CS 466
Introduction to Bioinformatics
Lecture 12

Mohammed El-Kebir

October 17, 2018

Course Announcements

10/17/2018	12	MSA 4	HW3 -- MSA
10/22/2018	14	Phylogeny I	
10/24/2018	15	Phylogeny II	
10/29/2018	16	Phylogeny III	
10/31/2018	17	Phylogeny IV	HW4 -- Phylogeny
11/5/2018	18	Assembly I	
11/7/2018	19	Assembly II	
11/12/2018	20	HMM I	
11/14/2018	21	HMM II	HW5 -- HMM, implementation Project proposal deadline
11/19/2018		Thanksgiving	
11/21/2018		Thanksgiving	
11/26/2018	22	Pattern Matching I	
11/28/2018	23	Pattern Matching II	
12/3/2018	24	Review	
12/5/2018	25	Final	
12/10/2018	26		
12/14/2018	27	Project deadline	

Course Project

Project

There are three kinds of projects.

1. Implement an algorithm discussed in class, and make it available on Github.
2. Benchmark algorithms discussed in class that solve the same problem on simulated or real data. Write a report about your findings.
3. Write a small survey paper, summarizing state-of-the-art algorithms for a specific computational biology problem.

Project proposal due on Nov. 14
(Motivation, Datasets/papers, Planned method/experiments, Timeline)

Project report due on Dec. 14

Outline

- Scoring matrices
- Tree/star alignment
- Progressive alignment methods

Reading:

- Material based on Chapter 14.6 in book “Algorithms on Strings, Trees and Sequences” by Dan Gusfield
- Chapter 6.7 in Jones and Pevzner
- Lecture notes

Substitution Matrices

- Given a pair $(\mathbf{v}, \mathbf{w})$ of aligned sequences, we want to assign a score that measure the **relative likelihood** that the sequences are **related** as opposed to being **unrelated**
- We need two models:
  - Random model R: each letter $a \in \Sigma$ occurs independently with probability q_a
  - Match model M: aligned pair $(a, b) \in \Sigma \times \Sigma$ occur with joint probability $p_{a,b}$

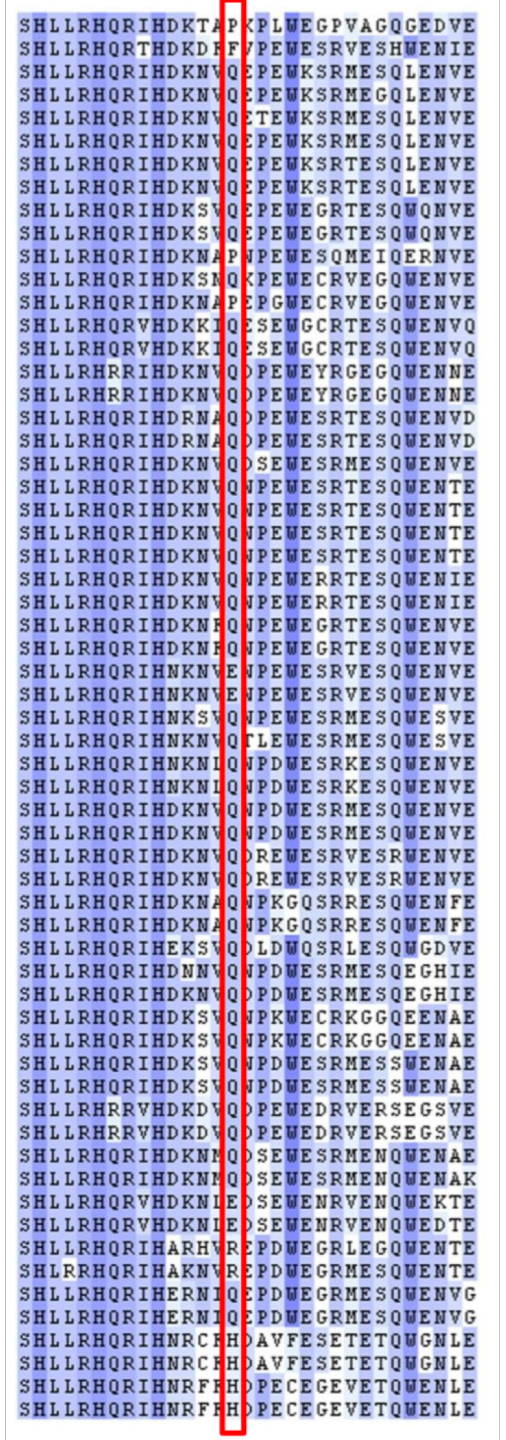
$$\Pr(\mathbf{v}, \mathbf{w}|\, R) = \prod_i q_{v_i} \cdot \prod_i q_{w_i}$$

