Example files

Last update 25th January

Will contain problems and examples not examinable

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0						
C 1							
A 2							
T 3							
G 4							
T 5							

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	← -1					
C 1							
A 2							
T 3							
G 4							
T 5							

A
–

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	← -1	← -2	← -3	← -4	← -5	← -6
C 1							
A 2							
T 3							
G 4							
T 5							

ACGCTG

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	-1	-2	-3	-4	-5	-6
C 1	-1						
A 2	-2						
T 3	-3						
G 4	-4						
T 5	-5						

CATGT

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	-1	-2	-3	-4	-5	-6
C 1	-1	-1					
A 2	-2						
T 3	-3						
G 4	-4						
T 5	-5						

A
C

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	-1	-2	-3	-4	-5	-6
C 1	-1	-1	1				
A 2	-2						
T 3	-3						
G 4	-4						
T 5	-5						

AC
-C

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	-1	-2	-3	-4	-5	-6
C 1	-1	-1	1	0			
A 2	-2						
T 3	-3						
G 4	-4						
T 5	-5						

ACG
-C-

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	-1	-2	-3	-4		
C 1	-1	-1	1	0	-1		
A 2	-2						
T 3	-3						
G 4	-4						
T 5	-5						

ACGC
---C

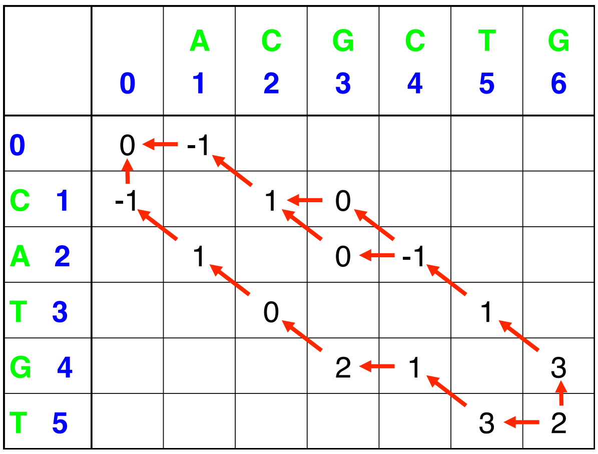
ACGC
-C--

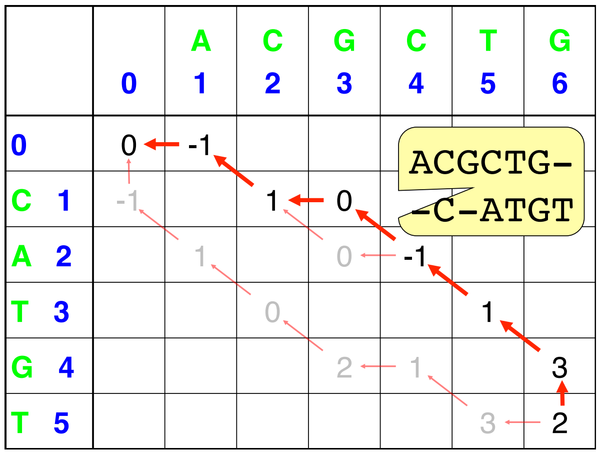
	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	-1	-2	-3	-4	-5	-6
C 1	-1	-1	1	0	-1	-2	-3
A 2	-2	1	0	0			
T 3	-3						
G 4	-4						
T 5	-5						

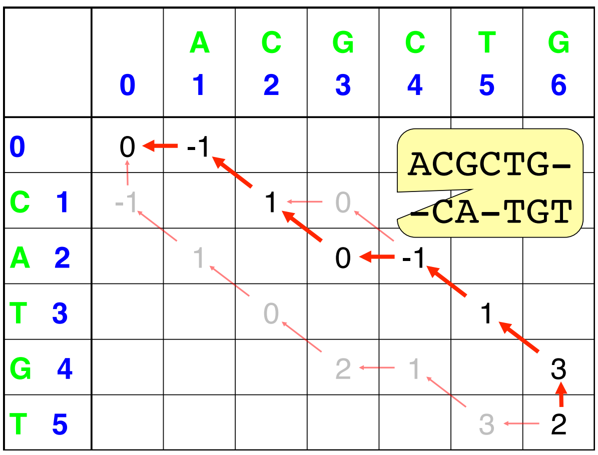
ACG
-CA

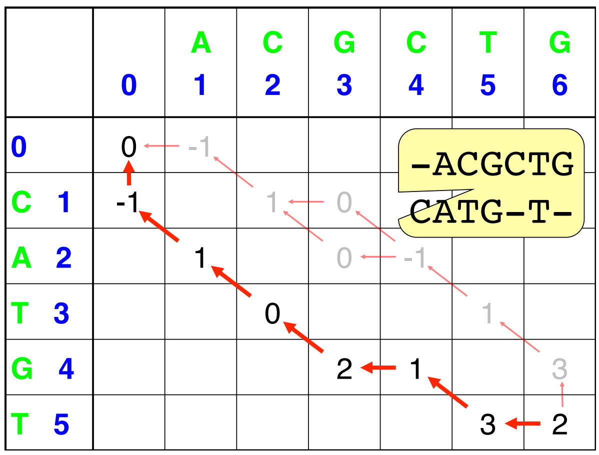
	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	-1	-2	-3	-4	-5	-6
C 1	-1	-1	1	0	-1	-2	-3
A 2	-2	1	0	0	-1	-2	-3
T 3	-3	0	0	-1	-1	1	0
G 4	-4	-1	-1	2	1	0	3
T 5	-5	-2	-2	1	1	3	2

