have been resolved.
- Testing the **Validity of Traditional Morphological Characters** by Seeing Whether They Are **Consistent** With Molecular Trees
- To be consistent with a given phylogenetic tree, a character must map onto the tree with few changes in character state.

Examples for **Consistency**

- For example, **bilateral symmetry** is consistent with the traditional morphology based cladogram for the Big Nine phyla (those with more than 5,000 named species), since it requires only one change in character state.

Example: Bilateral Symmetry

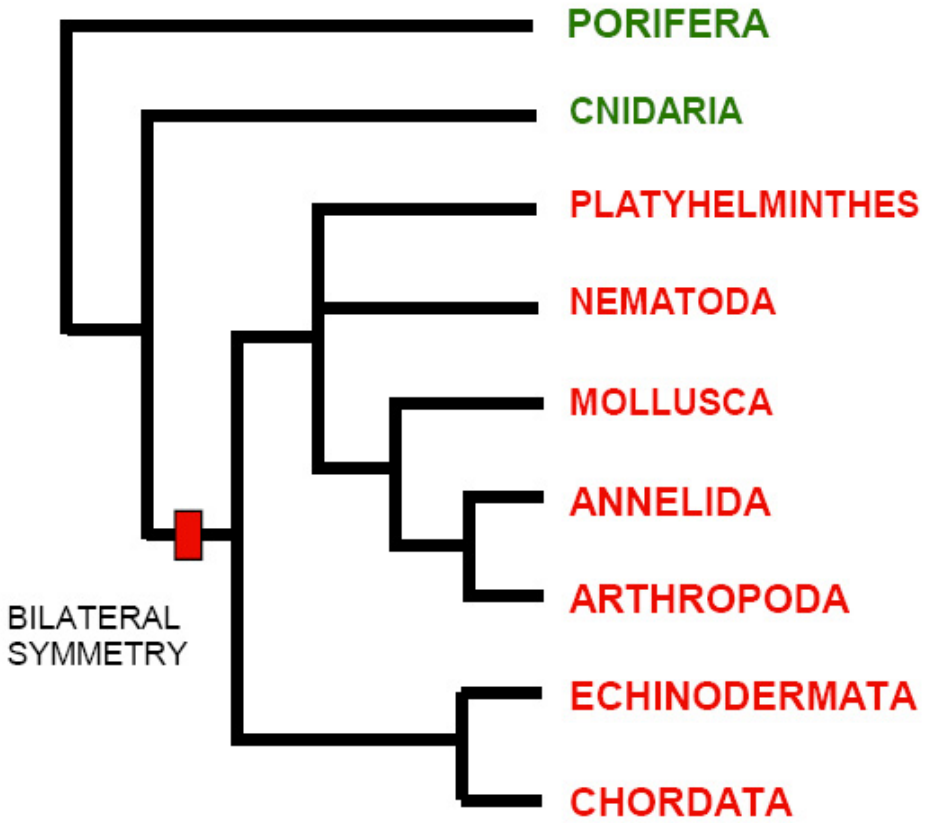


Figure 14. The traditional morphology-based phylogeny of the major animal phyla, based on the "hypothetical diagram" by Hyman (1940, vol. 1, p. 38). Bilateral symmetry is consistent with this phylogeny because it requires only one change in character state to account for its distribution among the phyla.

Examples for **Consistency**

- **Segmentation**, however, is less consistent with traditional morphology based phylogenetic trees, because it requires at least two changes in character state: one for annelids and arthropods and one for chordates.

Example: Segmentation

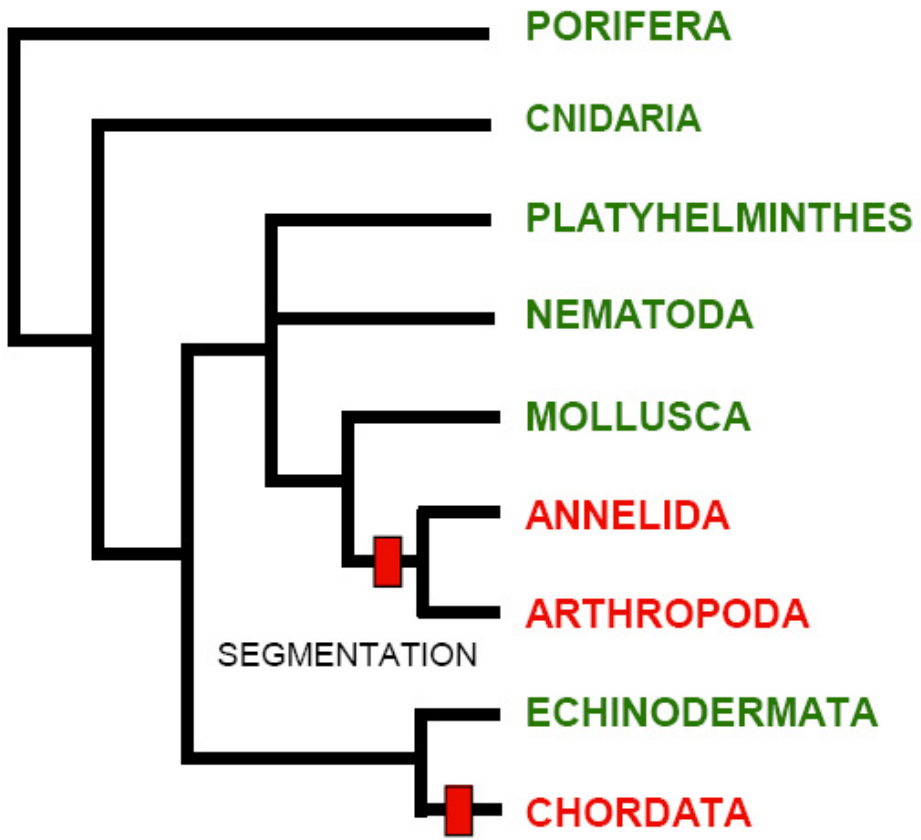


Figure 15. Traditional phylogeny of the Big Nine showing that segmentation is less consistent, because it requires at least two changes in character state to account for its distribution.

Examples for **Consistency**

- Lack of consistency implies that either the character is not **synapomorphic** (homologous), or the phylogenetic tree is incorrect.

Combining Evidence

- A character that is consistent with both a morphological and a molecular phylogeny is more likely to be **phylogenetically informative**
- Morphological characters have not led to a consensus phylogeny.
- The following morphological characters traditionally used in phylogenetics are also consistent with the widely accepted molecular phylogenetic tree of the Big Nine Phyla: bilateral symmetry and triploblasty, deuterostomy, and spiral cleavage pattern.

Example: Morpholocial Markers that fit Molecular Analysis

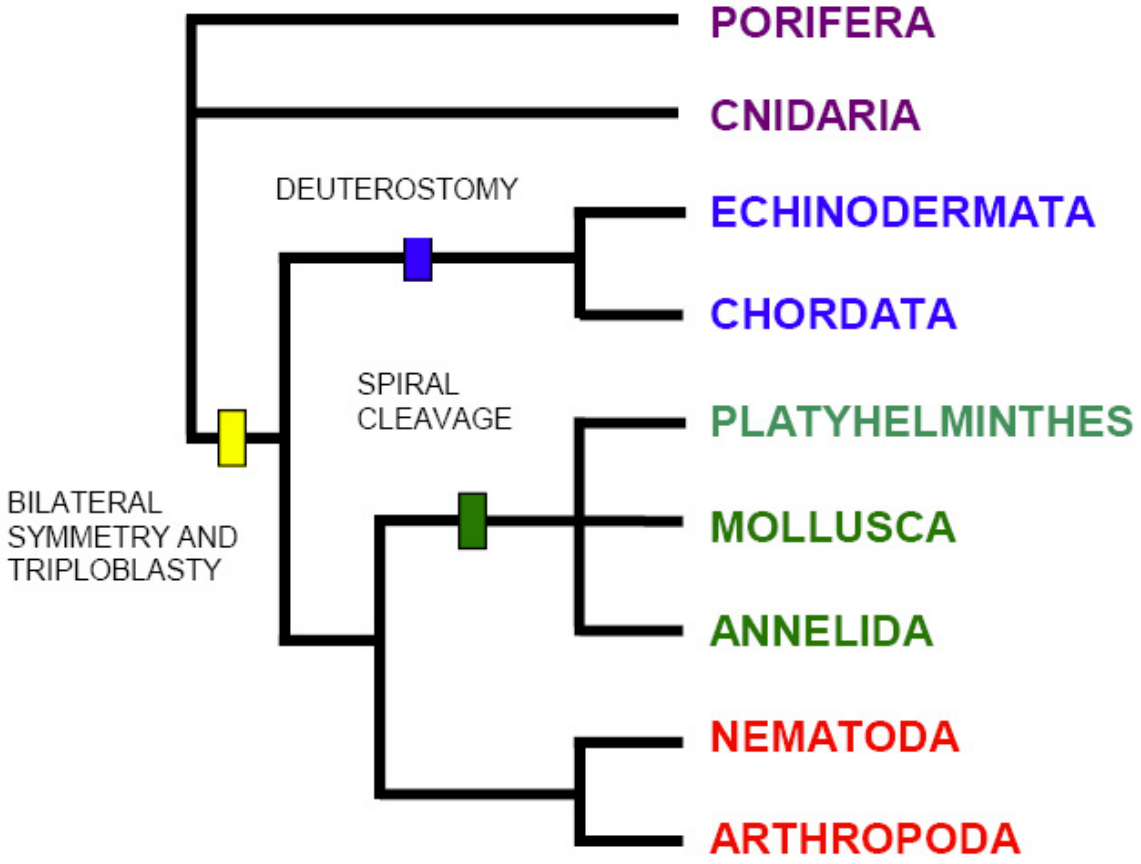


Figure 16. Bilateral symmetry and triploblasty, deuterostomy, and spiral cleavage pattern are

Combining Evidence

- **Bilateral symmetry** and **triploblasty** are also consistent with a molecular phylogenetic tree that includes all animal phyla.
- **Deuterostomy** in Echinodermata, Hemichordata, and Chordata is also consistent in a molecular phylogenetic tree of all animal phyla.
- The **spiral-cleavage** pattern is somewhat consistent with a molecular phylogenetic tree that includes all animal phyla.

How Much Information??

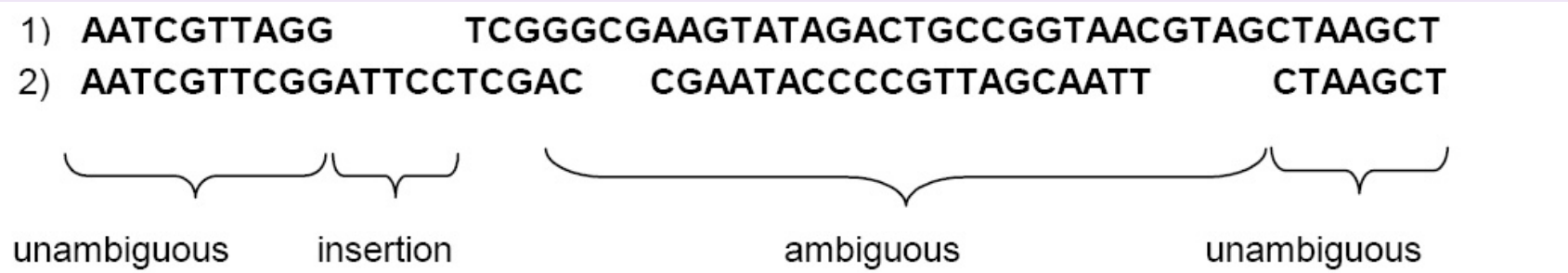


Figure 2. Homologous sequences of DNA bases from two taxa (1 and 2). An insertion, deletions, and a large number of mutations in the middle portion of the sequence make alignment ambiguous for that region.

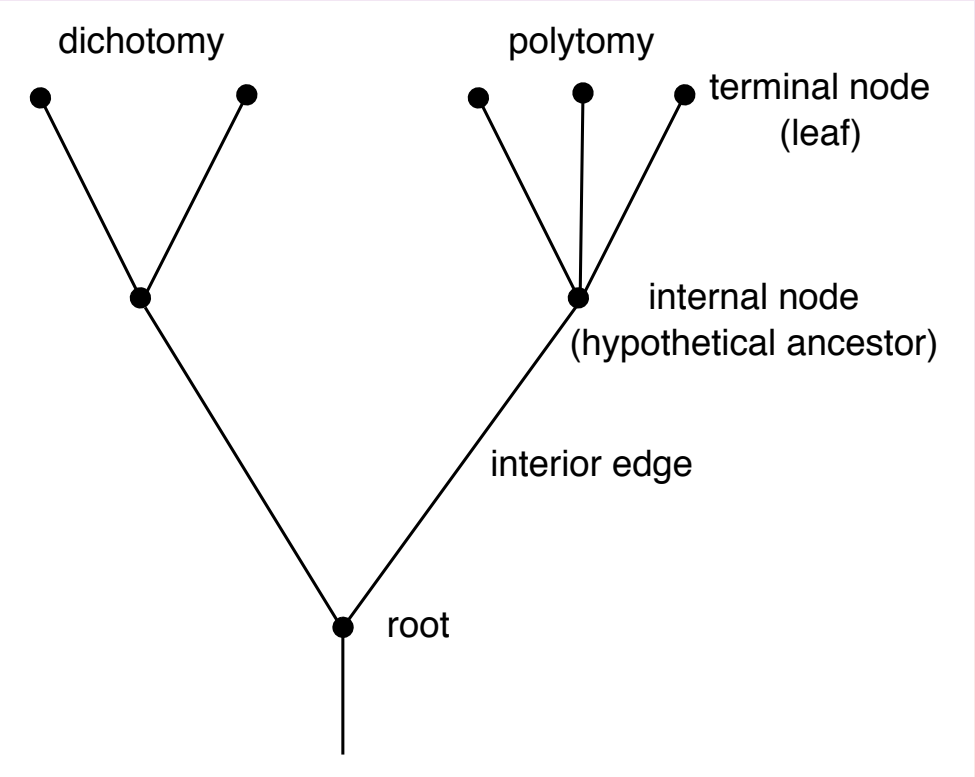
Informative Sites in Molecular Data

1)	T	T	C	G	A	C	C	G	T
2)	C	T	T	A	A	C	T	G	T
3)	C	T	A	T	G	C	T	G	G
4)	C	T	G	T	G	C	C	G	G
					x		y		z

Figure 7. Aligned homologous DNA sequences from four taxa. Informative sites are indicated by letters x, y, and z. Only positions with two or more different bases, at least two of which occur in more than one taxon, are informative.

A small tree nursery

A tree consists of **nodes** connected by **branches** (also called **edges**). Terminal nodes (also called **leaves**, OTUs) represent sequences or species for which we have data. Internal nodes represent hypothetical ancestors. If an internal node has three branches, it has



Phylogenetic trees

In the following, we will use $X = \{x_1, x_2, \ldots, x_n\}$ to denote a set of **taxa**, where a **taxon** x_i is simply a representative of a group of individuals defined in some way.
A **phylogenetic tree** (on X) is a system $T = (V, E, \lambda)$ consisting of a connected graph (V, E) without cycles, together with a **labeling** λ of the leaves by elements of X, such that:

1. every leaf is labeled by exactly one taxon, and
2. every taxon appears exactly once as a label.

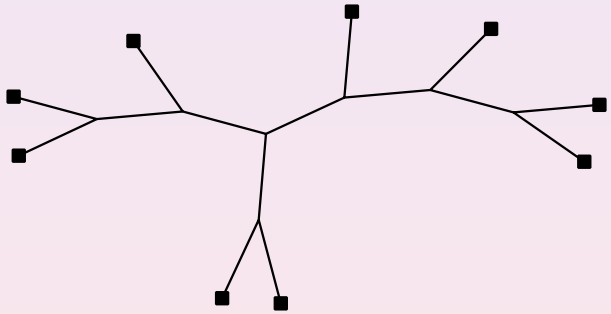
Phylogenetic trees

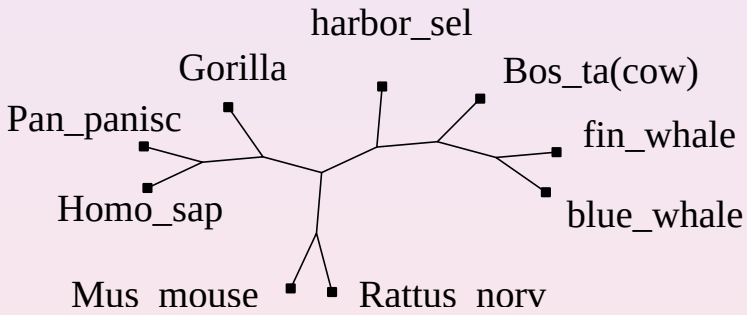
- Further, we require that either all internal nodes have degree ≥ 3, in which case T is called **unrooted**,
- or there exists precisely one internal **root** node ρ of degree 2, and T is called **rooted**.
- A phylogenetic tree T is called **binary**, if every internal node v has degree 3, except ρ, if T is rooted.
- An X**-tree** is obtained by relaxing the definition to allow labels on internal nodes, or nodes to carry multiple labels.
- Occasionally we will allow an **arbitrary** node to be the root.

Unrooted trees

An unrooted phylogenetic tree is obtained by placing a set of taxa on the leaves of a tree:

Pan_panisc
Gorilla
Homo_sap
Mus_mouse
Rattus_norv
harbor_sel
Bos_ta(cow)
fin_whale
blue_whale



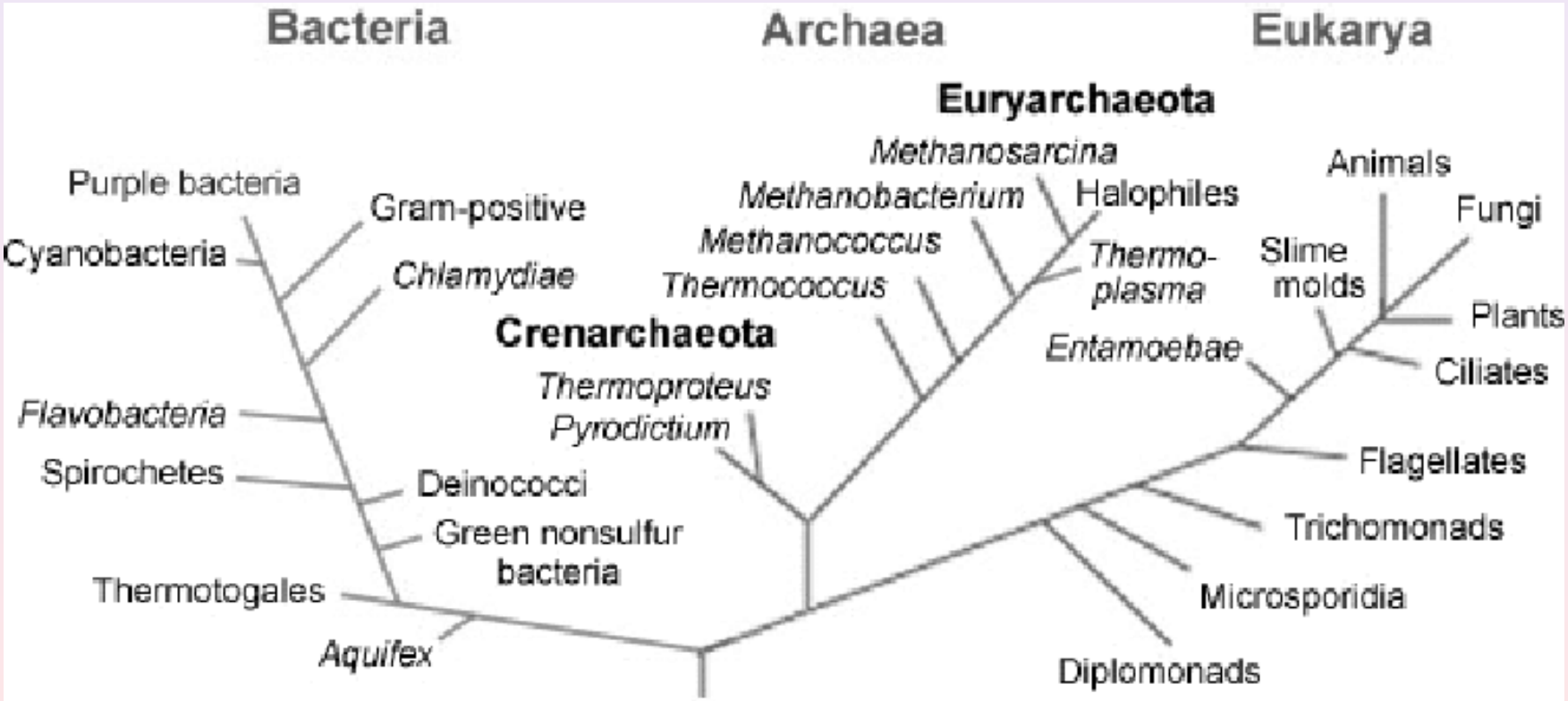


Taxa X $+$ tree $\Rightarrow$ phylogenetic tree T on X

Unrooted trees are often displayed using this type of **circular** layout.

Rooted trees

A rooted tree is usually drawn with the root placed at the bottom, top or left of the figure:



Modern version of the tree of life.

Edge lengths

Consider a phylogenetic tree T on X. The **topology** of the tree describes the putative order of speciation events that gave rise to the extant taxa.
Additionally, we can assign **lengths** to the edges of the tree. Ideally, these lengths should represent the amount of time that lies between two speciation events. However, **in practice the edge lengths usually represent quantities obtained by some given computation and only correspond very indirectly to time**.

Edge lengths

In the following, we will use $\omega : E \to \mathbb{R}_{\geq 0}$ to specify edge lengths and will use $T = (V, E, \lambda, \omega)$ to denote a phylogenetic tree with edge lengths.

The number of edges and nodes of an unrooted phylogenetic tree

Let T be an unrooted phylogenetic tree on n taxa, i.e., with n leaves. How many nodes and edges does T have? Let us assume that T is binary. Any non-binary tree on n taxa will have less nodes and edges.

Lemma

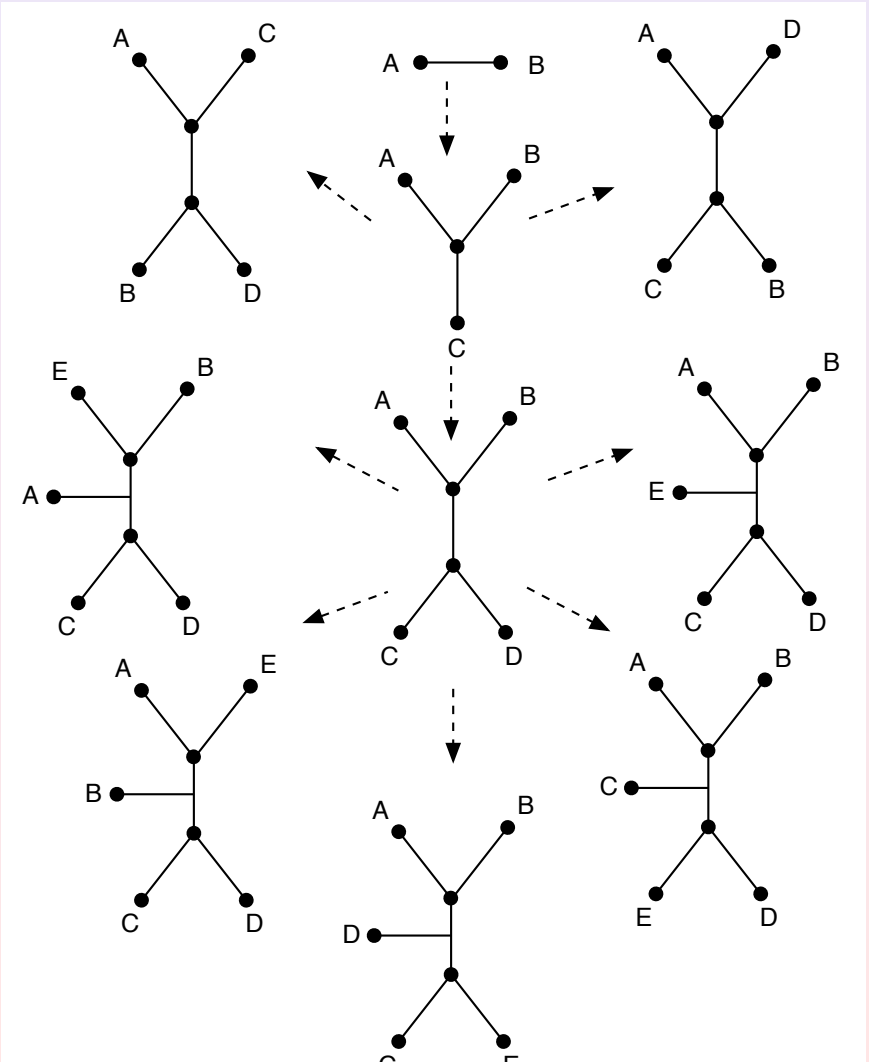
A binary phylogenetic tree T on n taxa has $2n-2$ nodes, $2n-3$ edges and $n-3$ interior edges for all $n \geq 3$.

The number of edges and nodes of an unrooted phylogenetic tree

Proof.

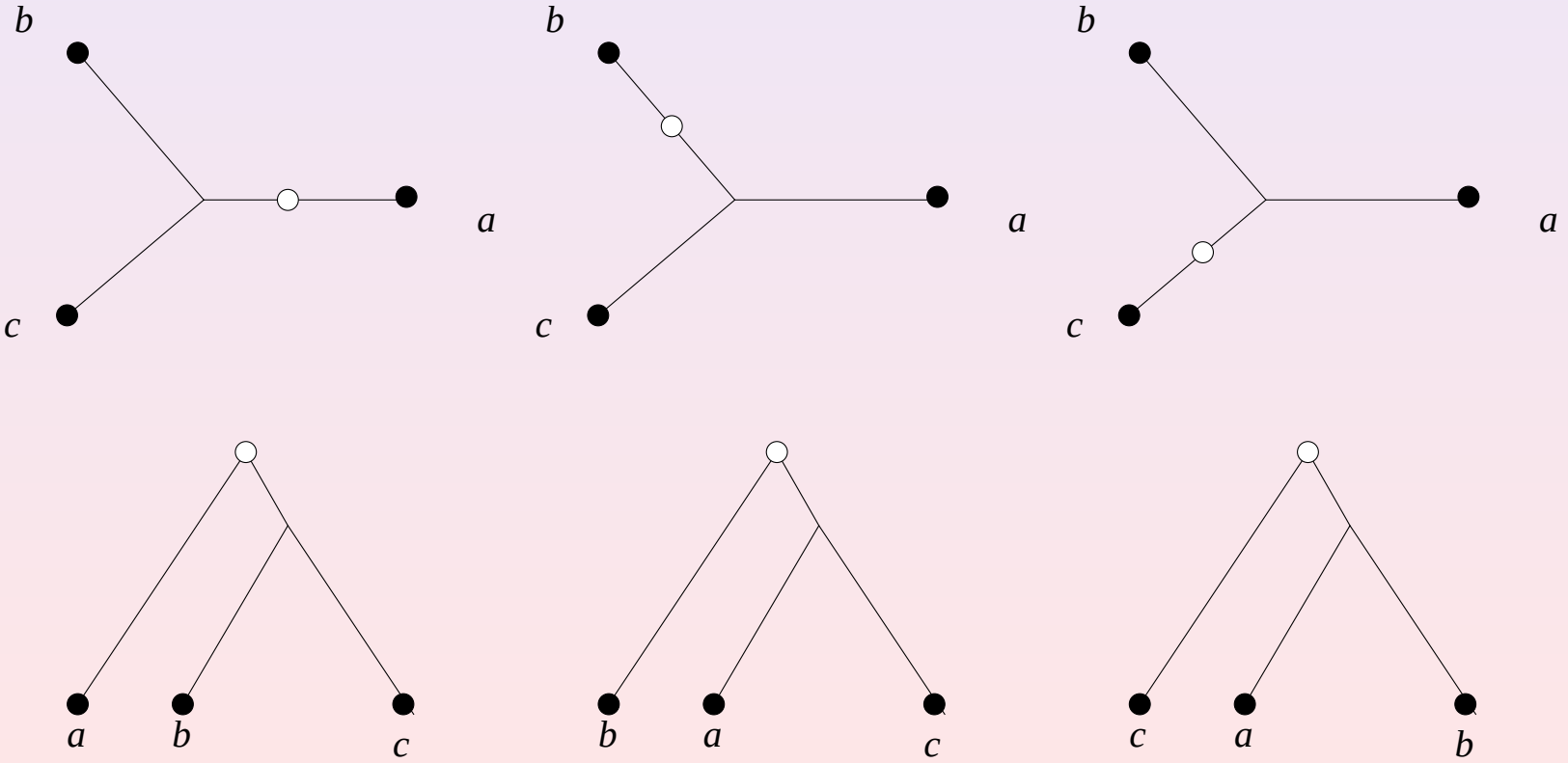
by induction. For $n = 3$ there is exactly one tree: it has 3 outer edges and zero interior edges. Also it has 3 leaves and one interior node.
We assume that the lemma has been proven for n. Any tree T' with $n+1$ leaves can be obtained from some tree T with n leaves by inserting a new node v into some edge e of T and connecting v to a new leaf w. This increases both the number of nodes and the number of edges by 2. Thus T' has $2n-3+2 = 2n-1 = 2(n+1)-3$ edges. The number of interior edges is $(2n-3)-n = n-3$. □

The number of edges and nodes of an unrooted phylogenetic tree



The number of phylogenetic trees

An unrooted tree T with n leaves has $2n-2$ nodes and $2n-3$ edges. A root can be added in any of the $2n-3$ edges, thus producing $2n-3$ different rooted trees from T:



Number of phylogenetic trees

For $n = 3$ there are three ways of adding a root. Similarly, there are 3 different ways of adding an extra edge with a new leaf to obtain an unrooted tree on 4 leaves. This new tree has $(2n-3) = 5$ edges and there are 5 ways to obtain a new tree with 5 leaves etc.
Continuing this, we see that there are

$$U(n) = (2n-5)!! := 3 \cdot 5 \cdot 7 \cdot \cdots \cdot (2n-5)$$

unrooted trees on n leaves. Similarly, there are

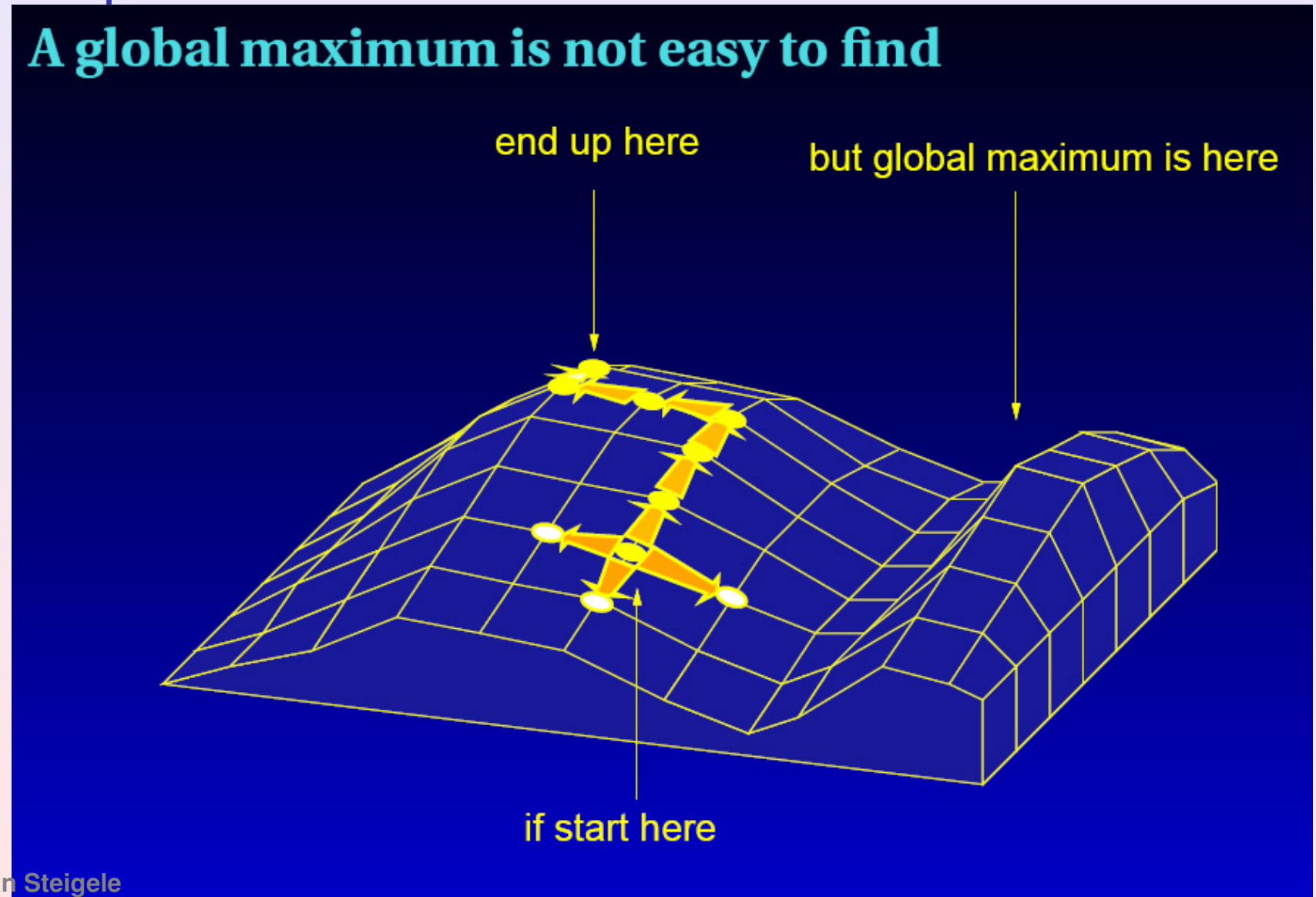
$$R(n) = (2n-3)!! = U(n) \cdot (2n-3) = 3 \cdot 5 \cdot \cdots \cdot (2n-3)$$

rooted trees.
These numbers grows very rapidly with n, for example, $U(10) \approx 2$ million and $U(20) \approx 2.2 \times 10^{20}$.

