Computational Genomics  גנומיקה חישובית

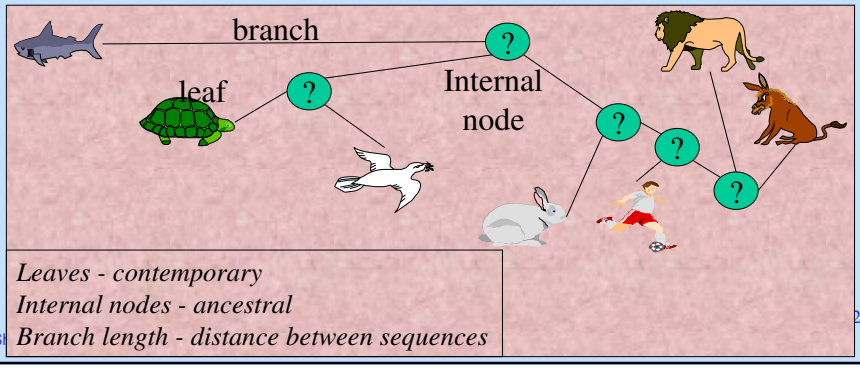
Phylogeny

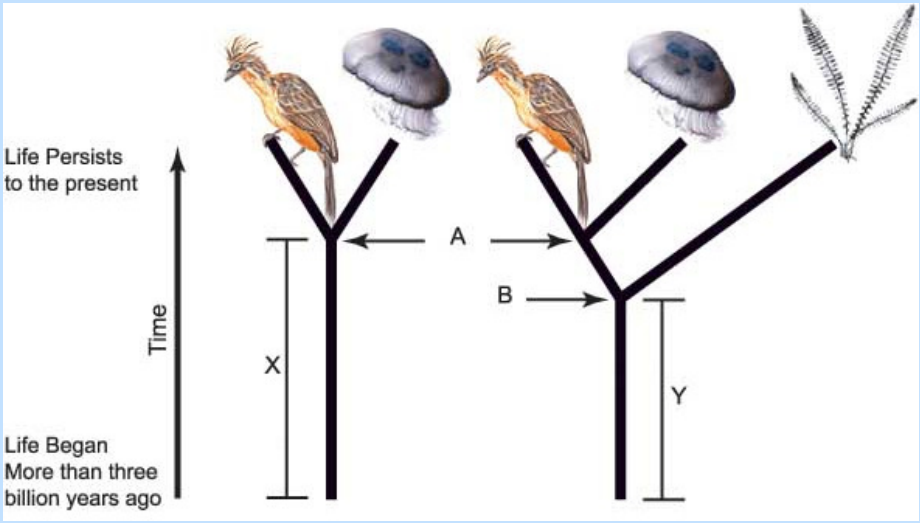
Nov, 2018

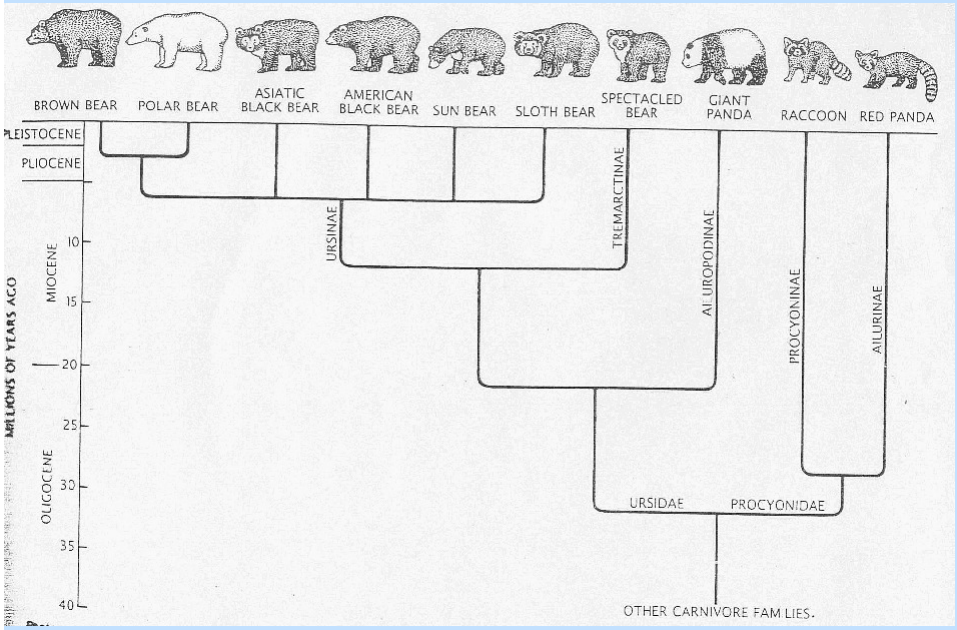
- Slides:
- Adi Akavia
- Nir Friedman's slides at HUJI (based on ALGMB 98)
- Anders Gorm Pedersen,Technical University of Denmark
- Sources: **Joe Felsenstein "Inferring Phylogenies" (2004)**

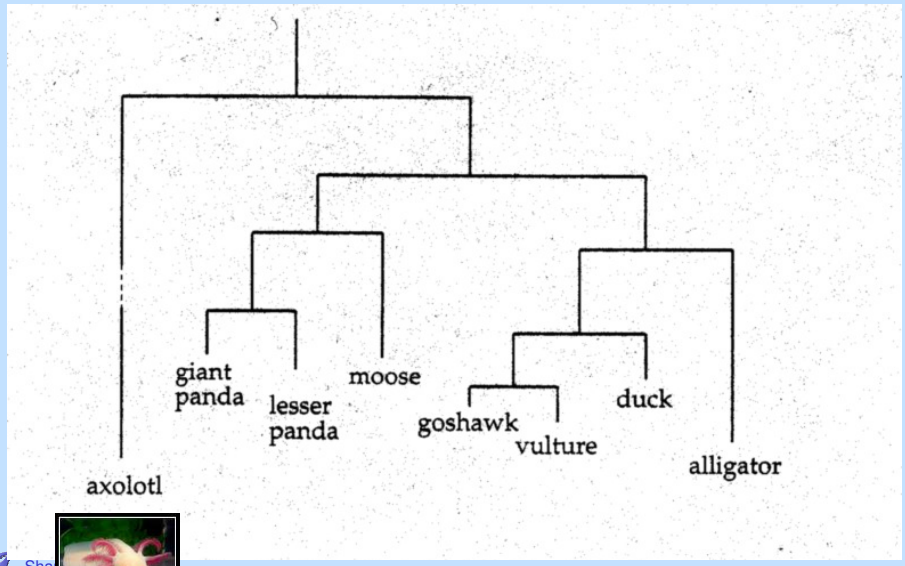
Phylogeny

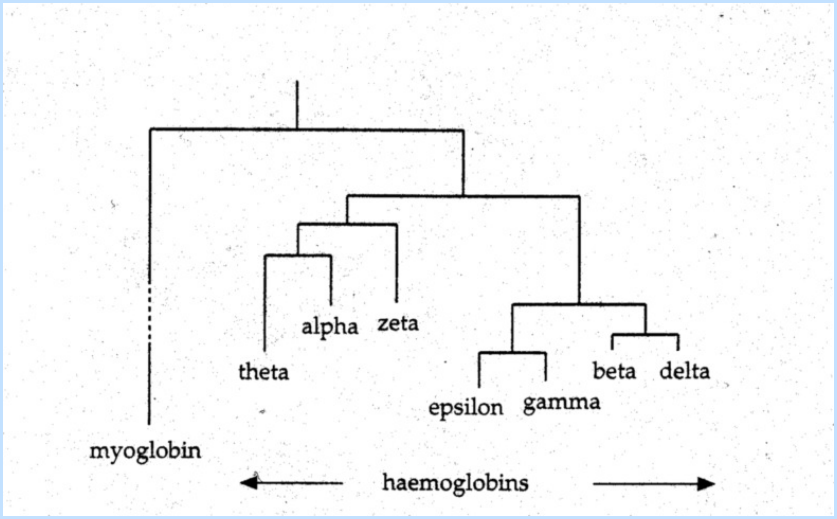
- Phylogeny: the ancestral relationship of a set of species.
- Represented by a phylogenetic tree