	0	A 1	C 2	G 3	C 4	T 5	G 6
0	0	-1	-2	-3	-4	-5	-6
C 1	-1	-1	1	0	-1	-2	-3
A 2	-2	1	0	0	-1	-2	-3
T 3	-3	0	0	-1	-1	1	0
G 4	-4	-1	-1	2	1	0	3
T 5	-5	-2	-2	1	1	3	2




ACGCTG-
-C-ATGT


ACGCTG-
-CA-TGT


-ACGCTG
CATG-T-

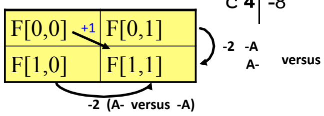
m = 1
s = 1
d = 2

y \ x	0	A 1	G 2	C 3
0	0	-2	-4	-6
A 1	-2			
A 2	-4			
A 3	-6			
C 4	-8			

We continue to fill the matrix using the recurrence rule

17

y \ x	0	A 1	G 2	C 3
0	0	-2	-4	-6
A 1	-2	1		
A 2	-4			
A 3	-6			
C 4	-8			



18

y \ x	0	A 1	G 2	C 3
0	0	-2	-4	-6
A 1	-2	1	-1	-3
A 2	-4	-1	0	
A 3	-6	-3		
C 4	-8	-5		

19

y \ x	0	A 1	G 2	C 3
0	0	-2	-4	-6
A 1	-2	1	-1	-3
A 2	-4	-1	0	-2
A 3	-6	-3	-2	-1
C 4	-8	-5	-4	-1

Conclusion: d(AAAC,AGC) = -1

20

- To reconstruct the best alignment, we record which case(s) in the recursive rule maximized the score

y \ x	0	A 1	G 2	C 3
0	0	← -2	← -4	← -6
A 1	-2	1	← -1	← -3
A 2	-4	-1	0	← -2
A 3	-6	-3	-2	-1
C 4	-8	-5	-4	-1

21

- We now trace back a path that corresponds to the best alignment

AAAC
AG-C

y \ x	0	A 1	G 2	C 3
0	0	← -2	← -4	← -6
A 1	-2	1	← -1	← -3
A 2	-4	-1	0	← -2
A 3	-6	-3	-2	-1
C 4	-8	-5	-4	-1

22

- Sometimes, more than one alignment has the best score

AAAC
A-GC

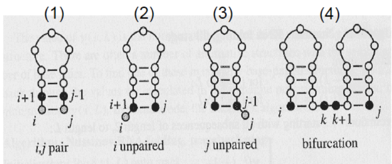
AAAC
-AGC

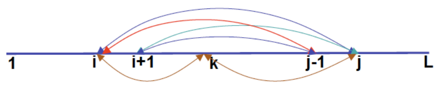
AAAC
AG-C

y \ x	0	A 1	G 2	C 3
0	0	← -2	← -4	← -6
A 1	-2	1	← -1	← -3
A 2	-4	-1	0	← -2
A 3	-6	-3	-2	-1
C 4	-8	-5	-4	-1

23

"Grow" from substructures





$$S(i,j) = max \begin{cases} S(i+1,j-1) + w(i,j) & (1) \\ S(i+1,j) & (2) \\ S(i,j-1) & (3) \\ max_{i<k<j} S(i,k) + S(k+1,j) & (4) \end{cases}$$

$w(i,j) = 1$ if i, j are complementary (i.e., GC, CG, AU or UA); 0 otherwise

Initialization

	G	G	G	A	A	A	U	C	C
G	0								
G	0	0							
G		0	0						
A			0	0					
A				0	0				
A					0	0			
U						0	0		
C							0	0	
C								0	0

Example:

GGGAAAUCC

$S(i,i) = 0 \quad \forall \quad 1 \le i \le L$ → the main diagonal

$S(i,i-1) = 0 \quad \forall \quad 2 \le i \le L$ → the diagonal below

L: the length of input sequence

Recursion

Fill up the table (DP matrix) -- diagonal by diagonal

→ j, ↓ i

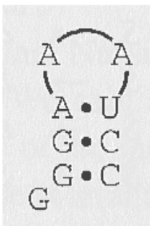
	G	G	G	A	A	A	U	C	C
G	0	0	0	0					
G	0	0	0	0	0				
G		0	0	0	0	0			
A			0	0	0	0	?		
A				0	0	0	1		
A					0	0	1	1	
U						0	0	0	0
C							0	0	0
C								0	0

$$S(i,j) = max \begin{cases} S(i+1,j-1) + w(i,j) & (1) \\ S(i+1,j) & (2) \\ S(i,j-1) & (3) \\ max_{i<k<j} S(i,k) + S(k+1,j) & (4) \end{cases} \qquad w(i,j) = \begin{cases} 1 & i,\ j \text{ are complementary} \\ 0 & otherwise \end{cases}$$

Traceback

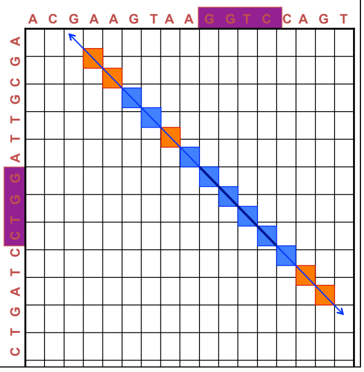
	G	G	G	A	A	A	U	C	C
G	0	0	0	0	0	0	1	2	3
G	0	0	0	0	0	0	1	2	3
G		0	0	0	0	0	1	2	2
A			0	0	0	0	1	1	1
A				0	0	0	1	1	1
A					0	0	1	1	1
U						0	0	0	0
C							0	0	0
C								0	0