Counting of Trees

species	number of trees
1	1
2	1
3	3
4	15
5	105
6	945
7	10,395
8	135,135
9	2,027,025
10	34,459,425
11	654,729,075
12	13,749,310,575
13	316,234,143,225
14	7,905,853,580,625
15	213,458,046,676,875
16	6,190,283,353,629,375
17	191,898,783,962,510,625
18	6,332,659,870,762,850,625
19	221,643,095,476,699,771,875
20	8,200,794,532,637,891,559,375
30	4.9518×10^{38}
40	1.00985×10^{57}
50	2.75292×10^{76}

Tree Space


A global maximum is not easy to find
end up here
but global maximum is here
if start here

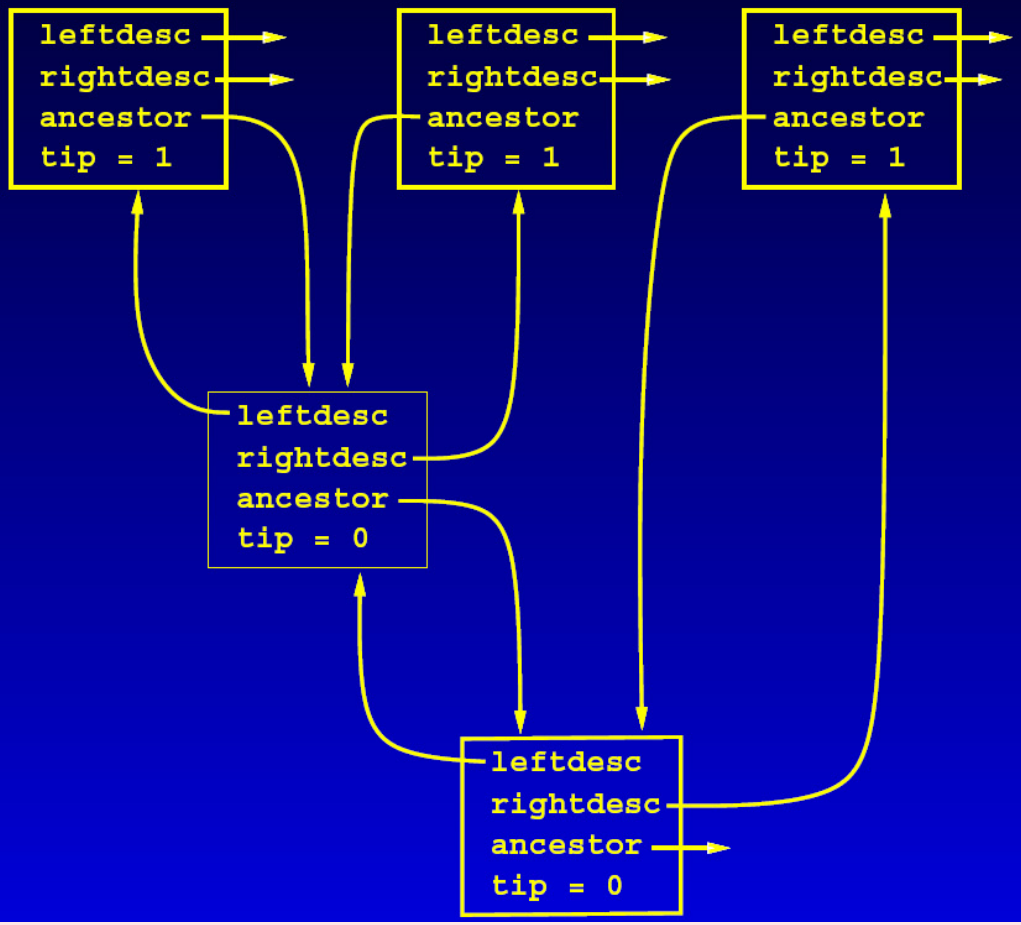
Computer representation of a phylogenetic tree

Let X be a set of taxa and T a phylogenetic tree on X.
To **represent a phylogenetic tree in a computer we maintain a set of nodes V and a set of edges E**.

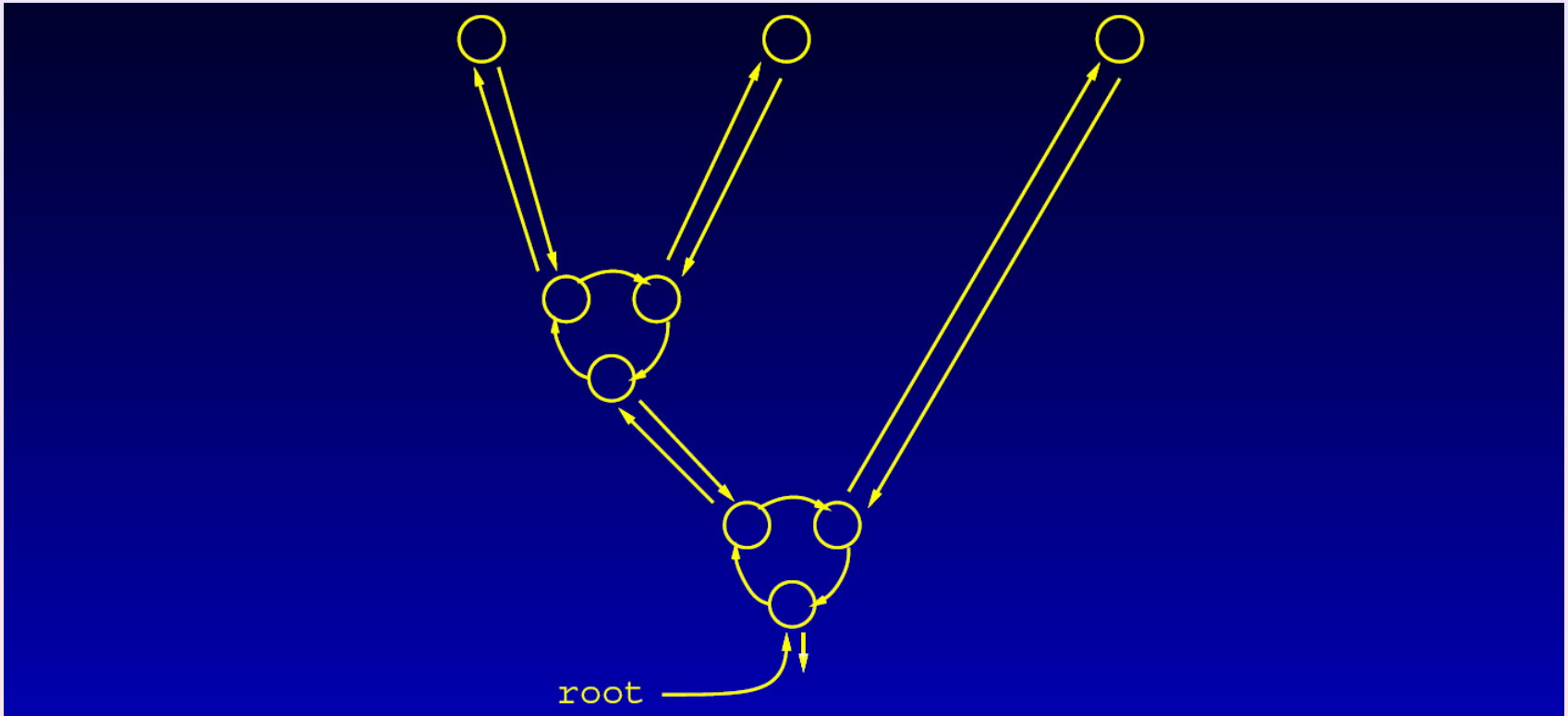
- Each edge $e \in E$ maintains a reference to its source node $s(e)$ and target node $t(e)$.
- Each node $v \in V$ maintains a list of references to all incident edges.
- Each node $v \in V$ maintains a reference $\lambda(v)$ to the taxon that it is labeled by, and, vice versa, $\nu(x)$ maps a taxon x to the node that it labels.
- Each edge $e \in E$ maintains its length $\omega(e)$.

Computer Representation

Using records (in C: structures, in Java and C++: classes) and pointers:
Here is one record-pointer structure representing a small tree:



Computer Representation allowing Multifurcations



This one allows multifurcations and is more easily rerootable. Each small circle represents a record with two pointers, "next" and "out", and a boolean variable "tip".

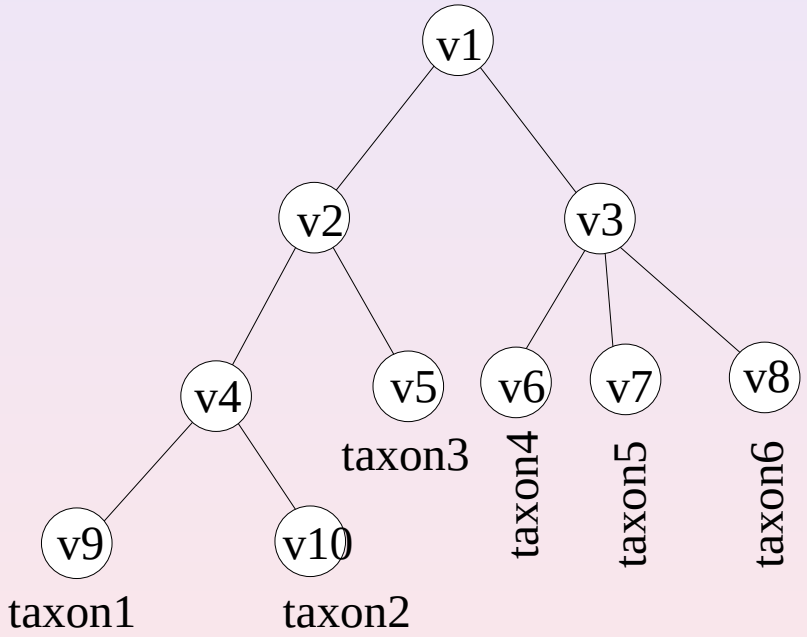
Nested structure

A rooted phylogenetic tree $T = (V, E, \lambda)$ is a **nested** structure:

- Consider any node $v \in V$.
- Let T_v denote the subtree rooted at v.
- Let $v_1, v_2, \ldots, v_k$ denote the children of v.
- Then T_v is obtainable from its subtrees $T_{v_1}, T_{v_2}, \ldots, T_{v_k}$ by connecting v to each of their roots.

Such a nested structure can be written using nested brackets:

Phylogenetic tree



nested description

$$v_1$$
$$\downarrow$$
$$(v_2, v_3)$$
$$\downarrow$$
$$((v_4, v_5), (v_6, v_7, v_8))$$
$$\downarrow$$
$$(((v_9, v_{10}), v_5), (v_6, v_7, v_8))$$

Description:

$$(((taxon_1, taxon_2), taxon_3), (taxon_4, taxon_5, taxon_6))$$

Printing a phylogenetic tree

Phylogenetic trees are usually printed using the so-called **Newick** format. The Newick Standard for representing trees in computer-readable form makes use of the correspondence between trees and nested parentheses, noticed in 1857 by the famous English mathematician Arthur Cayley.

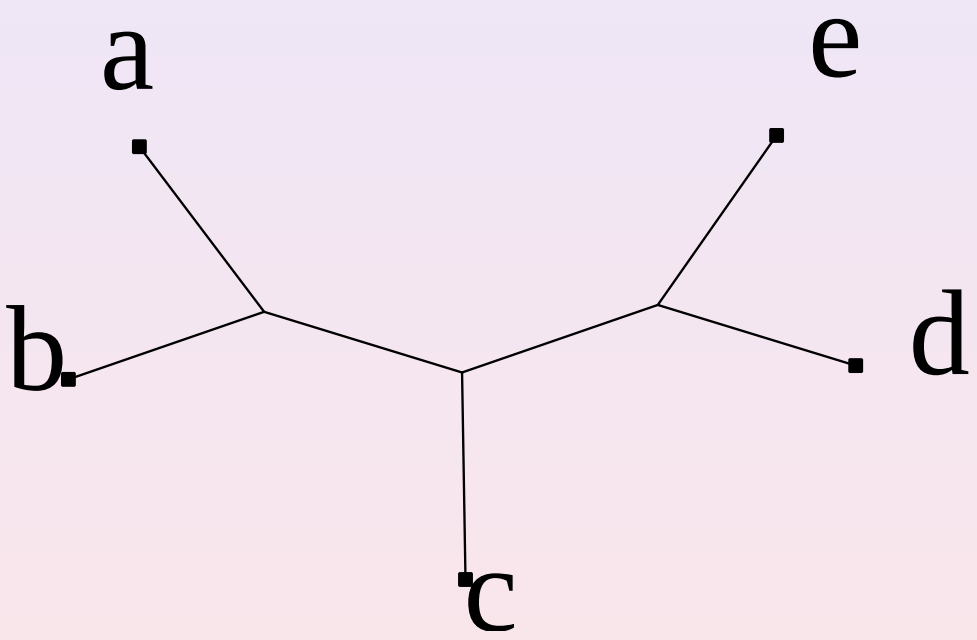
Newick Format

- The tree ends with a semicolon.
- Interior nodes are represented by a pair of matched parentheses.
- Between them are representations of the nodes that are immediately descended from that node, separated by commas.
- Tips are represented by their names.
- Trees can be multifurcating at any level.
- Branch lengths can be incorporated into a tree by putting a real number after a node and preceded by a colon. This represents the length of the branch immediately below that node.

Printing a phylogenetic tree

Here are two examples:

unrooted tree



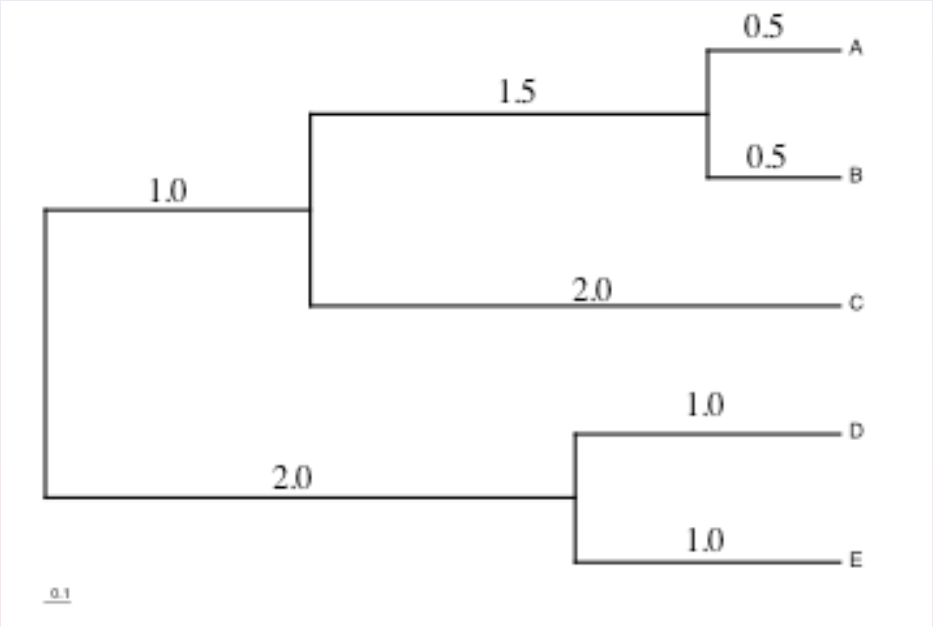
Newick string:

```
((a,b),c,(d,e));
```

Printing a phylogenetic tree

Here are two examples:

rooted tree with edge lengths



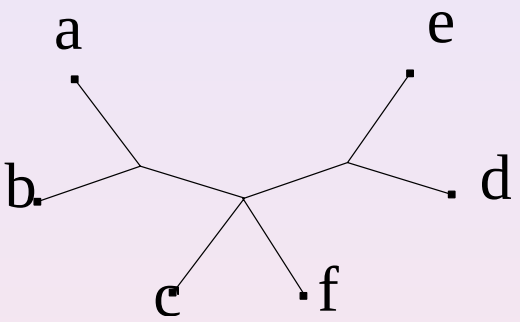
Newick string:

```
(((A:0.5,B:0.5):1.5,C:2.0):1.0,(D:1.0,E:1.0):2.0);
```

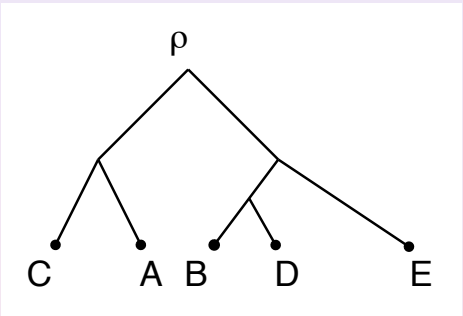
Printing a phylogenetic tree

Here are two more examples:

unrooted tree



rooted tree



Newick string:

`((a,b),c,f,(e,d))`

or e.g.:

`((a,b),(c,f,(e,d)))`

Newick string:

`((C,A),((B,D),E))`

...

Algorithm for printing a phylogenetic tree

The following algorithm recursively prints a tree in Newick format. It is initially called with $e = \mathit{null}$ and $v = \rho$, if the tree is rooted, or v set to an arbitrary internal node, else.

Algorithm for printing a phylogenetic tree

Algorithm toNewick(e, v)
Input: Phylogenetic tree $T = (V, E)$ on X, with labeling $\lambda : V \to X$
Output: Newick description of tree
begin
if v is a leaf **then**
    print $\lambda(v)$
else // v is an internal node
    print '('
    for each edge $f \neq e$ adjacent to v **do**
        If this is not the first pass of the loop, **print** ','
        Let $w \neq v$ be the other node adjacent to f
        call toNewick(f, w)
    print ')'
print ';'
end

Parsing a phylogenetic tree

We need to be able to read a phylogenetic tree into a program. The following algorithm parses a tree in Newick format from a (space-free) string $s = s_1 s_2 \ldots s_m$. The initial call is parseNewick$(1, m, \mathit{null})$.

Algorithm parseNewick(i,j,v)
Input: a substring $s_i \ldots s_j$ and a root node v
Output: a phylogenetic tree $T = (V, E, \lambda)$
begin
while $i \leq j$ **do**
    Add a new node w to V
    if $v \neq$ *null* **then** add a new edge $\{v, w\}$ to E

    if $s_i =$ '(' **then** // hit a subtree
        Set k to the position of the balancing close-bracket for i
        call parseNewick($i + 1, k - 1, w$) // strip brackets and recurse

    else // hit a leaf
        Set $k = \min\{k \geq i \mid s_{k+1} =$ ',' or $k + 1 = j\}$
        Set $\lambda(w) = s_i \ldots s_k$ // grab label for leaf

    Set $i = k + 1$ // advance to next ',' or j
    if $i < j$ **then**
        Check that $i + 1 \leq j$ and $s_{i+1} =$ ','
        Increment i // advance to next token **end**

Parsing a phylogenetic tree

- In the Newick format, each pair of matching brackets corresponds to an internal node and each taxon label corresponds to an external node.
- To add edge lengths to the format, specify the length *len* of the edge along which given node we visited by adding :*len* behind the closing bracket, in the case of an internal node, and behind the label, in the case of a leaf.
- For example, consider the tree `(a,b,(c,d))`. If all edges have the same length 1, then this tree is written as `(a:1,b:1,(c:1,d:1):1)`.

Drawing a phylogenetic tree

An **embedding** of a phylogenetic tree is given by an assignment of coordinates $(x(v), y(v))$ to each node v, such that

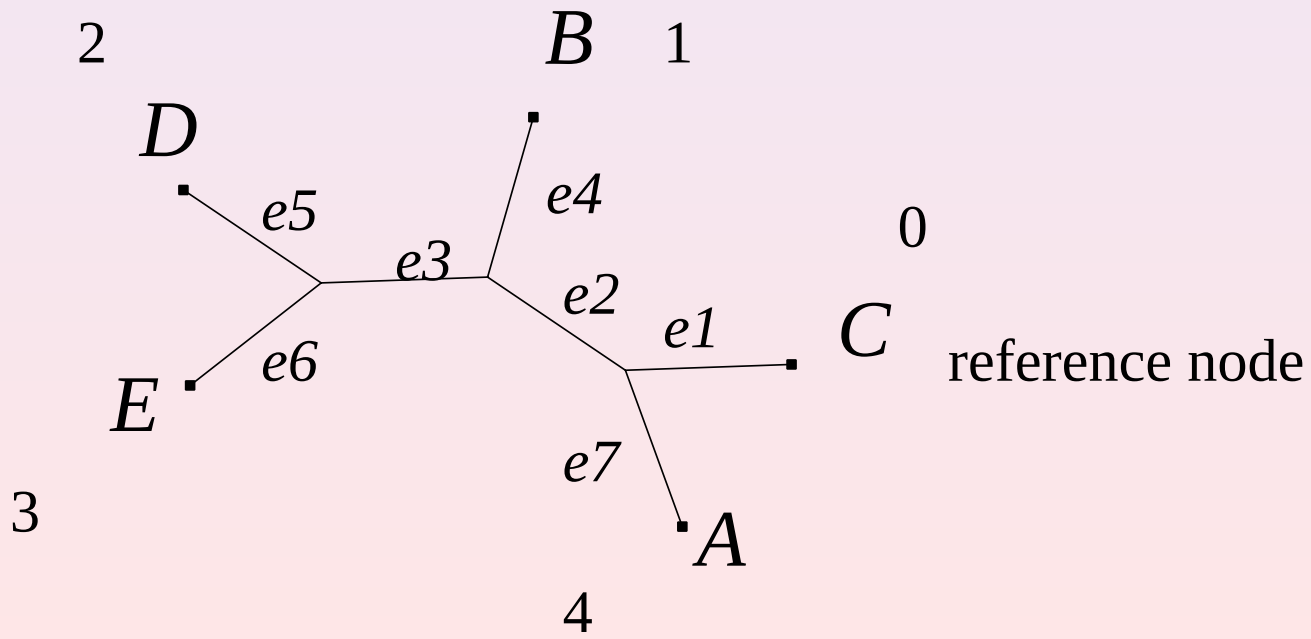
- any edge $e = (v, w)$ is represented by the line segment connecting points $(x(v), y(v))$ and $(x(w), y(w))$,
- no two such line segments cross, and
- for every edge e, the length of the line segment representing e equals $\omega(e)$.

There are many different ways to draw a phylogenetic tree, both unrooted and rooted. We will describe a method for drawing an unrooted tree, called the **circular embedding** method.

Computing a circular embedding

A circular embedding of a phylogenetic tree T is computed in two stages:

1. each edge e is assigned an angle $\alpha(e)$, and then,
2. based on the edge angles and lengths, each node v is assigned coordinates $(x(v), y(v))$.



Computing the edge angles

Given a phylogenetic tree T with n leaves. Choose an arbitrary leaf r and call it the **reference** node. Each leaf v is assigned a **rank** $h(v) \in \{0, 1, \ldots, n-1\}$ defined as follows:

- first set $h(r) = 0$ and then set $h(v) = i$, if v is the i-th leaf visited in a **depth-first search** (DFS) [1] of the tree from r.

We imagine the n leaves arranged around the unit circle in counter-clockwise order, with the i-th rank leaf v positioned at the angle $\alpha_i = \frac{i}{n} 2\pi$.

[1] Recall DFS: start from any node, go as far as possible (thus the name Depth First), until it can go no further. In this case, backtrack one step and repeat the process.

Computing the edge angles

Consider a fixed edge e and let U denote the set of all leaves that are separated from r by e. By definition of DFS, there exist two numbers p, q, with $1 \leq p \leq q \leq n-1$, such that $h(U) = \{p, p+1, \ldots, q\}$. We define $\alpha(e)$ as the **mean angle** associated with the leaves that are separated from r by e:

$$\alpha(e) = \frac{1}{q-p+1} \sum_{i=p}^{q} \alpha_i = \frac{\pi}{n}(p+q)$$

Computing the edge angles

This algorithm assigns an angle to every edge in the tree. The initial call is $\mathrm{setAngles}(0, \mathit{null}, r)$, where r is the reference leaf.

Algorithm setAngles(h, e, v)
Input: Number h of nodes placed, arrival edge e, current node v
Output: Angle $\alpha(e)$ for every edge $e \in E$.
Returns: New number of placed nodes.
begin
if v is a leaf and $v \neq r$ **then**
    return $h + 1$
Set $a = h$ // number of nodes placed before recursion
Set $b = 0$ // number of nodes placed after recursion
for all edges $f \neq e$ adjacent to v **do**
    Let $w \neq v$ be the other node adjacent to f
    Set $b = \mathrm{setAngles}(a, f, w)$
    Set $\alpha(f) = \frac{\pi(a+b)}{n}$
    Set $a = b$
return b.

Computing the edge angles

Lemma

The algorithm computes the edge angles of a circular embedding in linear time.

Proof.

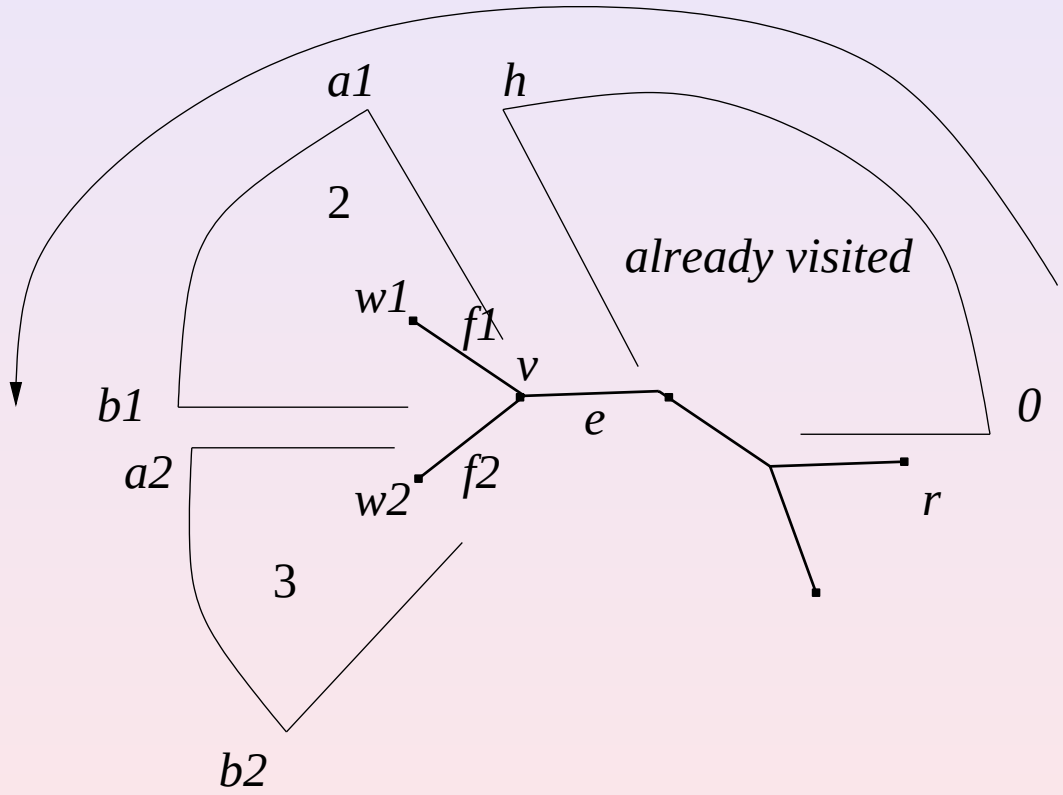
Every edge is visited exactly once, hence the runtime is linear in the number of edges $|E|$ (which is linear in the number of leaves).

To see that the algorithm produces the correct result, consider the situation for some arbitrary edge e and node v.

- The parameter h corresponds to the highest index assigned so far.
- Let $f_1, \ldots, f_k$ be the list of edges adjacent to v that are considered by the algorithm,
- let w_i denote the corresponding other node,
- and let a_i and b_i be the values of a and b directly after processing f_i.
- The latter two numbers denote the range of indices recursively assigned to the set of leaves in the subtree rooted at w

Computing the edge angles

The situation when processing node v:

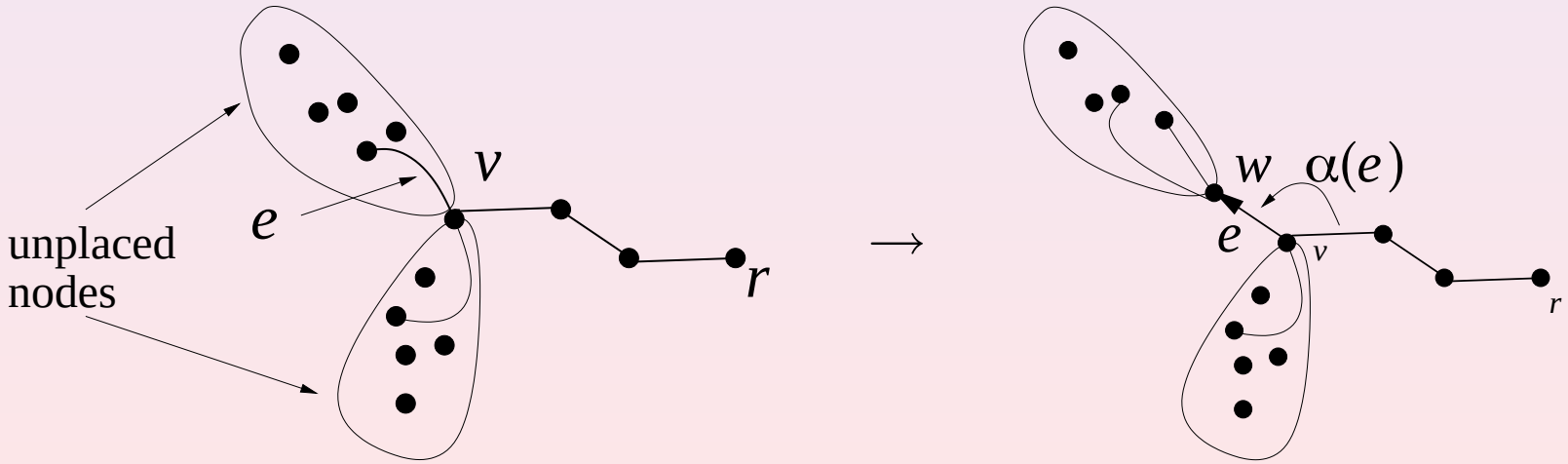


We then compute
$\alpha(f_i) = \frac{\pi(a_i+b_i)}{n} = \frac{2\pi}{n} \cdot \frac{a_i+b_i}{2} = \frac{2\pi}{n} \cdot \frac{a_i+a_{i+1}+\cdots+b_i}{b_i-a_i+1}$, as required in the definition of a circular embedding. □

Determining coordinates

Given a phylogenetic tree T and an angle function $\alpha : E \to [0, 2\pi]$, how do we obtain coordinates for the nodes?

Idea: In a depth-first traversal of the tree starting at the reference node r, for every edge e, simply push the opposite node away from the current node in the direction specified by $\alpha(e)$ by the amount specified by the length $\omega(e)$:



before processing e | after processing e

Determining coordinates

The following algorithm does the pushing. Initially, we set $\epsilon(r) = (0, 0)$ (the coordinates of the reference node r) and invoke $\mathrm{setCoordinates}(null, r)$.

Algorithm setCoordinates(e, v)
Input: Phylogenetic tree T and edge angles α
Output: Circular embedding ϵ of T
begin
Set $p = \epsilon(v)$
for each edge $f \neq e$ adjacent to v **do**
    Let $w \neq v$ be the other node adjacent to f
    Obtain p' by translating p in direction $\alpha(f)$ by amount $\omega(f)$
    Set $\epsilon(w) = p'$
    call $\mathrm{setCoordinates}(f, w)$
end

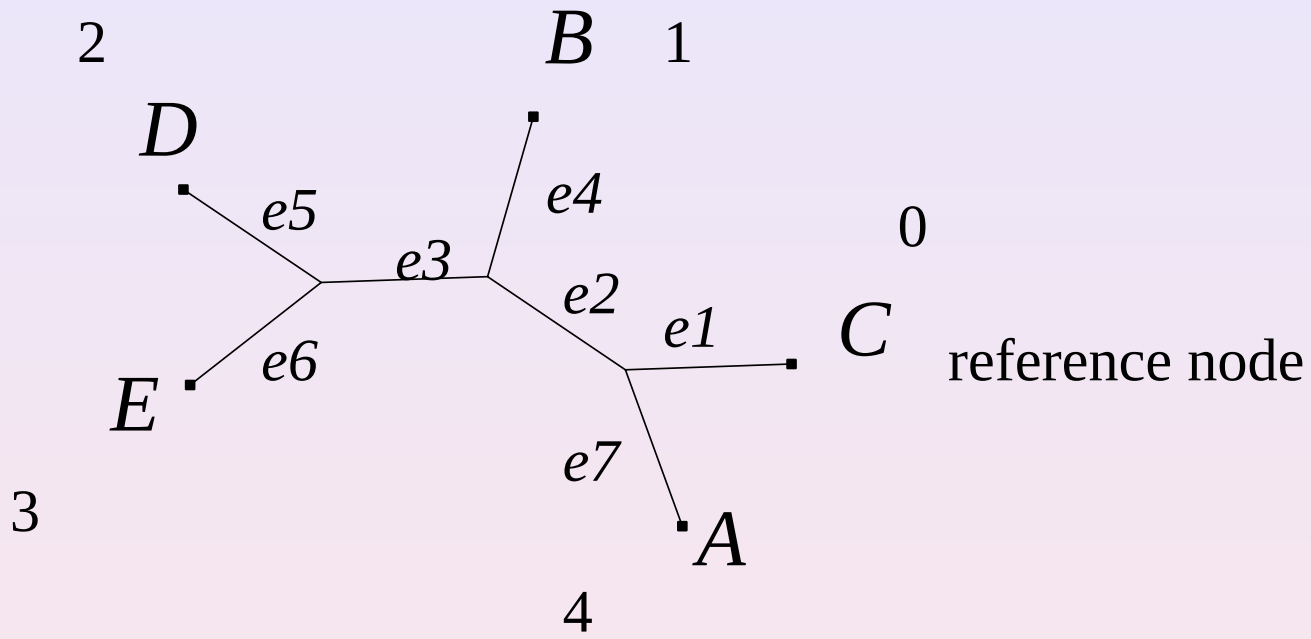
Determining coordinates

Theorem

A circular embedding is computable in linear time.

Challenge*: prove that the resulting configuration is indeed a* **planar** *embedding, that is, that no two edges can cross.*

Example



This is a circular layout with reference node C, which is verified as follows:

- $\alpha(e_1) = \frac{2\pi}{5 \cdot 4}(1 + 2 + 3 + 4) = \pi$,
- $\alpha(e_2) = \frac{2\pi}{5 \cdot 3}(1 + 2 + 3) = \frac{4}{5}\pi$,
- $\alpha(e_3) = \frac{2\pi}{5 \cdot 2}(2 + 3) = \pi$,
- ...

Part II

Distance-Based Tree Reconstruction

Constructing phylogenetic trees

There are three main approaches to constructing phylogenetic trees from molecular data.

1. Using a **distance method**, one first computes a distance matrix from a given set of biological data and then one computes a tree that represents these distances as closely as possible.
2. **Maximum parsimony** takes as input a set of aligned sequences and attempts to find a tree and a labeling of its internal nodes by auxiliary sequences such that the number of mutations along the tree is minimum.
3. Given a probabilistic model of evolution, **maximum likelihood** approaches aim at finding a phylogenetic tree that maximizes the likelihood of obtaining the given sequences.