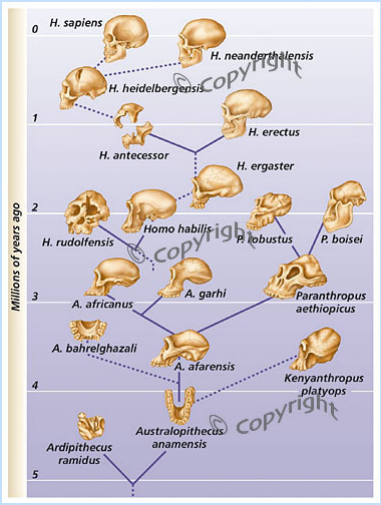



giant
panda
lesser
panda
moose
goshawk
vulture
duck
alligator
axolotl

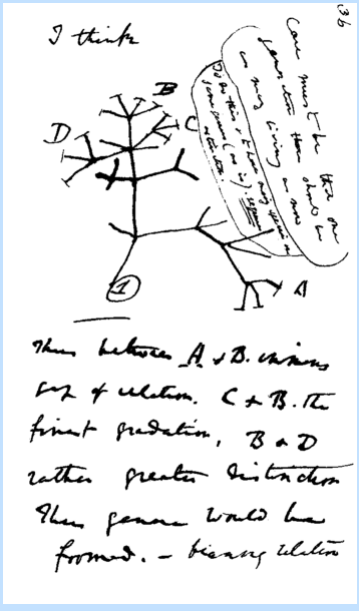

alpha
zeta
theta
beta
delta
epsilon
gamma
myoglobin
haemoglobins

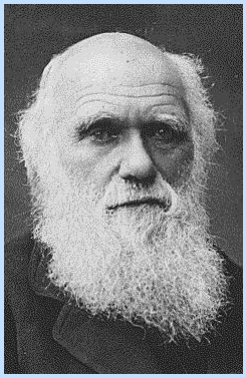
Classical vs. Modern Phylogeny schools

- Classical – morphological characters
- Modern - molecular sequences.



http://www.scientific-art.com/GIF%20files/Palaeontological/hominidtree.jpg





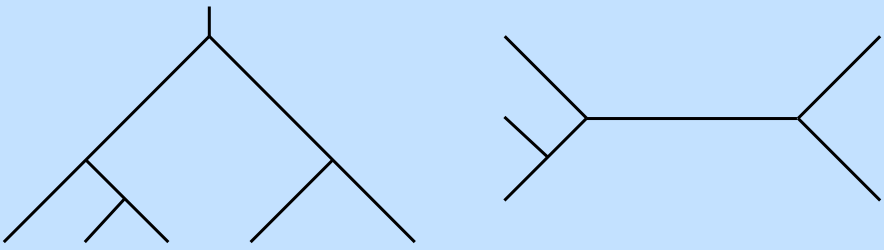
Page from Darwin's notebooks around July 1837 showing the first-known sketch by Charles Darwin of an evolutionary tree describing the relationships among groups of organisms.

Trees and Models

- rooted / unrooted
- topology / distance
- binary / general

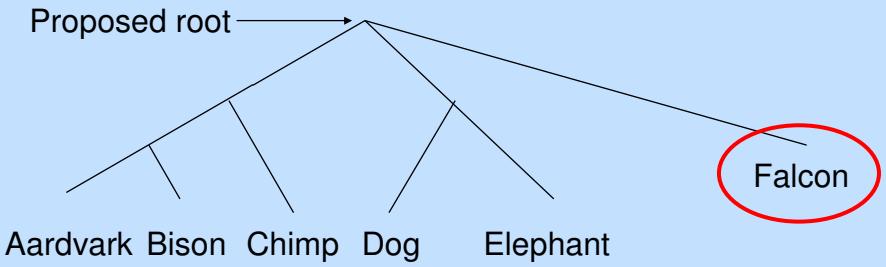
To root or not to root?

- **Unrooted** tree: phylogeny without direction.



Rooting an Unrooted Tree

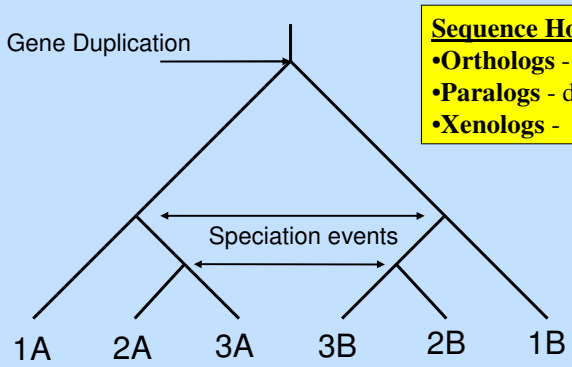
- We can estimate the position of the root by introducing an outgroup:
  - a species that is definitely most distant from all the species of interest



HOW DO WE FIGURE OUT THESE TREES? TIMES?

Dangers of Paralogs

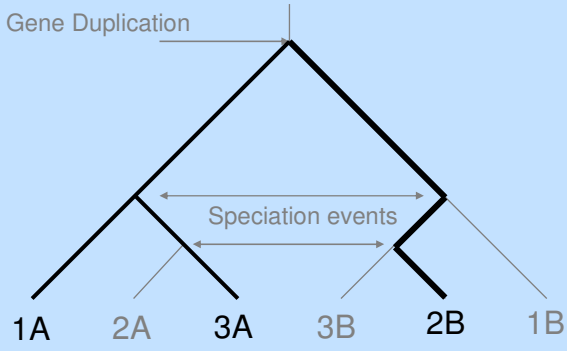
- Right species topology: (1,(2,3))



Sequence Homology Caused By:
- **Orthologs** - speciation,
- **Paralogs** - duplication
- **Xenologs** - horizontal (e.g., by virus)

Dangers of Paralogs

- Right species distance: (1,(2,3))
- If we only consider 1A, 2B, and 3A: ((1,3),2)



Type of Data

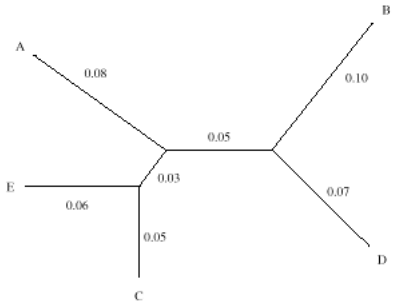
- **Distance**-based
  - Input: matrix of distances between species
  - Distance can be
    - fraction of residue they disagree on,
    - alignment score between them,
    - …
- **Character**-based
  - Examine each character (e.g., residue) separately

Distance Based Methods

Tree based distances

- d(i,j)=sum of arc lengths on the path i←→j
- Given d, can we find
  - an exactly matching tree?
  - An approximately matching tree?

species	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



The Problem

The least squares criteria

Input: matrix **d** of distances between species

Goal: Find a tree with leaves=chars and edge distances, matching d best.