The structure is:



Original BLAST: Example

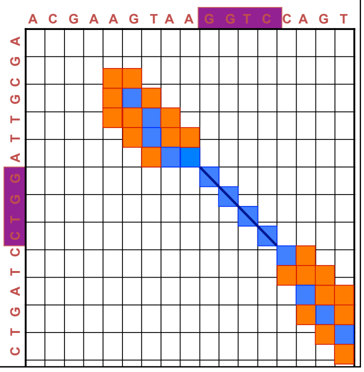
- *w* = 4
- Exact keyword match of GGTC
- Extend diagonals with mismatches until score is under 50%
- Output result
  GTAAGGTCC
  GTTAGGTCC



From lectures by Serafim Batzoglou (Stanford)

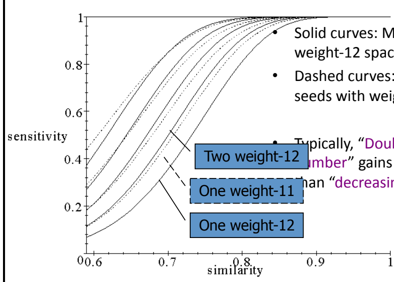
Gapped BLAST : Example

- Original BLAST exact keyword search, THEN:
- Extend with gaps around ends of exact match until score < *threshold*
- Output result
  GTAAGGTCCAGT
  GTTAGGTC-AGT



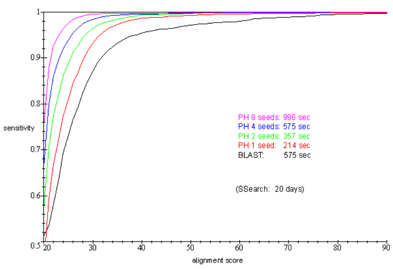
From lectures by Serafim Batzoglou (Stanford)

Simulated sensitivity curves:



- Solid curves: Multiple (1, 2, 4, 8, 16) weight-12 spaced seeds.
- Dashed curves: Optimal spaced seeds with weight = 11, 10, 9, 8.
- Typically, "Doubling the seed number" gains better sensitivity than "decreasing the weight by 1".

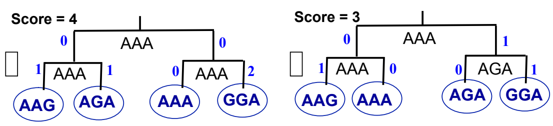
Sensitivity curves:



Most Parsimonious Tree

Parsimony-score:
Number of character-changes (mutations) along the evolutionary tree (tree containing labels on internal vertices)

Example:



Most parsimonious tree:
→ Tree with minimal parsimony score

Minimal Evolution Principle

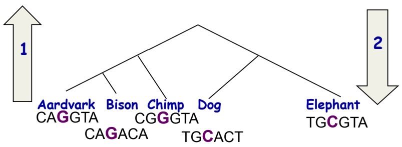
32

Fitch's Algorithm

Execute independently for each character:

1. Bottom-up phase: Determine set of possible states for each internal node
2. Top-down phase: Pick states for each internal node

Dynamic Programming framework

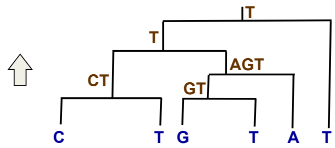


Fitch's Algorithm
Bottom-up phase

Determine set of possible states for each internal node

- Initialization: $R_i = \{s_i\}$
- Do a post-order (from leaves to root) traversal of tree
  - Determine R_i of internal node i with children j, k:

$$R_i = \begin{cases} R_j \cap R_k & if \ R_j \cap R_k \neq \phi \\ R_j \cup R_k & otherwise \end{cases}$$



Parsimony-score = # union operations

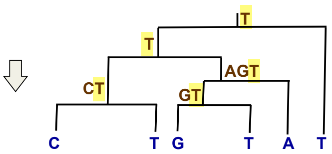
score = 3

Fitch's Algorithm
Top-down phase

Pick states for each internal node

- Pick arbitrary state in R_{root} for the root
- Do pre-order (from root to leaves) traversal of tree
  - Determine s_j of internal node j with parent i:

$$s_j = \begin{cases} s_i & if \quad s_i \in R_j \\ arbitrary \ \ state \in R_j & otherwise \end{cases}$$



Complexity: *O(mnk)*

#characters #states #taxa/nodes

score = 3

Weighted Parsimony
Sankoff's algorithm

- Each mutation **a↔b** costs differently - **S(a,b)**.
  1. Bottom-up phase: Determine $R_i(s)$ – cost of optimal state-assignment for subtree of i, when it is assigned state s.
  2. Top-down phase: Pick optimal states for each internal node

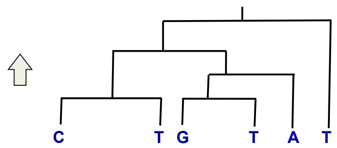
Fitch's algorithm as special case:
- R_i – set of states which yield minimal-cost subtree of i

