

Homework
Molecular Evolution and Phylogeny
CS 685, STA 695, PPA 784
Fall 2009

Prepared by J. W. Jaromczyk
Department of Computer Science
University of Kentucky

[FOR USE IN CS-685/STA-695/PPT-784 (FALL 2009). PLEASE DO NOT DISTRIBUTE.]

- Distributed: Monday, October 12, 2009 (last updated Sat Oct 17 17:53:27 EDT 2009 – see the comments section)
- Deadline: Tuesday, October 20, 2009
- Group work
- One submission from a group
- Useful sites:
 - <http://www.ncbi.nlm.nih.gov/>
 - <http://www.ch.embnet.org/>
 - <http://www.ebi.ac.uk/Tools/>
 - <http://www.molecularevolution.org/si/software/>
- Questions
 - Complexity 3-1 – 3-3
 - Parsimony 3-4 – 3-7
 - Pairwise Sequence Alignment 3-8 – 3-11
 - Multiple Sequence Alignment 3-12 – 3-15
- Comments, Clarifications, Corrections, Hints (last updated Sat Oct 17 17:53:27 EDT 2009)

Complexity

3-1 Most cells of sexually reproducing organisms have two sets of chromosomes; one of each from its parents. Before the fertilization, in a specialized cell division process the number of chromosomes is halved to a single chromosome set. What would have happened, if instead, the entire sets of genes from both parents were combined to double the number of chromosomes in each generation? Discuss and provide analysis, supported with numbers, of what would have been the result of such a process in a few hundreds of generations.

3-2 One of the steps in the maximum parsimony algorithm minimizes the score for a given topology, i.e., a binary tree whose leaves are labeled with states corresponding to the given character (site). Assume that a reconstruction (an assignment of states to the internal nodes) is processed in an exhaustive fashion; all possibilities are being tried and scored. Let us assume that the evaluation of the score for a fixed tree topology and one reconstruction of internal nodes takes 1 unit of time. How many units of time would it take to evaluate (compute the score over all possible reconfigurations) for one fixed topology for 5 taxa? For 10 taxa? For n taxa? Assume that each character can be in one of four states A, C, G, T. How many units of time is needed to evaluate the given topology for all the sites? What is the total cost of such an approach if all the topologies of size n are considered? If the time unit is 0.001 sec how long would it take to find the maximum parsimony tree for 10 taxa?

Remark: With a dynamic programming method each topology for one site can be scored in time proportional to size of the alphabet multiplied by the number of taxa, which is substantially faster than the exhaustive approach.

3-3 In the context of algorithms you will often see the following statement:

the running time of algorithm A is $O(n^2)$ for an input of size n .

Choose the most accurate meaning of this statement:

1. For all input instances of size n the running time grows asymptotically as fast as n^2 (i.e., both the running time and n^2 grow equally fast).
2. For some instances of size n the running time grows asymptotically as fast as n^2 (i.e., both the running time and n^2 grow equally fast for some instances).
3. The running time grows asymptotically as fast as or more slowly than n^2 and there are instances of size n for which the running time is asymptotically proportional to n^2 (i.e., not faster than n^2 but for some instances as fast as n^2).
4. The running time grows asymptotically as fast or more slowly than n^2 (i.e., not faster than n^2).

Justify your answer.

Parsimony

Parsimony is preference for the simplest explanation for an observation or phenomenon. In the context of phylogenetic trees it is expressed by finding a tree that explains the relation for a set of sequences using a minimal number of substitutions.

A high-level organization of the approach is:

scoring tree topologies: compute a cost for the given topology and possible reconstructions

finding the minimum evolution: search for the tree that minimizes the overall cost for all sites in the sequence.

There are various strategies for each of the steps. In particular, good strategies or heuristics are employed in the second step because the exhaustive search is not viable due to the fast growing running time.

3-4 Define the concept of an informative site for a multiple alignment considered for the maximum parsimony.

3-5 List all the informative sites (positions) in the following multiple alignment:

```
tx_1 CATTATTATTCTGCCTAGCAAACCTCAAATTATGAACGCACCCACAGTCGC
tx_2 CCCTGCTATTCTGCCTTGCAAACCTCAAACCTACGAACGAACTCACAGCCGC
tx_3 AATTATTATTCTGCCTAGCAAACCTCAAACCTACGAACGAACCCACAGCCGC
tx_4 CATTACTATTCTGCCTAGCAAACCTCAAACCTACGAACGCACTCACAGTCGC
tx_5 TCCTACTGTTCTGCCTAGCAAACCTCAAACCTACGAACGAACCCACAGCCGC
```

3-6 Based on the informative sites and how few there are, can you suggest a reasonable candidate for the maximum parsimony tree (without making all the calculations)?