Quality measure: sum of squares:

$$SSQ(T) = \sum_{i} \sum_{j \neq i} w_{ij} (d_{ij} - t_{ij})^2$$

t_{ij}: distance in the tree

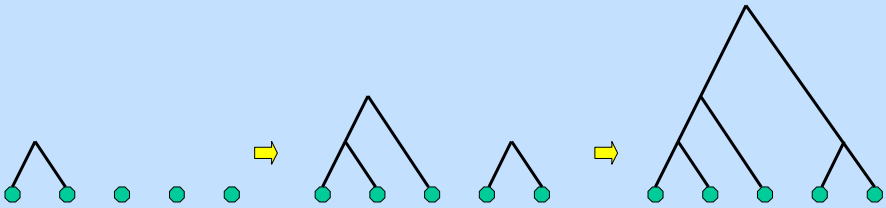
w_{ij}: pair weighting. Options: (1) ≡1 (2) $1/d_{ij}$ (3) $1/d_{ij}^2$

NP-hard (Day '86). We'll describe common heuristics

UPGMA Clustering (Sokal & Michener 1958)

(Unweighted pair-group method with arithmetic mean)

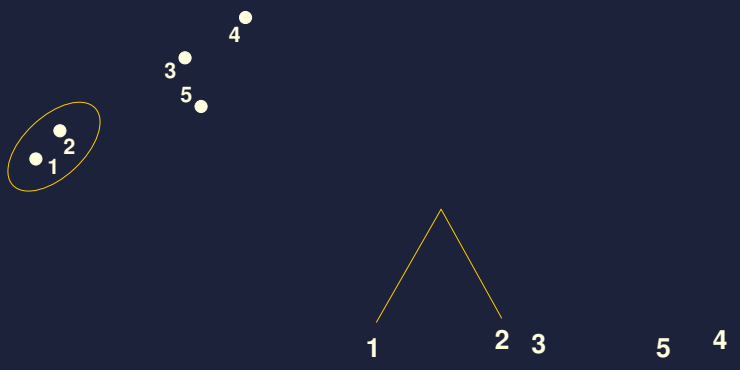
- Approach: Form a tree; closer species according to input distances should be closer in the tree
- Build the tree bottom up, each time merging two smaller trees
- All leaves are at same distance from the root



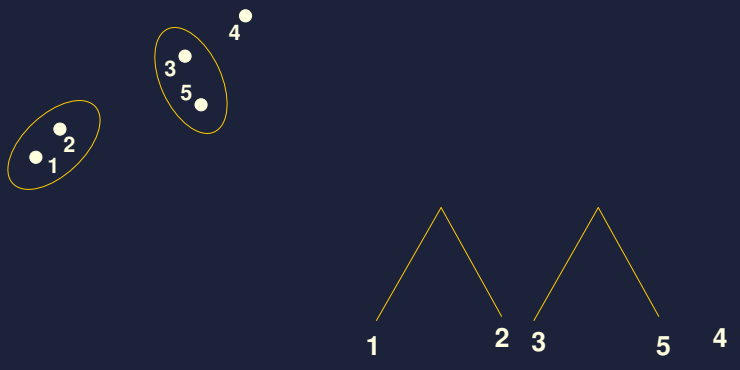
UPGMA Algorithm

1 2 3 5 4

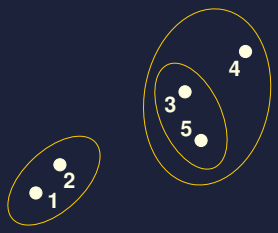
UPGMA Algorithm

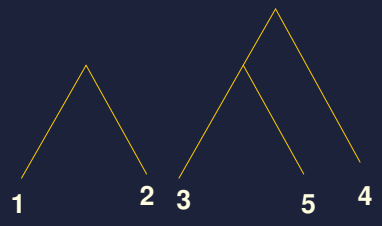


UPGMA Algorithm

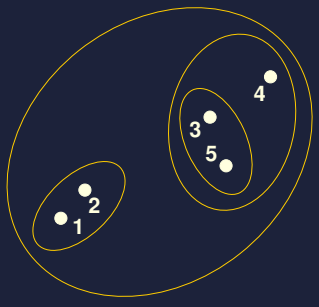


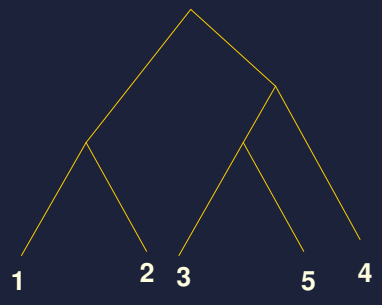
UPGMA Algorithm





UPGMA Algorithm





Efficiency lemma

- Approach: gradually form clusters: sets of species
- Repeatedly identify two clusters and merge them.
- For clusters C_i C_j , define the distance between them to be the average dist betw their members:

$$d(C_i,C_j) = \frac{1}{|C_i||C_j|}\sum_{p\in C_i}\sum_{q\in C_j} d(p,q)$$

- Lemma: If C_k is formed by merging C_i and C_j then for every other cluster C_l

  $d(C_k,C_l) = (|C_i|*d(C_i,C_l) + |C_j|*d(C_j,C_l)) / (|C_i| + |C_j|)$

→ Can update distances between clusters in time prop. to the number of clusters.

UPGMA algorithm

Initialize: each node is a cluster Ci={i}. d(Ci,Cj)=d(i,j) set height(i) = 0 ∀i

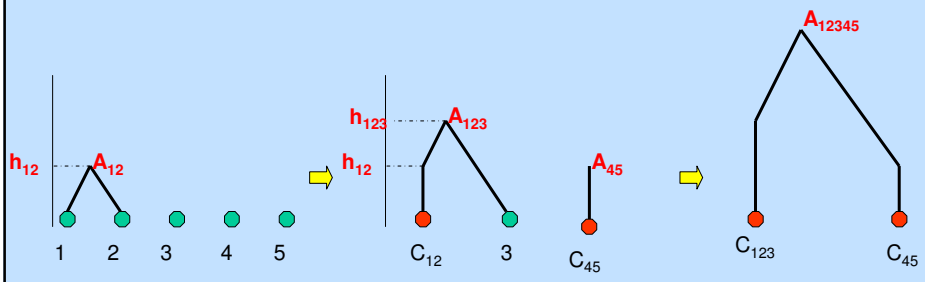
Iterate:

- Find Ci,Cj with smallest d(Ci,Cj)
- Introduce a new cluster node Ck that replaces Ci and Cj
  //Ck represents all the leaves in clusters Ci and Cj
- Introduce a new tree node Aij with height(Aij) = d(Ci,Cj)/2
  // d(Ci,Cj) is the average dist among leaves of Ci and Cj
- Connect the corresponding tree nodes Ci,Cj to Aij with
  length(Ci, Aij) = height(Aij) - height(Ci)
  length(Cj,Aij) = height(Aij) - height(Cj)
- For all other Cl:
  d(Ck,Cl) = (|Ci|*d(Ci,Cl) + |Cj|*d(Cj,Cl)) / (|Ci| + |Cj|)
  //dist to any old cluster is the ave dist between its leaves and leaves in Ci, Cj

Time: Naïve: $O(n^3)$; Can show $O(n^2 \log n)$ (ex.) and $O(n^2)$ (harder ex.)