$$\Pr(\mathbf{v}, \mathbf{w}|\, M) = \prod_i p_{v_i, w_i}$$

$$\log \frac{\Pr(\mathbf{v}, \mathbf{w}|\, M)}{\Pr(\mathbf{v}, \mathbf{w}|\, R)} = \sum_i s(v_i, w_i) \text{ where } s(a, b) = \log \frac{p_{a,b}}{q_a q_b}$$

BLOSUM (Blocks Substitution Matrices)

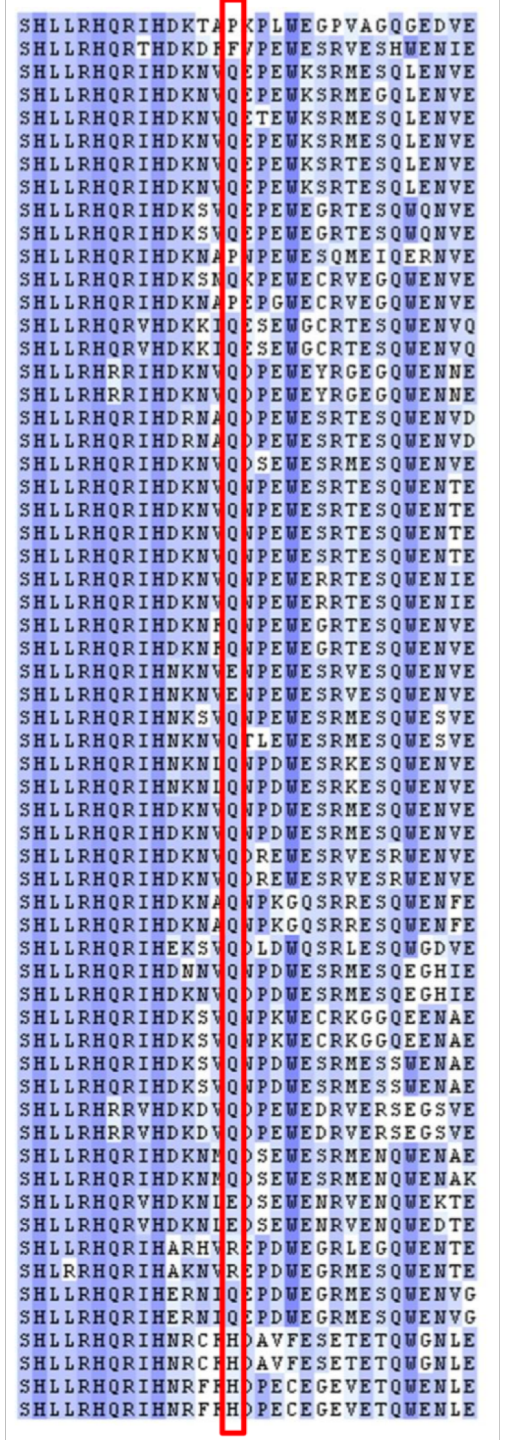
- Henikoff and Henikoff, 1992
- Computed using **ungapped** alignments of protein segments (blocks) from BLOCKS database
- Thousands of such blocks go into computing a single BLOSUM matrix
- Example of a one such block (right):
  - 31 positions (columns)
  - 61 sequences (rows)
- Given threshold L, block is pruned down to largest set C of sequences that have at least $L\%$ sequence identity to another sequence in C
  - How to compute C?