Distances

Given a set $X = \{x_1, x_2, \ldots, x_n\}$ of taxa. The input to a distance method is a dissimilarity matrix $D : X \times X \to \mathbb{R}_{\geq 0}$ that associates a **distance** $d(x_i, x_j)$ with every pair of taxa $x_i, x_j \in X$. Sometimes we will abbreviate $d_{ij} := d(x_i, x_j)$ or $D_{ij} := d(x_i, x_j)$. We usually require that

1. the matrix is **symmetric**, that is, $d(x, y) = d(y, x)$ for all $x, y \in X$, and
2. $d(x, x) = 0$ for all $x \in X$ and $d(x, y) > 0$ for all $x \neq y$, and
3. the **triangle inequalities** are satisfied:

$$d(x, z) \leq d(x, y) + d(y, z) \text{ for all } x, y, z \in X.$$

Hamming distance

Let a collection of taxa be given by a set of distinct sequences $A = \{a_1, a_2, \ldots, a_n\}$ and assume we are given a multiple sequence alignment A^* of the sequences.

- We define **sequence dissimilarity** as the (normalized) **Hamming distance** $Ham(a_i, a_j)$ between two taxa a_i and a_j as the number of mismatch positions in a_i^* and a_j^*, divided by the number of comparisons.
- We ignore any column in which both sequences contain a gap. If only one sequence has a gap in a column then we can either ignore the column, or treat it as a match, or as a mismatch, depending on the type of data.
- Usually, one ignores **all** columns in which any of the n sequences contains the gap.

Hamming distance

Lemma

If the alignment is gap-less, then the corresponding distance matrix is a metric on A.

Proof.

Consider three distinct sequences $a_i, a_j, a_k \in A$. If $a_i^* \neq a_k^*$, then either $a_i^* \neq a_j^*$, or $a_k^* \neq a_j^*$, and hence $Ham(a_i, a_k) < Ham(a_i, a_j) + Ham(a_j, a_k)$. □

Hamming distance

For protein data, it makes sense to relax the definition of **sequence dissimilarity** to the number of "non-synonymous" residues divided by the number of sequence positions compared.

- For example, we may choose to ignore "conservative substitutions" by pooling amino acids with similar properties into six groups: acidic (D,E), aromatic (F,W,Y), basic (H,K,R), cysteine (C), non-polar (A,G,I,L,P,V), and polar (M,N,Q,S,T).
- Two residues are considered synonymous, if they are contained in the same group, and non-synonymous, otherwise.

Hamming distance

Example:

a_1	C	A	A	C	C	C	C	C	A	A	A	A	A
a_2	T	A	A	T	T	T	-	C	A	A	A	A	A
a_3	C	G	G	T	T	T	-	-	A	A	A	A	A

Distances:

$$Ham(a_1, a_2) = \frac{4}{12} = 0.\overline{33}$$

$$Ham(a_1, a_3) = \frac{5}{11} = 0.\overline{45}$$

$$Ham(a_2, a_3) = \frac{3}{11} = 0.\overline{27}$$

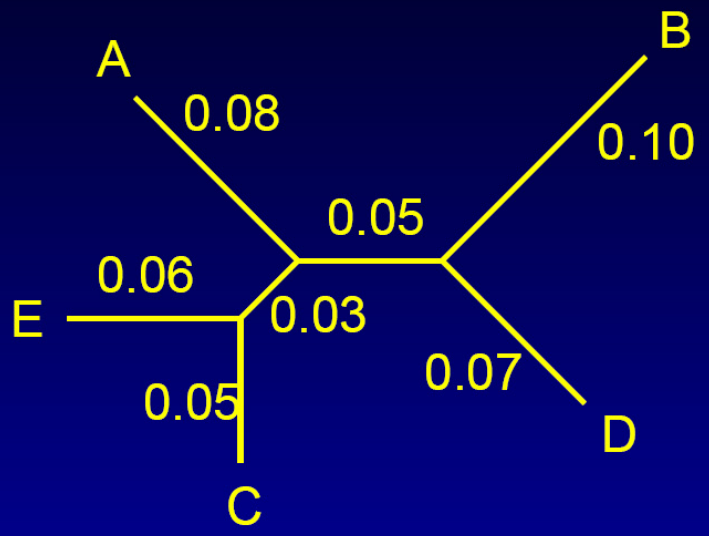
Hamming distances are only suitable for closely related sequences. Below we will discuss more sophisticated distances.

Question: What is the average Hamming distance between two random gap-free DNA sequences of the same length?

Distance-Based Tree Reconstruction

Distance-Based Phylogeny Problem:
Reconstruct an evolutionary tree on n leaves from an $n \times n$ distance matrix
Input: an $n \times n$ distance matrix $D = (d_{ij})$ and a labeling λ
Output: a phylogenetic tree T (rooted or unrooted) with n leaves and edge lengths.

A phylogeny with branch lengths



	A	B	C	D	E
A	0	0.23	0.16	0.20	0.17
B	0.23	0	0.23	0.17	0.24
C	0.16	0.23	0	0.15	0.11
D	0.20	0.17	0.15	0	0.21
E	0.17	0.24	0.11	0.21	0

and the pairwise distances it predicts

Least squares trees

Least squares methods minimize

$$Q = \sum_{i=1}^{n} \sum_{j \neq i} w_{ij}(D_{ij} - d_{ij})^2$$

over all trees, using the distances d_{ij} that they predict. Cavalli-Sforza and Edwards suggested $w_{ij} = 1$, Fitch and Margoliash suggested $w_{ij} = 1/D_{ij}^2$.

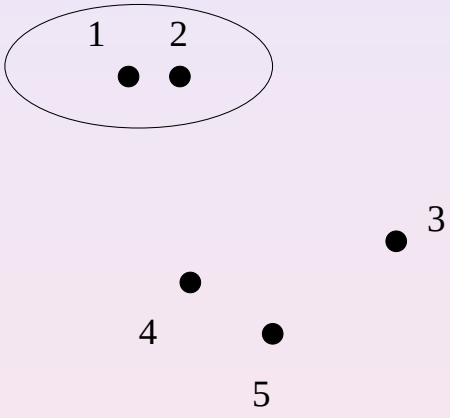
UPGMA

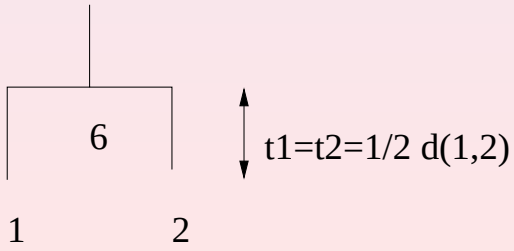
We will now discuss a simple distance method called UPGMA which stands for **unweighted pair group method using arithmetic averages** (Sokal & Michener 1958).

- Given a set of taxa X and a distance matrix D, UPGMA produces a rooted phylogenetic tree T with edge lengths.
- It operates by clustering the given taxa, at each stage merging two clusters and at the same time creating a new node in the tree.
- The tree is assembled **bottom-up**, first clustering pairs of leaves, then pairs of clustered leaves etc.
- Each node is given a height and the edge lengths are obtained as the difference of heights of its two end nodes.

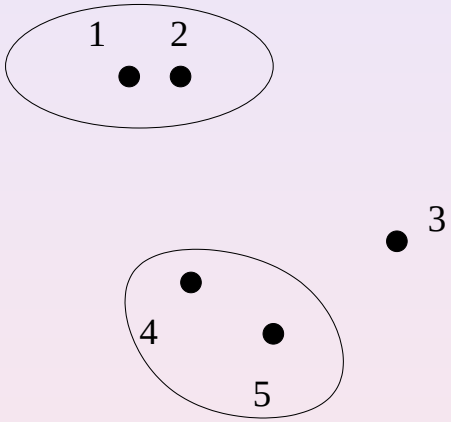
Example $X = \{1, 2, 3, 4, 5\}$, distances given by distance in the plane:

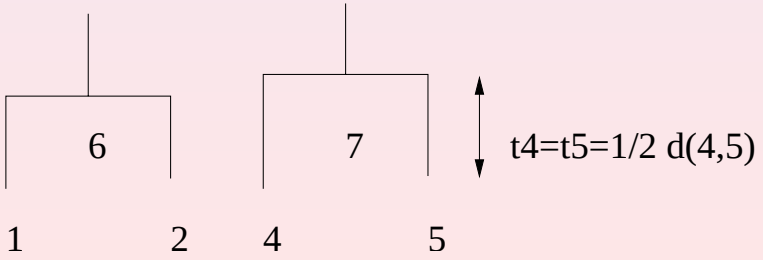
cluster 1 and 2:



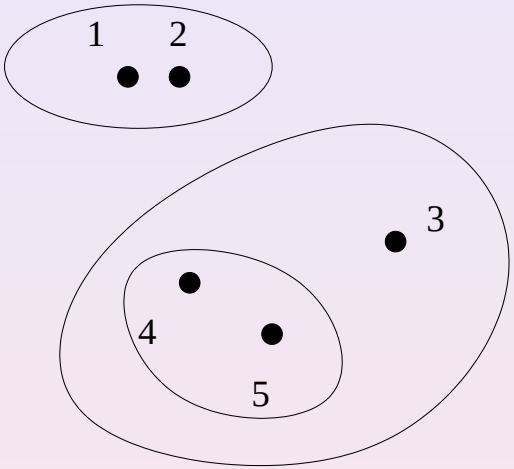


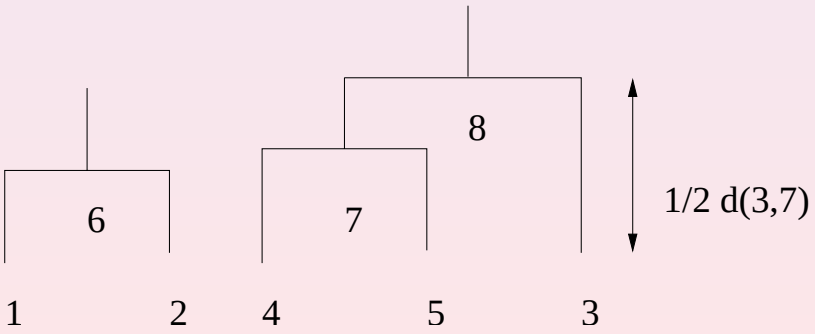
cluster 4 and 5:



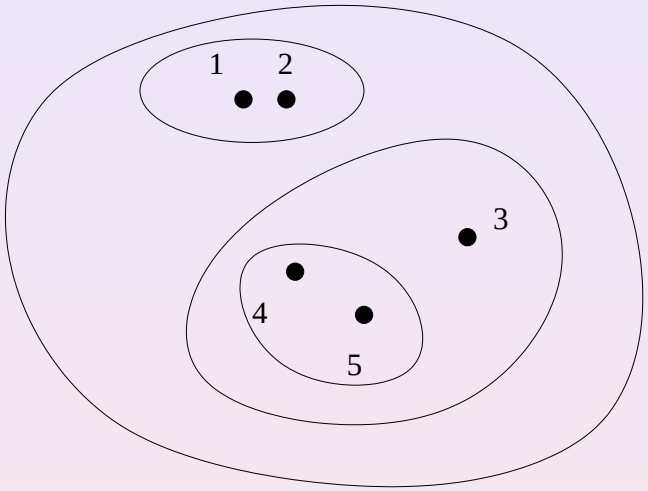


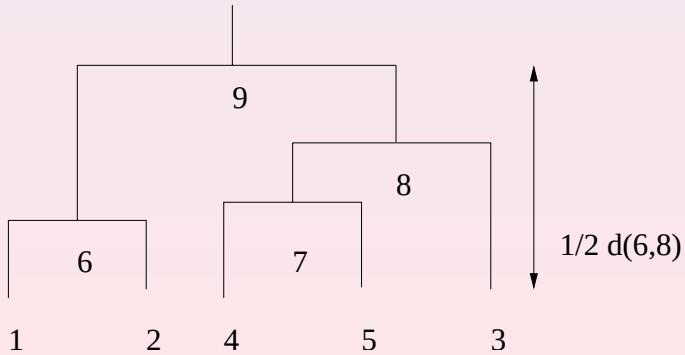
cluster 7 and 3:





cluster 6 and 8:





UPGMA produces a **rooted, binary phylogenetic tree**.

The distance between two clusters

Initially, we are given a distance $d(x, y)$ between any two taxa, i.e. leaves, x and y.
We define the **distance** $d(i, j) := d(C_i, C_j)$ between two clusters $C_i \subseteq X$ and $C_j \subseteq X$ **to be the average distance between pairs of taxa from each cluster**:

$$d(i,j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x,y).$$

The distance between two clusters

Note that, if C_k **is the union of two clusters** C_i **and** C_j, and C_l is any other cluster, then

$$d(k,l) = \frac{d(i,l)|C_i| + d(j,l)|C_j|}{|C_i| + |C_j|}.$$

This is a useful **update** formula, because using it in the algorithm, we can obtain the distance between two clusters in constant time.

The UPGMA algorithm

Algorithm UPGMA

Input: A set of taxa $X = \{x_1, \ldots, x_n\}$ and a corresponding distance matrix D
Output: A binary, rooted phylogenetic UPGMA tree $T = (V, E, \omega)$ on X

Initialization

Set $\mathcal{C} = \{C_1 = \{x_1\}, \ldots, C_n = \{x_n\}\}$
Assign each taxon x_i to its own cluster C_i
Set $h(\{x_i\}) = 0$ //Define one leaf of T for each taxon, placed at height zero
Set $V = \mathcal{C}$ and $E = \emptyset$

The UPGMA algorithm

Iteration

while $|\mathcal{C}| \geq 2$ **do**

Determine two clusters C_i and C_j for which $d(i,j)$ is minimal
Define a new cluster k by $C_k = C_i \cup C_j$
Set $\mathcal{C} = (\mathcal{C} - \{C_i, C_j\}) \cup \{C_k\}$
Set $d(k,l)$ for all clusters l using the update formula
Define a node k with daughter nodes i and j, and place it at height $h = \frac{d(i,j)}{2}$
Set $V = V \cup \{k\}$ and $E = E \cup \{(i,k),(j,k)\}$
Set $\omega(i,k) = h(k) - h(i)$ and $\omega(j,k) = h(k) - h(j)$

Termination

When only two clusters C_i and C_j remain, place the root at height $\frac{d(i,j)}{2}$.

Example of UPGMA applied to 5S rRNA data:

Original distances:

Abbreviations:
Bsu: *Bacillus subtilis*
Bst: *Baclillus stearothermophilus*
Lvi: *Lactobacillus viridescens*
Amo: *Acholeplasma modicum*
Mlu: *Micrococcus luteus*

	Bsu	Bst	Lvi	Amo	Mlu
Bsu	–	0.1715	0.2147	0.3091	0.2326
Bst		–	0.2991	0.3399	0.2058
Lvi			–	0.2795	0.3943
Amo				–	0.4289
Mlu					–

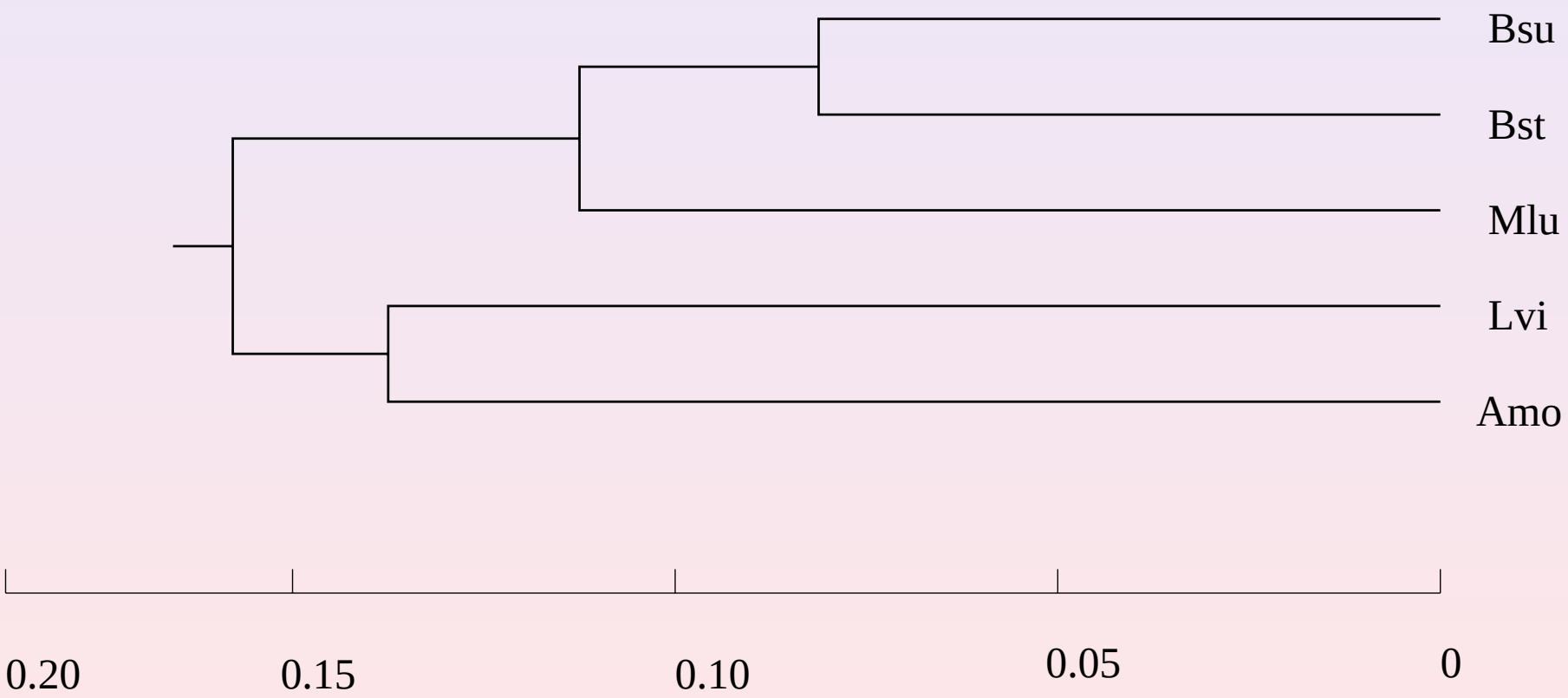
		Bsu + Bst	Lvi	Amo	Mlu
	Bsu + Bst	–	0.2569	0.3245	0.2192
→	Lvi		–	0.2795	0.3943
	Amo			–	0.4289
	Mlu				–

		Bsu + Bst + Mlu	Lvi	Amo
	Bsu + Bst + Mlu	–	0.3027	0.3593
→	Lvi		–	0.2795
	Amo			–

		Bsu + Bst + Mlu	Lvi + Amo
	Bsu + Bst + Mlu	–	0.3310
→	Lvi + Amo		–

Application of the UPGMA algorithm

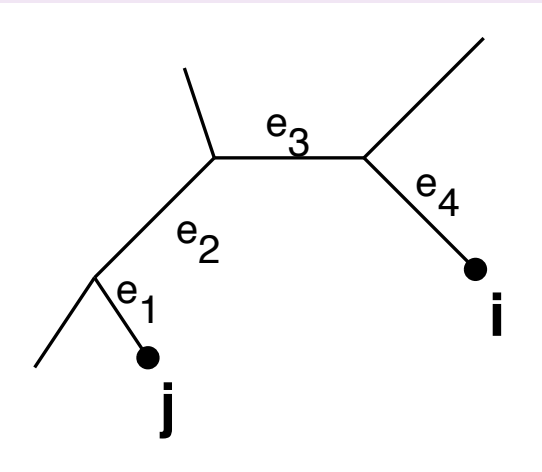
The resulting tree:



This tree is biologically incorrect, as we will see later.

Tree metrics

Definition: Given a phylogenetic tree T on X. We say that a distance function $d : X \times X \to \mathbb{R}^{\geq 0}$ is **directly obtainable** from T, if each d_{ij} was obtained by adding up the edge lengths of the path between the leaves i and j.



$$d_{ij} = \omega(e_1) + \omega(e_2) + \omega(e_3) + \omega(e_4)$$

Definition: Given a distance function $d : X \times X \to \mathbb{R}^{\geq 0}$ fulfilling the properties of a metric. Then d is a **tree metric** if there exists a phylogenetic tree T on X, such that d is directly obtainable from T.

Tree metrics

From this two fundamental questions arise:

- Is each distance function a tree metric?
- If d is a tree metric, does a unique tree and edge weight ω exist?

As we will see the answer to question one is “no”, while positive answers for the second question exist.

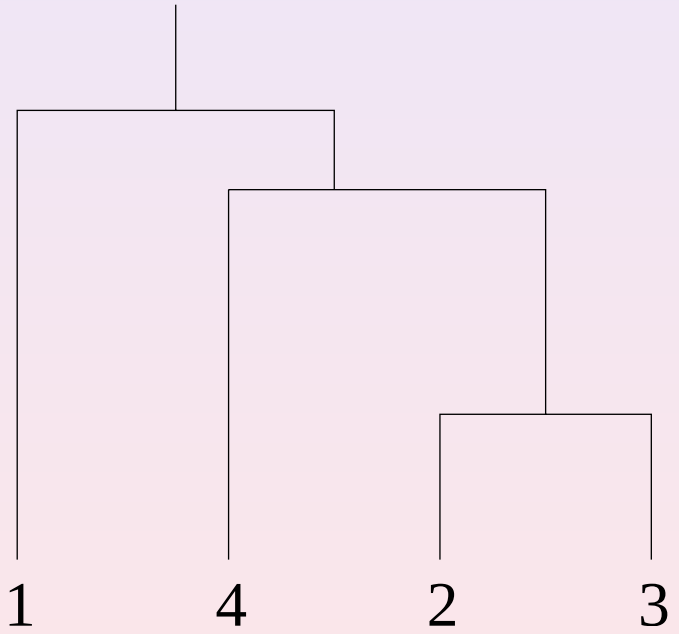
Tree metrics

The goal of phylogenetic analysis is usually to **reconstruct** a phylogenetic tree from data, such as distances or sequences, that was produced by some **generating** or **true** tree.

- When reconstructing trees from real data, the generating tree is the path of events that evolution actually took.
- In this case, the true tree is unknown and the objective is, of course, to try and reconstruct it.
- In contrast, in simulation studies, a **known** tree T_0 is used to generate artificial sequences and/or distances, under some specified model of evolution.
- A tree reconstruction method is then applied and its performance can be evaluated by comparing the resulting tree T with the true tree T_0.

The molecular clock hypothesis

Given a distance matrix D, the UPGMA method aims at building a rooted tree T with the property that all leaves have the same distance from the root ρ:



This approach is suitable for sequence data that has **evolved under circumstances in which the rate of mutations of sequences is constant over time and equal for all lineages in the tree.**

The molecular clock hypothesis

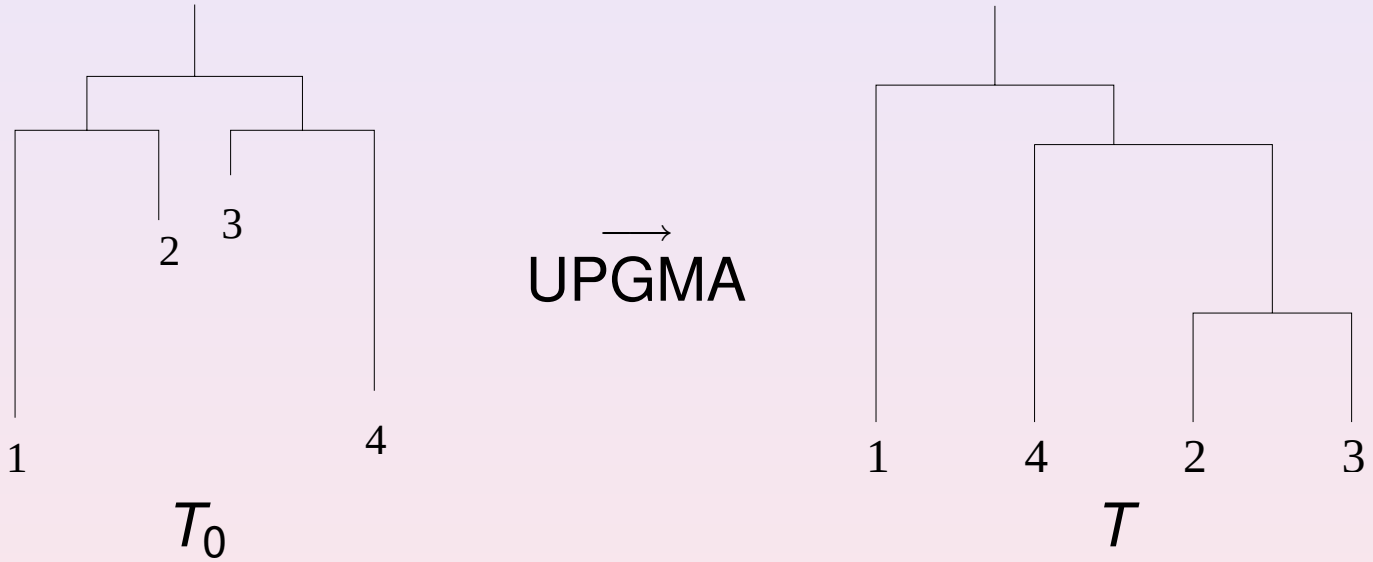
Definition The assumption that evolutionary events happen at a constant rate is called the **molecular clock** hypothesis.

UPGMA and the molecular clock

- If the input distance matrix D was directly obtained from a phylogenetic tree T_0 that fulfills the molecular clock assumption, then the tree T reconstructed by UPGMA from D will equal T_0.
- Otherwise, if T_0 does not do so, then UPGMA may fail to reconstruct the tree correctly

for example:

UPGMA and the molecular clock

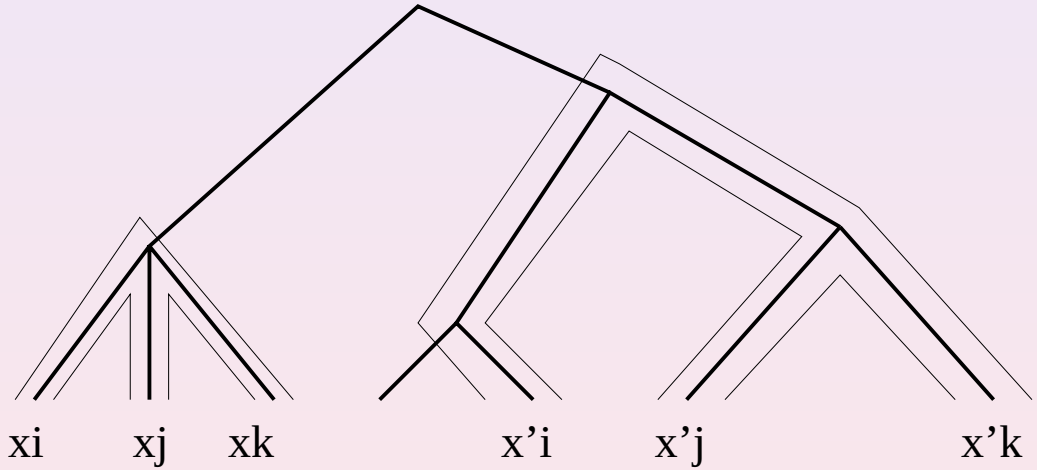


T_0 T

The problem here is that the closest leaves in T_0 are not neighboring leaves: they do not have a common parent node.

Definition: A distance matrix D is called an **ultrametric**, if for **every triplet** of taxa $x_i, x_j, x_k \in X$, the three distances $d(x_i, x_j)$, $d(x_i, x_k)$ and $d(x_j, x_k)$ have the property that either:

1. all three distances are equal, or
2. two are equal and the remaining one is smaller.



condition (1) condition (2)

Lemma

if D was directly obtained from some tree T that satisfies the molecular clock hypothesis, then D is an ultrametric.

The ultrametric property

We say that a rooted phylogenetic tree T is **ultrametric**, if every leaf has the same distance from the root.
One can show the following result:

Theorem

If D is a distance matrix directly obtainable from some ultrametric tree T, then UPGMA applied to D will produce that tree T.

Consistency

So, UPGMA computes the correct tree T, when given distances $D = D(T)$ directly obtained from a ultrametric **model tree** T.

- In applications, we do not know T (of course) and thus cannot "directly obtain" D from T.
- We usually have an alignment of sequences and use it to compute an **estimation** $\hat{D}$ of $D(T)$.

Consistency

Given a suitable model of evolution (such as the Jukes-Cantor model discussed below) that specifies how the sequences evolve on T. As the length n of the aligned sequences increases, $\hat{D}$ will converge to D.

- A tree construction method is called **consistent** on a model tree T, if the tree that it produces "converges" toward T, as n increases.

Lemma

UPGMA is consistent on the set of ultrametic trees.

Estimating the deviation from a molecular clock

Given a distance matrix D obtained by comparison of sequences generated along some unknown tree T_0.

- Biologically, it may be of interest to know how well the molecular clock hypothesis holds.

In other words, how close is D to being an ultrametric?

Estimating the deviation from a molecular clock

To answer this question, we define the **stretch of an ultrametric** U with respect to D as follows:

$$stretch_D(U) = \max_{i,j \in X} \left\{ \frac{D_{ij}}{U_{ij}}, \frac{U_{ij}}{D_{ij}} \right\}.$$

The **stretch of** D is defined as the minimum stretch over all possible ultrametrics:

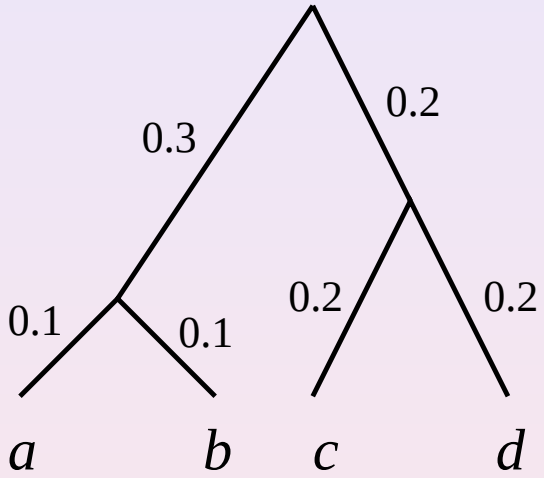
- $stretch(D) = \min_U \{stretch_D(U)\}$ and gives a lower bound for the stretch $stretch_D(T)$ of any tree T obtained from D.

This value can be computed in $O(n^2)$ time, see [2]

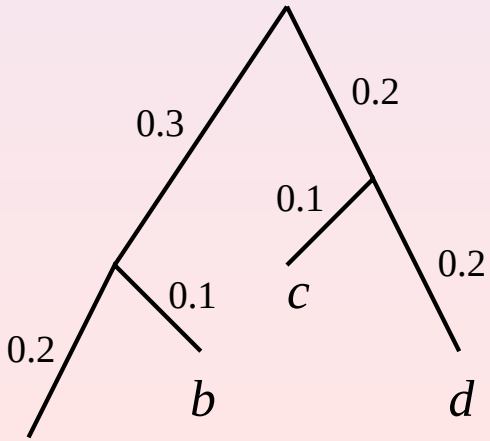
[2] L. Nakhleh, U. Roshan, L. Vawter and T. Warnow, LNCS 2452:287-299 (2002).

Estimating the deviation

Example of an ultrametric tree metric U and a non-ultrametric tree metric D:



$$\rightarrow \quad U = \begin{array}{c|cccc} & a & b & c & d \\ a & - & 0.2 & 0.8 & 0.8 \\ b & & - & 0.8 & 0.8 \\ c & & & - & 0.4 \\ d & & & & - \end{array}$$



$$\rightarrow \quad D = \begin{array}{c|cccc} & a & b & c & d \\ a & - & 0.3 & 0.8 & 0.9 \\ b & & - & 0.7 & 0.8 \\ c & & & - & 0.3 \\ d & & & & - \end{array}$$

Estimating the deviation from a molecular clock

The stretch of U w.r.t. D is:

$$stretch_D(U) = \max\left\{\frac{0.3}{0.2}, \frac{0.9}{0.8}, \frac{0.8}{0.7}, \frac{0.4}{0.3}\right\} = \frac{3}{2}.$$

Additivity and the four-point condition

As we have seen a distance matrix often does not fulfill the ultrametric property. Now we ask if we can define a property of non-ultrametric tree metrics.

Definition: Given a set of taxa X. A distance matrix D on X fulfills the so-called **four-point condition** if for all $i, j, k, l \in X$

$$d(i,j) + d(k,l) \leq \max\{d(i,k) + d(j,l), d(i,l) + d(j,k)\}$$

holds.

That is, **two of the three expressions** $d(x_i, x_j) + d(x_k, x_l)$, $d(x_i, x_k) + d(x_j, x_l)$, and $d(x_i, x_l) + d(x_j, x_k)$ are **equal and are not smaller than the third**.

Additivity and the four-point condition

The following theorem now tells us that the four-point condition and tree metric are equivalent properties, a result due to Peter Buneman (1971):

Theorem

A distance matrix D on X is a tree metric, if and only if it fulfills the four-point condition.

Additivity and the four-point condition

We want to at least prove one direction of the theorem:

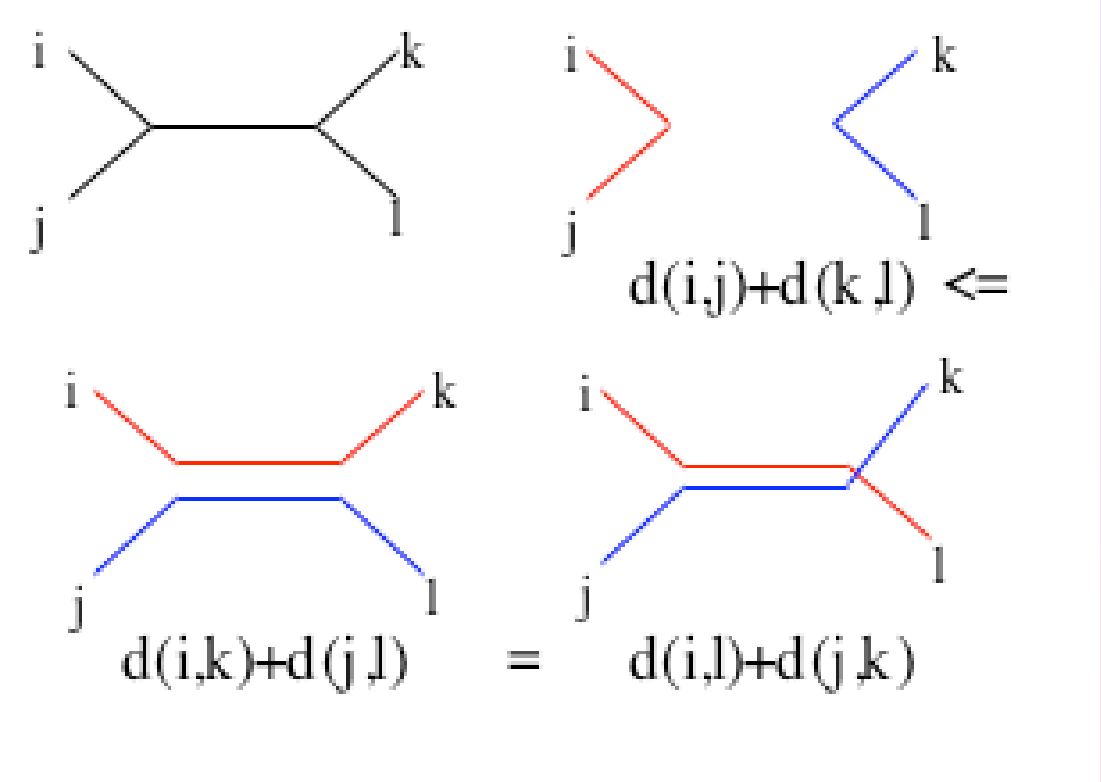
- Let D be a tree metric on X.
- Then there exists a binary phylogenetic tree T.
- Let i, j, k, l be elements from X. If they are all different, then w.l.o.g we assume that (i, j) and (k, l) are neighbors respectively.