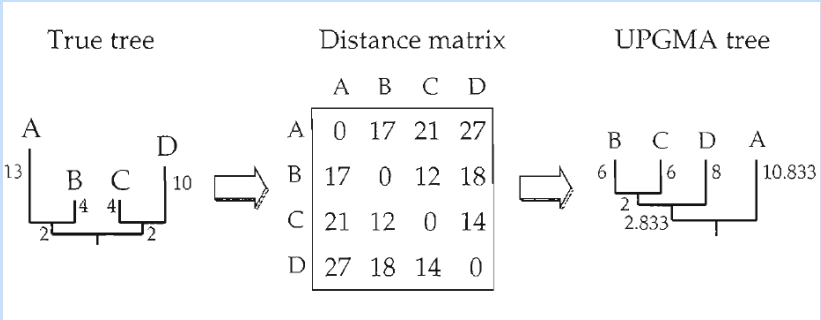
UPGMA alg (2)

- Orange nodes represent the groups of nodes that they replaced, and maintain the average dist of the set from other leaf nodes/clusters



Molecular Clock

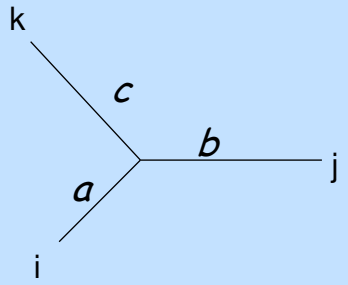
- UPGMA assumes the tree has equal leaf-root distances => common uniform clock. Such tree is called (a particular type of) ultrametric



- Works reasonably well for nearby species

Additivity

- An **additivity assumption**: distances between species are the sum of distances between intermediate nodes (even if the tree is not ultrametric)

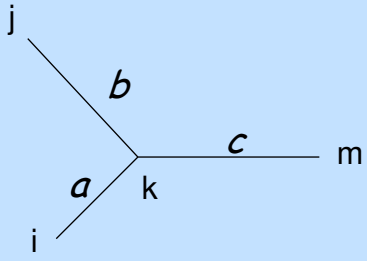


$$d(i,j) = a + b$$
$$d(i,k) = a + c$$
$$d(j,k) = b + c$$

If the distance matrix is an exact reflection of a true tree, then additivity holds

Consequences of Additivity

- Suppose input distances are additive
- For any three leaves



$$d(i,j) = a + b$$
$$d(i,m) = a + c$$
$$d(j,m) = b + c$$

- Thus $$d(i,k) = \frac{1}{2}(d(i,m) + d(i,j) - d(m,j))$$

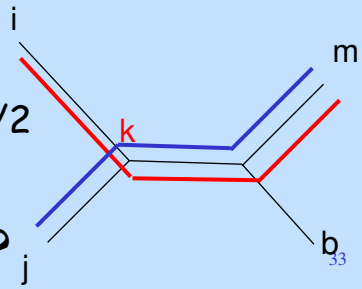
$$d(m,k) = \frac{1}{2}(d(i,m) + d(j,m) - d(i,j))$$

Consequences of Additivity II

- If we can identify neighbor leaves, then can use pairwise distances to reconstruct the tree:
- Remove neighbors i, j from the leaf set
- Add k
- Set $d_{km} = (d_{im} + d_{jm} - d_{ij})/2$

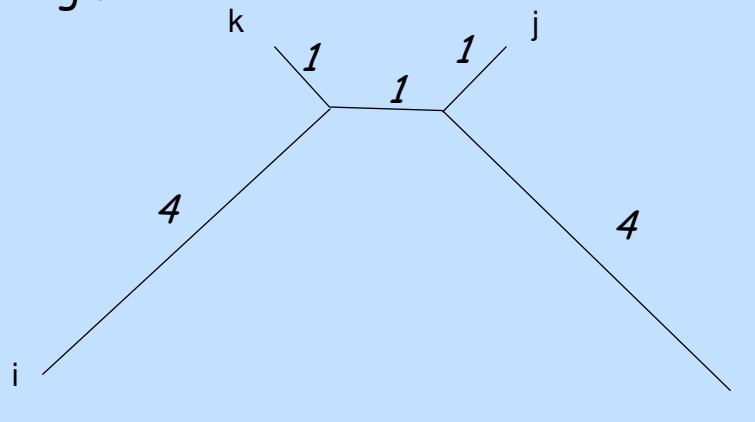
  $d_{ik} = d_{im} - d_{km} = (d_{im} - d_{jm} + d_{ij})/2$

Can we find neighbor leaves?



Closest pairs may not be neighbors!

- Closest pair: k and j, but they are not neighbors



Neighbor Joining (Saitou-Nei '87)

- Let $D(i,j) = d(i,j) - (r_i + r_j)$ where

$$r_i = \frac{1}{|L|-2} \sum_k d(i,k)$$

"Corrected" average distance of i from all other nodes

Theorem: if *D(i,j)* is minimum among all pairs of leaves, then *i* and *j* are neighbors in the tree

Neighbor Joining algorithm

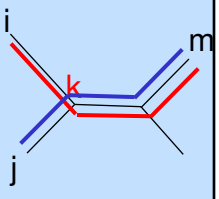
$$r_i = \frac{1}{|L|-2} \sum_k d(i,k)$$
$$D(i,j) = d(i,j) - (r_i + r_j)$$

- Set *L* to contain all leaves

Iteration:

- Choose *i,j* such that *D(i,j)* is minimum
- Create new node *k*, and set

$$d(i,k) = (d(i,j) + r_i - r_j)/2$$
$$d(j,k) = (d(i,j) + r_j - r_i)/2$$
$$d(k,m) = (d(i,m) + d(j,m) - d(i,j))/2$$



- remove *i,j* from *L*, and add *k*
- *Update r, D*
- **Termination:** when */L/ =2* connect the two nodes

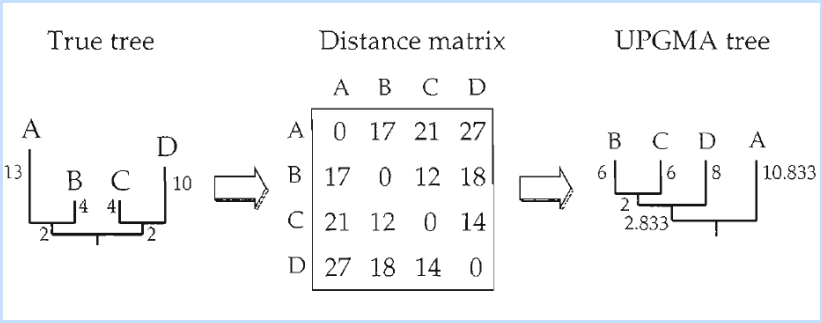
Time:O(n^3)

Ex.