Sankoff's Algorithm
Bottom-up phase

Determine $R_i(s)$ for each internal node

- Initialization: $R_i(s) = \begin{cases} 0 & if\ s_i = s \\ \infty & otherwise \end{cases}$
- Do a post-order (from leaves to root) traversal of tree
  - Determine R_i of internal node i with children j, k:

$$R_i(s) = \min_{s'}\{R_j(s') + S(s',s)\} + \min_{s'}\{R_k(s') + S(s',s)\}$$



Natural generalization
For non-binary trees

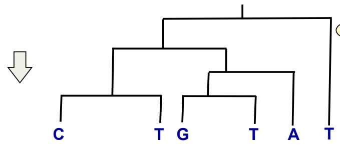
Remember pointers
$s \rightarrow s'$

37

Sankoff's Algorithm
Top-down phase

Pick states for each internal node

- Select minimal cost character for root (s minimizing $R_{root}(s)$)
- Do pre-order (from root to leaves) traversal of tree:
  - For internal node j, with parent i, select state that produced minimal cost at i (use pointers kept in 1st stage)



$$R_i(s) = \min_{s'}\{R_j(s') + S(s',s)\} + \min_{s'}\{R_k(s') + S(s',s)\}$$

Complexity: $O(mnk^2)$

#characters #states #taxa/nodes

38

Large Parsimony Problem

- Input: An n x m matrix M describing n species, each represented by an m-character string
- Output: A tree T with n leaves labeled by the n rows of matrix M, and a labeling of the internal vertices such that the parsimony score is minimized over all possible trees and all possible labelings of internal vertices
- Possible search space is huge, especially as n increases
- (2n – 3)!! possible rooted trees
- (2n – 5)!! possible unrooted trees
- Problem is NP-complete; Exhaustive search only possible w/ small n(< 10)

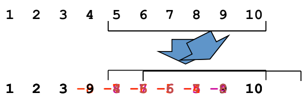
39

Solving NP-hard problems exactly is … unlikely

- Number of (unrooted) binary trees on n leaves is (2n-5)!!
- If each tree on **1000** taxa could be analyzed in **0.001** seconds, we would find the best tree in **2890 millennia**

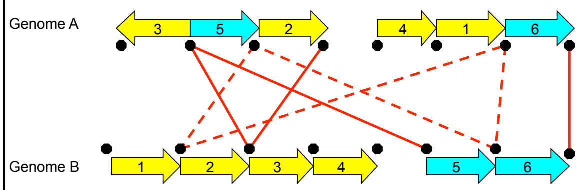
#leaves	#trees
4	3
5	15
6	105
7	945
8	10395
9	135135
10	2027025
20	2.2×10^{20}
100	4.5×10^{190}
1000	2.7×10^{2900}

Genomes Evolve by Rearrangements



- Inversion (Reversal)
- Transposition
- Inverted Transposition

The adjacency graph and the distance equation



C = number of cycles
I = number of odd paths
G = number of "genes"

$D = G - (C + I/2)$

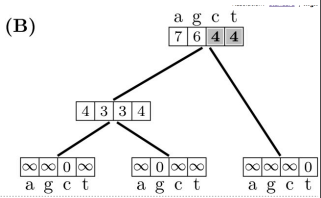
D = 6 - (1 + 2/2) = 4

Joint work with Julia Mixtacki and Jens Stoye

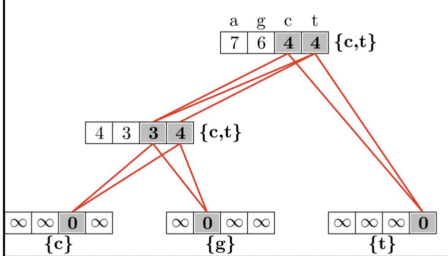
(A)

C_{ij}	a	g	c	t
a	0	1	3	3
g	1	0	3	3
c	3	3	0	1
t	3	3	1	0

(B)



Original Sankoff parsimony: up phase. (A) A matrix defining transition costs between states. (B) A sample phylogenetic tree with cost vectors calculated for all nodes using Algorithm 1.
Clemente *et al. BMC Bioinformatics* 2009 **10**:51 doi:10.1186/1471-2105-10-51



Original Sankoff parsimony: down phase. Ancestral states for the phylogeny, cost matrix and observed states of Figure 1, as obtained by Algorithm 2. Without solving ties, ancestral nodes are written next to each inner node. Red lines indicate the most parsimonious reconstructions after solving ties.
Clemente *et al. BMC Bioinformatics* 2009 **10**:51 doi:10.1186/1471-2105-10-51

Algorithm 1 (Original Sankoff algorithm: Up phase). *A procedure call Sankoff_Up(T, C, S) calculates the cost vector $S^{(p)}$ of all nodes p of the phylogeny T, given a cost matrix $C = (c_{ij})$.*

procedure *Sankoff_Up(T, C, S)*
  for all *nodes p of T in postorder* **do**
    if *p is a leaf* **then**
      for all *i in* 1, ..., *n* **do**
        if *state i observed at leaf p* **then**
          $S_i^{(p)} \leftarrow 0$
        else
          $S_i^{(p)} \leftarrow \infty$
    else
      $\{q, r\} \leftarrow$ *children of p*
      for all *i in* 1, ..., *n* **do**
        $S_i^{(p)} \leftarrow cost(q, i) + cost(r, i)$
function *cost(x, i)*
  min $\leftarrow \infty$
  for all *j in* 1, ..., *n* **do**
    if $c_{ij} + S_j^{(x)} <$ min **then**
      min $\leftarrow c_{ij} + S_j^{(x)}$
  return min

Algorithm 2 (Original Sankoff algorithm: Down phase). *A procedure call Sankoff_Down(x, T, C, S, S_{anc}) calculates the ancestral states $S_{anc}^{(p)}$ of all nodes p of the phylogeny T, given the root x of T, a cost matrix $C = (c_{ij})$ of transition costs between states, and the cost vectors $S^{(p)}$ for all nodes p of T as calculated by Sankoff_Up(T, C, S).*