Then it follows that

$$d(i,j) + d(k,l) \leq d(i,k) + d(j,l) = d(i,l) + d(j,k)$$

as can be seen from the following figure:

Additivity and the four-point condition



and thus D fulfills the four-point condition.

Remark: A tree metric is often called an **additive** metric.

The four-point condition

Example additive metric:

$$D = \begin{pmatrix} & B & C & D \\ A & 7 & 6 & 5 \\ B & & 3 & 6 \\ C & & & 5 \end{pmatrix}$$

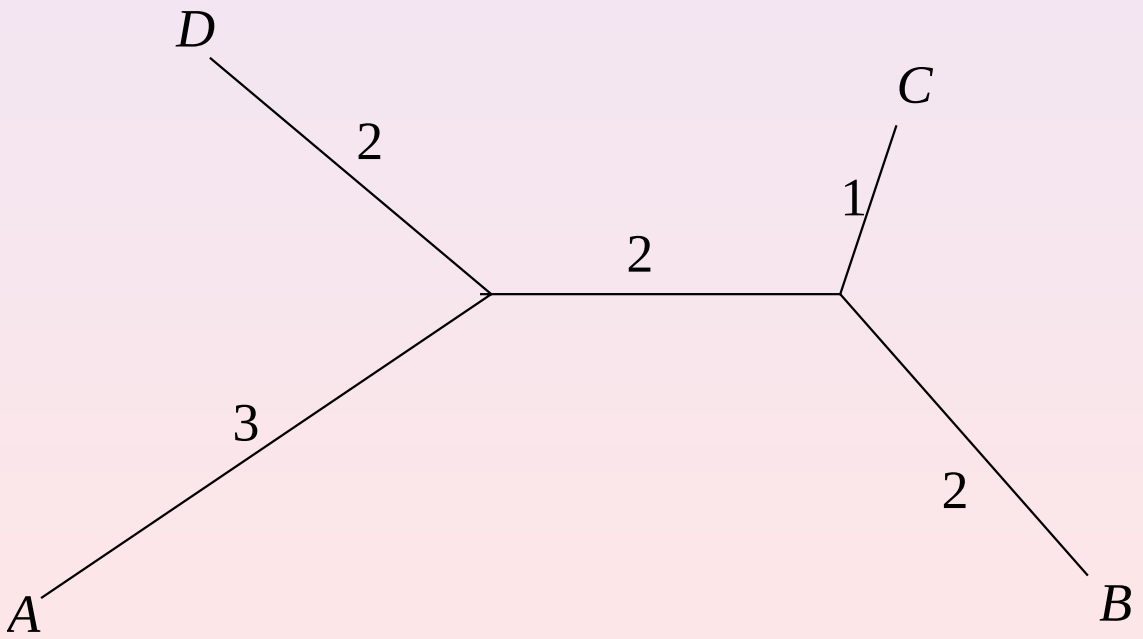
Check the four-point condition:
$d(A,B) + d(C,D) = 7 + 5 = 12$
$d(A,C) + d(B,D) = 6 + 6 = 12$
$d(A,D) + d(B,C) = 5 + 3 = 8$
$\Rightarrow$ the four-point condition holds.

The four-point condition

Example additive metric:

$$D = \begin{pmatrix} & B & C & D \\ A & 7 & 6 & 5 \\ B & & 3 & 6 \\ C & & & 5 \end{pmatrix}$$

Check the four-point condition:
$d(A,B) + d(C,D) = 7 + 5 = 12$
$d(A,C) + d(B,D) = 6 + 6 = 12$
$d(A,D) + d(B,C) = 5 + 3 = 8$
$\Rightarrow$ the four-point condition holds.



The four-point condition

A non-additive metric:

$$D = \begin{pmatrix} & B & C & D \\ A & 7 & 7 & 6 \\ B & & 4 & 7 \\ C & & & 5 \end{pmatrix}$$

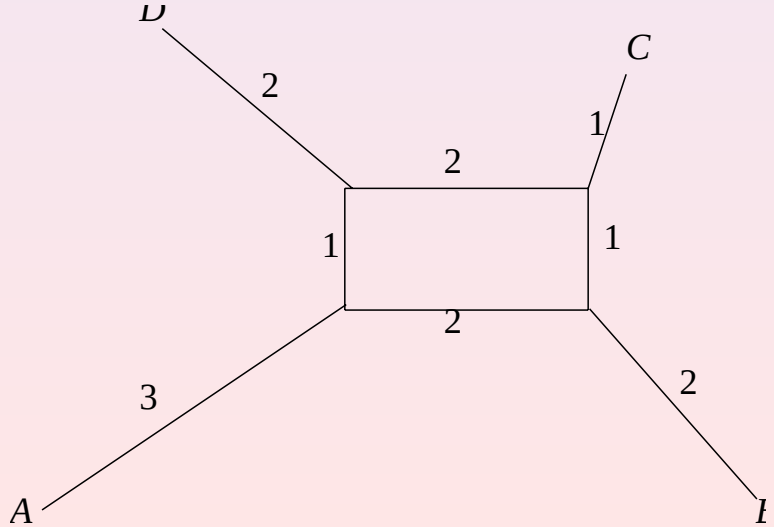
Check the four-point condition:
$d(A,B) + d(C,D) = 7 + 5 = 12$
$d(A,C) + d(B,D) = 7 + 7 = 14$
$d(A,D) + d(B,C) = 6 + 4 = 10$
$\Rightarrow d(A,C) + d(B,D) \not\leq$
$\max\left(d(A,B) + d(C,D), d(A,D)\right)$
$\Rightarrow$ 4-point condition doesn't hold.

The four-point condition

A non-additive metric:

$$D = \begin{pmatrix} & B & C & D \\ A & 7 & 7 & 6 \\ B & & 4 & 7 \\ C & & & 5 \end{pmatrix}$$

Check the four-point condition:
$d(A,B) + d(C,D) = 7 + 5 = 12$
$d(A,C) + d(B,D) = 7 + 7 = 14$
$d(A,D) + d(B,C) = 6 + 4 = 10$
$\Rightarrow d(A,C) + d(B,D) \not\leq$
$\max\left(d(A,B) + d(C,D), d(A,D)\right)$
$\Rightarrow$ 4-point condition doesn't hold.



Neighbor-Joining

The most widely used distance method is **Neighbor-Joining (NJ)**, originally introduced by Saitou and Nei (1987)[3], and modified by Studier and Keppler (1988).

- Given a distance matrix D, Neighbor-Joining produces an unrooted phylogenetic tree T with edge lengths.
- It is more widely applicable than UPGMA, as it does not assume a molecular clock.

[3] Saitou, N., Nei, Y. (1987) **SIAM J. on Comp.** 10:405-421

Neighbor-Joining

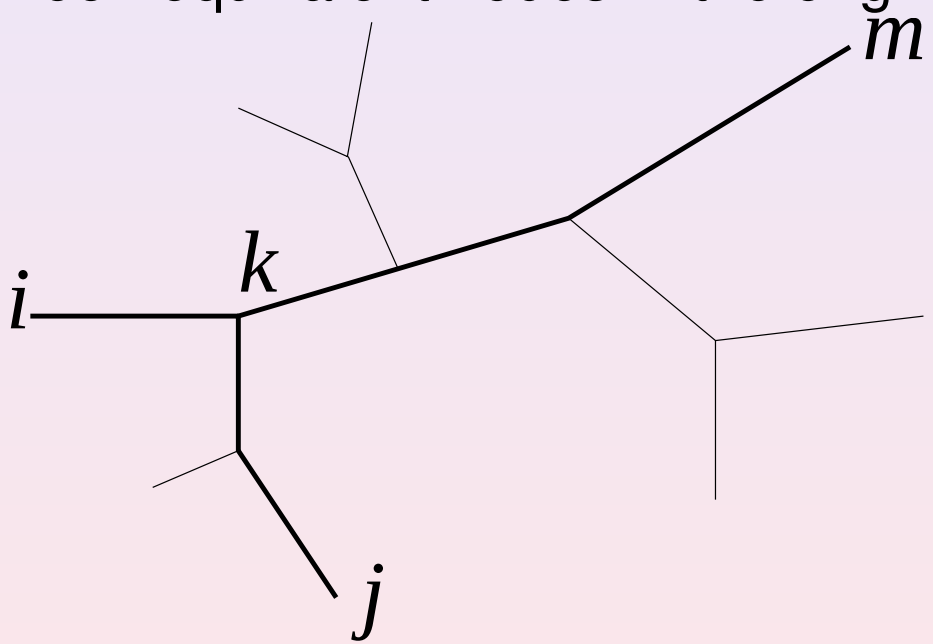
Let D be a distance matrix directly obtainable from some (unknown) tree T.
Assume that we are building a tree based on D by repeatedly pairing "neighboring" taxa.
The following step reduces the number of leaves by one and we can repeatedly apply it until we arrive at a single pair of leaves:

- Let i and j be two **neighboring leaves** that have the same parent node, k.
- Remove i, j from the list of nodes and add k to the current list of nodes.

How do we have to set its distance to any given leaf m?

Neighbor-Joining

By **additivity of** D, we can compute the distances d_{km} from those between equivalent nodes in the original tree:



In other words, for any three leaves i, j, m there is a node k where the paths to them meet.

Neighbor-Joining

By additivity,

$$d_{im} = d_{ik} + d_{km},\ d_{jm} = d_{jk} + d_{km} \text{ and } d_{ij} = d_{ik} + d_{jk},$$

thus

$$d_{im} + d_{jm} = d_{ik} + d_{km} + d_{jk} + d_{km} = d_{ij} + 2d_{km},$$

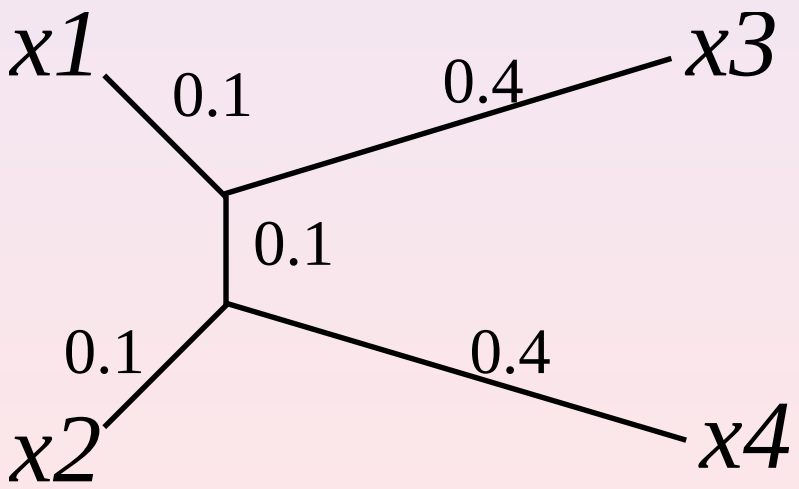
which implies $d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$.

Neighbor-Joining

How to determine which nodes are neighbors?

The Neighbor-Joining method is based on the fact that we can decide which nodes are neighbors, using only the distance matrix.

- However, it does **not** suffice simply to pick the two closest leaves, i.e. a pair i, j with d_{ij} minimal, for example:



Given distances generated by this tree.

Leaves x_1 and x_2 have minimal distance, but are not neighbors.

Neighbor-Joining

To avoid this problem, the trick is to **subtract the "averaged distances"[4] to all other leaves**, thus compensating for long edges. We define:

$$N_{ij} := d_{ij} - (r_i + r_j),$$

where

$$r_i = \frac{1}{|L| - 2} \sum_{k \in L} d_{ik},$$

and L denotes the set of leaves.

[4] Note that this is not precisely the average, as the number of summands is $|L|$, not $|L - 2|$.

Neighbor-Joining

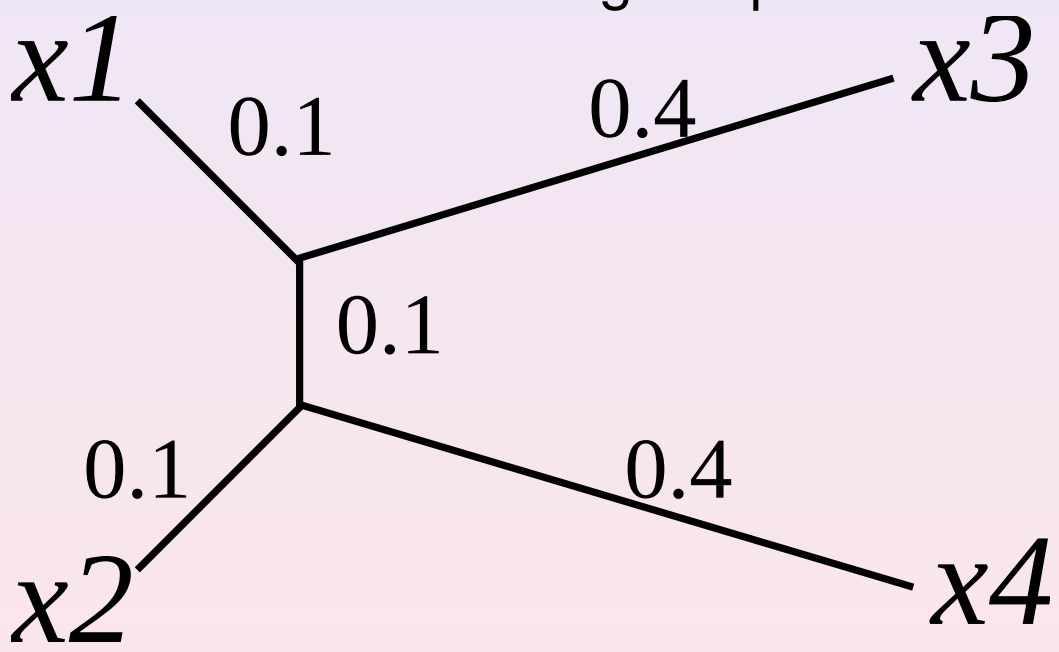
Lemma

If D is directly obtainable from some tree T, then the 2 leaves x_i and x_j for which N_{ij} is minimal are neighbors in T.

This result ensures that the Neighbor-Joining algorithm will correctly reconstruct a tree from its additive distances.

Neighbor-Joining

Let us illustrate this result using the previous example:



Neighbor-Joining

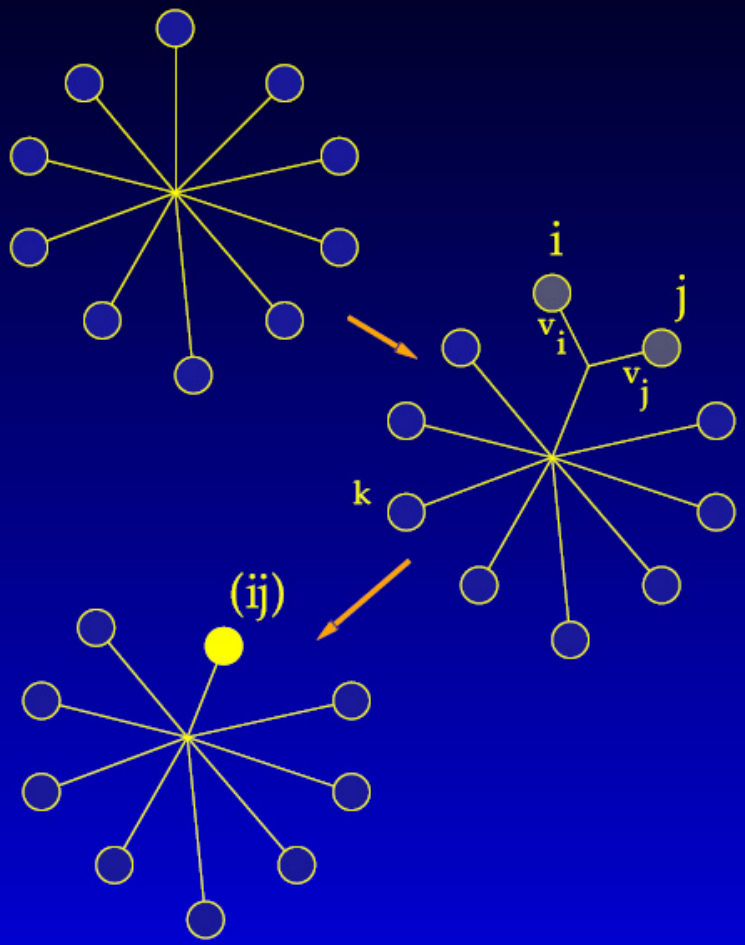
Here, $r_1 = 1/2(0.5 + 0.3 + 0.6) = 0.7$, and equivalently we compute $r_2 = 0.7$, $r_3 = 1.0$ and $r_4 = 1.0$. And so,

$$N = \begin{array}{c|ccc} & x_2 & x_3 & x_4 \\ x_1 & -1.1 & \mathbf{-1.2} & -1.1 \\ x_2 & & -1.1 & \mathbf{-1.2} \\ x_3 & & & -1.1 \end{array}$$

The matrix N attains a minimum value for the pair $i = 1$ and $j = 3$ and for the pair $i = 2$ and $j = 4$, as required.

Algorithm (Neighbor-Joining)
Input: Distance matrix D
Output: Phylogenetic tree T
Initialization:
Define T to be the set of leaf nodes, one for each taxon.
Set $L = T$.
Iteration:
Pick a pair $i, j \in L$ for which N_{ij} is minimal.
Define a new node k and
set $d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$, for all $m \in L$.
Add k to T with edges of lengths $d_{ik} = \frac{1}{2}(d_{ij} + r_i - r_j)$ and
$d_{jk} = d_{ij} - d_{ik}$, joining k to i and j, respectively.
Remove i and j from L and add k.
Termination:
When L consists of two leaves i and j, add the remaining
edge between i and j, with length d_{ij}.

Star decomposition search

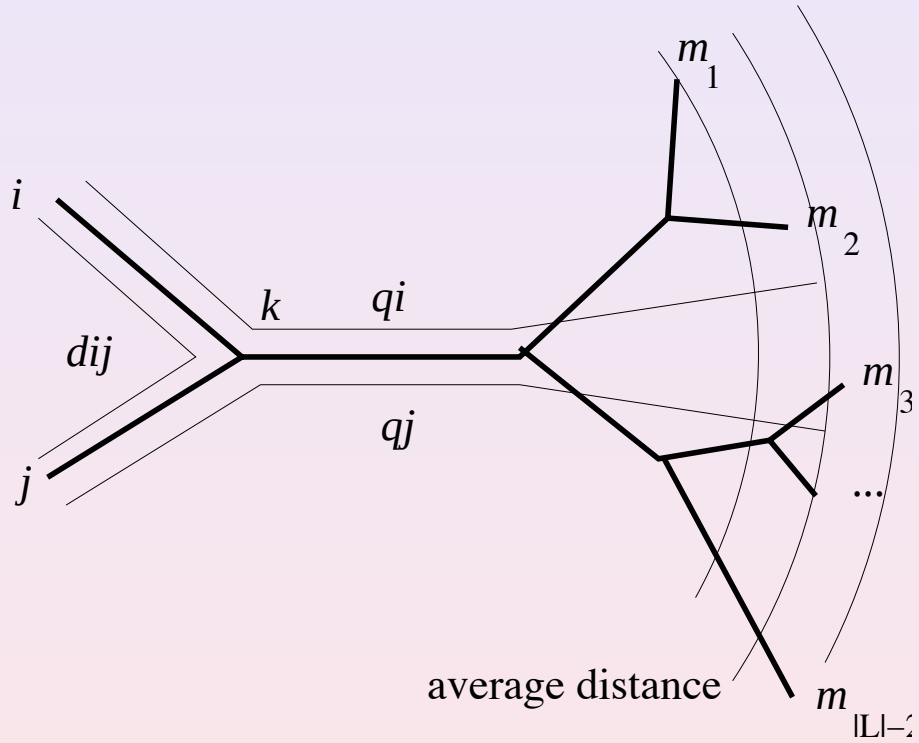


"Star decomposition" tree search method used in Neighbor-Joining method

Neighbor-Joining algorithm

Why can we use $d_{ik} = \frac{1}{2}(d_{ij} + r_i - r_j)$ to update distances?
By definition, $r_i = \frac{1}{|L|-2} \sum_{k \in L} d_{ik}$
equals the average distance $q_i = \frac{1}{|L|-2} \sum_{k \in L, k \neq i,j} d_{ik}$
from i to all other nodes $m \neq i, j$, plus $\frac{d_{ij}}{|L|-2}$:

Neighbor-Joining algorithm



As we see from this figure:

$$2d_{ik} = d_{ij} + q_i - q_j = d_{ij} + q_i - q_j + \frac{d_{ij}}{|L|-2} - \frac{d_{ij}}{|L|-2} = d_{ij} + r_i - r_j.$$

Application of Neighbor-Joining

- Given an additive distance matrix D directly obtained from a phylogenetic tree T, Neighbor-Joining is guaranteed to reconstruct T correctly.
- However, **in practice we are never given a matrix that was "directly obtained" from the generating tree**, but rather the distance matrix is usually obtained very indirectly by a comparison of finite sequence data generated along the tree. Such data is rarely additive.
- Nevertheless, the Neighboring-Joining method is often applied to such data and has proved to be a fast, useful and robust tree reconstruction method.

Moreover:

Lemma

Neighbor-Joining is consistent.

Example

Input matrix:

$$D_0 = \begin{array}{c|cccc|c} Taxa & A & B & C & D & r \\ \hline A & - & 8 & 7 & 12 & 27 \\ B & & - & 9 & 14 & 31 \\ C & & & - & 11 & 27 \\ D & & & & - & 37 \end{array} \quad \rightarrow$$

$$N_0 = \begin{array}{c|cccc} & A & B & C & D \\ A & - & \mathbf{-21} & -20 & -20 \\ B & & - & -20 & -20 \\ C & & & - & -21 \\ D & & & & - \end{array}$$

Example

Data after one merge of neighbors:

$$D_1 = \left\{\begin{array}{c|ccc|c} & A+B & C & D & r \\ A+B & - & 4 & 9 & 13 \\ C & & - & 11 & 15 \\ D & & & - & 20 \end{array}\right. \quad \rightarrow$$

$$N_1 = \left\{\begin{array}{ccccc} & A+B & C & D \\ A+B & - & -\mathbf{10} & -7.5 \\ C & & - & -4.5 \\ D & & & - \end{array}\right.$$

Example

Data after two merges of neighbors:

$$D_2 = \left\{ \begin{array}{c|ccc|c} & & A+B+C & D & r \\ A+B+C & & - & 8 & 8 \\ D & & & - & 8 \end{array} \right. \qquad \rightarrow$$

$$N_2 = \left\{ \begin{array}{ccc} & A+B+C & D \\ A+B+C & - & \\ D & & - \end{array} \right.$$

Example

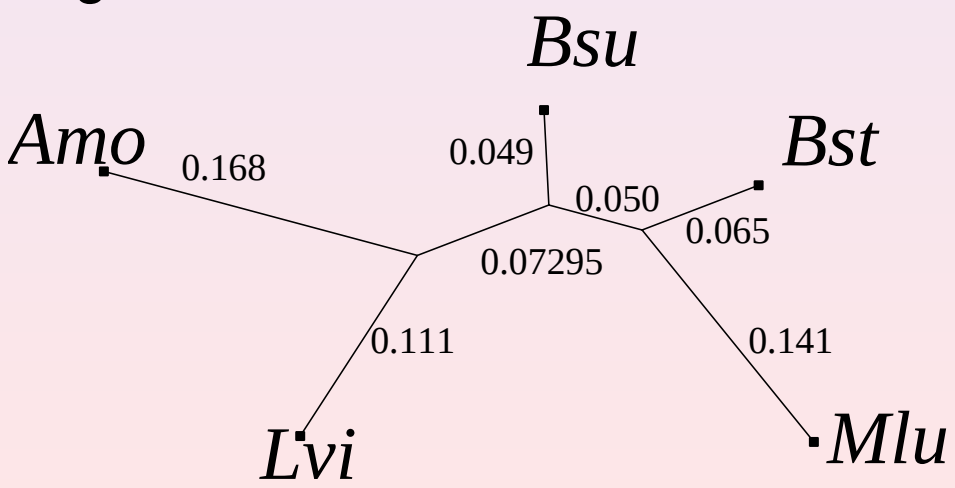
Example of Neighbor-Joining applied to 5S rRNA data:

Original distances:

	Bsu	Bst	Lvi	Amo	Mlu
Bsu	–	0.1715	0.2147	0.3091	0.2326
Bst		–	0.2991	0.3399	0.2058
Lvi			–	0.2795	0.3943
Amo				–	0.4289
Mlu					

Abbreviations:
Bsu: *Bacillus subtilis*
Bst: *Baclillus stearothermophilus*
Lvi: *Lactobacillus viridescens*
Amo: *Acholeplasma modicum*
Mlu: *Micrococcus luteus*

The resulting tree:



Rooting unrooted trees

In contrast to UPGMA, most tree reconstruction methods produce an **unrooted** tree.

- Indeed, determining the root of a tree using computational methods is very difficult.
- In practice, the question of rooting a tree is addressed by adding an **outgroup** to the set of taxa under consideration.
- This is a taxon that is more distantly related to all other taxa than any other of the taxa.
- The root is then assumed to be on the branch attaching the outgroup taxon to the rest of the tree.

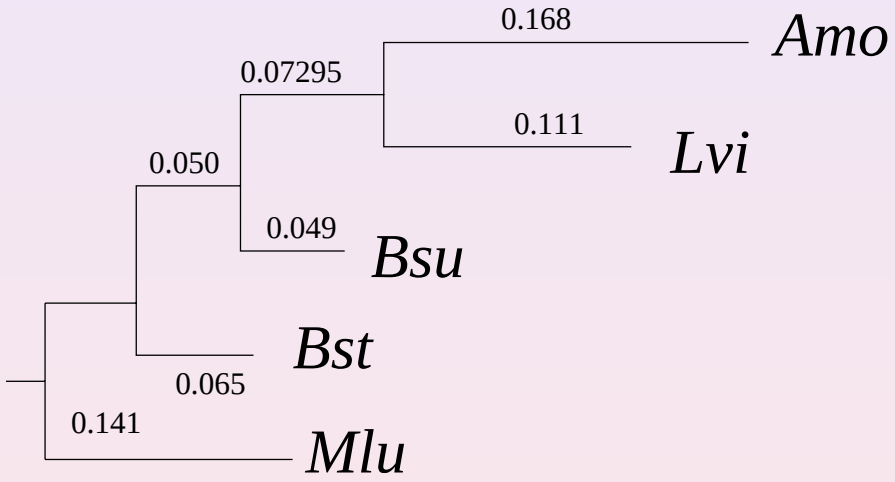
Rooting unrooted trees

In practice, selecting an appropriate outgroup can be difficult:

- if it is too similar to the other taxa, then it might be more related to some than to others.
- If it is too distant, then there might not be enough similarity to the other taxa to perform meaningful comparisons.

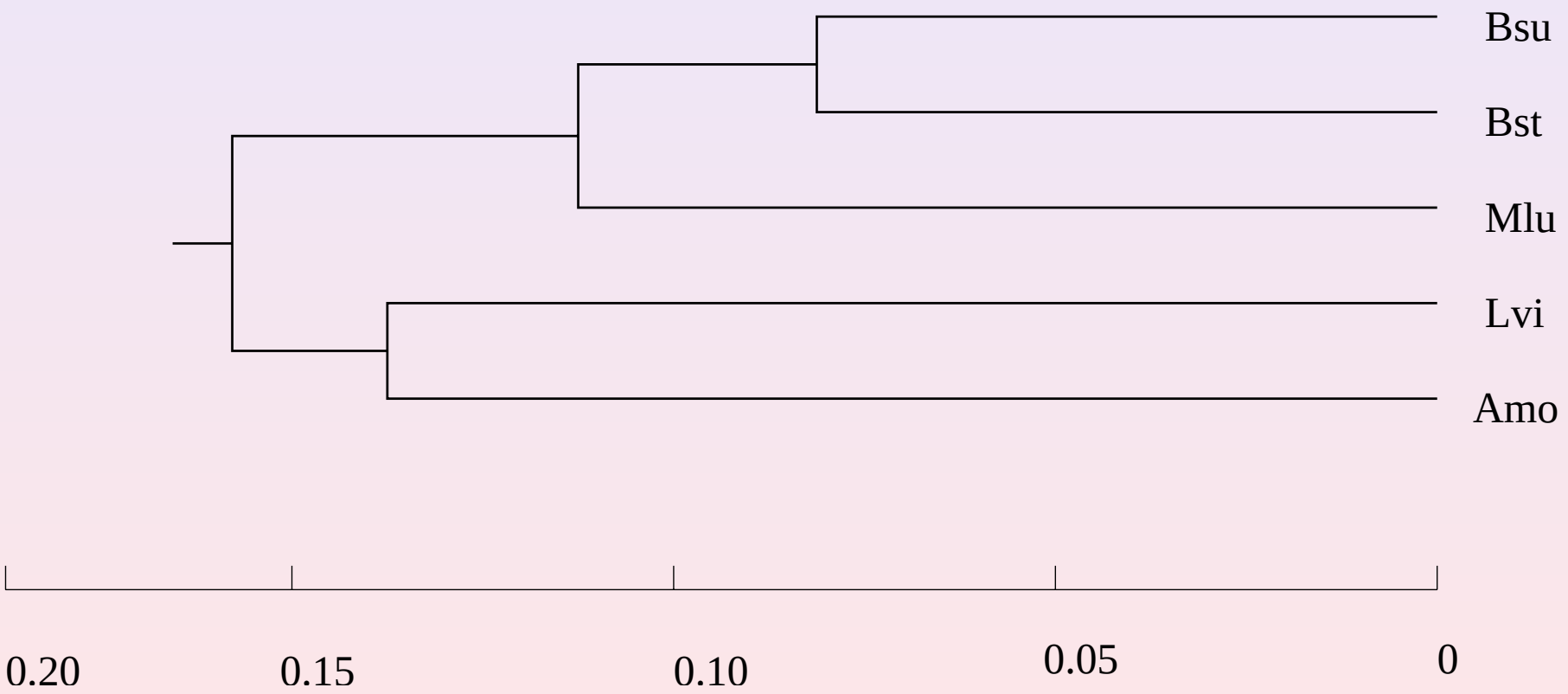
Rooting unrooted trees

In the above neighboring tree, Mlu is the outgroup, Hence, the rooted version of this tree looks something like this:



- This tree is believed to be closer to the correct tree than the one produced earlier using UPGMA.

Rooting unrooted trees



Rooting unrooted trees

- In particular, the UPGMA tree does not separate the outgroup from all other taxa by the root node.
- The reason why UPGMA produces an incorrect tree is that two of the sequences, those of **L.viridescens** (Lvi) and **A.modicum** (Amo) are very much more diverged than the others.

Part III

Models of evolution

Models of evolution

- Jukes-Cantor
- Jukes-Cantor-Model
- Distance Transformations
- Accounting for superimposed events
- More general transformations

Models of evolution

In phylogenetic analysis, a **model of evolution** is given by a rooted tree T, called the **model-**, **true-** or **generating** tree, together with a procedure for generating sequences along the model tree.

- Usually, the procedure must determine how to generate an initial sequence at the root of the tree and specify how to "evolve" sequences along the edges of the tree.
- This involves obtaining intermediate sequences for all internal nodes of the tree, and producing a set of aligned sequences A at the leaves of the tree.

Observed and expected Distances

- p-distance

Describes the proportion of different homologous sites, it is expressed as the **number of nucleotide differences per site**

- g-distance

Accounts for the effects of homoplasy (e.g. $A \rightarrow G \rightarrow C$ or $A \rightarrow G \rightarrow A$) Here, the p-distance underestimates the true genetic distance.

Nucleotide Substitutions as homogeneous Markov Process

- Nucleotide Substitutions can be generalized as a Markov Process
- The relative rates of change of each nucleotide is specified in a Q**-Matrix**
- Rate of change is independent from one site to another (**Markov Property**)
- Substitution rates do not change (**homogeneity**)
- The relative frequencies of A, C, G, T are at equilibrium (**stationary**)

The Jukes-Cantor model of evolution

T. Jukes and C. Cantor (1969)[5] introduced a very simple model of DNA sequence evolution.
Definition Let T be a rooted phylogenetic tree. The **Jukes-Cantor** model of evolution makes the following assumptions:

1. The possible states for each site are `A`, `C`, `G` and `T`.
2. The sites evolve identically and independently (i.i.d.) down the edges of the tree from the root at a fixed rate u.
3. With each edge $e \in E$ we associate a duration $t = t(e)$. The probabilities of change to each of the 3 remaining states are equal.

[5] T.H. Jukes and C.R. Cantor. (1969) Evolution of protein molecules. In Mammalian Protein Metabolism, H.N. Munro, ed., Academic Press, New York, NY, pp. 21-132.

Jukes-Cantor model

Under the Jukes-Cantor model, the evolutionary event that a nucleotide changes to any base occurs with a rate α.

- Let P and Q denote the base pair present at a given site before and after a given time period t.
- **What is the probability** $\mathrm{Prob}(Q = P \mid P, t)$ **that** P **equals** Q**?** (and hence become unchanged)

To answer this we make use of differential equations:

When to mutate ?

For $t = 0$ it is $\text{Prob}(P|t = 0) = 1$.

At the next time point $t = 1$ P either changes with rate α into one of the other 3 bases or it stays in the same state. Thus

$$\text{Prob}(Q = P|P, t = 1) = 1 - 3\alpha.$$

Then it follows

$$\text{Prob}(Q = P|P, t = 2) =$$

$$(1 - 3\alpha)\text{Prob}(Q = P|P, t = 1) + \alpha(1 - \text{Prob}(Q = P|P, t = 1))$$

Exponential Function

After some transformations of the former equation we finally get the *exponential function* as solution. The probability that no mutation occured in the time interval $(0, t)$.

$$P_0(t) = exp(-\mu t)$$

The Probability of *at least* one event is then given by

$$P_1(t) = 1 - exp(-\mu t)$$

Note that the *Poisson Distribution* contains the exponential function.

$$P_\lambda(X = k) = \frac{\lambda^k}{k!}\, \mathrm{e}^{-\lambda}$$

Probabilities of change from the **Q-Matrix**

A soon as the relative rates of change of each nucleotide are specified in a *Q*-**Matrix** it is possible to calculate the probability of change during evolutionary time t.

$$P(t) = exp(Qt)$$

Normally, detailed solutions require the use of matrix algebra, in case of Jukes-Cantor the result is straightforward:

Jukes-Cantor model

For $\mathrm{Prob}(P = P|P, t)$ we get the solution

$$\mathrm{Prob}_{ii}(t) = \frac{1}{4} + \frac{3}{4}e^{-\frac{4}{3}\alpha t}$$

while for $\mathrm{Prob}(P \neq Q|P, t)$ we get the solution

$$\mathrm{Prob}_{ij}(t) = \frac{1}{4} - \frac{1}{4}e^{-\frac{4}{3}\alpha t}$$

For $t \to \infty$ $\mathrm{Prob}(P|t)$ tends against $1/4$. Ie., under the Jukes-Cantor model in equilibrium the probability of all four nucleotides is equal to $1/4$.