Thm: Opt tree guaranteed if distances match a tree

Does not assume a clock

An example



Neighbor Joining Algorithm

	A	B	C	D
A	-	17	21	27
B		-	12	18
C			-	14
D				-

Neighbor Joining Algorithm

	A	B	C	D
A	–	17	21	27
B		–	12	18
C			–	14
D				–

i	r_i
A	(17+21+27)/2=32.5
B	(17+12+18)/2=23.5
C	(21+12+14)/2=23.5
D	(27+18+14)/2=29.5

Neighbor Joining Algorithm

	A	B	C	D
A	–	17	21	27
B		–	12	18
C			–	14
D				–

i	r_i
A	(17+21+27)/2=32.5
B	(17+12+18)/2=23.5
C	(21+12+14)/2=23.5
D	(27+18+14)/2=29.5

	A	B	C	D
A	–	–39	–35	–35
B		–	–35	–35
C			–	–39
D				–

$d_{ij}-r_i-r_j$

Neighbor Joining Algorithm

	A	B	C	D
A	–	17	21	27
B		–	12	18
C			–	14
D				–

i	r_i
A	(17+21+27)/2=32.5
B	(17+12+18)/2=23.5
C	(21+12+14)/2=23.5
D	(27+18+14)/2=29.5

	A	B	C	D
A	–	–39	–35	–35
B		–	–35	–35
C			–	–39
D				–

$d_{ij}-r_i-r_j$

Neighbor Joining Algorithm

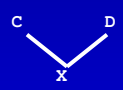
CENTER FOR BIOLOGICAL SEQUENCE ANALYSIS

	A	B	C	D
A	–	17	21	27
B		–	12	18
C			–	14
D				–

i	r_i
A	(17+21+27)/2=32.5
B	(17+12+18)/2=23.5
C	(21+12+14)/2=23.5
D	(27+18+14)/2=29.5

	A	B	C	D
A	–	–39	–35	–35
B		–	–35	–35
C			–	–39
D				–

$d_{ij}-r_i-r_j$



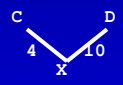
Neighbor Joining Algorithm

	A	B	C	D
A	–	17	21	27
B		–	12	18
C			–	14
D				–

i	r_i
A	(17+21+27)/2=32.5
B	(17+12+18)/2=23.5
C	(21+12+14)/2=23.5
D	(27+18+14)/2=29.5

	A	B	C	D
A	–	-39	-35	-35
B		–	-35	-35
C			–	-39
D				–

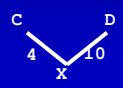
$d_{ij}-r_i-r_j$



$v_C = 0.5 \times 14 + 0.5 \times (23.5-29.5) = 4$
$v_D = 0.5 \times 14 + 0.5 \times (29.5-23.5) = 10$

Neighbor Joining Algorithm

	A	B	C	D	X
A	–	17	21	27	
B		–	12	18	
C			–	14	
D				–	
X					–



Neighbor Joining Algorithm

	A	B	C	D	X
A	–	17	21	27	17
B		–	12	18	8
C			–	14	
D				–	
X					–

$d_{XA} = (d_{CA} + d_{DA} - d_{CD})/2$
$= (21 + 27 - 14)/2$
$= 17$

$d_{XB} = (d_{CB} + d_{DB} - d_{CD})/2$
$= (12 + 18 - 14)/2$
$= 8$

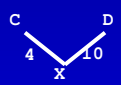
CENTER FOR BIOLOGICAL SEQUENCE ANALYSIS

Neighbor Joining Algorithm

	A	B	X
A	–	17	17
B		–	8
X			–

$d_{XA} = (d_{CA} + d_{DA} - d_{CD})/2$
$= (21 + 27 - 14)/2$
$= 17$

$d_{XB} = (d_{CB} + d_{DB} - d_{CD})/2$
$= (12 + 18 - 14)/2$
$= 8$

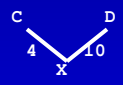


CENTER FOR BIOLOGICAL SEQUENCE ANALYSIS

Neighbor Joining Algorithm

	A	B	X
A	–	17	17
B		–	8
X			–

i	r_i
A	(17+17)/1 = 34
B	(17+8)/1 = 25
X	(17+8)/1 = 25



Neighbor Joining Algorithm

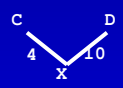
CENTER FOR BIOLOGICAL SEQUENCE ANALYSIS

	A	B	X
A	–	17	17
B		–	8
X			–

i	r_i
A	(17+17)/1 = 34
B	(17+8)/1 = 25
X	(17+8)/1 = 25

	A	B	X
A	–	-42	-28
B		–	-28
X			–

$d_{ij}-r_i-r_j$



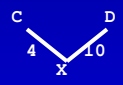
Neighbor Joining Algorithm

	A	B	X
A	–	17	17
B		–	8
X			–

i	r_i
A	(17+17)/1 = 34
B	(17+8)/1 = 25
X	(17+8)/1 = 25

	A	B	X
A	–	–42	–28
B		–	–28
X			–

$d_{ij} - r_i - r_j$



Neighbor Joining Algorithm

CENTER FOR BIOLOGICAL SEQUENCE ANALYSIS

	A	B	X
A	–	17	17
B		–	8
X			–

i	r_i
A	(17+17)/1 = 34
B	(17+8)/1 = 25
X	(17+8)/1 = 25

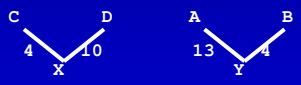
	A	B	X
A	–	–42	–28
B		–	–28
X			–

$d_{ij} - r_i - r_j$

$v_A = 0.5 \times 17 + 0.5 \times (34-25) = 13$
$v_D = 0.5 \times 17 + 0.5 \times (25-34) = 4$

Neighbor Joining Algorithm

	A	B	X	Y
A	–	17	17	
B		–	8	
X			–	
Y				



Neighbor Joining Algorithm

	A	B	X	Y
A	–	17	17	
B		–	8	
X			–	4
Y				

$$d_{YX} = (d_{AX} + d_{BX} - d_{AB})/2$$
$$= (17 + 8 - 17)/2$$
$$= 4$$

Neighbor Joining Algorithm

CENTER FOR BIOLOGICAL SEQUENCE ANALYSIS

	X	Y
X	–	4
Y		–

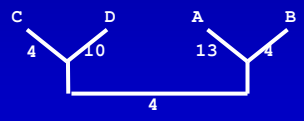
$d_{YX} = (d_{AX} + d_{BX} - d_{AB})/2$
$= (17 + 8 - 17)/2$
$= 4$

Neighbor Joining Algorithm

CENTER FOR BIOLOGICAL SEQUENCE ANALYSIS

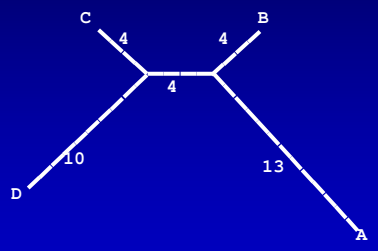
	X	Y
X	–	4
Y		–

$d_{YX} = (d_{AX} + d_{BX} - d_{AB})/2$
$= (17 + 8 - 17)/2$
$= 4$



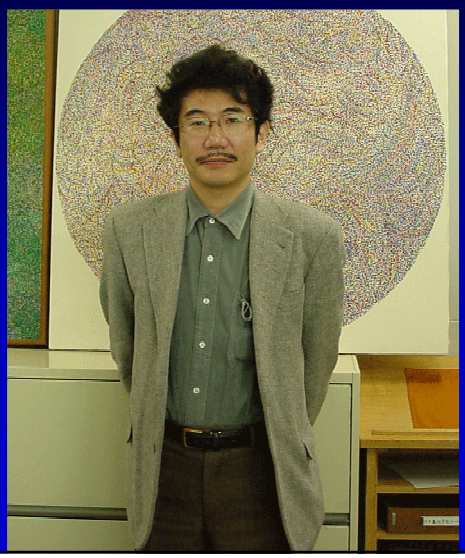
Neighbor Joining Algorithm

	A	B	C	D
A	–	17	21	27
B		–	12	18
C			–	14
D				–



Naruya Saitou

- Division of Population Genetics, National Institute of Genetics
& Department of Genetics, Graduate University for Advanced Studies (Sokendai)
Mishima, 411-8540, Japan



Character Based Methods

Inferring a Phylogenetic Tree

Generic problem: Optimal Phylogenetic Tree:

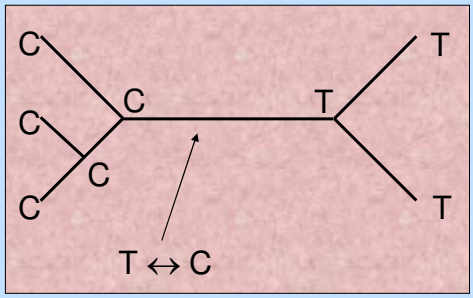
- Input:
  - *n* species,
  - set of characters,
  - for each species, the state of each of the characters.
  - (parameters)
- Goal: find a fully-labeled phylogenetic tree that best explains the data. (maximizes a target function).