BLOSUM (Blocks Substitution Matrices)

$$\log \frac{\Pr(\mathbf{v}, \mathbf{w} | R)}{\Pr(\mathbf{v}, \mathbf{w} | M)} = \sum_i s(v_i, w_i) \text{ where } s(a,b) = \frac{1}{\lambda} \log \frac{p_{a,b}}{q_a q_b}$$

- Null model frequencies $q_a q_b$ of letters a and b:
  - Count the number of occurrences of a (b) in all blocks
  - Divide by sum of lengths of each block (sequences * positions)
- Match model frequency $p_{a,b}$:
  - Count the number of pairs (a, b) in all columns of all blocks
  - Divide by the total number of pairs of columns:
    - $\sum_C n(C) \binom{m(C)}{2}$
    - $m(C)$ is the number of sequences in block C
    - $n(C)$ is the number of positions in block C



BLOSUM (Blocks Substitution Matrices)

$$\log \frac{\Pr(\mathbf{v}, \mathbf{w}| R)}{\Pr(\mathbf{v}, \mathbf{w}| M)} = \sum_i s(v_i, w_i) \text{ where } s(a,b) = \frac{1}{\lambda} \log \frac{p_{a,b}}{q_a q_b}$$

- Null model frequencies $q_a q_b$ of letters a and b:
  - Count the number of occurrences of a (b) in all blocks
  - Divide by sum of lengths of each block (sequences * positions)
- Match model frequency $p_{a,b}$:
  - Count the number of pairs (a, b) in all columns of all blocks
  - Divide by the total number of pairs of columns:
    - $\sum_C n(C)\binom{m(C)}{2}$
    - $m(C)$ is the number of sequences in block C
    - $n(C)$ is the number of positions in block C

Example: ($\lambda = 0.5$)

```
A A T
S A L
T A L
T A V
A A L
```

$$q_A = \frac{7}{15}$$

$$q_T = \frac{3}{15}$$

$$p_{A,T} = \frac{4}{30}$$

$$s(A,T) = 2 \cdot \log \frac{\frac{4}{30}}{\frac{7}{15} \cdot \frac{3}{15}} \approx 0.3$$

BLOSUM62

$$\log \frac{\Pr(\mathbf{v}, \mathbf{w}| R)}{\Pr(\mathbf{v}, \mathbf{w}| M)} = \sum_i s(v_i, w_i) \text{ where } s(a,b) = \frac{1}{\lambda} \log \frac{p_{a,b}}{q_a q_b}$$

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	−1	5																		
Asn	−2	0	6																	
Asp	−2	−2	1	6																
Cys	0	−3	−3	−3	9															
Gln	−1	1	0	0	−3	5														
Glu	−1	0	0	2	−4	2	5													
Gly	0	−2	0	−1	−3	−2	−2	6												
His	−2	0	1	−1	−3	0	0	−2	8											
Ile	−1	−3	−3	−3	−1	−3	−3	−4	−3	4										
Leu	−1	−2	−3	−4	−1	−2	−3	−4	−3	2	4									
Lys	−1	2	0	−1	−3	1	1	−2	−1	−3	−2									
Met	−1	−1	−2	−3	−1	0	−2	−3	−2	1	2	−1	5							
Phe	−2	−3	−3	−3	−2	−3	−3	−3	−1	0	0	−3	0	6						
Pro	−1	−2	−2	−1	−3	−1	−1	−2	−2	−3	−3	−1	−2	−4	7					
Ser	1	−1	1	0	−1	0	0	0	−1	−2	−2	0	−1	−2	−1	4				
Thr	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	1	5			
Trp	−3	−3	−4	−4	−2	−2	−3	−2	−2	−3	−2	−3	−1	1	−4	−3	−2	11		
Tyr	−2	−2	−2	−3	−2	−1	−2	−3	2	−1	−1	−2	−1	3	−3	−2	−2	2	7	
Val	0	−3	−3	−3	−1	−2	−2	−3	−3	3	1	−2	1	−1	−2	−2	0	−3	−1	4