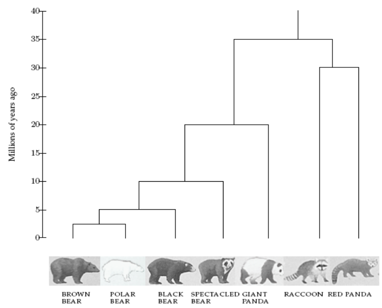
procedure *Sankoff_Down(x, T, C, S, S_{anc})*
  $S_{anc}^{(x)} \leftarrow \arg\min_i S_i^{(x)}$
  for all *j in* $S_{anc}^{(x)}$ **do**
    for all *child y of x* **do**
      Sankoff_Down(j, y, T, C, S, S_{anc})
procedure *Sankoff_Down(i, x, T, C, S, S_{anc})*
  min_states(i, x, C, S, S_{anc})
  for all *j in* $S_{anc}^{(x)}$ **do**
    for all *child y of x* **do**
    Sankoff_Down(j, y, T, C, S, S_{anc})
procedure *min_states(i, x, C, S, S_{anc})*
  min $\leftarrow \infty$
  for all *j in* 1, ..., *n* **do**
    if *x = root(T)* **then**
      trans_cost $\leftarrow S_j^{(x)}$
    else
      trans_cost $\leftarrow c_{ij} + S_j^{(x)}$
    if *trans_cost* < min **then**
      min $\leftarrow$ *trans_cost*
      $S_{anc}^{(x)} \leftarrow \{j\}$
    else if *trans_cost* = min **then**
      $S_{anc}^{(x)} \leftarrow S_{anc}^{(x)} \cup \{j\}$

The Giant Panda Riddle

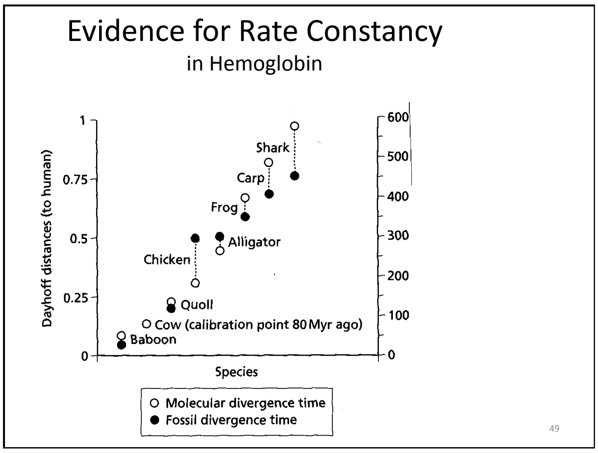
- Giant pandas look like bears but have features that are unusual for bears and typical for raccoons, e.g., they do not hibernate
- **Is the Giant panda closer to a bear or to a raccoon?**

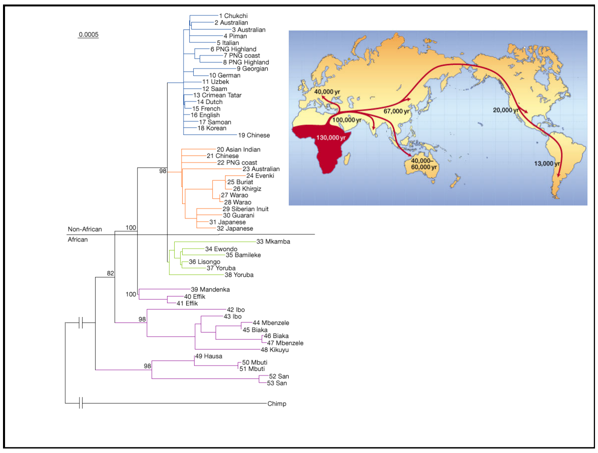
47

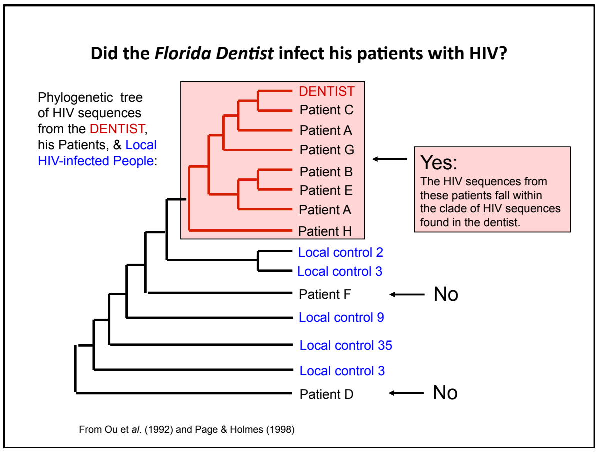
Evolutionary Tree of Bears and Raccoons

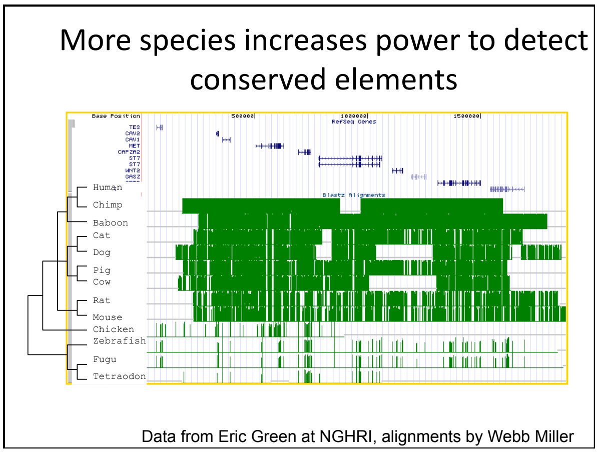


48


Evidence for Rate Constancy
in Hemoglobin
Dayhoff distances (to human)
1
0.75
0.5
0.25
0
600
500
400
300
200
100
0
Shark
Carp
Frog
Alligator
Chicken
Quoll
Cow (calibration point 80 Myr ago)
Baboon
Species
Molecular divergence time
Fossil divergence time
49