A:	CAGGTA
B:	CAGACA
C:	CGGGTA
D:	TGCACT
E:	TGCGTA

Assumptions:
- characters are mutually independent
- species evolve independently

A Simple Example

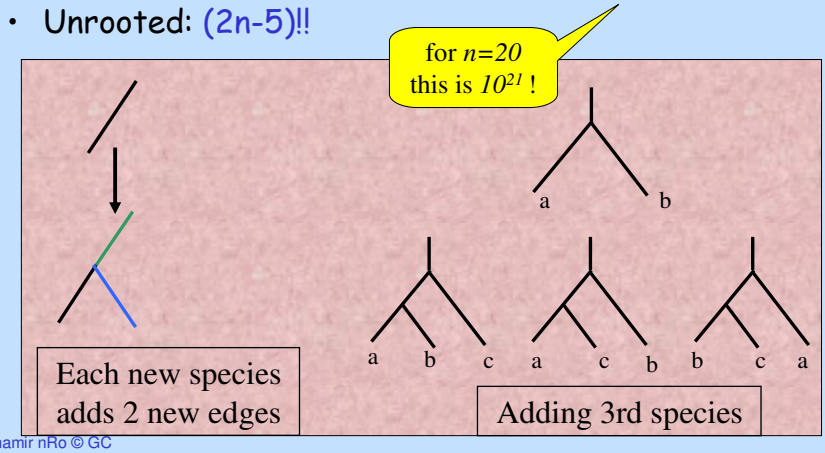
- Five species, three have 'C' and two 'T' at a specified position.
- A minimal tree has one evolutionary change:



Inferring a Phylogenetic Tree

Naive Solution - Enumeration:

- No. of non-isomorphic, labeled, binary, rooted trees, containing *n* leaves: $(2n-3)!! = \Pi_{i=3\ldots n}(2i-3)$
- Unrooted: $(2n-5)!!$



Parsimony

- **Goal**: explain data with min. no. of evolutionary changes ("mutations", or mismatches)
- **Parsimony**: $S(T) \equiv \sum_j \sum_{\{v,u\} \in E(T)} |\{j: v_j \neq u_j\}|$
- **"Small parsimony problem"**:
  - Input: leaf sequences + a leaf-labeled tree T
  - Goal: Find ancestral sequences implying minimum no. of changes (*most parsimonious*)
- **"Large parsimony problem"**:
  - Input: leaf sequences
  - Gaol: Find a most parsimonious tree (topology, leaf labeling and internal seqs.)

Algorithm for the Small Parsimony Problem (Fitch `71)

- Consider each site in a sequence separately
- Initialization: scan T in post-order, assign:
  - leaf vertex *m*: $S_m = \{\text{state at node } m\}$
  - internal node m with children *l*, *r*:

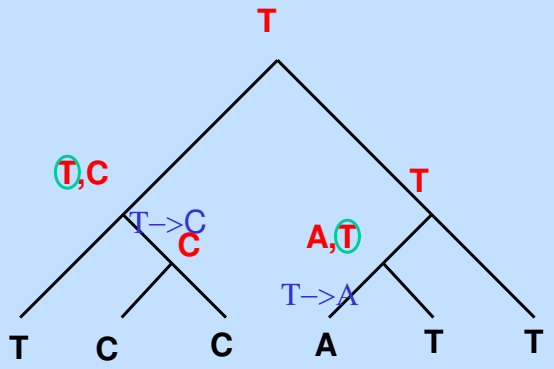
$$S_m = \begin{cases} S_l \cup S_r & \text{if } S_l \cap S_r = \phi \quad (i) \\ S_l \cap S_r & o/w \quad (ii) \end{cases}$$

- Solution Construction: scan tree in preorder, choose:
  - for the root choose $x \in S_{root}$
  - at node *m* with parent *k (already constructed*

    pick same state as k if possible; o/w - pick arbitrarily

time: O(m·n·k),
k = #states s; n=#nodes; m=#sites

Fitch's Alg for small parsimony



Walter Fitch

May 21, 1929 - March 10, 2011

- One of the most influential evolutionary biologists in the world, who established a new scientific field: **molecular phylogenetics**. He was a member in the National Academy of Sciences, the American Academy of Arts and Sciences and the American Philosophical Society. He co-founded and was the first president of the Society for Molecular Biology, which established the annually awarded Fitch Prize. Additionally, he was a founding editor of *Molecular Biology and Evolution*.
- Fitch was at the University of California, Irvine, until his death, preceded by three years at the University of Southern California and 24 years University of Wisconsin-Madison.

Weighted Small Parsimony

In Ex.

k states $s_1, \ldots, s_k$

C_{ij} = cost of changing from state i to j

Algorithm (Sankoff-Cedergren '88):

- Need: $S_k(x)$ – *best cost for the subtree rooted at x if state at x is k*
- For leaf x,

  $S_k(x) = 0$ *if state of x is k*

  ∞ *o/w*
- Scan T in postorder. At node a with children l, r
- $S_k(a) = \min_m(S_m(l)+C_{mk}) + \min_m(S_m(r)+C_{mk})$
- $\text{Opt}=\min_m(S_m(\text{root}))$

time: O(n·k)

David Sankoff

Over the past 30 years, Sankoff formulated and contributed to many of the fundamental problems in computational biology.
In **sequence comparison**, he introduced the quadratic version of the Needleman-Wunsch algorithm, developed the first statistical test for alignments, initiated the study of the limit behavior of random sequences with Vaclav Chvatal and described the multiple alignment problem, based on minimum evolution over a phylogenetic tree. In the study of **RNA secondary structure**, he developed algorithms based on general energy functions for multiple loops and for simultaneous folding and alignment, and performed the earliest studies of parametric folding and automated phylogenetic filtering.
Sankoff and Robert Cedergren collaborated on the first studies of the evolution of the genetic code based on tRNA sequences. His contributions to **phylogenetics** include early models for horizontal transfer, a general approach for optimizing the nodes of a given tree, a method for rapid bootstrap calculations, a generalization of the nearest neighbor interchange heuristic, various constraint, consensus and supertree problems, the computational complexity of several phylogeny problems with William Day, and a general technique for phylogenetic invariants with Vincent Ferretti. Over the last fifteen years he has focused on the evolution of genomes as the result of **chromosomal rearrangement** processes. Here he introduced the computational analysis of genomic edit distances, including parametric versions, the distribution of gene numbers in conserved segments in a random model with Joseph Nadeau, phylogeny based on gene order with Mathieu Blanchette and David Bryant, generalizations to include multi-gene families, including algorithms for analyzing genome duplication and hybridization with Nadia El-Mabrouk, and the statistical analysis of gene clusters with Dannie Durand. Sankoff is also well known in **linguistics** for his methods of studying grammatical variation and change in speech communities, the quantification of discourse analysis and production models of bilingual speech.