3-7 Show a few different high-score topologies that have equal score for the first non-informative site in the above multiple alignment. Justify your answer. Assume score 1 for edges with matching endpoint labels, otherwise the edge score is 0.

Pairwise Sequence Alignment (PSA)

First efficient algorithms for computing pairwise alignments were designed by Needleman-Wunsch (global alignment) and Smith-Waterman (local alignment).

Both algorithms use the dynamic programming approach, a paradigm for solving optimization problems. In our case the objective is to maximize the score for two alignments.

The method hinges on the so-called *optimal subproblem structure* property of the PSA; the decision on how to best extend the current alignment for the next positions v_i and w_j can be made based on the quality of previously identified alignments for the shorter prefixes $v_1 \dots v_{i-1}$ and $w_1 \dots w_{j-1}$ of V and W , respectively.

A high-level organization of the algorithm is:

initialization: Initialize the table (the 0th row and columns) with multiples of the gap penalty

scoring: Apply the dynamic programming scoring formula

traceback: read the best alignment starting from the highest scores

The basic formula that assumes fixed gap penalties (for delete/insert is operations) and the cost of match/mismatch $\delta(v_i, w_j)$ given by a scoring matrix, for example Blosum62, is given below:

The directions of arrows indicate where the maximum is coming from, and are used in tracing back the optimal solutions.

	...	w_{j-1}	w_j	...
\vdots	\vdots	\vdots	\vdots	\vdots
v_{i-1}	...	$\leftarrow s(i-1, j-1)$	$\uparrow s(i-1, j)$...
v_i	...	$\swarrow s(i, j-1)$	$s(i, j)$...
\vdots	\vdots	\vdots	\vdots	\vdots

where

$$s(i, j) = \max \begin{cases} s(i-1, j) + \text{gap penalty} & (\uparrow) \\ s(i-1, j-1) + \delta(v_i, w_j) & (\swarrow) \\ s(i, j-1) + \text{gap penalty} & (\leftarrow). \end{cases}$$

The arrows in the above table are for the sake of example only. In the actual computation they are dictated by the formula that maximizes $s(i, j)$.

Note that when $s(i, j)$ is being evaluated, all the values on the right-hand side of the formula are known. The above formula is used for the global alignment, i.e., the cost of transforming the entire string V into the entire string W . The arrows can be placed together with the selected maximum (or maxima, if there is a tie); they help with tracing back optimal solutions. The initial (ground) conditions for the formula (the 0th column and row) is set depending on the gap model. In this case their are consecutive multiples of the (negated) gap penalty.

If, instead of aligning the entire string, one is interested in high-scored mutations of substrings of V and W into one another (local alignment) then the above formula changes into

$$s(i, j) = \max \begin{cases} 0 \\ s(i-1, j) + \text{gap penalty} & (\uparrow) \\ s(i-1, j-1) + \delta(v_i, w_j) & (\nearrow) \\ s(i, j-1) + \text{gap penalty} & (\leftarrow). \end{cases}$$

For example, global and local alignments for two sequences from Durbin *et al.* textbook but with different scoring matrix (Blosum62) and gap penalty (4) are given in Table 1 and 2. Try to trace back the optimal solutions by adding arrows. Typically, the arrows are added when the scoring matrix is computed. However, they can also be found by moving backwards from the lower-right score matrix position for the global alignments or any highest score position for the local alignment. Then, from the current position (i, j) we move left, up, or North-West to the position that returned the current score $s(i, j)$ as the maximum when $s(i, j)$ was computed. The process is continued until the position $(0, 0)$ for global, or score 0 (for local) is reached. Each path corresponds to an alignment: for example \nearrow next to $s(i, j)$ means that v_i and w_j are aligned (a match or mismatch). The entire alignment can be reconstructed following the path. There may be more than one path for the optimal score, resulting in more than one optimal alignment.