This explains some details in BLOSUM62 that may seem counterintuitive at first glance. For instance, tryptophan (W/W) pairs score +11, while leucine (L/L) pairs only score +4; why shouldn't all identitites get the same score? The rarer the amino acid is, the more surprising it would be to see two of them align together by chance. In the homologous alignment data that BLOSUM62 was trained on, leucine/leucine (L/L) pairs were in fact more common than tryptophan/tryptophan (W/W) pairs (p_{LL} = 0.0371, p_{WW} = 0.0065), but tryptophan is a much rarer amino acid (f_L = 0.099, f_W = 0.013). Run those numbers (with BLOSUM62's original λ = 0.347) and you get +3.8 for L/L and +10.5 for W/W, which were rounded to +4 and +11.

https://doi.org/10.1038/nbt0804-1035

Outline

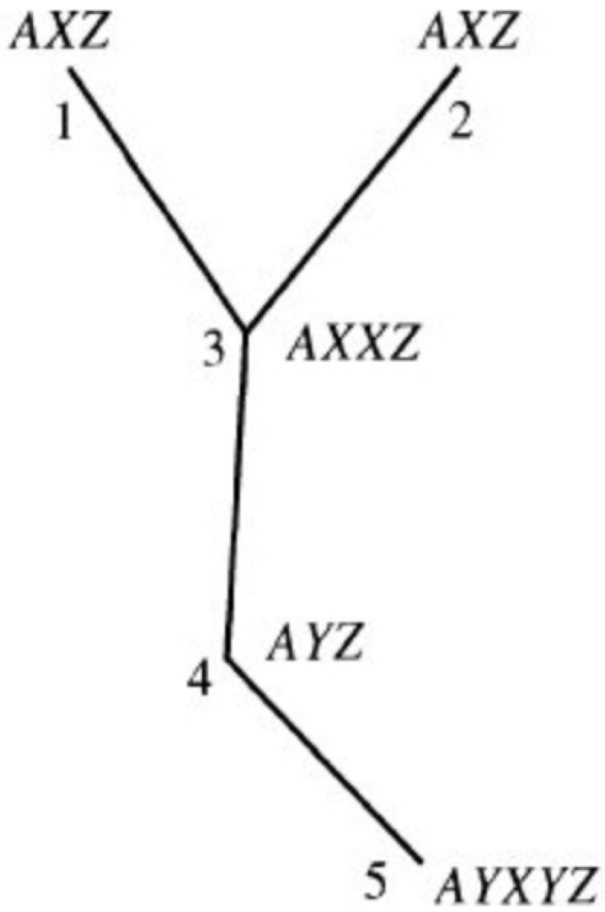
- Scoring matrices
- Tree/star alignment
- Current progressive alignment methods

Reading:

- Material based on Chapter 14.6 in book “Algorithms on Strings, Trees and Sequences” by Dan Gusfield
- Chapter 6.7 in Jones and Pevzner
- Lecture notes

Example – Tree Alignment

a)



b)

3	A	X	X	_	Z
1	A	X	_	_	Z
2	A	_	X	_	Z
4	A	Y	_	_	Z
5	A	Y	X	X	Z

Figure 14.6: a. A tree with its nodes labeled by a (multi)set of strings, b. A multiple alignment of those strings that is consistent with the tree. The pairwise scoring scheme scores a zero for each match and a one for each mismatch or space opposite a character. The reader can verify that each of the four induced alignments specified by an edge of the tree has a score equal to its respective optimal distance. However, the induced alignment of two strings which do not label adjacent nodes may have a score greater than their optimal pairwise distance.