Interpretation of the Jukes-Cantor model

Think about: how do we "evolve" a sequence down an edge e under the Jukes-Cantor model ?

- The evolutionary event ($C_3 =$) *nucleotide change to one of the other three bases* ocurs at a fixed rate u
- The event ($C_4 =$) *nucleotide change to any base* is taken to be $\frac{4}{3}u = 4\frac{u}{3}$
- The **probability** that event C_4 **does not** occur within time t is: $e^{-\frac{4}{3}ut}$ (Poisson distribution with $k = 0$)
- this leads also to a probablity of $\frac{1}{4}$ to end in any particular base

Jukes-Cantor model

Let P and Q denote the base pair present at a given site before and after a given time period t. The probability that an observable substitution is detected (all changes from $P \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ to base Q within time t is:

$$\text{Prob}(Q \neq P \mid P, t) = \frac{1}{4}\left(1 - e^{-\frac{4}{3}ut}\right).$$

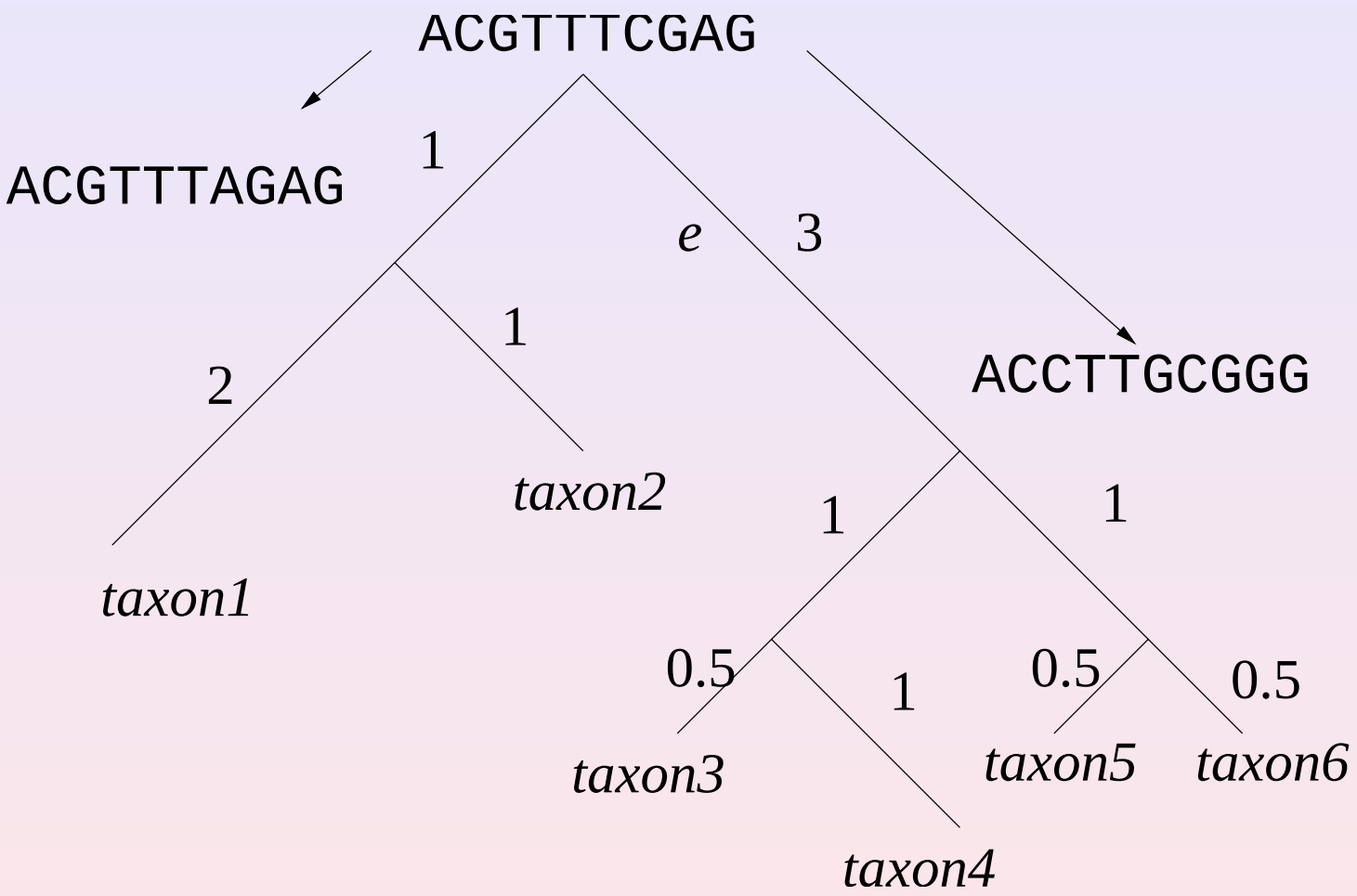
However, there are three additional nucleotides, such that we have to sum over three such quantities to yield the **probability-of-change formula** for the probability of an **observable** change occurring at any given site in time t:

$$\text{Prob}(\text{change} \mid t) = \frac{3}{4}\left(1 - e^{-\frac{4}{3}ut}\right).$$

Jukes-Cantor model

We can use this model to generate artificial sequences along a model tree.

- E.g., set the sequence length to 10 and the mutation rate to $u = 0.1$.
- Initially, the root node is assigned a random sequence.
- Then the sequences are evolved down the tree, at each edge using the probability-of-change formula to decide whether a given base is to change:



The probability of change along edge e is
$0.75(1 - e^{-\frac{4}{3} \times 0.1 \times 3}) = 0.75(1 - e^{-0.4}) = 0.247.$

The Jukes-Cantor distance transformation

- Given sequences generated under the Jukes-Cantor model of evolution.
- We would like to calculate the expected number of mutations for any given site on the path in a Jukes-Cantor tree between the leaves a_i and a_j.

Lemma

The maximum likelihood distance between a pair of sequences a_i, a_j (that is, the most likely ut to have generated the observed sequences) is given by the following formula:

$$JC(a_i, a_j) = -\frac{3}{4} \ln \left(1 - \frac{4}{3} Ham(a_i, a_j)\right).$$

This is called the **Jukes-Cantor distance transformation**.

Accounting for superimposed events

Hamming distances are suitable for inferring phylogenies between closely related species, given long sequences.

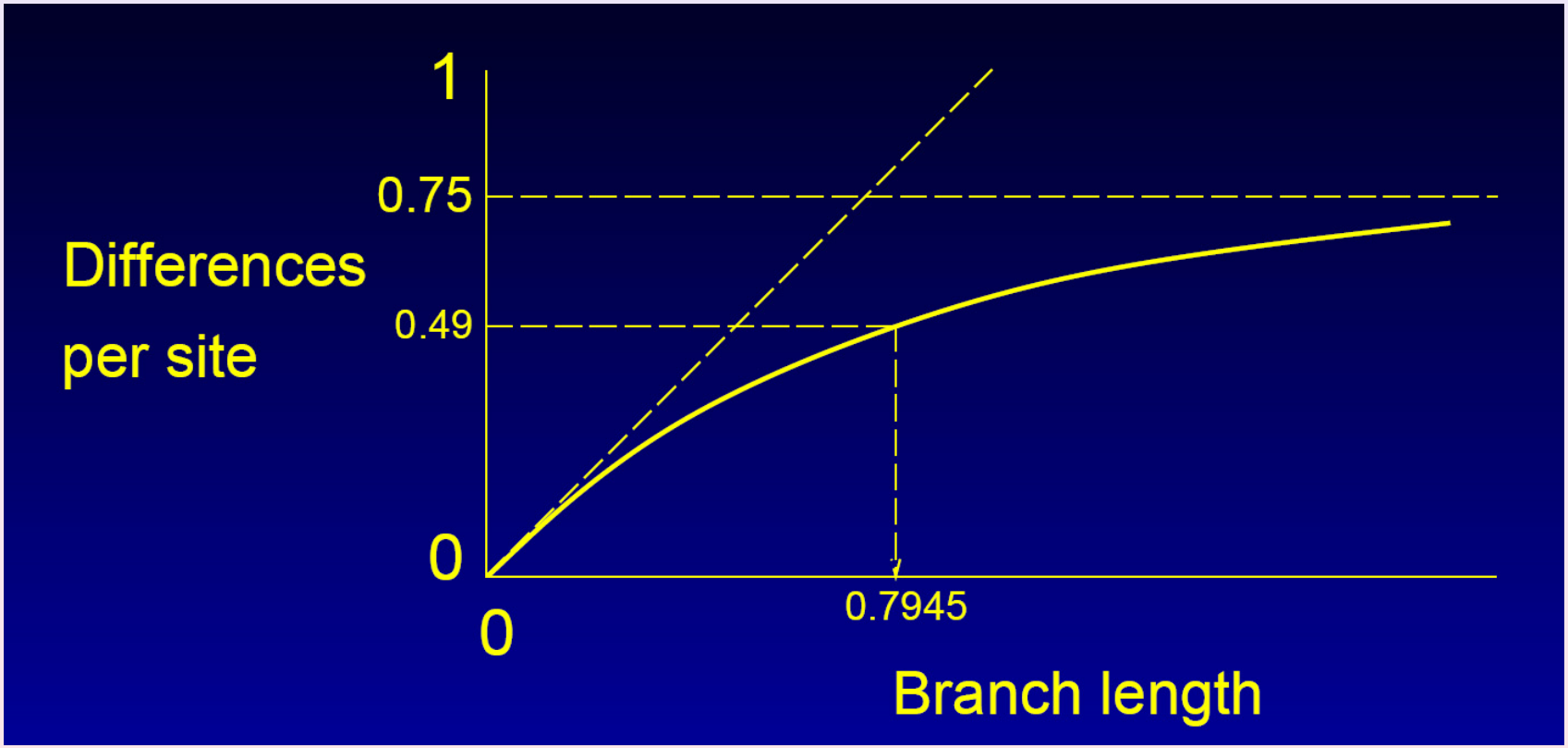
- For more distantly related sequences, the problem arises that mutation events will take place more than once at the same site.
- These **superimposed events** will not contribute to the Hamming distances and thus will go undetected.

Accounting for superimposed events

- Note that the expected Hamming distance between two random sequences is $\frac{3}{4}$. Any two sequences a_i, a_j for which $Ham(a_i, a_j) \geq \frac{3}{4}$ holds are called **saturated** w.r.t. each other.
- Note that the Jukes-Cantor transformation is undefined for any pair of saturated sequences.

In practice, when a saturated pair of taxa is encountered, their *JC* value is simply set to a large number which may be a fixed-factor times the largest value obtained between any two non-saturated taxa, for example.

genetic VS. observed distance



More general transformations

The Jukes-Cantor model assumes that all nucleotides occur with the same frequency.

- This assumption is relaxed under the **Felsenstein 81** model introduced by Joe Felsenstein (1981), which has the following distance transformation:

$$F81(a_i, a_j) = -B \ln(1 - Ham(a_i, a_j)B),$$

where $B = 1 - (\pi_{\mathtt{A}}^2 + \pi_{\mathtt{C}}^2 + \pi_{\mathtt{G}}^2 + \pi_{\mathtt{T}}^2)$ and π_Q is the frequency of base Q.

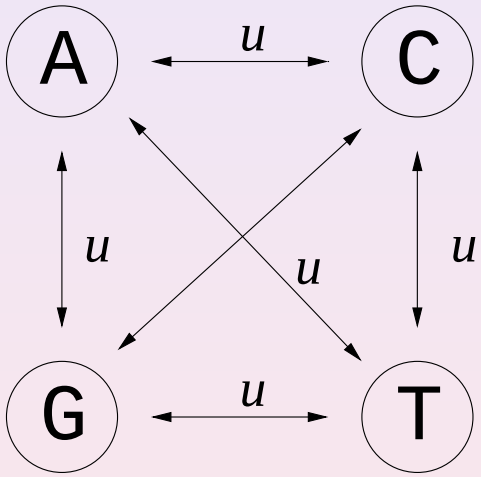
- The base frequency is obtained from the pair of sequences to be compared, or better, from the complete set of given sequences.

More general transformations

- Note that this **contains the Jukes-Cantor transformation as a special case** with equal base frequencies $\pi_{\mathtt{A}} = \pi_{\mathtt{C}} = \pi_{\mathtt{G}} = \pi_{\mathtt{T}} = 0.25$, thus $B = \frac{3}{4}$.
- Unlike the Jukes-Cantor transformation, this transformation can also be applied to protein sequences, setting $B = \frac{19}{20}$ if all amino acids are generated with the same frequency, and $B = \sum_{i=1}^{20} \pi_i^2$, in the proportional case.

More general transformations

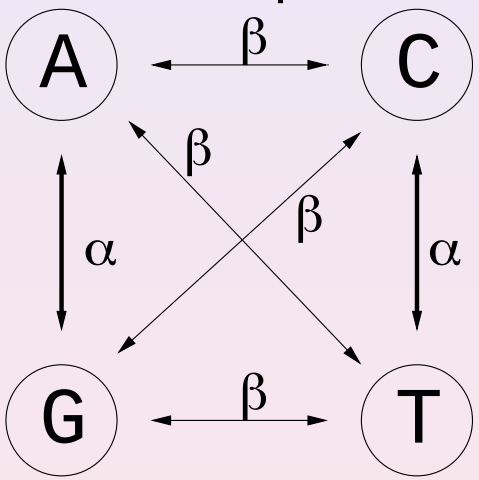
Both of these transformations assume that all changes of states are equally likely:



However, this is a very unrealistic assumption. For example, in DNA sequences, we observe many more **transitions**, which are purine-to-purine or pyrimidine-to-pyrimidine substitutions, than **transversions**, which change the type of the nucleotide.

More general transformations

We need to model two separate substitution rates α and β:



(Transitions: `A` $\leftrightarrow$ `G`, `C` $\leftrightarrow$ `T`, transversions: `A` $\leftrightarrow$ `T`, `G` $\leftrightarrow$ `T`, `A` $\leftrightarrow$ `C`, and `C` $\leftrightarrow$ `G`.)

More general transformations

Given equal base frequencies, but different proportions P and Q of transitions and transversions between a_i and a_j, the distance for the **Kimura 2 parameter** model is computed as:

$$K2P(a_i, a_j) = \frac{1}{2} \ln\left(\frac{1}{1-2P-Q}\right) + \frac{1}{4} \ln\left(\frac{1}{1-2Q}\right).$$

If we drop the assumption of equal base frequencies, then we obtain the **Felsenstein 84** transformation:

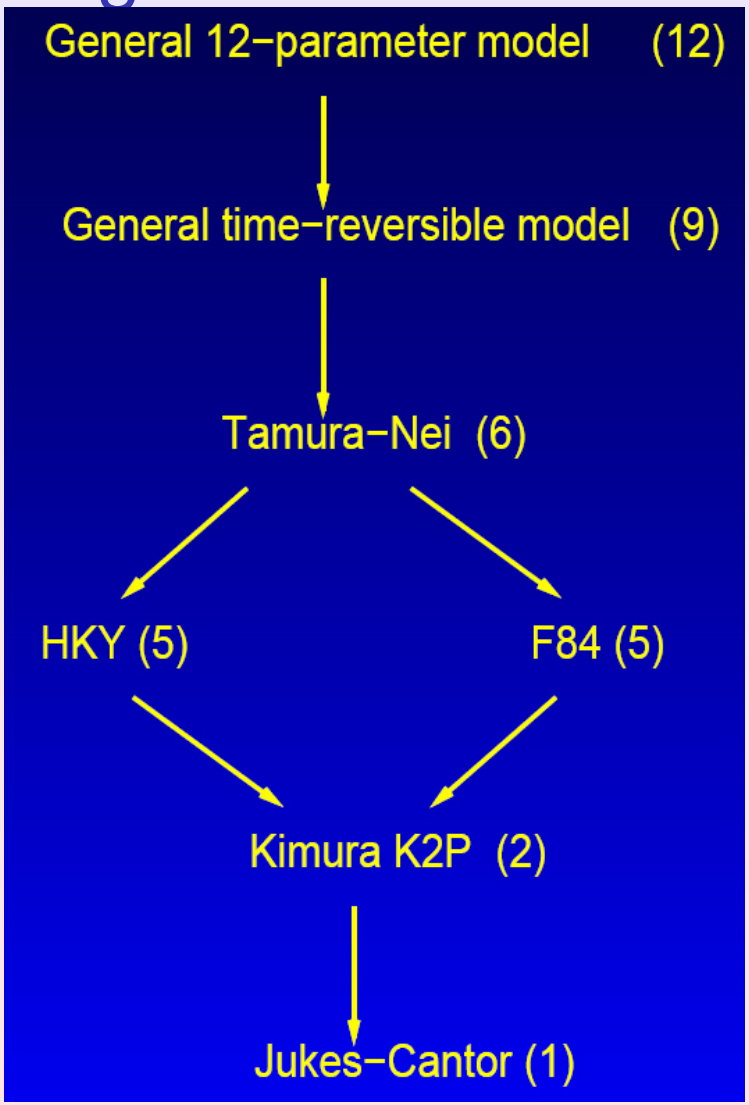
$$F84(a_i, a_j) = -2A \ln\left(1 - \frac{P}{2A} - \frac{(A-B)Q}{2AC}\right) + 2(A-B-C) \ln\left(1 - \frac{Q}{2C}\right),$$

where $A = \pi_C\pi_T/\pi_Y + \pi_A\pi_G/\pi_R$, $B = \pi_C\pi_T + \pi_A\pi_G$, and $C = \pi_R\pi_Y$ with $\pi_Y = \pi_C + \pi_T$, $\pi_R = \pi_A + \pi_G$.

More general transformations

Even more general models exist, but we will skip them. The following figure summarizes the complete overview of the time reversible models:

More general transformations



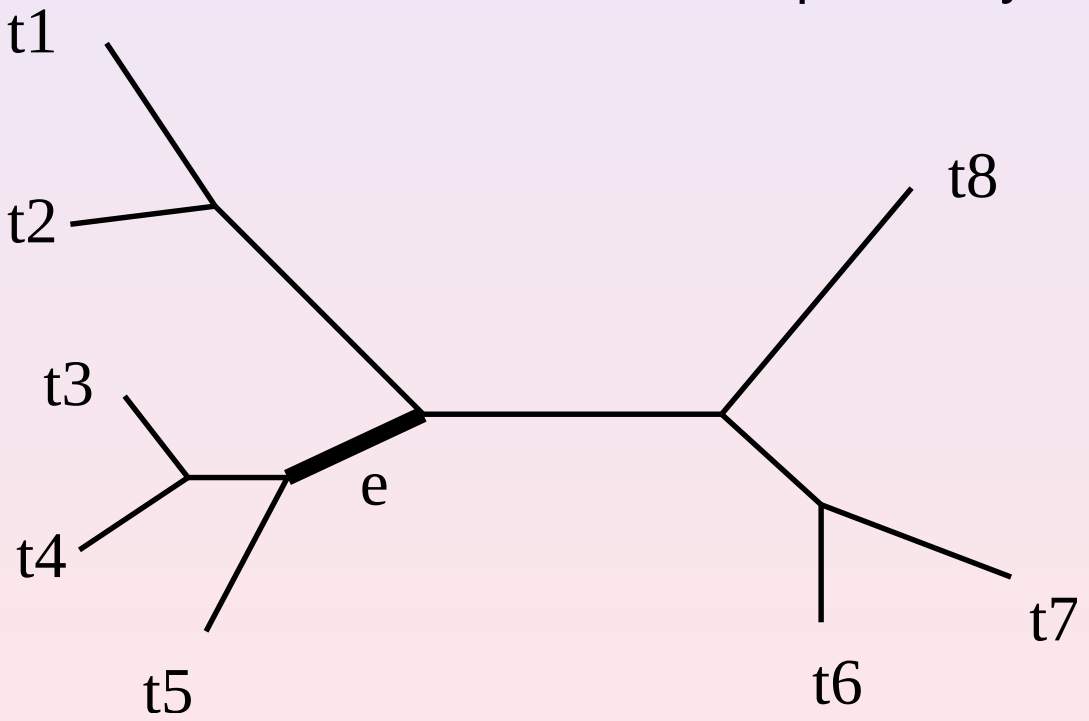
Part IV

Trees and splits

Trees and splits

Any edge e of T defines a **split** $S = \{A, \bar{A}\}$ of X, that is, a partitioning of X into two non-empty sets A and $\bar{A}$, consisting of all taxa on the one side and other side of e, respectively.

For example:



Here, $A = \{t_3, t_4, t_5\}$ and $\bar{A} = \{t_1, t_2, t_6, t_7, t_8\}$.

Trees and splits

- We will use $\Sigma(T)$ to denote the **split encoding of** T, i.e. the set of all splits obtained from T.
- If $x \in X$ and $S \in \Sigma$, then we use $S(x)$ or $\bar{S}(x)$ to denote the split part that contains x, or doesn't contain x, respectively.
- We define the **size** of a split $S = \{A, \bar{A}\}$ as $\text{size}(S) = \min(|A|, |\bar{A}|)$.
- A split of size 1 is called a **trivial** split.

Ideally, each edge of the tree separates a monophyletic group from the rest and this is reflected by the corresponding split.

Compatible splits

Given a set of taxa X. Let Σ be a set of splits of X. Two splits $S_1 = \{A_1, \bar{A}_1\}$ and $S_2 = \{A_2, \bar{A}_2\}$ are called **compatible**, if one of the four following intersections

$$A_1 \cap A_2, \quad A_1 \cap \bar{A}_2, \quad \bar{A}_1 \cap A_2, \quad \text{or} \quad \bar{A}_1 \cap \bar{A}_2,$$

is empty.

Compatible splits

A set Σ of splits of X is called **compatible**, if every pair of splits in Σ is compatible.

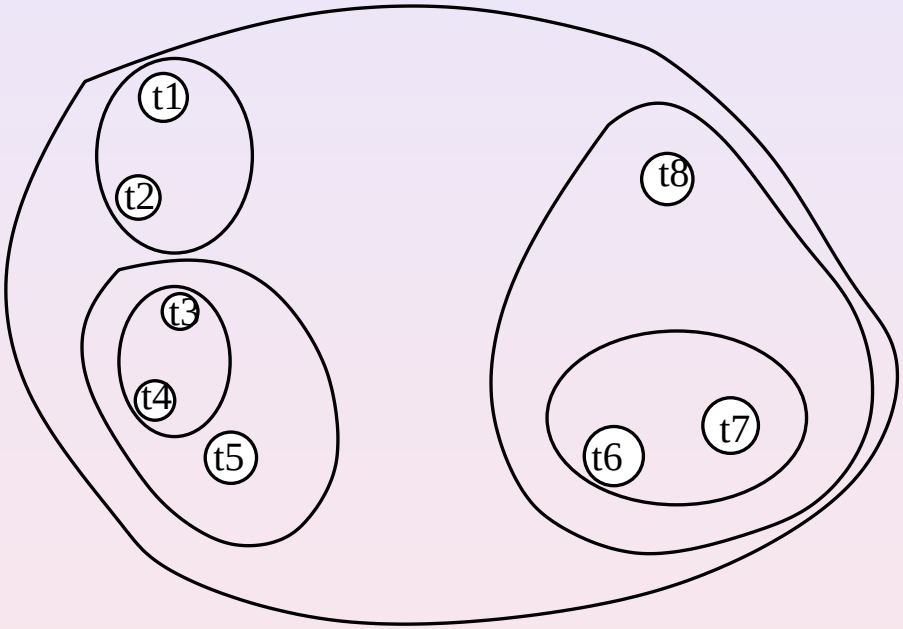
Example

- Given the taxa set $X = \{a, b, c, d, e\}$.
- The splits $S_1 = \{\{a, b\}, \{c, d, e\}\}$, $S_2 = \{\{a, b, c\}, \{d, e\}\}$ and $S_3 = \{\{e\}, \{a, b, c, d\}\}$ are all compatible with each other.
- However, $S_4 = \{\{a, c\}, \{b, d, e\}\}$ is not compatible with the first one.
- Hence, the set $\Sigma = \{S_1, S_2, S_3\}$ is compatible, but $\Sigma' = \{S_1, S_2, S_3, S_4\}$ is not.

Compatible splits

- The compatibility condition states that any split S subdivides either the one side, or the other side, of any other split S', but not both sides.
- Hence, any set of compatible splits can be drawn as follows, without crossing lines:

Compatible splits

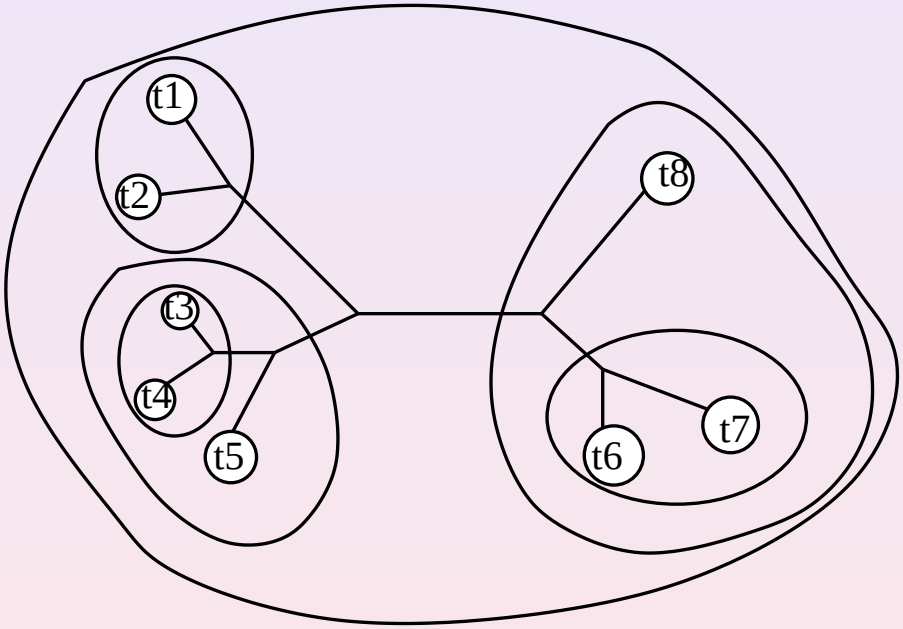


This figure also shows the relationship between compatible splits and a **hierarchical clustering**[6]

[6] A **hierarchical clustering** of a set X is a system $\mathcal{H}$ of subsets of X such that $\bigcup_{A \in \mathcal{H}} A = X$ and $A, B \in \mathcal{H} \Rightarrow$ either $A \cap B = \emptyset$, or $A \subset B$ or $B \subset A$.

Compatible splits and trees

Any compatible set of splits Σ gives rise to a phylogenetic tree T, for example:



Note: To obtain a one-to-one correspondence between trees and compatible split systems, we now use the relaxed definition of a phylogenetic tree that allows any leaf to carry more than one label and also allows labeling of internal nodes, too.

Compatible splits and trees

In summary:

Theorem

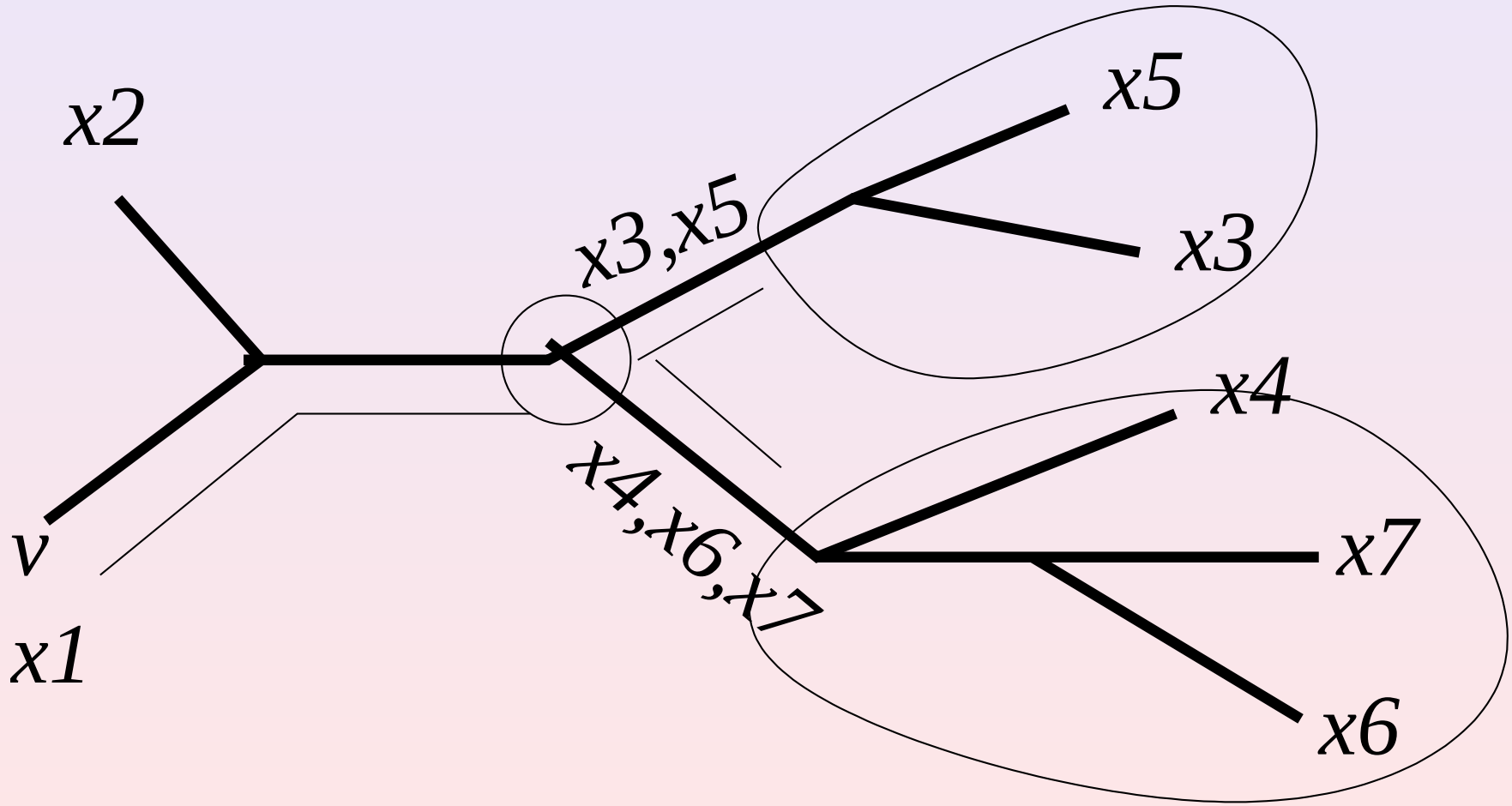
A set of splits Σ is compatible, iff there exists a phylogenetic tree T such that $\Sigma = \Sigma(T)$.

Splits from a tree

Given an unrooted phylogenetic tree T on $X = \{x_1, \ldots, x_n\}$. We can easily compute the set $\Sigma(T)$ in $O(n)$, where n is the number of taxa,

- by starting at the leaf labeled x_1,
- visit all nodes in a post-order traversal,
- for each edge maintaining and reporting the set of leaves reached after crossing the edge.

Splits from a tree



Splits from a tree

The following algorithm recursively visits all nodes and prints out a split after visiting all nodes reachable over a particular edge. Initially, the algorithm is called with $e = \mathit{null}$ and v equal to the node labeled x_1.

Algorithm TreeToSplits(e, v)
Input: A phylogenetic tree T on X
Output: The corresponding compatible splits $\Sigma(T)$
Returns: The set $\lambda(e)$ of all taxa separated from x_1 by e
begin Set $\lambda(e) = \emptyset$
for each edge $f \neq e$ adjacent to v **do**
    Let w be the opposite node adjacent to f
    if w is a leaf **then**
        add the label of w to $\lambda(e)$
    else
        call TreeToSplits(f, w) and add the returned labels to $\lambda(e)$
print the split $\{\lambda(e), \overline{\lambda(e)}\}$
return $\lambda(e)$

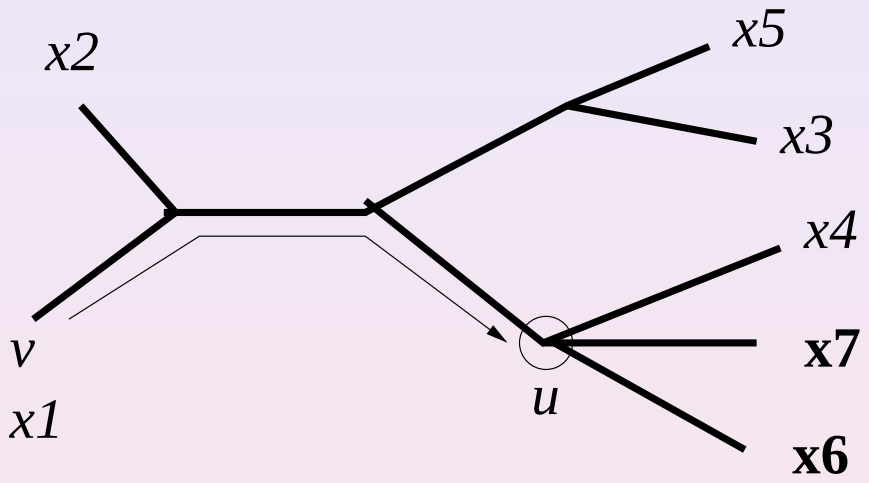
A tree from splits

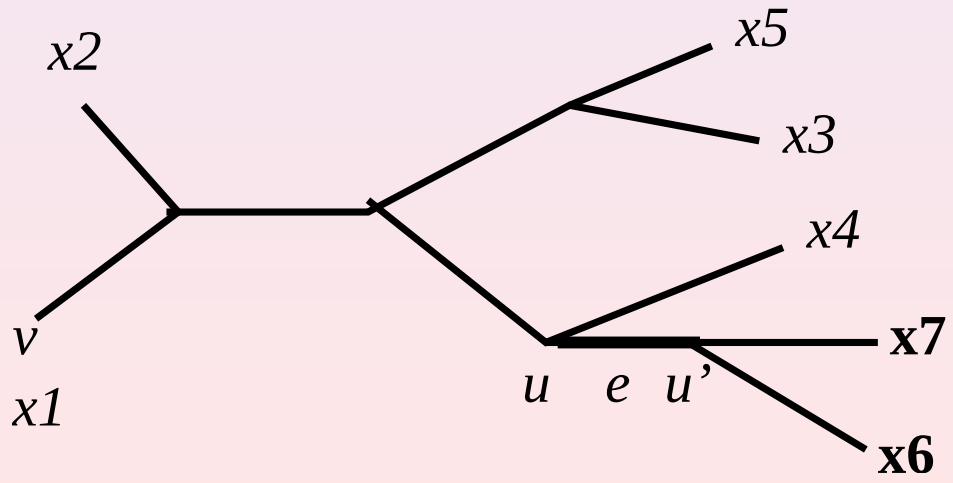
Given a compatible set of splits $\Sigma = \{S_1, S_2, \ldots, S_m\}$ of X.