		H	E	A	G	A	W	G	H	E	E
	0	$\leftarrow -4$	$\leftarrow -8$	$\leftarrow -12$	-16	-20	-24	-28	-32	-36	-40
P	-4	-2	-5	-9	$\nearrow -13$	-17	-21	-25	-29	-33	-37
A	-8	-6	-3	-1	-5	$\nearrow -9$	-13	-17	-21	-25	-29
W	-12	-10	-7	-5	-3	-7	$\nearrow 2$	$\leftarrow -2$	-6	-10	-14
H	-16	-4	-8	-9	-7	-5	-2	0	$\nearrow 6$	2	-2
E	-20	-8	1	-3	-7	-8	-6	-4	2	$\nearrow 11$	7
A	-24	-12	-3	5	1	-3	-7	-6	-2	$\uparrow 7$	10
E	-28	-16	-7	1	3	0	-4	-8	-6	3	$\nearrow 12$

HEAGAWGHE-E

--P-AW-HEAE

Table 1: Score Matrix for Global Alignment – Blosum62 and gap penalty = 4

For some of the problems you will need to use Blosum50 or Blosum62 scoring matrices. It is a part of this task to find them.

3-7 Compute the missing scores in the score matrix (Table 3) and find the optimal alignment.

3-8 Table 4 shows score matrix for the same sequences but this time for the local alignment. The best local alignment is provided. What are the second best local scores and their corresponding local alignments?

3-10 Assuming a constant gap penalty, how does the penalty affect the alignment? In particular, do you expect more indels for the gap penalty -10 or -4? Why?

		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	4	0	4	0	0	0	0	0
W	0	0	0	0	2	0	15	11	7	3	0
H	0	8	4	0	0	0	11	13	19	15	11
E	0	4	13	9	5	1	7	9	15	24	20
A	0	0	9	17	13	9	5	7	11	20	23
E	0	0	5	13	15	12	8	4	7	16	25

AWGHE-E

AW-HEAE

Table 2: Score Matrix for Local Alignment – Blosum62 and gap penalty = 4

		V	E	T	E	R	I	N	A	R	I	A	N
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24
S	-2	-2	-3	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20
T	-4	-2	-3	2	0	-2	-4	-6	-8	-10	-12	-14	-16
A	-6	-4	-3	0	1	-1	-3	-5	-1	-3	-5	-7	-9
T	-8	-6	-5	2	0	0	-2	-3	-3	-2	-4	-5	-7
I	-10	-4	-6	0	-2	-2	5	3	1	-1	3	1	-1
S	-12	-6	-5	-2	-1	-3	3	6	4	2	1	4	2
T	-14	-8	-7	0	-2	-2	1	4	6	4	2	2	4
I	-16	-10	-9	-2	-4	-4	3	2					
C	-18	-12	-11	-4	-5	-6	1	1					
I	-20	-14	-13	-6	-7	-8	-1	-1					
A	-22	-16	-15	-8	-7	-9	-3	-2					
N	-24	-18	-16	-10	-8	-8	-5	4					

Table 3: Partial Score Matrix – Blosum50, gap penalty = 2

3-11 Taking the above question to the extreme, what kind of alignment is generated with the global alignment dynamic programming algorithm if the gap penalty is 0 and mismatches are penalized with a prohibitively large penalty.

Hint: Look at the optimal alignment

```
-----VETERI-NARIAN
STATIS--T--IC---IAN
```

resulting from the scoring matrix in Table 5. Important are only the matching symbols. The scores are computed with the gap penalty 0 and $-\infty$ for mismatches. You may assume that all matches have the same score.

		V	E	T	E	R	I	N	A	R	I	A	N
	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	2	0	0	0	1	1	0	0	1	1
T	0	0	0	5	3	1	0	0	1	0	0	0	1
A	0	0	0	3	4	2	0	0	5	3	1	5	3
T	0	0	0	5	3	3	1	0	3	4	2	3	5
I	0	4	2	3	1	1	8	6	4	2	9	7	5
S	0	2	3	4	2	0	6	9	7	5	7	10	8
T	0	0	1	8	6	4	4	7	9	7	5	8	10
I	0	4	2	6	4	2	9	7	7	5	12	10	8
C	0	2	1	4	3	1	7	7	6	4	10	11	9
I	0	4	2	2	1	0	6	5	6	4	9	9	8
A	0	2	3	2	1	0	4	5	10	8	7	14	12
N	0	0	2	3	2	0	2	11	9	9	7	12	21