Did the Florida Dentist infect his patients with HIV?
Phylogenetic tree of HIV sequences from the DENTIST, his Patients, & Local HIV-infected People:
DENTIST
Patient C
Patient A
Patient G
Patient B
Patient E
Patient A
Patient H
Local control 2
Local control 3
Patient F
Local control 9
Local control 35
Local control 3
Patient D
Yes:
The HIV sequences from these patients fall within the clade of HIV sequences found in the dentist.
No
No
From Ou et al. (1992) and Page & Holmes (1998)


More species increases power to detect conserved elements
Human
Chimp
Baboon
Cat
Dog
Pig
Cow
Rat
Mouse
Chicken
Zebrafish
Fugu
Tetraodon
Data from Eric Green at NGHRI, alignments by Webb Miller

Approximate methods

- For larger data sets computing time becomes prohibitive and we only explore some subset of all possible trees (hoping that the optimal trees will be found in the subset explored)
- *Heuristic approaches* sacrifice the guarantee of optimality in favor of reduced computer time
- Use "hill climbing" methods. Initial tree starts the process, then we seek to improve its score
- When we can find no way to further improve the score, we stop.We don't know if we reached a *local* or a *global optimum*

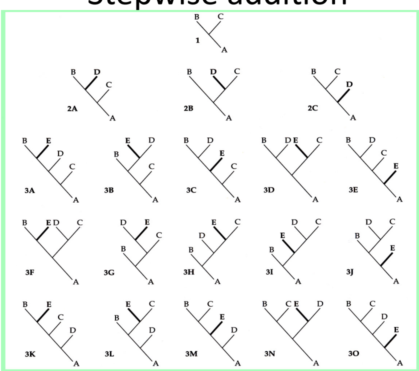
53

Initial trees

- May be obtained by *stepwise addition*, the most commonly used method
- Similar to exhaustive search but evaluate trees at every step, each time you add a new taxon and only follow the path derived from the optimal tree
- Which taxa do you choose first? Which do you connect next?
- These are "greedy algorithms"

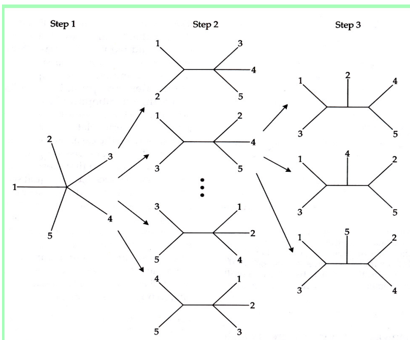
54

Stepwise addition



55

- Initial trees also may be obtained by *star decomposition*, another greedy algorithm

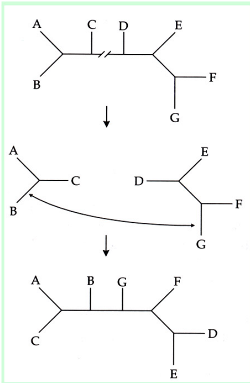


56

Branch swapping

- To improve the initial estimate we can perform sets of predefined rearrangements on the tree
- Any of these rearrangements amounts to a 'stab in the dark'
- Globally optimal trees may be several rearrangements away from the starting tree
- If a better tree is found, a new round of rearrangements is then performed in the new tree
- Several branch-swapping algorithms are available

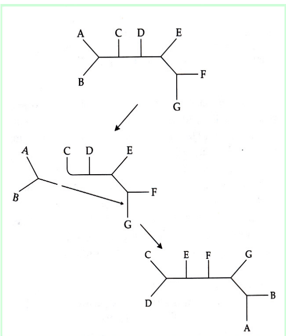
57



Branch swapping by tree bisection and reconnection (TBR)

1. Tree is bisected along a branch, yielding two disjunct subtrees
2. The subtrees are reconnected by joining a pair of branches, one from each subtree
3. All possible bisections and pairwise reconnections are evaluated

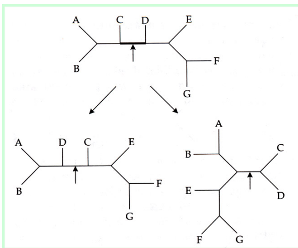
58



Branch swapping by subtree pruning and regrafting

1. A subtree is pruned from the tree (e.g. A,B)
2. The subtree is then regrafted to a different location on the tree
3. All possible subtree removals and reattachment points are evaluated

59



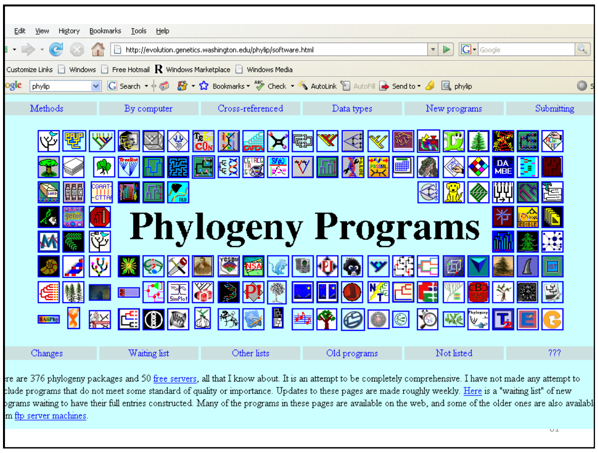
Branch swapping by nearest-neighbor interchanges (NNI)