- Assume we have already processed the first i splits and have obtained a tree T_i.
- To incorporate a new edge e to represent the next split $S_{i+1} \in \Sigma$,
- starting at the node v labeled x_1, we follow a path through the tree
- until we reach the node w at which e is to be inserted

A tree from splits

Example: insert split $\{x_1, x_2, x_3, x_4, x_5\}$ vs $\{x_6, x_7\}$:





A tree from splits

This node is found as follows:

- if there is only one edge leaving the current node u that **separates** x_1 from elements in $\bar{S}(x_1)$, then we follow this edge.
- If more than one separating edges exist, then $u = w$
- create a new edge e from u to a new node u' and all separating edges are moved from u to u'.

A tree from splits

Algorithm (Splits to tree)
Input: A set $\Sigma = \{S_1, \ldots, S_m\}$ of compatible splits of X,
(including all trivial ones)
Output: The corresponding phylogenetic tree $T = (V, E)$ on X
Initialization: Let T be a star tree with n leaves labeled $x_1, \ldots, x_n$
Orient all edges away from the node v labeled x_1
For $e \in E$, maintain the set $\tau(e)$ of all taxa separated from x_1 by e

begin
for each non-trivial split $S \in \Sigma$ **do**
    Set $u = v$
    while there is only edge e leaving u with $\tau(e) \cap \bar{S}(x_1) \neq \emptyset$ **do**
        Set u equal to the opposite node of e
    Let F be the set of all edges f leaving u with $\tau(f) \cap \bar{S}(x_1) \neq \emptyset$
    Create a new edge e from u to a new node u'
    Set $\tau(e) = \bigcup_{f \in F} \tau(f)$
    Make all edges in F leave from w' instead of w
end

A tree from splits

Lemma

The algorithm constructs the correct tree T in $O(n \log n)$ expected steps.

Counting Splits

There are **many** possible splits

$$2^n - 2n - 2$$

compared to the $2n - 3$ splits that are compatible.
For example:
When $n = 15$ there are 27 *compatible* splits within 16,383 splits.

Splitstree

How to select for compatible splits?

- the *Four-Point-Metric* condition tells whether distances within a quartet of species fit within a tree
- applaying the FPM condition to all quartets of the given taxa does not guarantee that **a unique** tree implies all valid quartets

How to select for compatible splits?

A good try:
Buneman index

- given the matrix D, Taxa X
- given split $S = \{A, B\}$ of X, some x, y in A and some u, v in B

put

$$\beta(xy|uv) = min\{d(x,u) + d(y,v), d(x,v) + d(y,u)\}$$

$$-(d(x,y) + d(u,v))$$

which defines the **Buneman index** $\beta_s = \frac{1}{2} min\beta(xy|uv)$

Remarkable Facts: Buneman Index

Lemma

The collection of splits, for which $\beta_S > 0$ holds, is compatible and, therefore, corresponds to a tree.

However, the selection is too strict and usually elects to discard too many splits because only the minimum value is choosen and only one value has to be negative to reject the split.

Isolation Index

- given the matrix D, Taxa X
- given split $S = \{A, B\}$ of X, some x, y in A and some u, v in B

put

$$\beta(xy|uv) = max\{d(x,u) + d(y,v), d(x,v) + d(y,u)\}$$

$$-(d(x,y) + d(u,v))$$

which defines the **Isolation index** $\alpha_s = \frac{1}{2} min\alpha(xy|uv)$

Consequences: Isolation index

- the Isolation Index corresponds to the weight of internal edges.
- **however**, the Splits S of the isolation index are not necesarily compatible
- Splits with positive isolation index are called $d - splits$

Lemma

if X has n elements, then the number of splits with positive isolation index is at most $n(n-1)/2$

Weak compatibility

all $d-$ *splits* are **weakly compatible**

Definition

a split is *weakly compatible*, if for every *three* splits

$$S = \{A, B\}, T = \{C, D\}, U = \{E, F\}$$

at least one of the intersections

$$A \cap C \cap E, A \cap D \cap F, B \cap C \cap F, B \cap D \cap E$$

is empty

Consequences: Weak compatibility

the FPM condition is also weakend by applying the weak compatibility

Lemma

when we look at the quartet ABCD we should have

$$max(D_{AB} + D_{DE}, D_{AE} + D_{DB}) > D_{AD} + D_{BE}$$

for the quartet to be compatible with the split $\{ACDF|BEG\}$.

- which means, that the distance sum of pairs of taxa *within* sets of the partition should not be the largets of the three sums

Drawing

The usual way to draw the networks is to utilize

- outer planar networks.

However, details of the drawing algorithm are beyond the scope of this lecture.

Part V

Analysis and Benchmarking

Comparison of two trees

Simulation studies

Algorithms vs optimality criteria

Comparison of two trees

Given two unrooted phylogenetic trees T_1 and T_2 on the same set of taxa X.

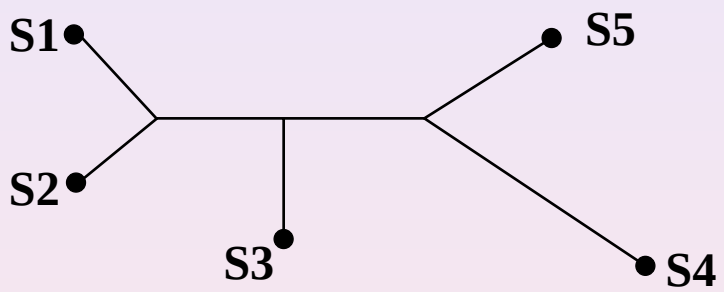
- We will call the first tree T_1 the **model** tree and T_2 the **reconstructed** tree.

How can we measure how similar their topologies are?

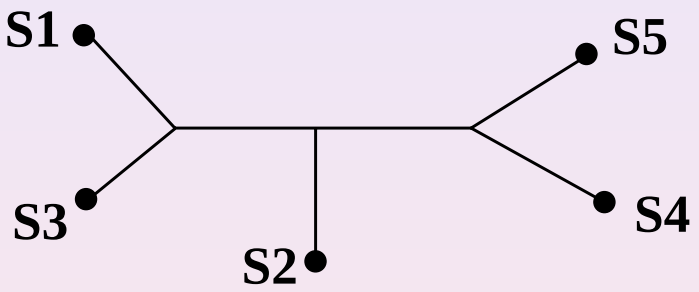
Comparison of two trees

- Let $\Sigma_1 = \Sigma(T_1)$ and $\Sigma_2 = \Sigma(T_2)$ be the split encodings of the two trees respectively.
- We define the set of all **false positives (FP)** as $\Sigma_2 \setminus \Sigma_1$.
- These are all splits contained in the estimated tree that are not present in the model tree.
- Similarly, we define the set of **false negatives (FN)** as $\Sigma_1 \setminus \Sigma_2$ as the set of all splits missing in the estimated tree, that are present in the model tree.

Comparison of two trees



Model Tree



Reconstructed Tree

In this example, there is one false positive $\{S_1, S_3\}$ vs $\{S_2, S_4, S_5\}$ and one false negative $\{S_1, S_2\}$ vs $\{S_3, S_4, S_5\}$.

Simulation studies

Simulation studies play an important role in bioinformatics. They are used to evaluate the performance of new algorithms on simulated datasets for which the correct answers are known.

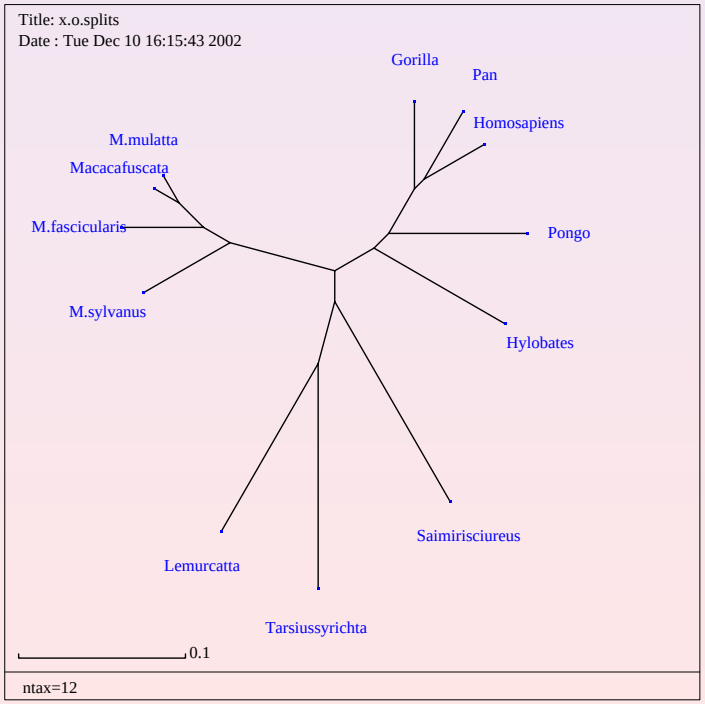
- For example, to evaluate the performance of a tree construction method, M, we can proceed as follows:

1. Select a model tree $T = (V, E, \omega)$ on X.
2. For a specified sequence length l and mutation rate u, generate a set of aligned sequences A along the tree T under the Jukes-Cantor model.
3. Apply the method M to A to obtain a tree $M(A)$.
4. Compute the number of false positives and false negatives.
5. Repeat many times for many different parameters and report the average performance of the method.

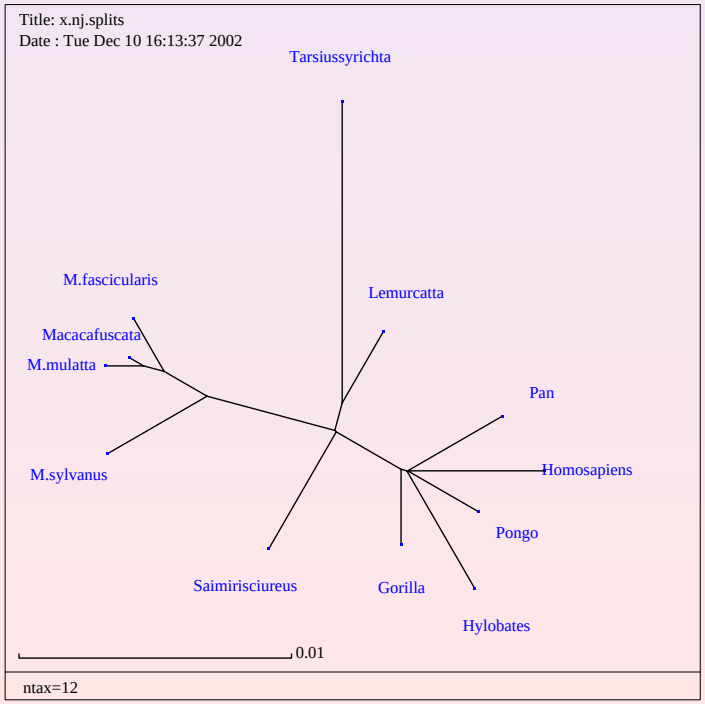
Simulation studies

Example Under the Jukes-Cantor model with $L = 1000$ and $u = 0.1$, sequences were generated on a model tree, then Hamming distances were computed, then **Neighbor-Joining** was applied:

Model tree



Neighbor-Joining tree

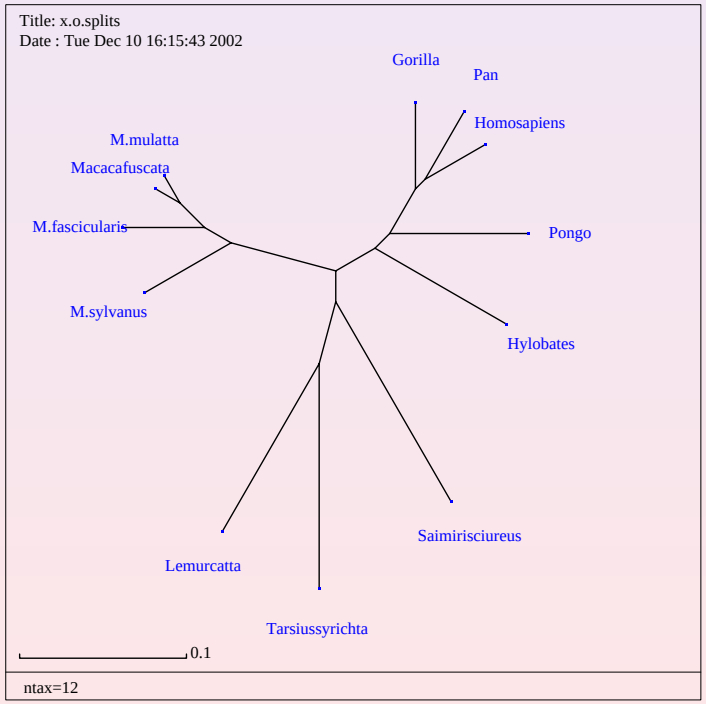


$FP = FN = 3$

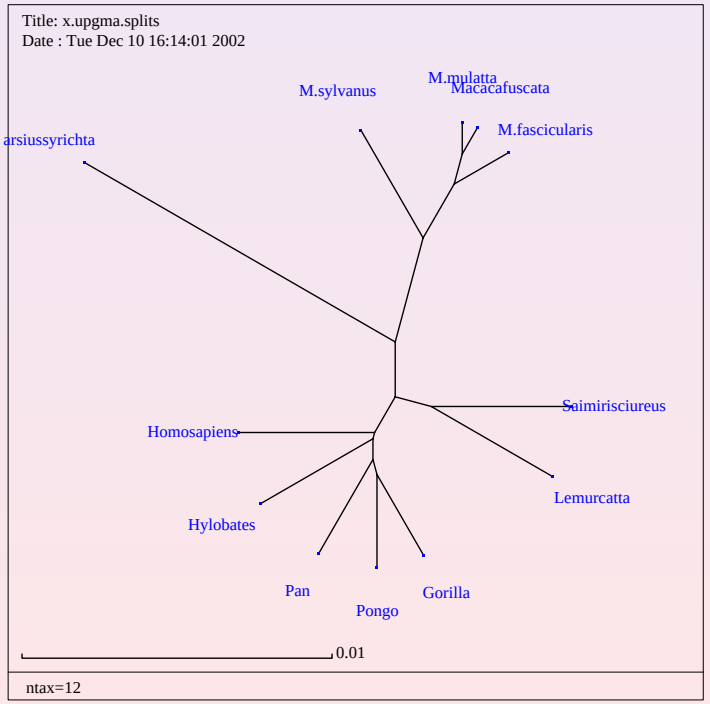
Simulation studies

Example Under the Jukes-Cantor model with $L = 1000$ and $u = 0.1$, sequences were generated on a model tree, then Hamming distances were computed, then **UPGMA** was applied:

Model tree

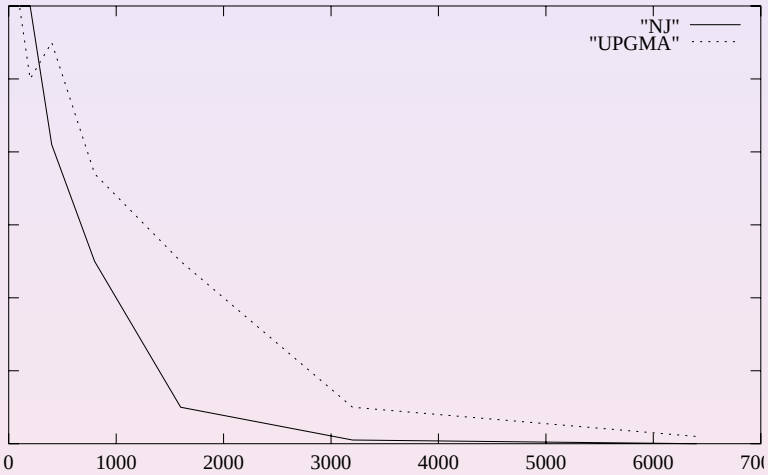


UPGMA tree

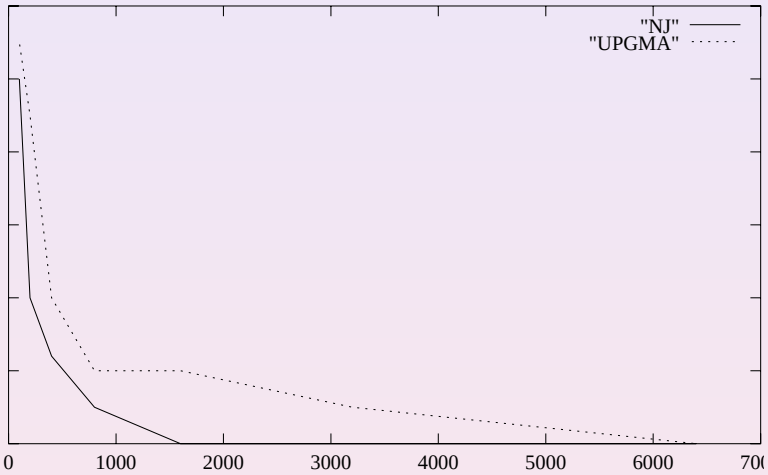


$$FP = FN = 5$$

Using the same tree and computations, here we plot[7] **the number of false positives** obtaining for $u = 0.1, 0.5$ and different sequence lengths L ranging from $100 - 6400$:



L vs FP, $u = 0.1$



L vs FP, $u = 0.4$

For this particular tree, **Neighbor-Joining displays generally a better performance than UPGMA**. However, to obtain significant results, many trees and many different parameters must be considered.

[7] Each point plotted was obtained as the average of five independent simulations

Simulation studies

SeqGen: Program for simulation studies
Given the tree in Newick format:
Generation of sequences using SeqGen[8]

[8] Rambaut, A. and Grassly, N. C. (1997) Seq-Gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. Comput. Appl. Biosci. 13: 235-238

Algorithms vs optimality criteria

- In phylogenetics, the goal is to **estimate** the true evolutionary tree that generated a set of extant taxa.
- Tree reconstruction methods attempt to accomplish this goal by **selecting** one of the many different possible topologies.

Algorithms vs optimality criteria

This selection process is performed in one of two ways:

1. The tree selected by an **algorithmic** method such as Neighbor-Joining or UPGMA is defined by the sequence of steps that make up the algorithm. This tree does not solve any explicitly formulated optimization problem.
2. The tree selected by an **optimization** method is a solution to an explicitly formulated optimization problem such as “maximum parsimony” or “maximum likelihood”. Here, algorithms play a secondary role as a means of obtaining or approximating a solution to the optimization problem.

Algorithms vs optimality

Distance methods are usually **algorithmically** defined and have polynomial run time. Sequence methods usually involve solving an **optimization** problem by finding an optimal tree with respect to some criterion and are usually NP-hard.

- **Algorithmic** methods combine the computation and definition of the preferred tree into a single statement.
- **Optimization** methods involve formulating and solving two problems:
  - computing the cost for a given tree T, and
  - searching through all trees to find a tree that minimizes the cost.

Part VI

Parsimony

wï¿½ga dem script

http://www-ab.informatik.uni-tuebingen.de/teaching/ws02/abi1/AIBiI-WS2002-3-Huson.pdf

Maximum parsimony

The maximum parsimony method is by far the most used sequence-based tree reconstruction method.

- In science, the principle of **maximum parsimony** is well known: always use the simplest, **most parsimonious** explanation of an observation, until new observations force one to adopt a more complex theory.
- In phylogenetic analysis, the **maximum parsimony problem** is to find a phylogenetic tree that explains a given set of aligned sequences using a minimum number of “evolutionary events”.

Assumptions for Maximum parsimony

- Changes in different sites is independent.
- Changes in different lineages is independent.
- The probability of a base substitution that changes the amino acid sequence is small over the lengths of time in a branch.
- The expected amounts of change in different branches do not vary by so much that two changes in a high-rate branch are more probable than one change in a low-rate branch.
- The expected amounts of change do not vary enough among sites that two changes in one site are more probable than one change in another.

The parsimony score of a tree

The **difference** between two sequences $x = (x_1, \ldots, x_L)$ and $y = (y_1, \ldots, y_L)$ is simply their non-normalized Hamming distance

$$\mathrm{diff}(x, y) = |\{k \mid x_k \neq y_k\}|.$$

The parsimony score of a tree

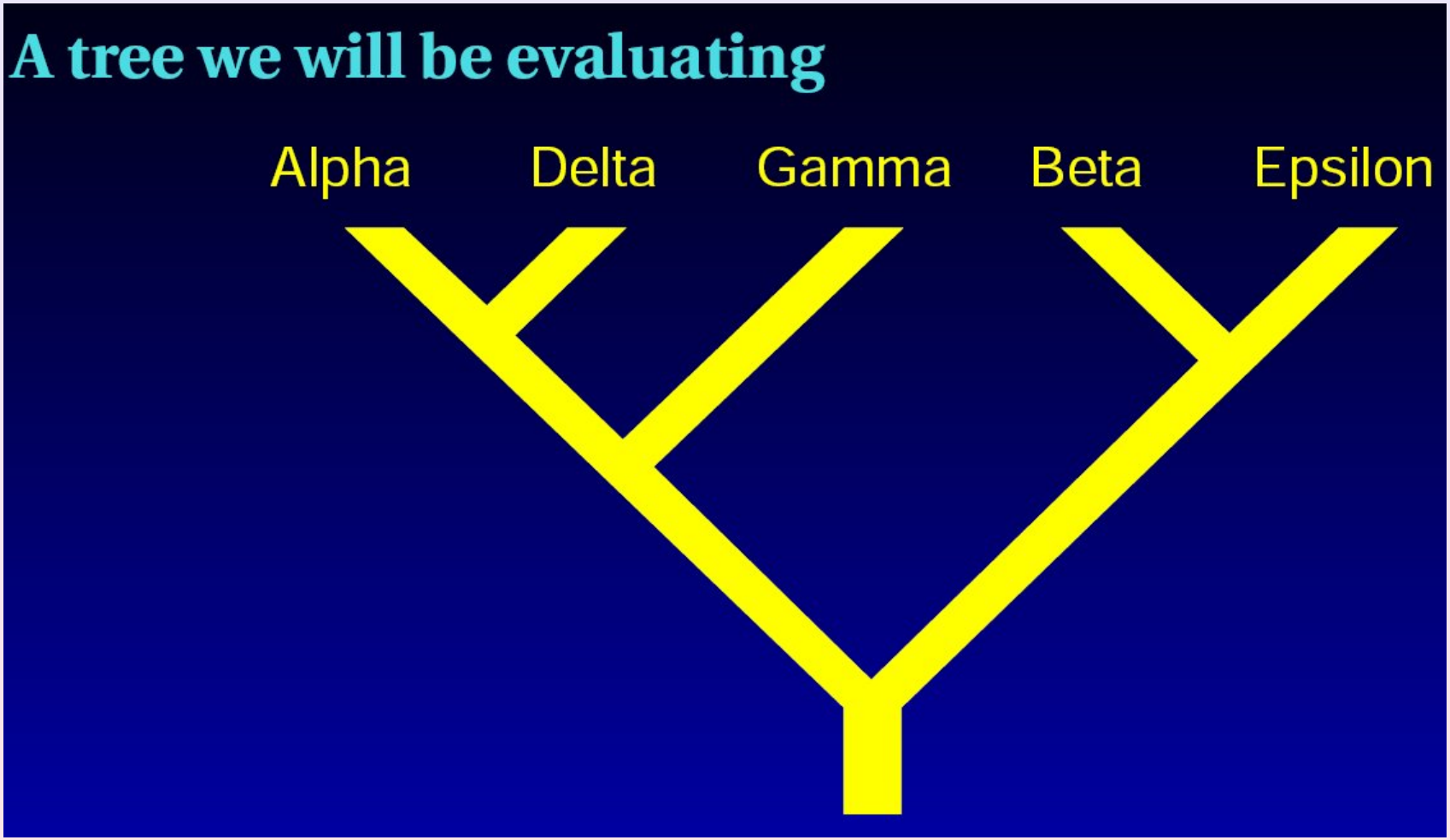
Given a multiple alignment of sequences $A = \{a_1, a_2, \ldots, a_n\}$ and a corresponding phylogenetic tree T, leaf-labeled by A.

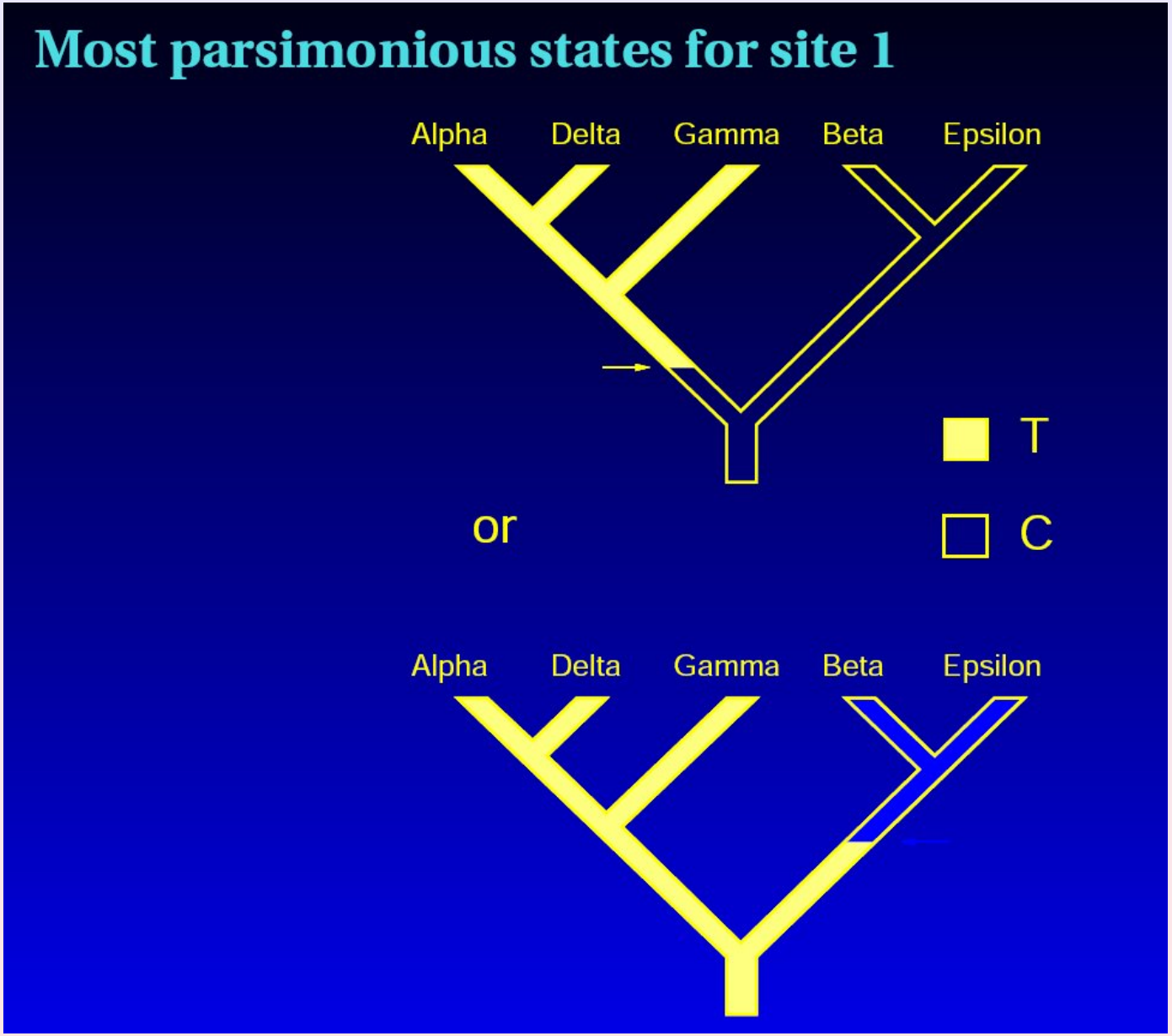
- If we assign a hypothetical ancestor sequence to every internal node in T,
- then we can obtain a score for T together with this assignment,
- by summing over all differences $\mathrm{diff}(x, y)$,
- where x and y are any two sequences labeling two nodes that are joined by an edge in T.

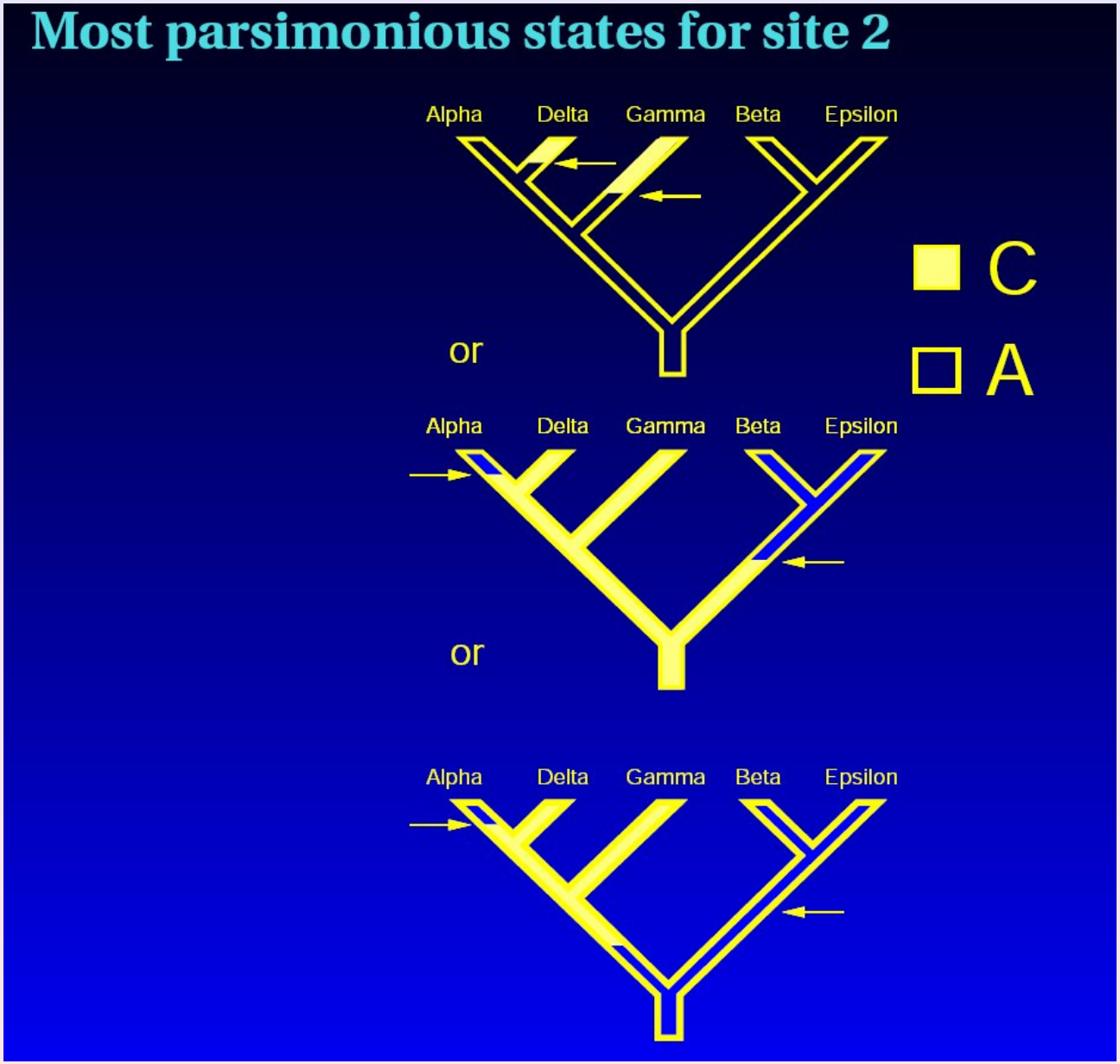
The minimum value obtainable in this way is called the **parsimony score** $PS(T, A)$ of T and A.