VETERINARIAN
IST--I-C-IAN

Table 4: Score Matrix for local alignment – Blosum50, gap penalty = 2

		V	E	T	E	R	I	N	A	R	I	A	N
	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	5	5	5	5	5	5	5	5	5	5
A	0	0	0	5	5	5	5	5	9	9	9	9	9
T	0	0	0	5	5	5	5	5	9	9	9	9	9
I	0	0	0	5	5	5	9	9	9	9	13	13	13
S	0	0	0	5	5	5	9	9	9	9	13	13	13
T	0	0	0	5	5	5	9	9	9	9	13	13	13
I	0	0	0	5	5	5	9	9	9	9	13	13	13
C	0	0	0	5	5	5	9	9	9	9	13	13	13
I	0	0	0	5	5	5	9	9	9	9	13	13	13
A	0	0	0	5	5	5	9	9	13	13	13	17	17
N	0	0	0	5	5	5	9	15	15	15	15	17	23

-----VETERI-NARIAN
STATIS--T--IC---IAN

Table 5: Score Matrix – gap penalty 0, mismatches $-\infty$

Multiple sequence alignments (MSA)

Consider the following five sequences:

```

tx1 CATTATTATTCTGCCTAGCAAACCTCAAATTATGAACGCACCCACAGTCGC
tx2 CCCTGCTATTCTGCCTTGCAAACCTCAAACCTACGAACGAACCTCACAGCCGC
tx3 CATTATTATTCTGCCTAGCAAACCTCAAACCTACGAACGAACCCACAGCCGC
tx3 CATTACTATTCTGCCTAGCAAACCTCAAACCTACGAACGCACTCACAGTCGC
tx4 CCCTACTGTTCTGCCTAGCAAACCTCAAACCTACGAACGAACCCACAGCCGC

```

	1	2	3	4	5
1		239	273	272	253
2			273	275	298
3				277	287
4					273

Table 6: Matrix of scores from pairwise alignments for the 5 taxa

	1	2	3	4	5
1		4.18	3.66	3.67	3.95
2			3.66	3.63	3.35
3				3.61	3.48
4					3.66

Table 7: Distance matrix for the 5 taxa

The scores for pairwise global alignments using Needleman-Wuensch algorithm with Blosum62 scoring matrix and the gap penalty 4 are tabulated in Table 6. Based on the scores, a distance matrix between the sequences was computed. In this case, the distance between two taxa is arbitrarily set to 1000 divided by the pairwise alignment score in Table 7. The higher the score the more related sequences are so their pairwise distance is smaller. From the distance matrix a guide tree, see Figure 1, was constructed with the NJ (Neighbor Joining) method.

Computing a distance matrix and a guide tree (albeit using more sophisticated methods) are two basic steps for a class of multiple alignment algorithms; that class includes ClustalW.

3-12 Use ClustalW to compute a multiple alignment for the given five sequences. Select and record the parameters, in particular the scoring matrix. In the ClustalW output to your experiment look for the following elements:

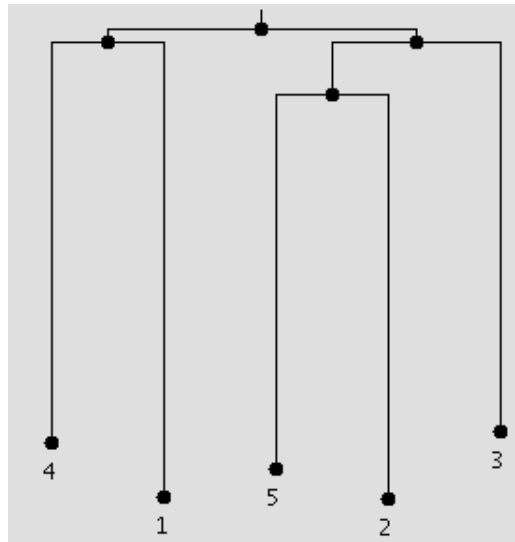


Figure 1: A guide tree for 5 sequences

- guide tree and the distance matrix from ClustalW – does the matrix agree, at least in the relative terms, with the above distance matrix computed from the pairwise alignment?
 - the phylogeny tree corresponding to the computed MSA. Does it agree with our guide tree from Figure 1?
- 3-13** Repeat the same process using a different set of parameters, specifically the scoring matrix. Describe differences in the outputs.
- 3-14** For the above five sequences compute their maximum parsimony phylogeny tree using Phylip software. Compare the tree to one obtained with ClustalW.
- 3-15** [This part of the homework assignment can be returned separately in the next class following the deadline for this homework]

Identify eight to twelve organisms or proteins and find their corresponding sequences in one of the data banks (e.g., NCBI) or other sources. Your selection of the set of sequences should be ideally guided by your interest (life sciences students in the group). Please do not select sequences commonly used and easily available in books, materials published on the Web or journal/conference papers.