Outline

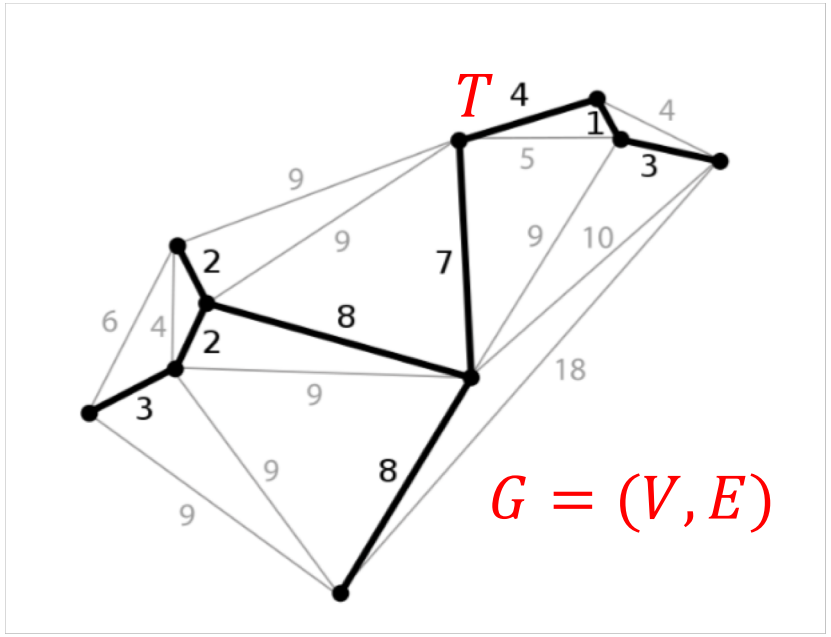
- Multiple sequence alignment
- Scoring matrices
- Tree/star alignment
- Current progressive alignment methods

Reading:

- Material based on Chapter 14.6 in book “Algorithms on Strings, Trees and Sequences” by Dan Gusfield
- Chapter 6.7 in Jones and Pevzner
- Lecture notes

Progressive Alignment – Feng and Doolittle (1987)

1. Compute pairwise sequence alignments of n sequences
2. Generate complete graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$
3. Compute a (rooted) minimum spanning tree T of G
4. Perform sequence-sequence, sequence-profile and profile-profile alignment to construct MSA according to guide tree T



Minimum spanning tree is a tree T spanning all vertices of G with minimum total weight

'Once a gap, always a gap'

Progressive Alignment – ClustalW (1994)

- Widely used alignment method by Thompson, Higgins and Gibson (1994)
- W stands for weighted:
  - Input sequences are weighted to compensate for biased representation
  - Different substitution matrices depending on expected similarity in guide tree (BLOSUM80 for closely related sequences, and BLOSUM50 for distant sequences)
  - Position-specific gap-open and gap-extend penalties depending on context (hydrophobic vs. hydrophilic)

Three steps:

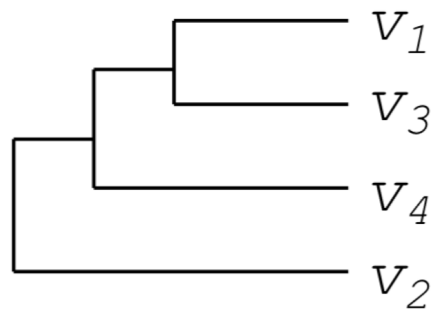
1. Construct pairwise alignments
2. Build guide tree T using neighbor joining*
3. Progressive alignment guided by T

ClustalW – Step 2: Guide Tree

Create Guide Tree using the similarity matrix

("cluster" distances. Details to come…)

	v_1	v_2	v_3	v_4
v_1	-			
v_2	.17	-		
v_3	.87	.28	-	
v_4	.59	.33	.62	-



ClustalW uses the neighbor-joining method

Guide tree roughly reflects evolutionary relationships

Calculate:

$v_{1,3}$ = alignment (v_1, v_3)

$v_{1,3,4}$ = alignment(($v_{1,3}$),v_4)

$v_{1,2,3,4}$ = alignment(($v_{1,3,4}$),v_2)