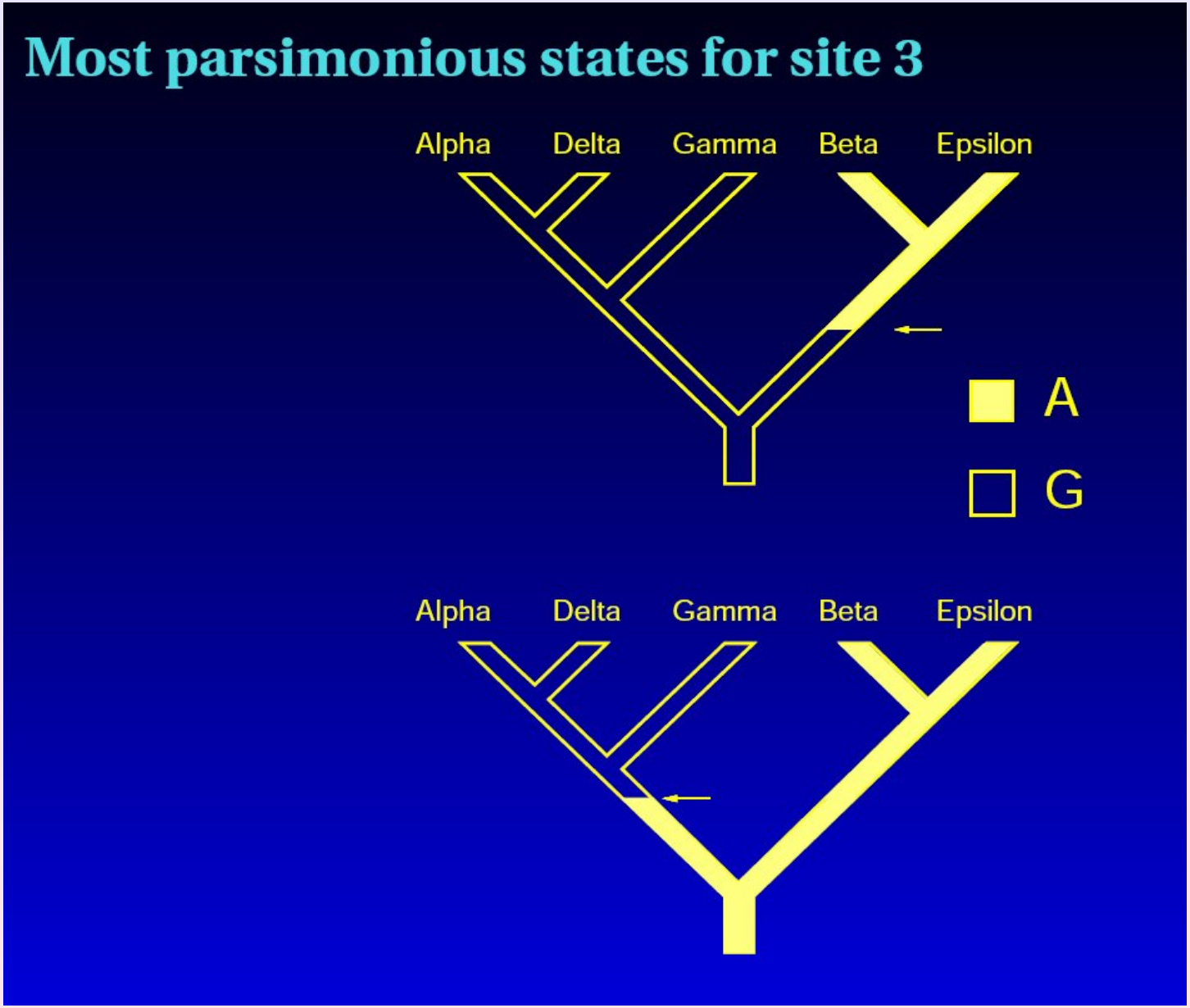
A simple data set with nucleotide sequences

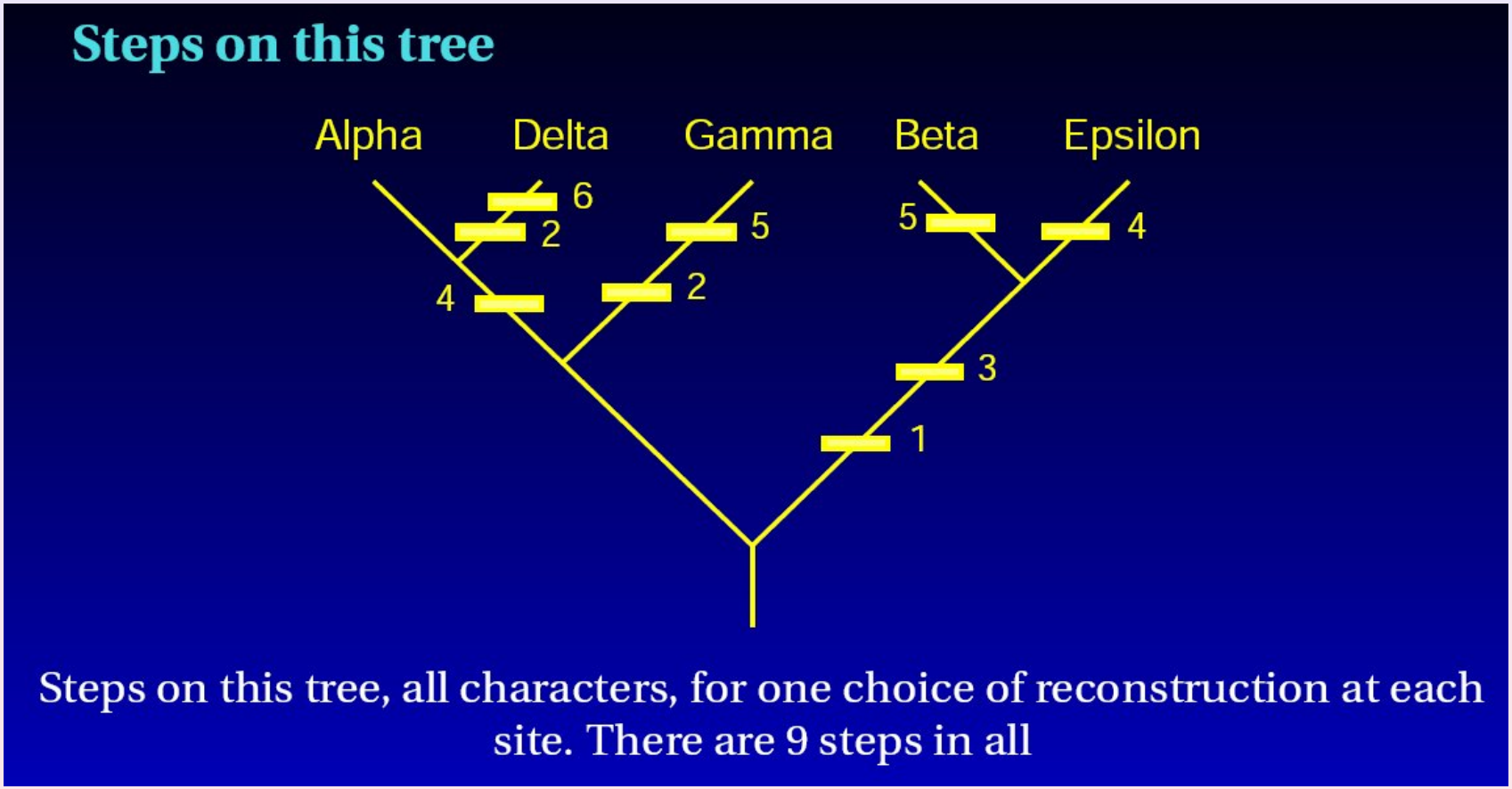
Species	Characters					
	1	2	3	4	5	6
Alpha	T	A	G	C	A	T
Beta	C	A	A	G	C	T
Gamma	T	C	G	G	C	T
Delta	T	C	G	C	A	A
Epsilon	C	A	A	C	A	T

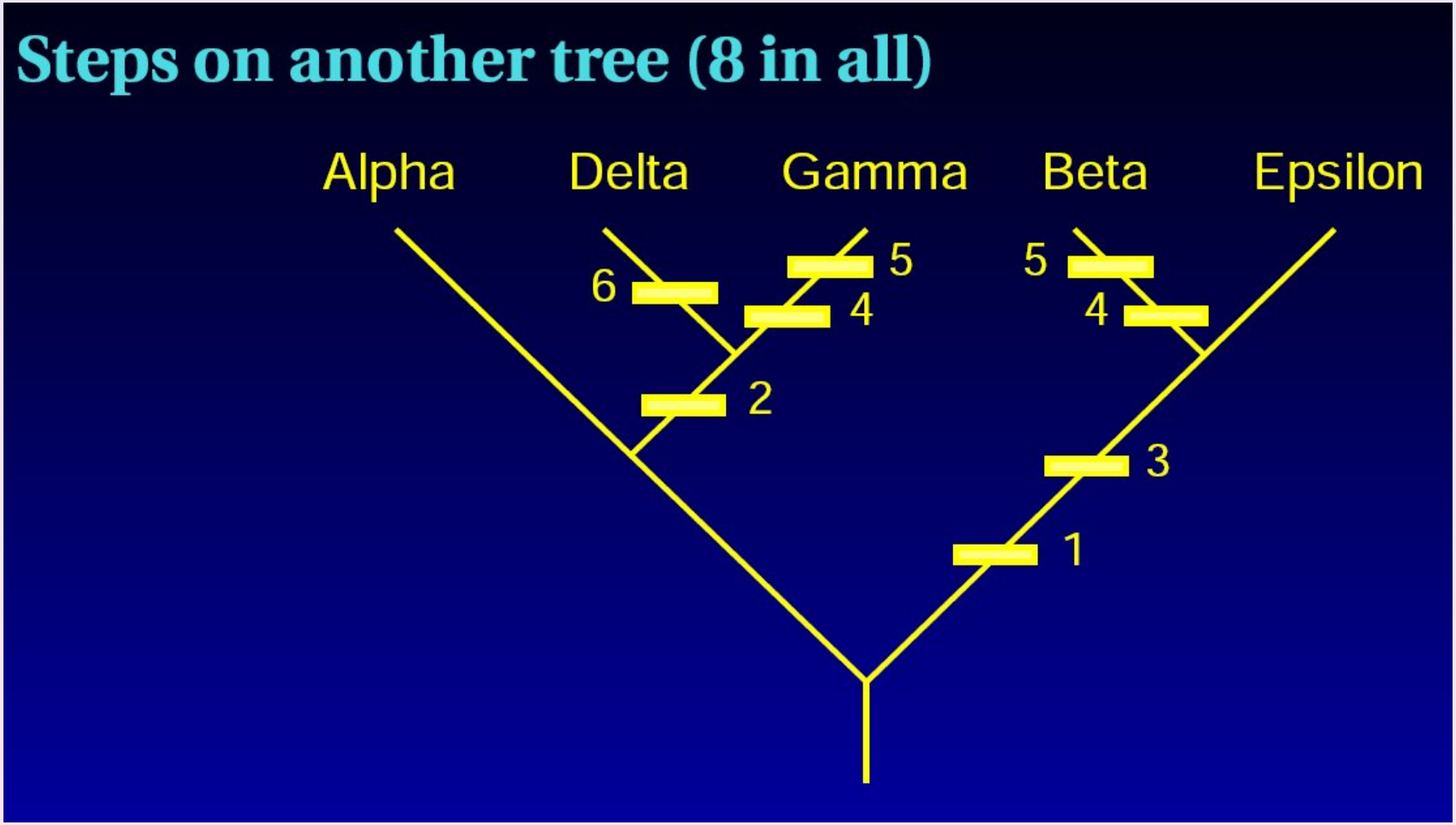

A tree we will be evaluating
Alpha
Delta
Gamma
Beta
Epsilon


Most parsimonious states for site 1
Alpha
Delta
Gamma
Beta
Epsilon
T
or
C
Alpha
Delta
Gamma
Beta
Epsilon


Most parsimonious states for site 2
Alpha
Delta
Gamma
Beta
Epsilon
C
A
or
Alpha
Delta
Gamma
Beta
Epsilon
or
Alpha
Delta
Gamma
Beta
Epsilon


Most parsimonious states for site 3
Alpha
Delta
Gamma
Beta
Epsilon
A
G
Alpha
Delta
Gamma
Beta
Epsilon

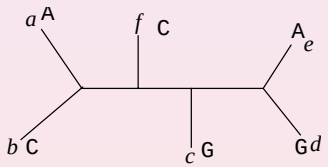

Steps on this tree
Alpha
Delta
Gamma
Beta
Epsilon
6
2
4
5
2
5
4
3
1
Steps on this tree, all characters, for one choice of reconstruction at each site. There are 9 steps in all

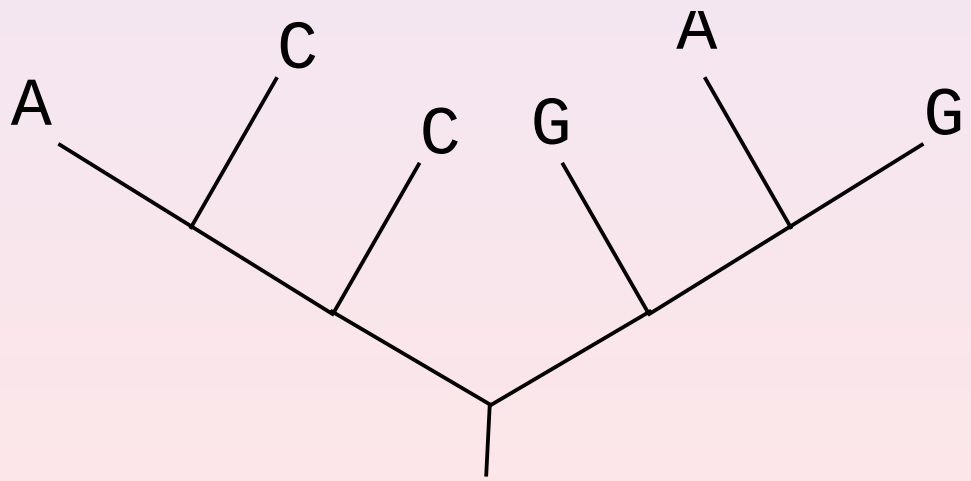

Steps on another tree (8 in all)
Alpha
Delta
Gamma
Beta
Epsilon
6
5
4
2
5
4
3
1

The small parsimony problem

The **small** parsimony problem is to compute the parsimony score for a given tree T. **Can it be solved efficiently?**

- As the parsimony score is obtained by summing over all columns, the columns are independent and so it suffices to discuss how to obtain an optimal assignment for one position:





unrooted tree

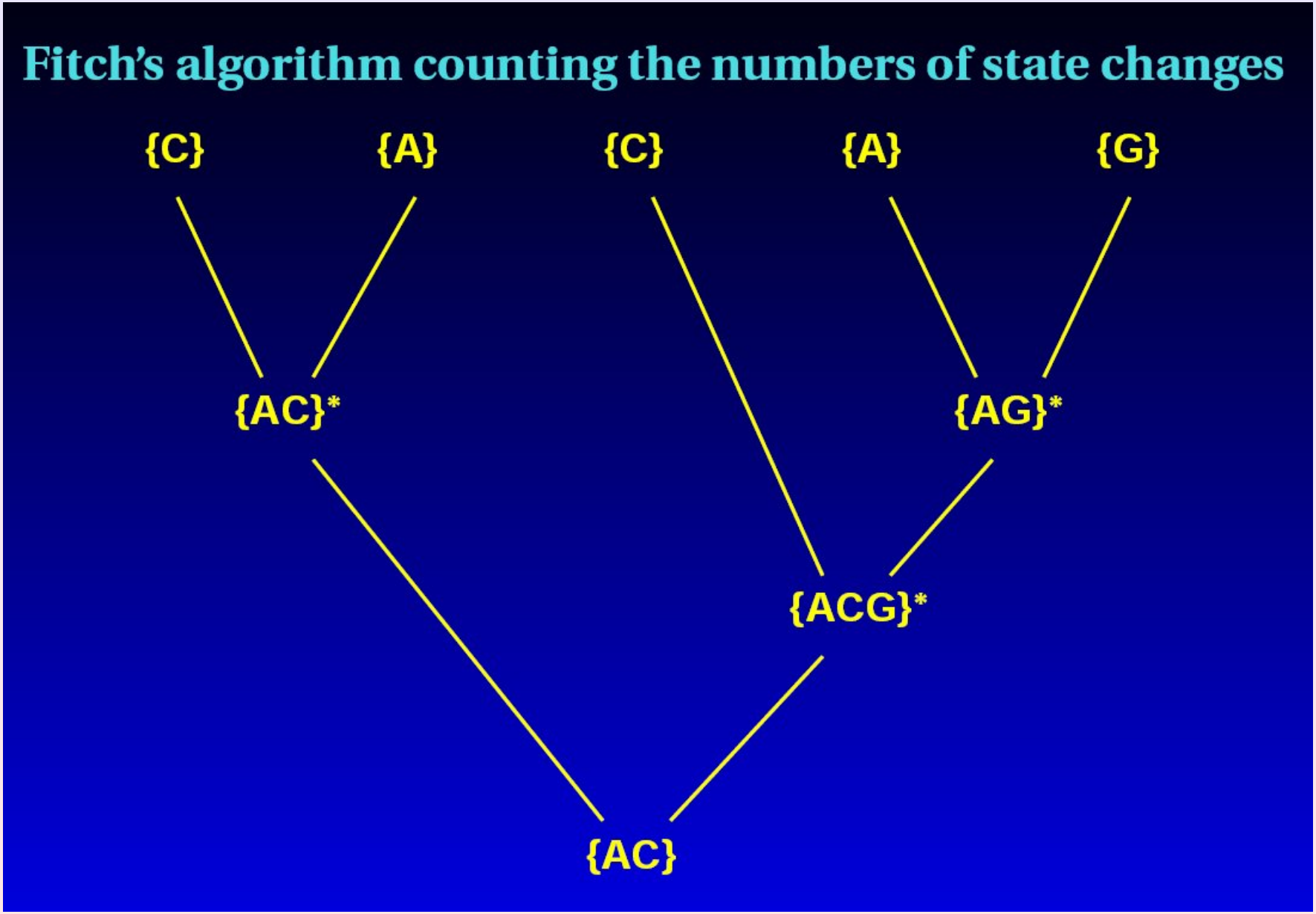
rooted tree

The Fitch algorithm

- The following algorithm computes the parsimony score for T and a fixed column c in the sequence alignment.
- It uses a dynamic programming algorithm and assigns sets of character assignments $S(v)$ to each node v in the tree.
- Initially it is called with $e = \mathit{null}$ and v the root node and $PS(T, c) = 0$.

The Fitch algorithm

Algorithm ParsimonyScore(e, v) (Walter Fitch 1971)
Input: A phylogenetic tree T and a character $c(v)$ for each leaf v
Output: The parsimony score $PS(T, c)$ for T and c
if v is a leaf node **then**
  Set $S(v) = \{c(v)\}$
else
  for each edge $f_1, f_2 \neq e$ adjacent to v **do**
    Let w_i be the opposite node of f_i
    Call ParsimonyScore(f_i, w_i) // to compute $S(w_i)$
  if $S(w_1) \cap S(w_2) \neq \emptyset$ **then**
    Set $S(v) = S(w_1) \cap S(w_2)$
  else
    Set $S(v) = S(w_1) \cup S(w_2)$ and $PS(T, c) = PS(T, c) + 1$
end


Fitch's algorithm counting the numbers of state changes
{C}
{A}
{C}
{A}
{G}
{AC}*
{AG}*
{ACG}*
{AC}

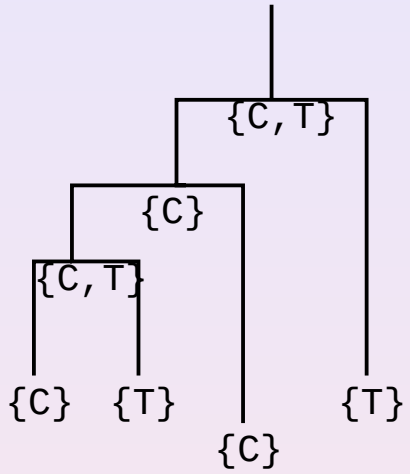
Traceback

The above algorithm computes the parsimony score. An optimal labeling of the internal nodes is obtained via traceback:

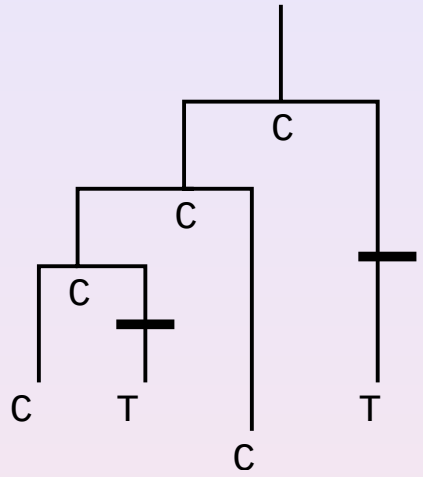
- starting at the root node r,
- we label r using any character in $S(r)$.
- Then, for each child w, we use the same letter, if it is contained in $S(w)$,
- otherwise we use any letter in $S(w)$ as label for w.
- We then visit the children von w etc.

Again, the algorithm requires $O(nL)$ steps, in total.

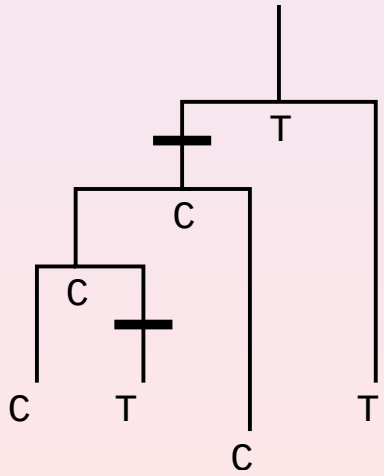
Example:



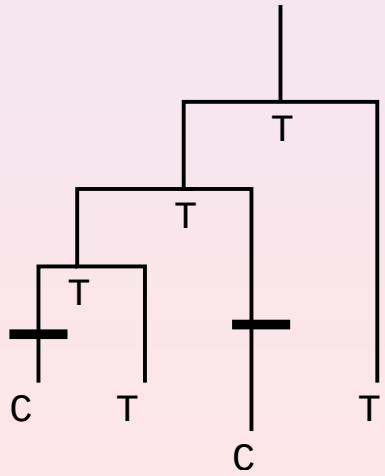
Fitch labeling



one traceback result



another traceback result



not obtainable by traceback

The Fitch algorithm on unrooted trees

Above, we formulated the Fitch algorithm for rooted trees. However, the minimum parsimony cost is independent of where the root is located in the tree T.

more Advanced: Sankoff-Algorithm

A dynamic programming algorithm for counting the smallest number of possible (weighted) state changes needed on a given tree

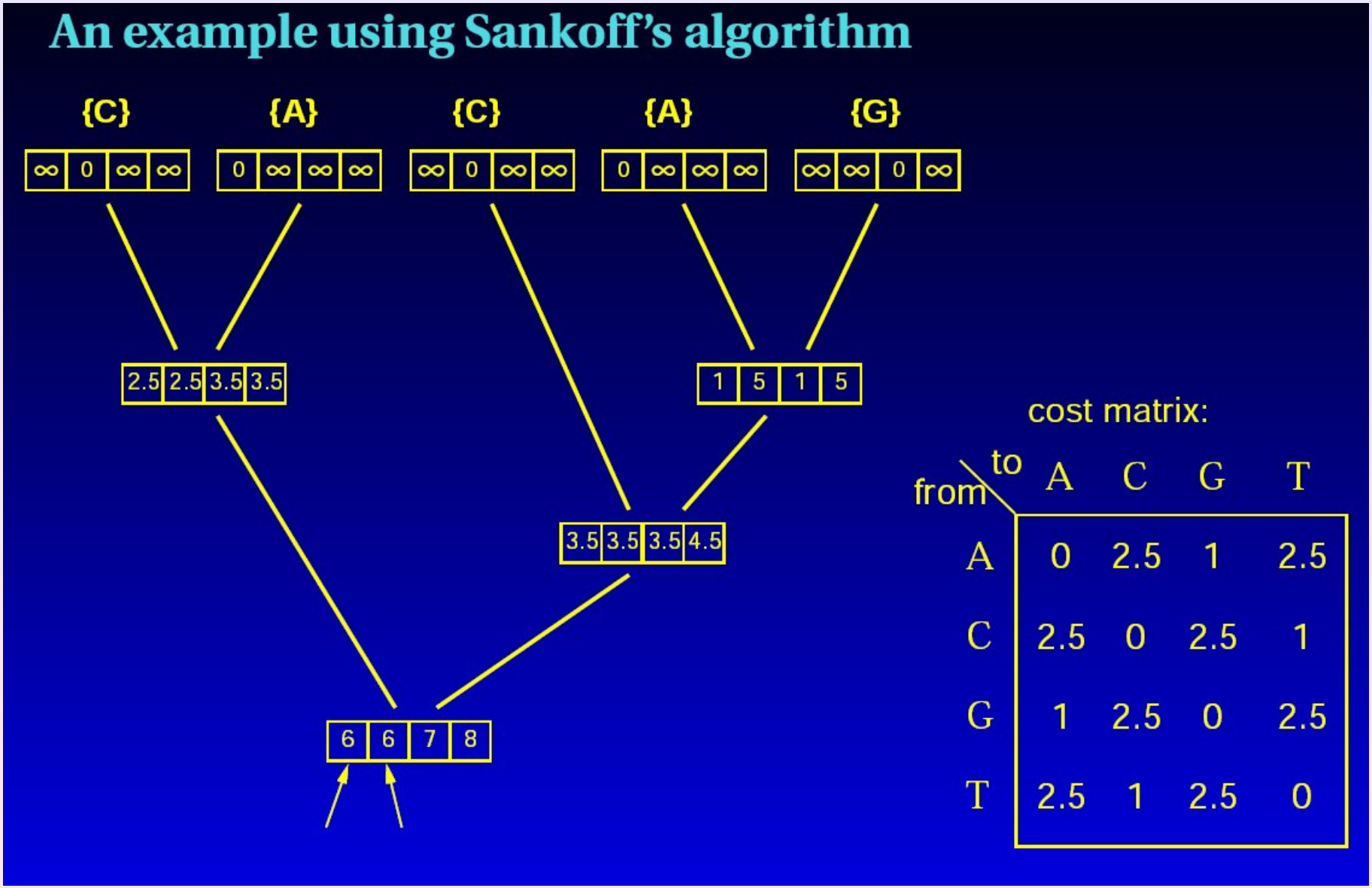
- Let Sj(i) be the smallest (weighted) number of steps needed to evolve the subtree at or above node j, given that node j is in state i. Suppose that cij is the cost of going from state i to state j.
- Initially, at tip (say) j

$$S_j(i) = \begin{cases} 0 & \text{if node } j \text{ has (or could have) state } i \\ \infty & \text{if node } j \text{ has any other state} \end{cases}$$

- Then proceeding down the tree (postorder tree traversal) for node a whose immediate descendants are l and r

$$S_a(i) = \min_j[c_{ij} + S_l(j)] + \min_k[c_{ik} + S_r(k)]$$

- The minimum number of (weighted) steps for the tree is found by computing at the bottom node (0) the $S_0(i)$ and taking the smallest of these.


An example using Sankoff's algorithm
{C}
{A}
{C}
{A}
{G}
∞ 0 ∞ ∞
0 ∞ ∞ ∞
∞ 0 ∞ ∞
0 ∞ ∞ ∞
∞ ∞ 0 ∞
2.5 2.5 3.5 3.5
1 5 1 5
3.5 3.5 3.5 4.5
6 6 7 8
cost matrix:
from \ to
A C G T
A 0 2.5 1 2.5
C 2.5 0 2.5 1
G 1 2.5 0 2.5
T 2.5 1 2.5 0

The large parsimony problem

Given a multiple alignment $A = \{a_1, \ldots, a_n\}$, its **parsimony score** is defined as

$$PS(A) = \min\{PS(T, A) \mid T \text{ is a phylogenetic tree on } A\}.$$

The **large parsimony problem** is to compute $PS(A)$.

- Potentially, we need to consider all $(n-5)!!$ possible trees.
- Unfortunately, in general this can't be avoided and the maximum parsimony problem is known to be NP-hard.
- Exhaustive enumeration of all possible tree topologies will only work for $n \leq 10$ or 11, say.

The large parsimony problem

Thus, we need more efficient strategies that either solve the problem exactly, such as the branch and bound technique, or return good approximations, such as heuristic searches[9].

[9] As with most biological problems, we are not only interested in the optimal solution, but would also like to know something about other near optimal solutions as well.

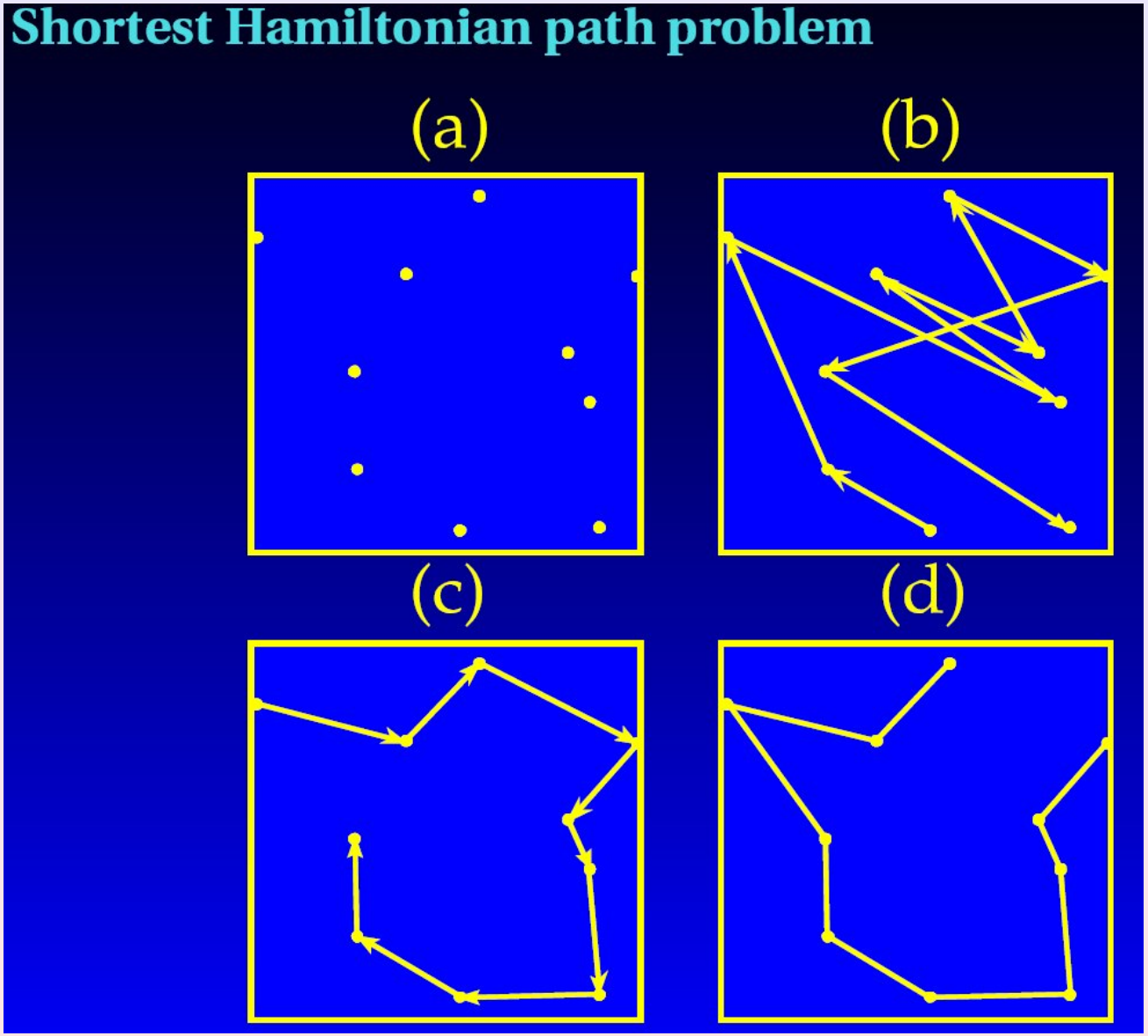
Part VII

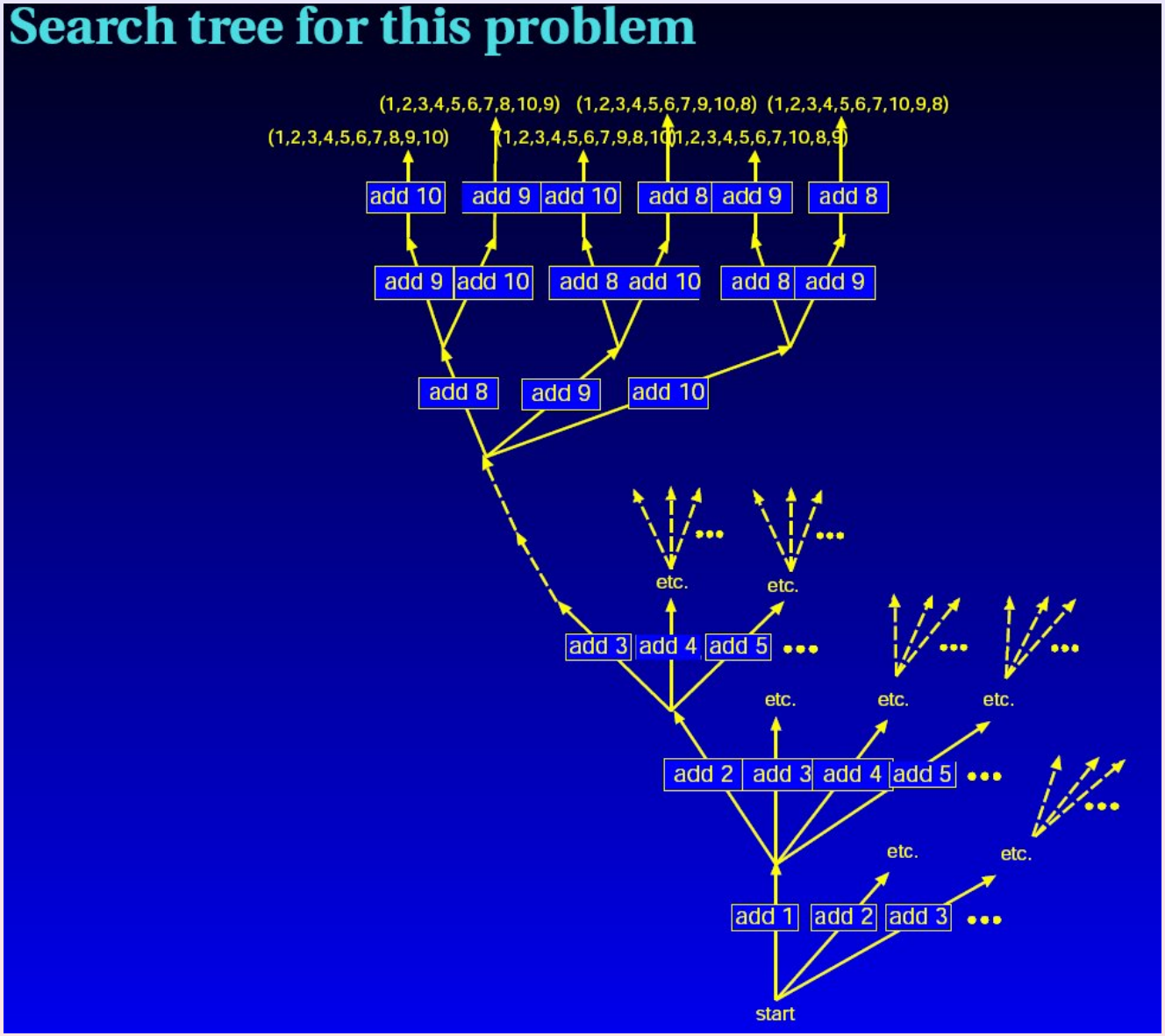
Tree Searching Methods

- Branch and bound
  - Shortest Hamiltonian Path Problem
  - But how to obtain an Branch and bound Algorithm for Phylogenetic Trees??
- The stepwise-additional heuristic
- The star-decomposition heuristic
- Branch swapping methods
  - Nearest-Neighbor-Interchange (NMI)
  - Subtree Pruning and Regrafting (SPR)
  - Tree bisection and Reattachment (TBR)
- Heuristic search
- Simulated annealing
- The Great Deluge method

Tree Searching Methods

- Exhaustive search (exact)
- Branch-and-bound search (exact)
- Heuristic search methods (approximate)
  - Stepwise addition
  - Branch swapping
  - Star decomposition


Shortest Hamiltonian path problem
(a)
(b)
(c)
(d)


Search tree for this problem
(1,2,3,4,5,6,7,8,10,9)
(1,2,3,4,5,6,7,9,10,8)
(1,2,3,4,5,6,7,10,9,8)
(1,2,3,4,5,6,7,8,9,10)
(1,2,3,4,5,6,7,9,8,10)
(1,2,3,4,5,6,7,10,8,9)
add 10
add 9
add 10
add 8
add 9
add 8
add 9
add 10
add 8
add 10
add 8
add 9
add 8
add 9
add 10
etc.
etc.
add 3
add 4
add 5
etc.
etc.
etc.
add 2
add 3
add 4
add 5
etc.
etc.
add 1
add 2
add 3
start

But how to obtain an Branch and bound Algorithm for Phylogenetic Trees??