For the set of the selected sequences, construct a multiple alignment and phylogeny trees. Use at least two different multiple alignment software packages (ClustalW and another one of your choice) plus the maximum parsimony method from Phylip. Collect results, analyze and describe them in a brief report. Conduct the experiments and report the results so they can be reproduced.

Comments, Clarifications, Corrections, Hints (last updated Oct 17, 2009)

- Answer question 3-2 *concurrently* with answering questions on Parsimony (this is a suggestion). Question 3-2 refers to unrooted trees. It requires understanding that the best tree is selected by selecting the topology (unrooted tree – a binary tree plus specific labeling of its leaves) that minimizes the score over all sites (positions in the sequences) and reconstructions (assigning states [e.g., letters A, C, G, T] to the internal nodes). Question 3-2 asks, among others, about the number of reconstructions to consider if one computes the score by considering each possible reconstruction. In other words, a naïve algorithm would select the best topology by iterating the following process for all sites, and all topologies, and each reconstruction. For the currently analyzed topology, currently analyzed site, and currently selected reconstruction the score is computed by adding the scores for all the edges. The score of an edge is 1 if the labels (states) of its endpoints are identical and 0 otherwise.

For example, if the sequences are

```

site 123 456 789
tx_A AAC TTC CAC
tx_B ACA TTC ATT
tx_C ACC TTT ACT
tx_D ACA TTC ATC
tx_E ACC TTC ATT

```

the 9th site is (C, T, T, C, T) for (tx_A, tx_B, tx_C, tx_D, tx_E), respectively. Assume that we are considering the following topology: (first topology from <http://www.cs.uky.edu/~jurek/cs685/cs685f09/notes/phylo5trees-DOC100109.pdf>)

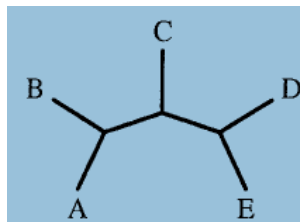


Figure 2: A sample topology for 5 taxa

Then the labels (A, B, C, D, E) corresponding to (tx_A, tx_B, tx_C, tx_D, tx_E) will be replaced with (C, T, T, C, T) (the 9th site). If the considered reconstruction places (T, C, T) in the internal nodes, from left to right, then the edges are: (CT, TT, TC, TC, CT, CT, TT) and the score is 2 (0+1+0+0+0+0+1). For the reconstruction (T, T, T) the score is 5 (why?). To answer question 3-2 one needs to know general formulas for the number of topologies for n taxa (leaves) and the number of internal nodes for a topology with n leaves (to compute the number of possible reconstructions). Those combinatorial formulas were given in class, and are available in our textbook and from numerous references that discuss the maximum parsimony.

Finally, the maximum parsimony (minimum evolution) tree is selected as the topology that minimizes the sum of scores over all the sites.

- typo (swapped arrows) corrected in the formulas for $s(i, j)$; page 4 and 5;
- The phrase “as fast as” is clarified in question 3-3, page 2 by adding comments (in parentheses.) In the context of algorithms, the big- O notation, such as $O(g(n))$, means that the running time of an analyzed algorithm is bounded asymptotically from above by $g(n)$. Equivalently, we can say that $g(n)$ is an upper bound for the running time of the algorithm. Ideally, tight upper bounds (ones that are met for some instances of the problem being solved by the algorithm) are preferred but the big- O notation does not require that even if the worst-case (the most difficult instances of the problem) running time is considered. If the running time grows asymptotically as fast as $g(n)$ the big- Θ notation is used; in this case $g(n)$ is more than an upper bound – it is an exact bound. So why is the big- O notations used instead of big- Θ ? There are two major reasons:
 - Although the worst case running time may be proportional to $g(n)$, there may also be instances that require much less time contrary to what big- Θ would imply.
 - In some cases exact analysis of the algorithm (that is, establishing a tight bound on the running time) is technically difficult, cumbersome, unnecessary, or unknown. In such cases a reasonable upper bound suffices; big- O provides an adequate notation.