Large Parsimony Problem

Input: n x m matrix M:

- M_{ij} = state of j^{th} character of species i.
- $M_{i.}$ = label of i *(all labels are distinct)*

Goal:

Construct a phylogenetic tree T with n leaves and a label for each node, s.t.

- 1-1 correspondence of leaves and labels
- cost of tree is minimum.

- NP-hard

Branch & Bound (Hendy-Penny '89)

In Ex.

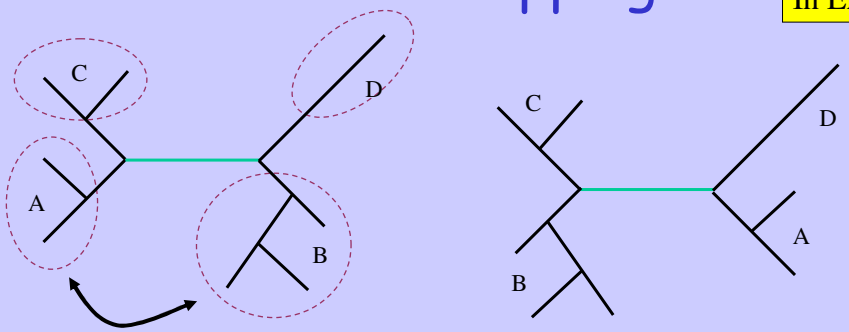
- enumerate all unrooted trees with increasing no. of leaves

Note: cost of tree with all leaves ≥ cost of subtree with some leaves pruned (and same labeling)
=> If cost of subtree ≥ best cost for full tree so far, then: can prune (ignore) all refinements of the subtree.

enumeration & pruning can be done in O(1) time per visited subtree.

Branch swapping

In Ex.



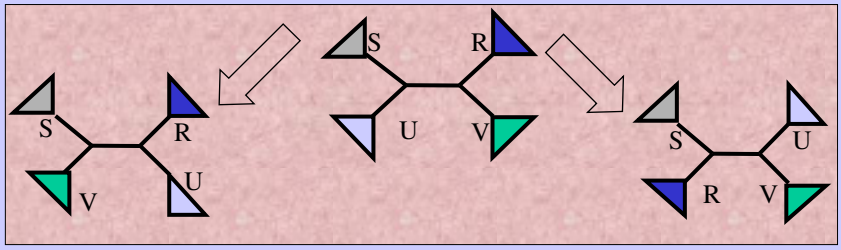
Each internal edge defines 4 sub-trees:

Can swap two such non-adjacent sub-trees

Nearest Neighbor Interchanges

In Ex.

- handles n-labeled trees
- T and T' are neighbors if one can get T' by following operation on T:



SV|RU **SU|RV** **SR|UV**

- Use the neighborhood structure on the set of solutions (all trees) via *hill climbing, annealing,* other heuristics...

Probabilistic approaches

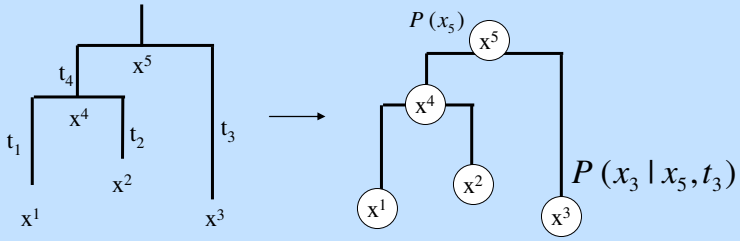
Likelihood of a Tree

- Given:
  - *n* aligned sequences $M = X^1,\ldots,X^n$
  - A tree *T*, leaves labeled with $X^1,\ldots,X^n$
- **reconstruction** *t*:
  - labeling of internal nodes
  - branch lengths
- Goal: Find optimal reconstruction t^* : One maximizing the likelihood $P(M/T,t^*)$

Likelihood (2)

- We need a model for computing *P(M/T,t*)*
- Assumptions:
  - Each character is independent
  - The branching is a Markov process:
    - The probability that a node x has a specific label is only a function of the parent node y and the branch length *t* between them.
- The probabilities *P(x/y,t) are known*

Modeling phylogeny as a Bayesian network



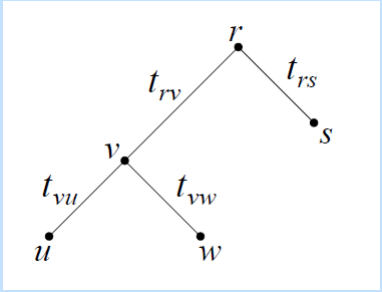
- BN with variables x1-x5 and local distributions $P(x_i \mid Pa_i, t_i) = P_{Pai \to x_i}(t_i)$

$$P(x^1,...,x^5 \mid T, t^*) =$$

$$P(x^1 \mid x^4, t_1)P(x^2 \mid x^4, t_2)P(x^3 \mid x^5, t_3)P(x^4 \mid x^5, t_4)P(x^5)$$

$$P(root) \prod_{\text{edge } u \to v} p_{u \to v}(t_{uv})$$

Calculating the Likelihood - Example



Inference in a BN

$$L = P(M|T) = \sum_r \sum_v P(r) \cdot P_{r\to s}(t_{rs}) \cdot P_{r\to v}(t_{rv}) \cdot P_{v\to u}(t_{vu}) \cdot P_{v\to w}(t_{vw})$$

Calculating the Likelihood – General equation

Assume that the branch lengths t_{uv} are known.

Let $\bar{t}$ be the branch lengths and R the rest of the reconstruction = the internal node labels

$$P(M|T,\bar{t}) = \prod_{\text{character } j} \left\{ \sum_{\text{reconstrcuction } R} P(M_{\bullet j}, R \mid T, \bar{t}) \right\}$$

$$= \prod_{\text{character } j} \left\{ \sum_{\text{reconstrcuction } R} \left(P(root) \prod_{\text{edge } u\to v} p_{u\to v}(t_{uv}) \right) \right\}$$

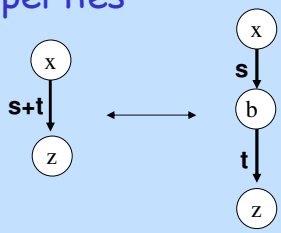
Independence of sites

Markov property independence of each branch

Additional Assumed Properties

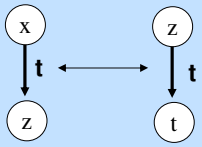
- Additivity:

$$P_{x\to z}(s+t) = \sum_b P_{x\to b}(s) P_{b\to z}(t)$$

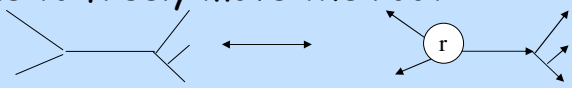


- Reversibility (symmetry):

$$P(x)P_{x\to y}(t) = P(y)P_{y\to x}(t)$$



- Provable under broad and reasonable assumptions
- Allows one to freely move the root



Efficient Likelihood Calculation
(Felsenstein '73)

Use dynamic programming

Define $S_j(a,v)$ = *Pr(subtree rooted in v | v_j = a)*

Initialization:

$\forall$ leaf *v* set $S_j(a,v) = 1$ if *v* is labeled by *a*, else $S_j(a,v) = 0$

Recursion:

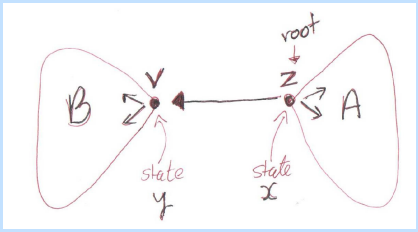
Traverse the tree in postorder: for each node *v* with children *u* and *w*, for each state *x*

$$S_j(x,v) = \left(\sum_y S_j(y,u) p_{x\to y}(t_{vu})\right)\left(\sum_y S_j(y,w) p_{x\to y}(t_{vw})\right)$$

Final Soln: $L = \prod_j \left(\sum_x S_j(x,root) P(x)\right)$

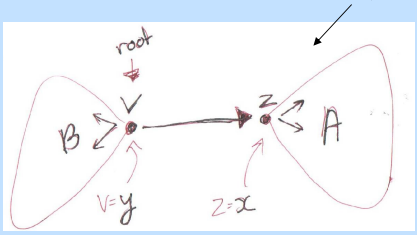
Complexity: $O(nmk^2)$
n species, m chars, k states

Finding Optimal Branch lengths



$$L = \sum_{x,y} P(B \mid v = y) P(v = y \mid z = x, t) P(z = x) P(A \mid z = x)$$

$S^z(y, v)$ $\quad p_{x \to y}(t)$ $\quad p(x)$ $\quad S^v(x, z)$

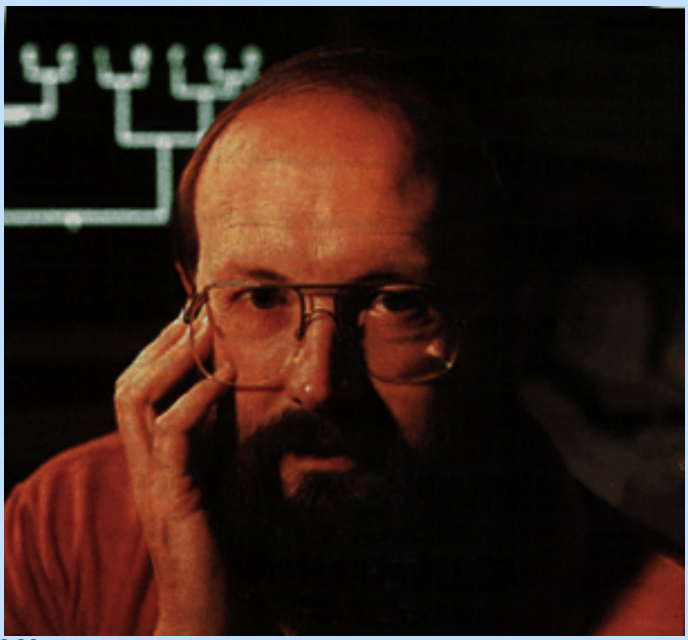


Finding Optimal Branch lengths

Optimizing the length of a single branch z-v can be done using standard optimization techniques

$$\log L = \sum_{j=1,\ldots,m} \log \sum_{x,y} S_j^z(y, v) P_{x \to y}(t) P(x) S_j^v(x, z)$$

- Under the symmetry assumption, each node can be made (temporarily) the root
- To heuristically optimize all the branch lengths: repeatedly optimize one branch at a time
  - No guaranteed convergence, but often works



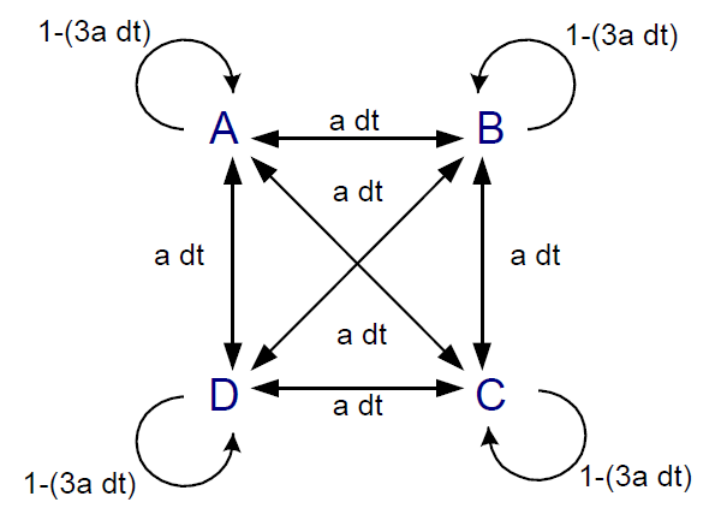
HOW DO WE FIGURE OUT THE TIMES?

Calculating $P_{u \to v}(t_{uv})$

Jukes-Cantor Model (J-K '69)

- Assumptions:
  - Each base in sequence has equal chance of changing
  - Changes to other 3 bases with equal probability
- Characteristics
  - Each base appears with equal frequency in DNA
  - The quantity a is the rate of change
  - During each infinitesimal time Δt a substitution occurs with probability $3a\Delta t$

Jukes-Cantor Model (J-K '69)



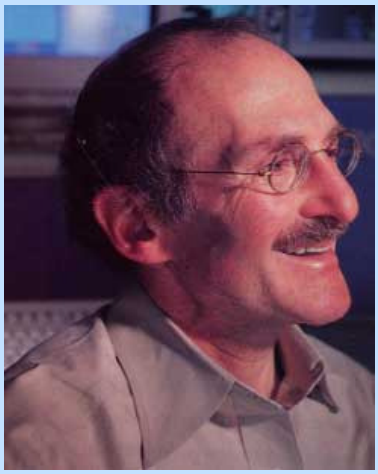
Jukes-Cantor Model (J-K '69)

- prob. that the nucleotide remains unchanged over t time units: $P_{\text{same}} = \frac{1}{4} + \frac{3}{4}e^{-4at}$
- Probability of specific change: $P_{A \to B} = \frac{1}{4} - \frac{1}{4}e^{-4at}$
- Probability of change: $P_{change} = \frac{3}{4} - \frac{3}{4}e^{-4at}$
- Note: For $t \to \infty$ $\quad P_{\text{change}} \to \frac{3}{4}$

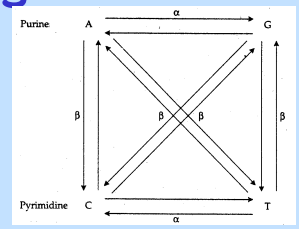
Charles Cantor

Boston University
Professor Emeritus, Biomedical Engineering
Professor of Pharmacology, School of Medicine
Ph.D., Biophysical Chemistry, University of California, Berkeley

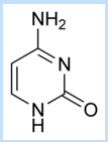
CSO Sequenom, San Diego, California.

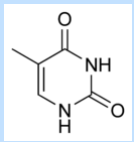


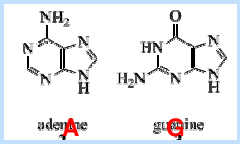
Other Models



- Kimura's 2-parameter model:
  - A,G - purines; C,T - pyrmidines
  - Two different rates
    - purine-purine or pyrmidine-pyrimidine (transitions)
    - purine-pyrmidine or pyrmidine-purine (transversions)
- Felsenstein '84 and Yano, Hasegawa & Kishino '85 extend the Kimura model to asymmetric base frequencies.


C


T



A G

FIN