Recall how we obtained an expression for the number $U(n)$ of unrooted phylogenetic tree topologies on n taxa:

- For $n = 3$ there are three ways of adding an extra edge with a new leaf to obtain an unrooted tree on 4 leaves.
- This new tree has $(2n - 3) = 5$ edges and there are 5 ways to obtain a new tree with 5 leaves etc.

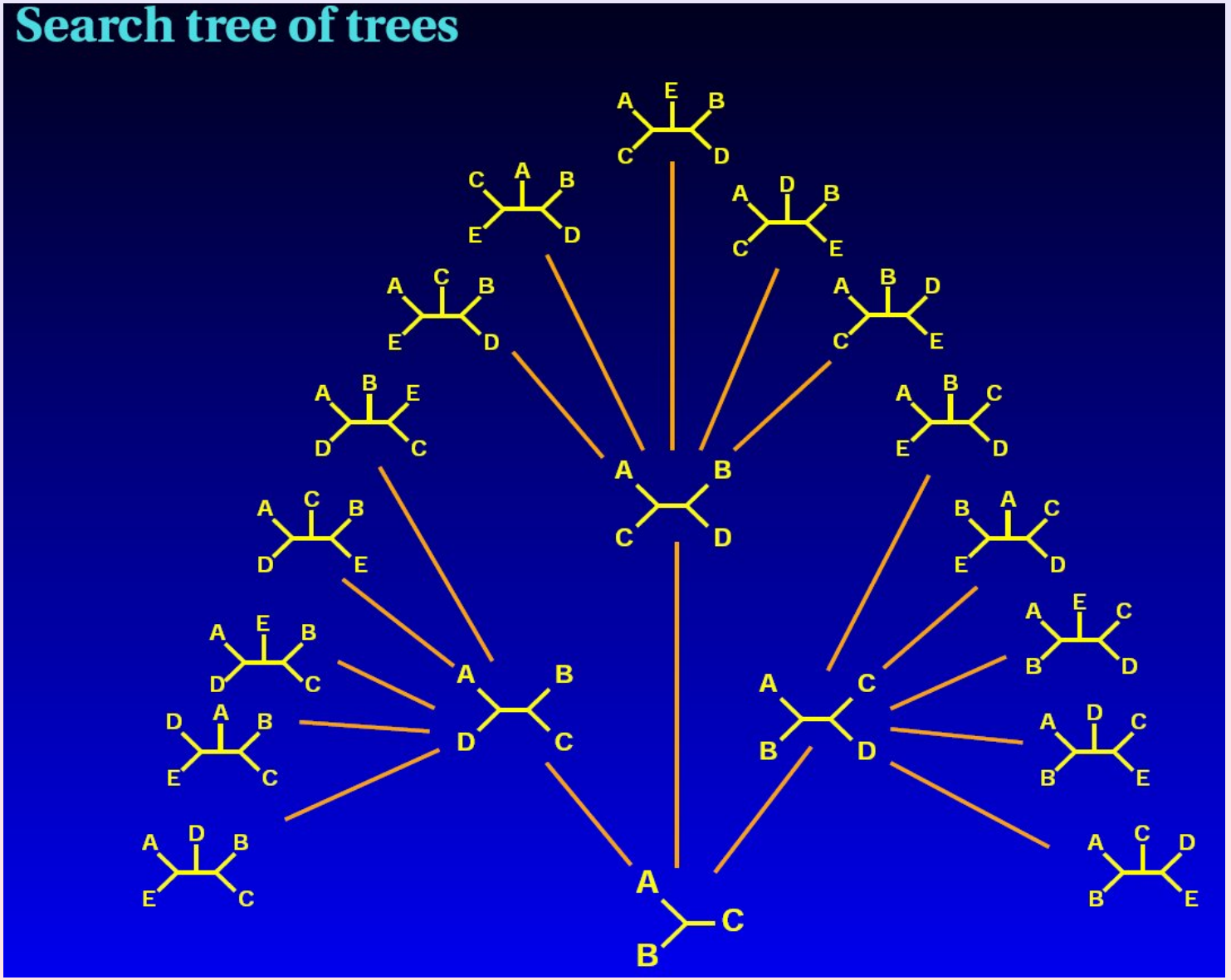
Branch and bound

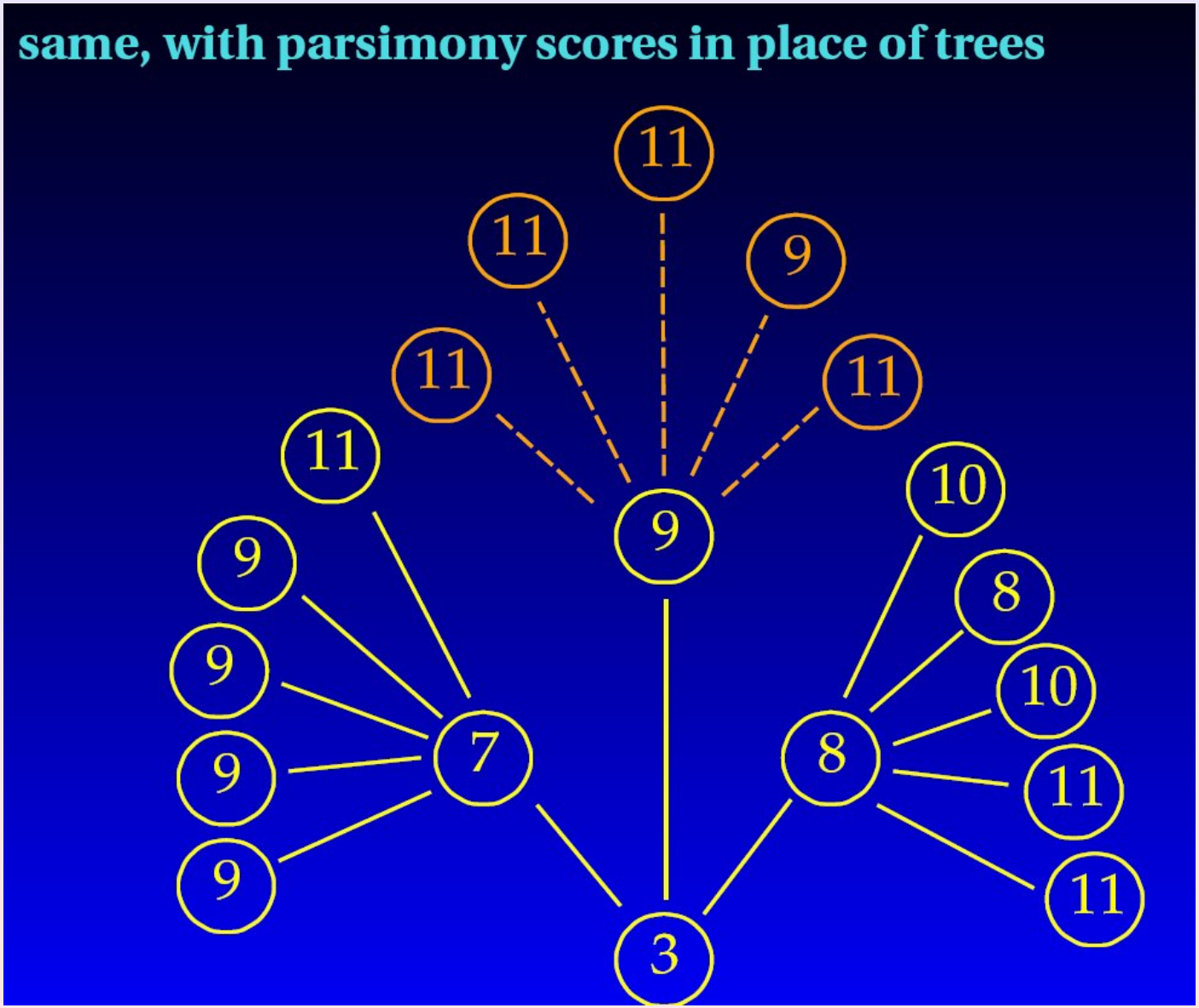
- To be precise, one can obtain any tree T_{i+1} on $\{a_1, \ldots, a_i, a_{i+1}\}$ by adding an extra edge with a leaf labeled a_{i+1} to some (unique) tree T_i on $\{a_1, \ldots, a_i\}$.
- In other words, we can produce the set of **all** possible trees on n taxa by adding one leaf at a time in all possible ways, thus systematically generating a complete **enumeration tree**.

Branch and bound

A simple, but crucial observation is that adding a new sequence a_{i+1} to a tree T_i to obtain a new tree T_{i+1} **cannot** lead to a smaller parsimony score.
This gives rise to the following bound criterion when generating the enumeration tree:

- if the *local* parsimony score of the current incomplete tree T' is larger or equal to the best *global* score for any complete tree seen so far, then *we do not* generate or search the enumeration subtree below T'.


Search tree of trees


same, with parsimony scores in place of trees
11
11
9
11
11
11
9
10
9
8
9
10
9
7
8
11
9
11

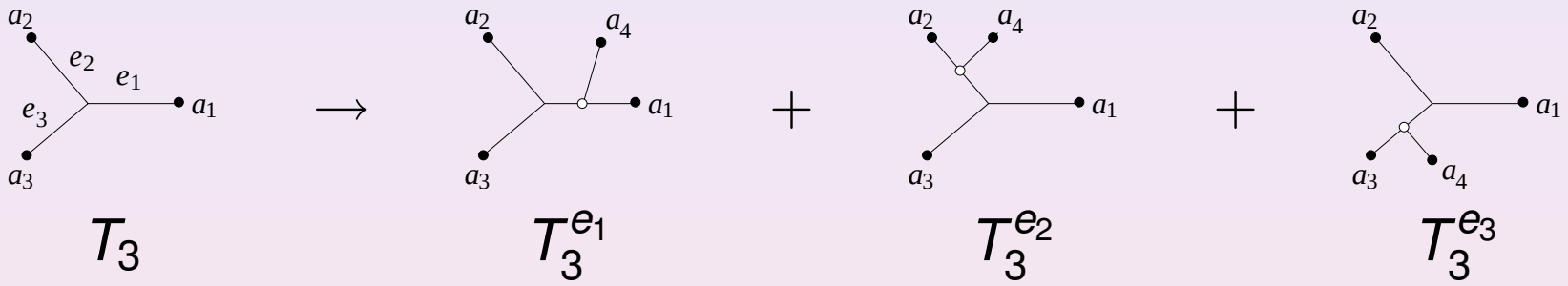
Branch and bound

- Application of branch-and-bound to evolutionary trees was first suggested by Mike Hendy and Dave Penny (1982).
- In practice, using branch and bound one can obtain exact solutions for data sets of twenty or more sequences, depending on the sequence length and the messiness of the data.
- A good starting strategy is to first compute a tree T_0 for the data, e.g. using Neighbor-Joining, and then to initialize the *global* bound to the parsimony score of T_0.

The stepwise-additional heuristic

- Now we discuss a simple greedy heuristic (Felsenstein 1981) for approximating the optimal tree or score.
- We build the tree T by adding one leaf after the other, in each step choosing the optimal position for the new leaf-edge:
- Given a multiple sequence alignment $A = \{a_1, a_2, \ldots, a_n\}$.
  1. Start with a tree T_2 consisting of two leaves labeled a_1 and a_2.
  2. Given T_i. For each edge e in T_i, obtain a new tree T_i^e as follows: Insert a new node v in e and join it via a new edge f to a new leaf w with label a_{i+1}.
  3. Set $T_{i+1} = \arg\min\{P(T_i^e, A)\}$.

The stepwise-additional heuristic



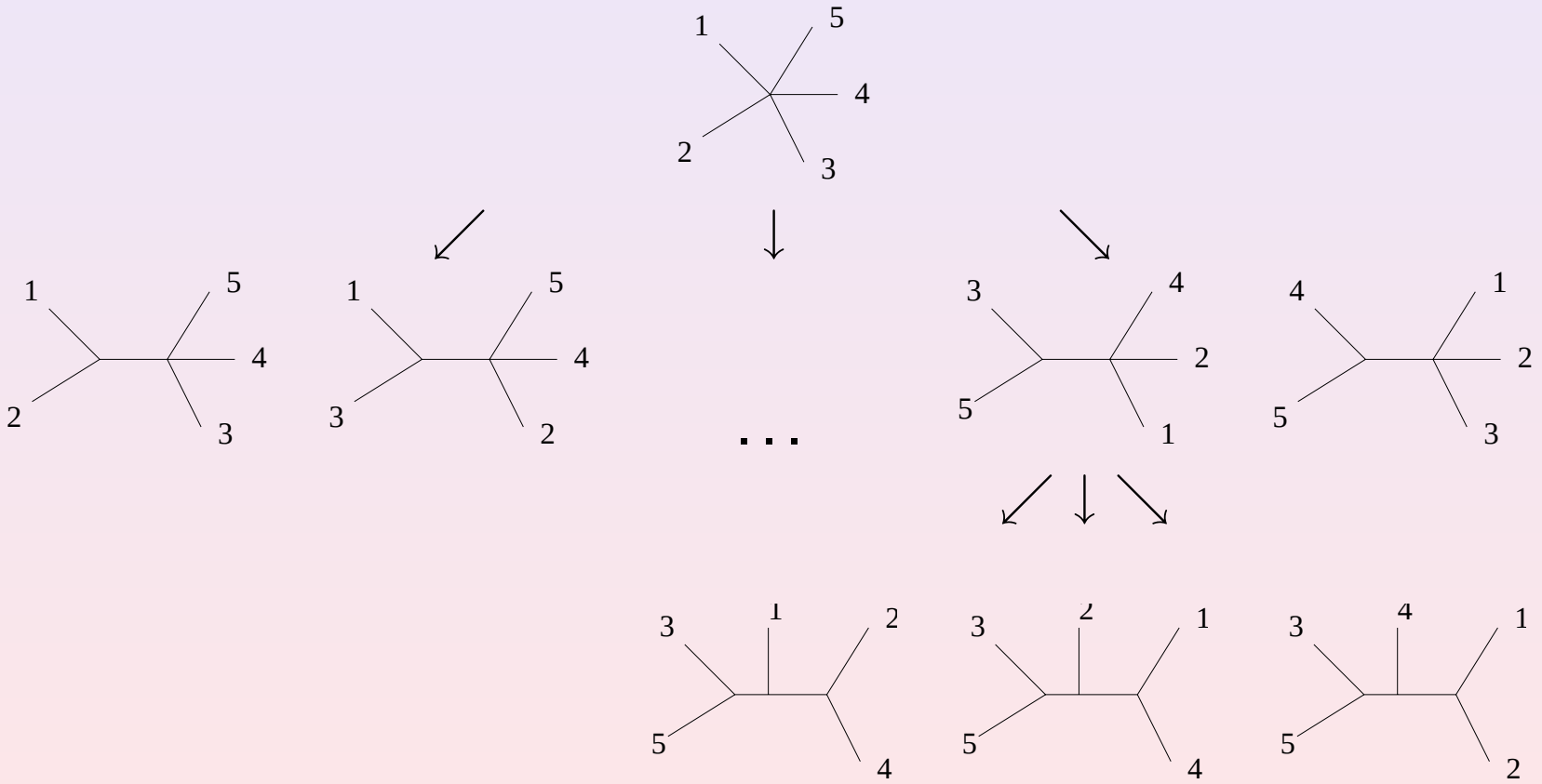
- Obviously, this approach is not guaranteed to obtain an optimal result.
- Moreover, the result obtained will depend on the order in which the sequences are processed.

The star-decomposition heuristic

This employs a similar strategy to Neighbor-Joining.

- We start with a star tree on all n taxa.
- At each step, the optimality criterion is evaluated for every possible joining of a pair of lineages incident to the central node.
- The best tree found at one step is then used as the basis of the next step:

The star-decomposition heuristic



Branch swapping methods

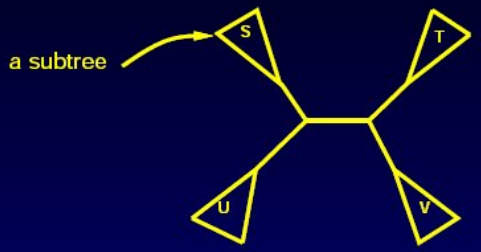
The two heuristics just described are both very susceptible to entrapment in local optima.

- We now discuss a number of **branch-swapping operations** that one can use to move through the space of all trees, hopefully jumping far enough to escape from local optima.

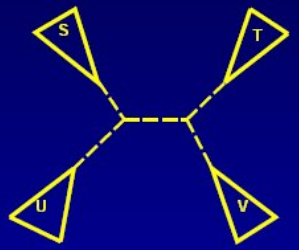
Nearest-Neighbor-Interchange (NMI)

In a **nearest-neighbor interchange (NNI)**, two of the four subtrees around an edge are swapped, in the two possible different ways:

Nearest-neighbor interchanges (NNIs)

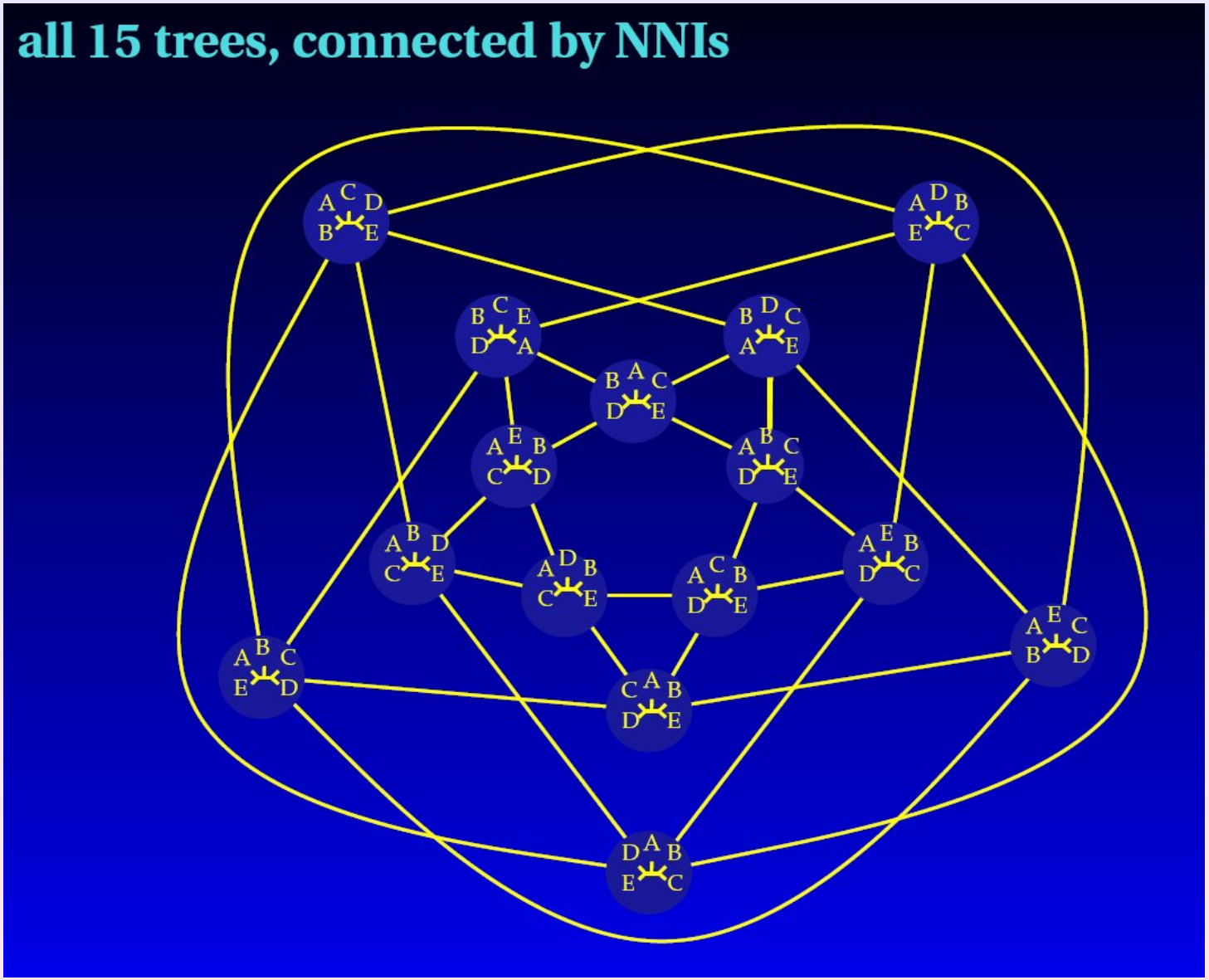


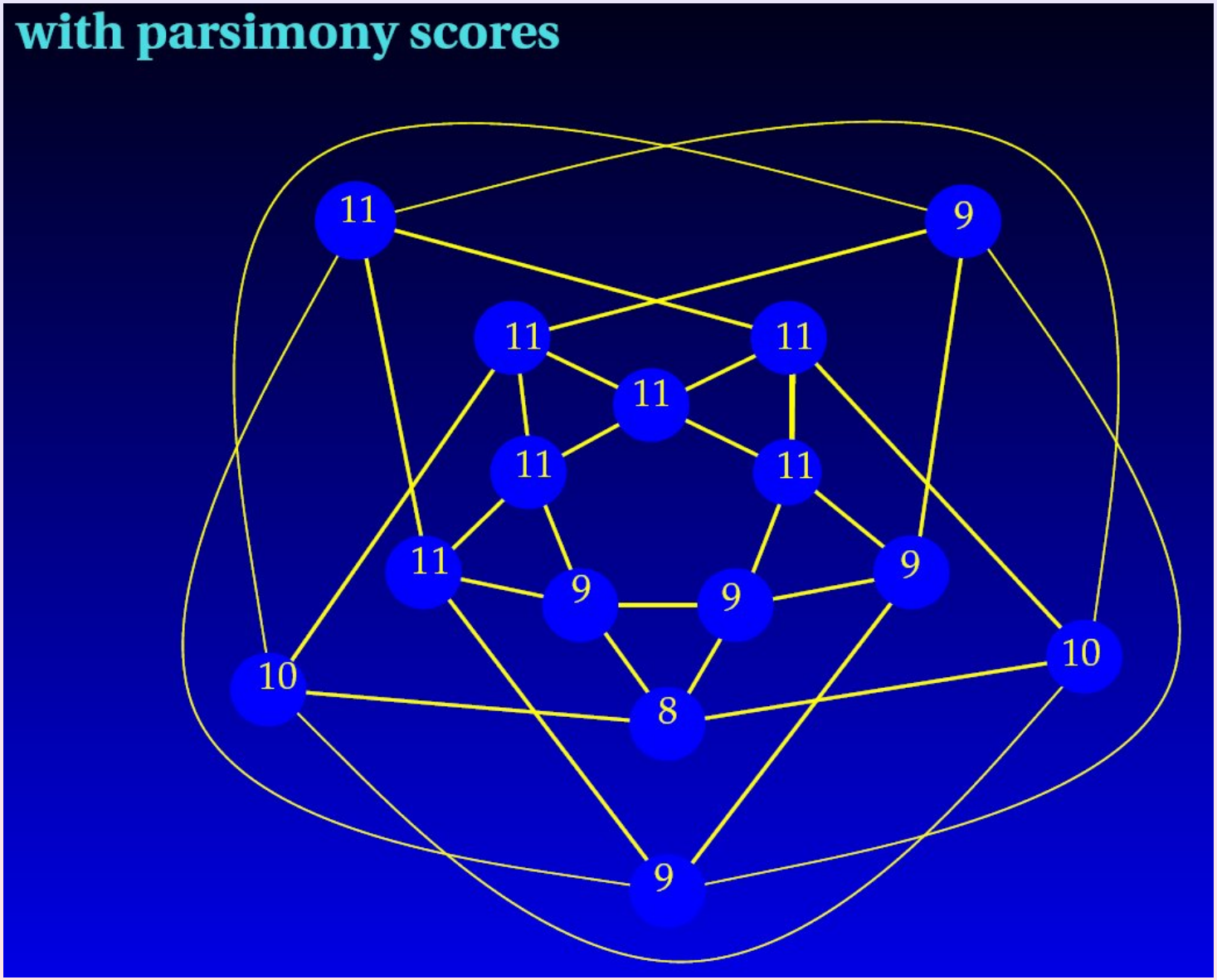
is rearranged by dissolving the connections to an interior branch:



and reforming them in one of the two possible alternative ways:

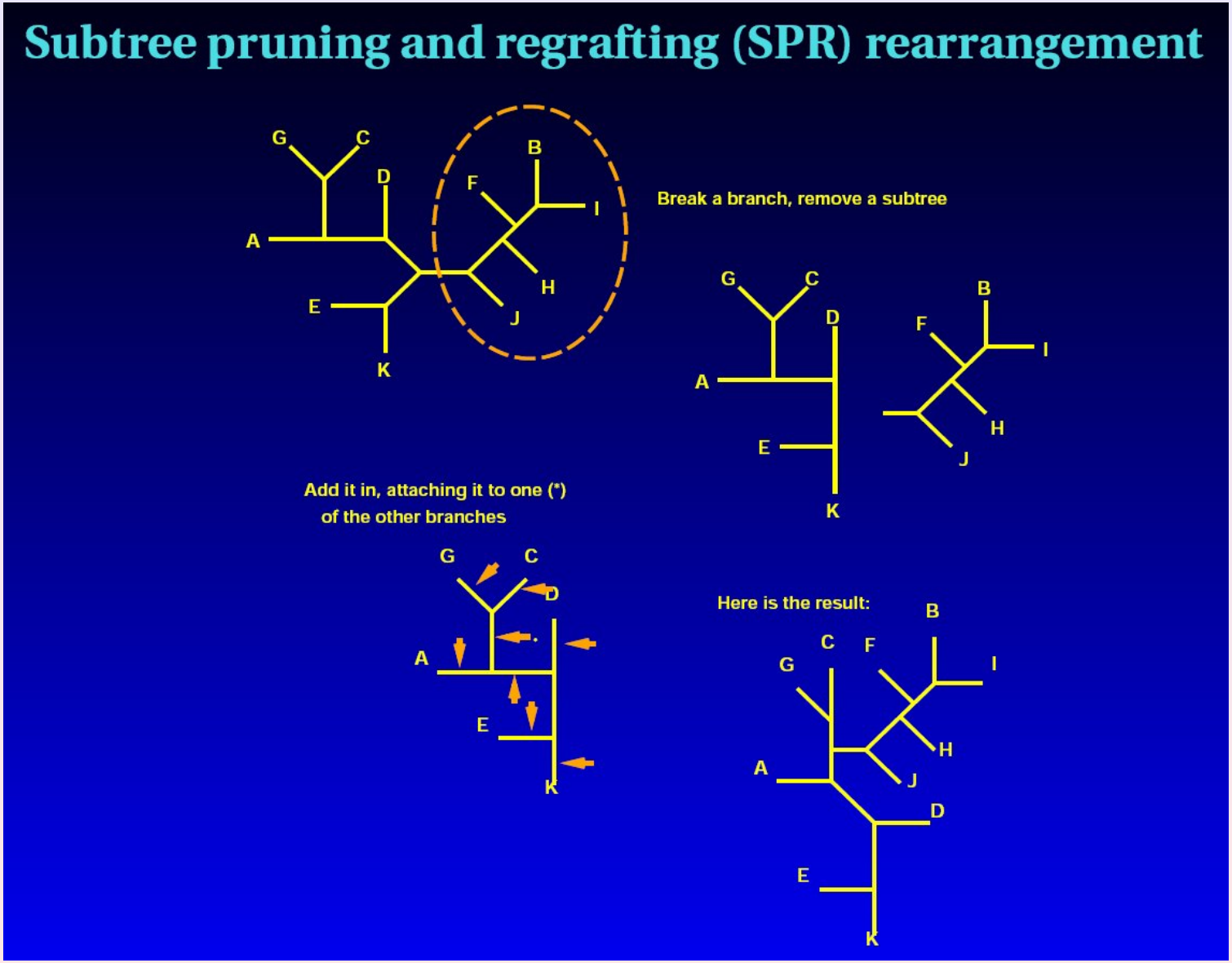
(The triangles are subtrees)


all 15 trees, connected by NNIs


with parsimony scores
11
9
11
11
11
11
11
11
9
9
9
10
10
8
9

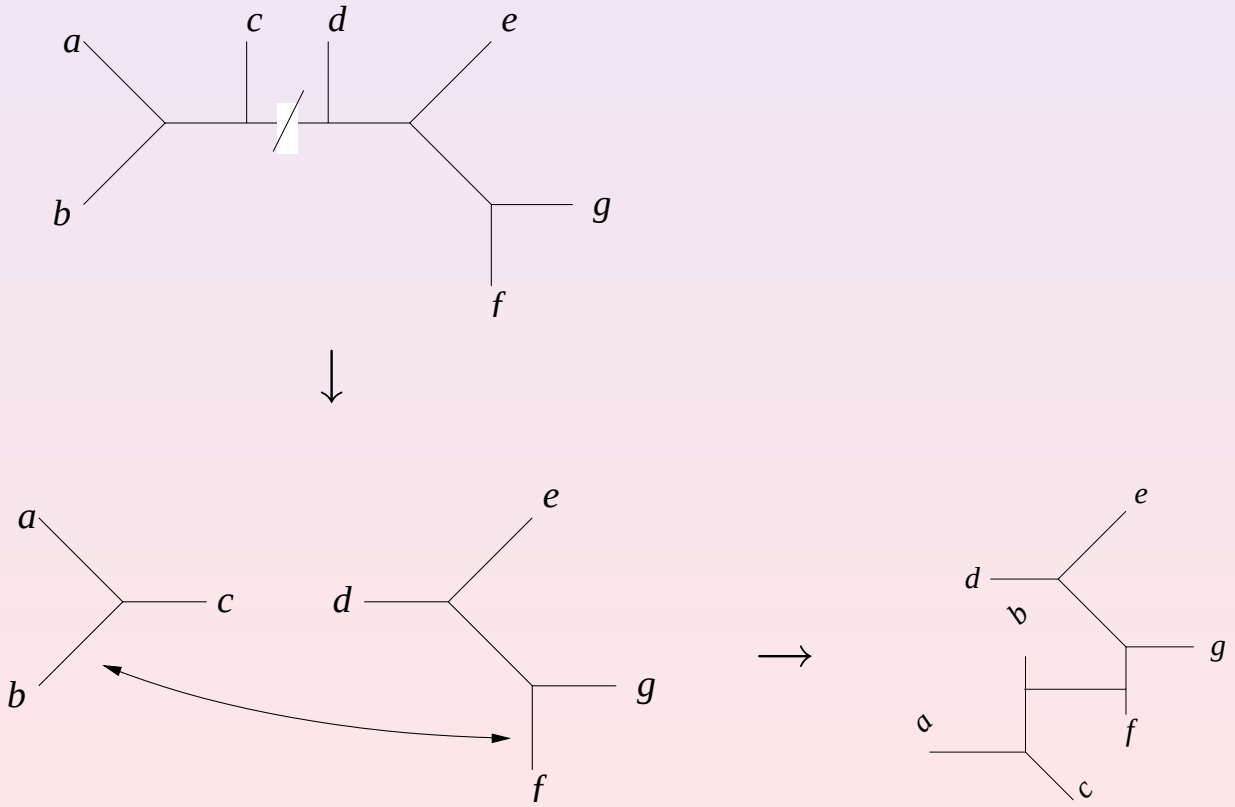
Subtree Pruning and Regrafting (SPR)

In branch swapping by **subtree pruning and re-grafting**, a subtree is pruned from the tree and re-grafted to a different location of the tree:


Subtree pruning and regrafting (SPR) rearrangement
G
C
D
A
E
K
B
F
I
H
J
Break a branch, remove a subtree
G
C
D
A
E
K
B
F
I
H
J
Add it in, attaching it to one (*)
of the other branches
G
C
D
A
E
K
Here is the result:
B
C
F
G
I
H
A
J
D
E
K

Tree bisection and Reattachment (TBR)

In branch swapping by **tree bisection and reattachment** (TBR), the tree is bisected at an edge, yielding two subtrees. The two subtrees are then reconnected at two new positions:



Some Notes

- Each NNI operation is a special case of SPR operation, and each SPR operation is again a special case of TBR operation.
- We also note that each of the branch swapping operations is reversible such that if T' is the result of a branch-swapping operation on T then the same operation applied to T' yields back T.

Some Notes

In addition we have the following

Theorem

Let T and T' be two binary trees. Then T' can be constructed from T by a series of NNI operations. The number of different trees, that can be constructed from T with a single NNI operation is $2(n-3)$ and with SPR $4(n-3)(n-2)$. For the TBR the exact number of single operations depends on tree shape (but usually more than by SPR).

Heuristic search

If the data set $A = \{a_1, \ldots, a_n\}$ is too big to be solved exactly via branch and bound, then we can use a heuristic search method in an attempt to find or approximate the optimal solution.

- This involves searching through the space of all unrooted phylogenetic trees on n labels and trying to proceed toward a globally optimal one.
- We “move” through tree space using one or more branching-swapping techniques.

Heuristic search

Heuristic searches employ **hill-climbing techniques**:

- we imagine that the "goodness" $-PS(T)$ of the solution as a landscape along which we move during the search.
- The general strategy is to always move upwards in the hope of reaching the top of the highest peak.

Even using the above described branch-swapping techniques, **any heuristic search is in danger of "climbing the wrong mountain" and getting stuck in a local optimum**. Different strategies have been developed to avoid this problem.

Simulated annealing

The **simulated annealing** method employs a **temperature** that cools over time (Van Laarhoven and Aarts, 1987).

- At high temperatures the search can move more easily to trees whose score is less optimal than the score of the current tree.
- As the temperature decreases, the search becomes more and more directed toward better trees.

I.e., let T_i denote the current tree at step i and let $z(T_i)$ denote the goodness of T_i (e.g., $-PS(T)$).

Simulated annealing

In hill climbing, a move to T_{i+1} is acceptable, if $z(T_{i+1}) \geq z(T_i)$. In simulated annealing, **any** new solution is accepted with a certain probability:

$$Prob(\text{accepting solution } T_{i+1})$$

$$= \begin{cases} 1 & \text{if } z(T_{i+1}) \geq z(T_i) \\ e^{-t_i(z(T_{i+1})-z(T_i))} & \text{otherwise,} \end{cases}$$

where t_i is called the **temperature** and decreases over time.

The Great Deluge method

- The **Great Deluge** method, introduced by Guenter Dueck and Tobias Scheuer (1990), employs a slowly rising water level and the search accepts any move that stays above the water level.
- The probability of accepting a new solution T_{i+1} is 1, if $z(T_{i+1}) > w_i$, where w_i is a bound that increases slowly with time.
- If T_{i+1} is accepted, then we update the water level by setting

$$w_{i+1} = c \times (z(T_{i+1}) - z(T_i)) .$$

- Typically, the constant c is usually about 0.01 to 0.05.

Another of the many heuristics is **tabu search** method (Glover, 1989) that maintains a **tabu list** of $5 - 10$ solutions recently visited and refrains from revisiting them.