1. Each interior branch of the tree defines a local region of four subtrees

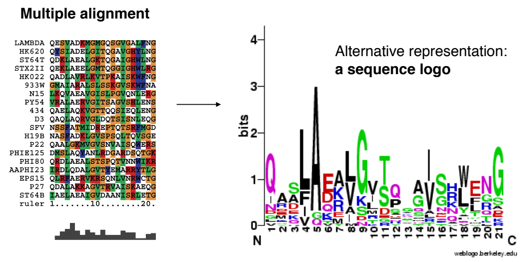
2. Interchanging a subtree on one side of the branch with one from the other constitutes an NNI

3. Two such rearrangements are possible for each interior branch (all interior branches are swapped)

60



Position-specific conservation and sequence variation



Logos represent sequence conservation in an easy to read format, with letter heights essentially representing the frequency with which a residue type occurs at a position in an alignment, relative to the frequency with which it would occur at random. The units of the y-axis are "bits" of information, which is to say that if a residue did not occur more often than expected at random, it would not offer us any information and the letter height would be zero. Note that the letter heights only become very high when a residue really dominates in the alignment, like Ala at the fifth position here.

weblogo server: http://weblogo.berkeley.edu
sequence logos paper: Schneider and Stevens, *Nucleic Acids Res* **18**, 6097 (1990).

Weight Matrices & Sequence Logos

Set of signal sequences:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	G	A	C	C	A	A	A	T	A	A	G	G	C	A
2	G	A	C	C	A	A	A	T	A	A	G	G	C	A
3	T	G	A	C	T	A	T	A	A	A	A	G	G	A
4	T	G	A	C	T	A	T	A	A	A	A	G	G	A
5	T	G	C	C	A	A	A	A	G	T	G	G	T	C
6	C	A	A	C	T	A	T	C	T	T	G	G	G	C
7	C	A	A	C	T	A	T	C	T	T	G	G	G	C
8	C	T	C	C	T	T	A	C	A	T	G	G	G	C

Position Frequency Matrix - PFM

A	0	4	4	0	3	7	4	3	5	4	2	0	0	4
C	3	0	4	8	0	0	0	3	0	0	0	0	0	4
G	2	3	0	0	0	0	0	0	1	0	6	8	5	0
T	3	1	0	0	5	1	4	2	2	4	0	0	1	0

Consensus sequence: B R M C W A W H R W G G B M

Position Weight Matrix - PWM

$$PWM: W_{b,i} = \log_2 \frac{p(b,i)}{p(b)}$$

A	-1.93	.79	.79	-1.93	.45	1.50	.79	.45	1.07	.79	.0	-1.93	-1.93	.79
C	.45	-1.93	.79	1.68	-1.93	-1.93	-1.93	.45	-1.93	-1.93	-1.93	-1.93	.0	.79
G	.0	.45	-1.93	-1.93	-1.93	-1.93	-1.93	-1.93	.66	-1.93	1.3	1.68	1.07	-1.93
T	.15	.66	-1.93	-1.93	1.07	.66	.79	.0	.79	-1.93	-1.93	-1.93	.66	-1.93
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Score for New Sequence

$$S = \sum_{l=1}^{w} W_{b,i}$$

T	T	G	C	A	T	A	A	G	T	A	G	T	C
.15	-.66	-1.93	1.66	.45	-.66	.79	.45	-.66	.79	.0	1.68	-.66	.79

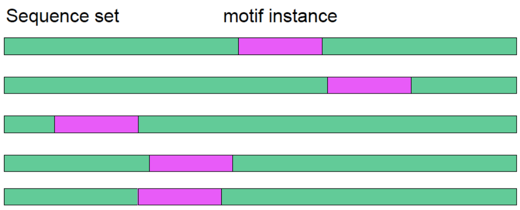
Sequence Logo & Information content

Wasserman and Sandelin (2004) 'Applied Bioinformatics for the Identification of Regulatory Elements" Nature Review

AlignAce by Hughes et al. 2000 http://atlas.med.harvard.edu/download/index.html,
BioProspector by Liu et al. 2001 http://motif.stanford.edu/distributions/r