ClustalW – Step 3: Progressive Alignment

- Start by aligning the two most similar sequences
- Following the guide tree, add in the next sequences, aligning to the existing alignment
- Insert gaps as necessary

```
FOS_RAT         PEEMSVTS-LDLTGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE       PEEMSVAS-LDLTGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK       SEELAAATALDLG----APSPAAAEEAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE      PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPP-----------------LPFQ
FOSB_HUMAN      PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPP-----------------LPFQ
                .   . :   ** .     :..  *:.*   *   . *                  **:
```

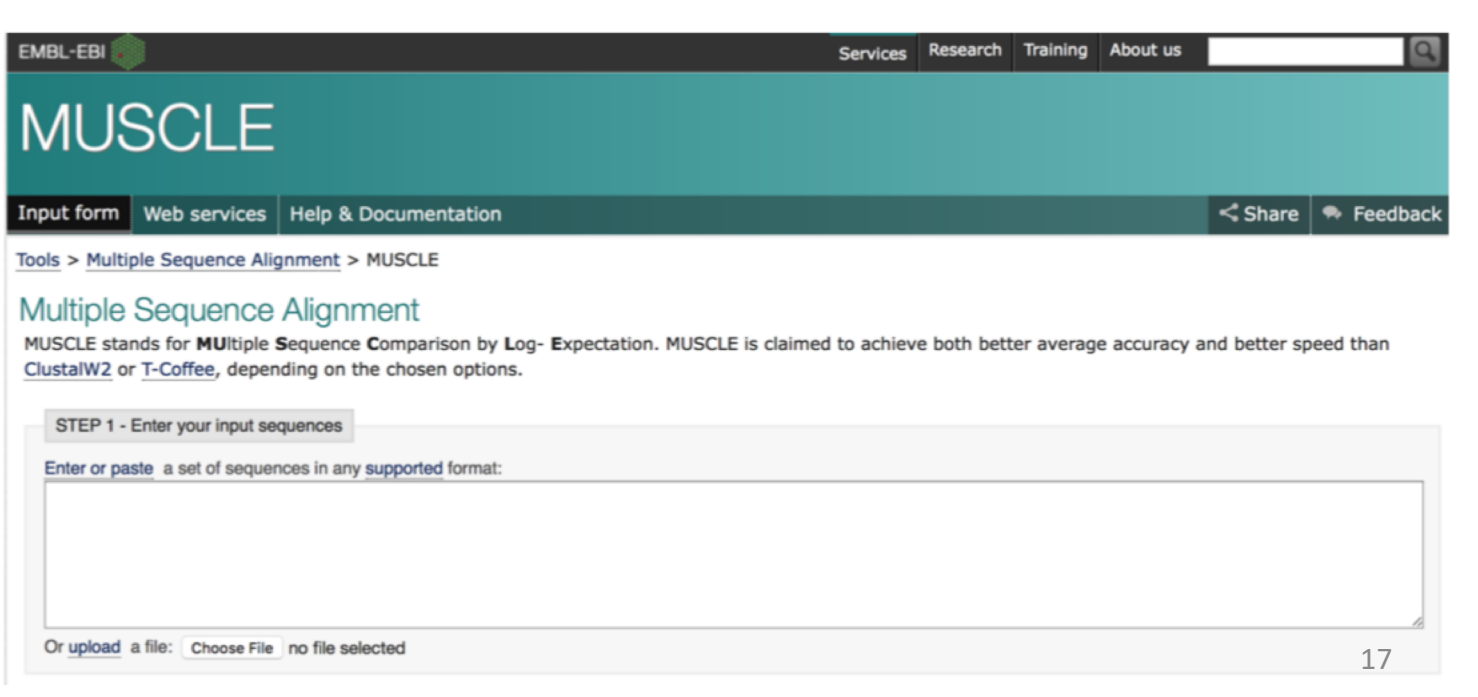
Dots and stars show how well-conserved a column is.