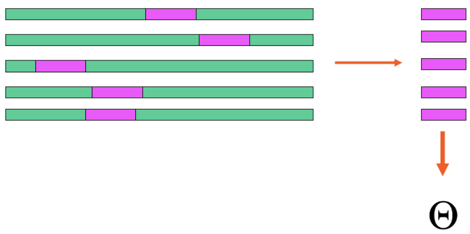
Gibbs Sampling Algorithm I

1. Select a random position in each sequence



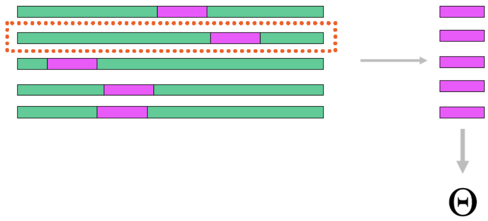
Gibbs Sampling Algorithm II

2. Build a weight matrix



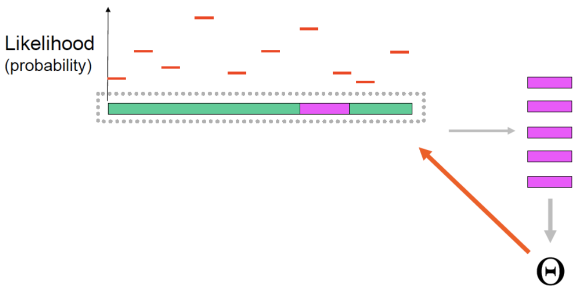
Gibbs Sampling Algorithm III

3. Select a sequence at random



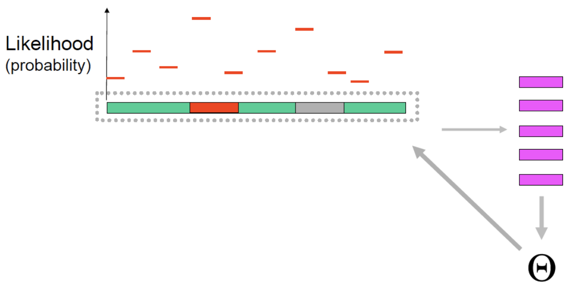
Gibbs Sampling Algorithm IV

4. Score possible sites in seq using weight matrix



Gibbs Sampling Algorithm V

5. Sample a new site proportional to likelihood

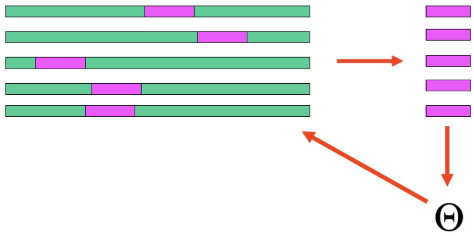


Gibbs Sampling Algorithm VI

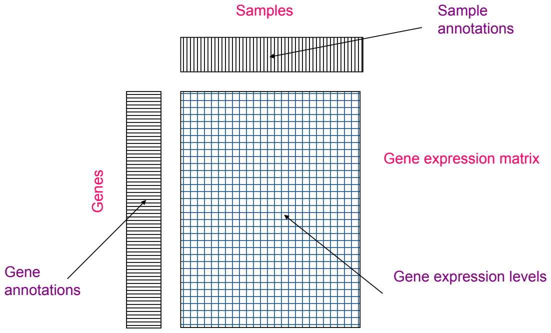
6. Update weight matrix

Gibbs Sampling Algorithm VII

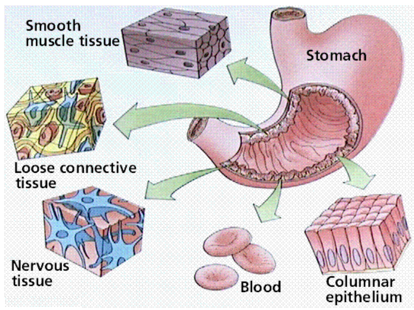
7. Iterate until convergence (no change in sites/Θ)



Gene expression database – a conceptual view

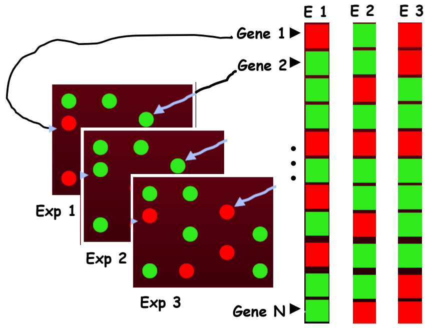


Networks of cells form tissues and organs

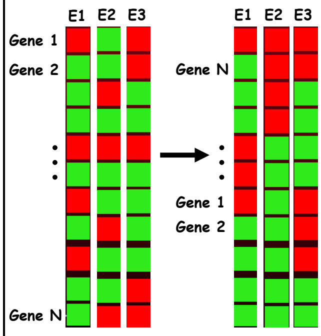


How is this variety encoded and expressed ?

Matrix of expression



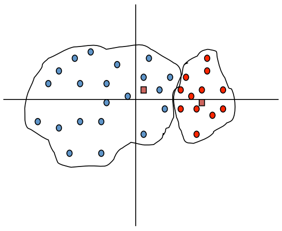
Why care about "clustering" ?



- **Discover functional relation**
  Similar expression→ functionally related
- **Assign function to unknown gene**
- **Find which gene controls which other genes**

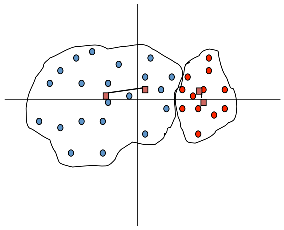
An example with k=2

1. We Pick k=2 centers at random
2. We cluster our data around these center points



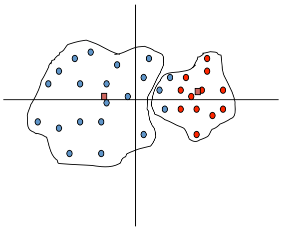
K-means example with k=2

3. We recalculate centers based on our current clusters



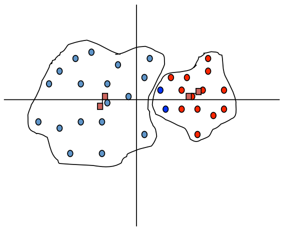
K-means example with k=2

4. We re-cluster our data around our new center points

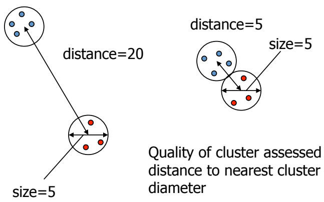


K-means example with k=2

5. We repeat the last two steps until no more data points are moved into a different cluster



Cluster Quality



Quality of cluster assessed by ratio of distance to nearest cluster and cluster diameter

PCA algorithm

- "Column-centered" matrix: A
- Covariance matrix: A^TA
- Eigenvalue Decomposition
  $A^TA = U \Lambda U^T$
  U: Eigenvectors (principle components)
  Λ: Eigenvalues
-