MUSCLE (Edgar, 2004)

Multiple Sequence Comparison by Log-Expectation

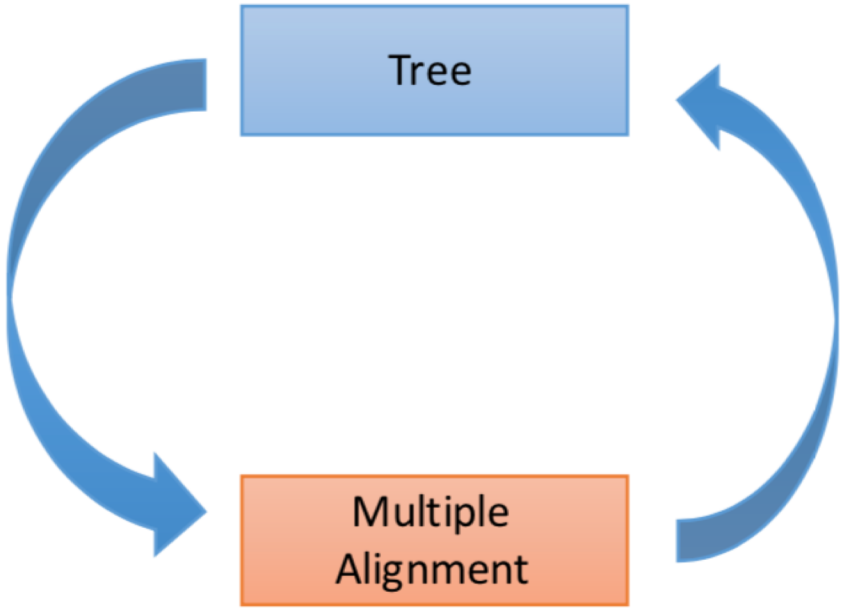
Three phases:

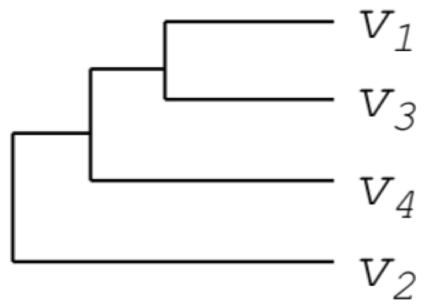
1. Draft progressive alignment: fast heuristic
2. Improved progressive: use tree derived in phase 1
3. Refinement of MSA
   - Remove sequence from MSA and realign to profile of remaining sequences
   - Repeat until convergence

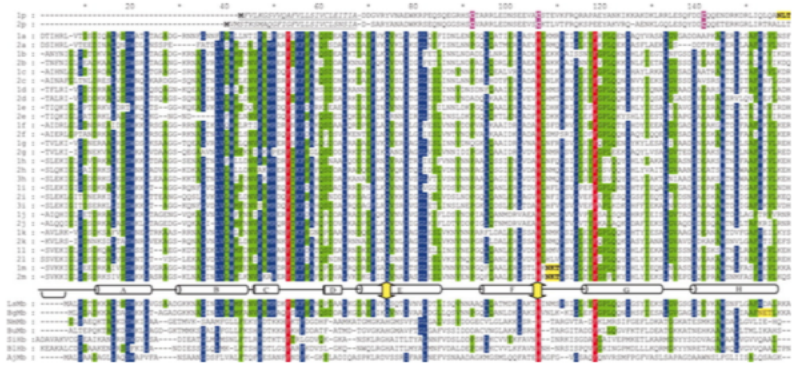


Progressive MSA

Progressive alignment







Circularity!

Ideally, want to derive alignment and tree simultaneously → Hard

Summary

1. Optimal pairwise alignment by dynamic programming in $O(n^2)$ time
2. Optimal multiple alignment with SP-score by dynamic programming in $O(k^2 2^k n^k)$ time
3. Multiple alignment with SP-score is NP-hard (Jiang and Wang, 1994)
4. Carrillo-Lipman enables us to decide whether alignment passes through a vertex (i_1, i_2, i_3) for $k = 3$ sequences (generalizes to $k > 3$)
5. Star alignment gives 2-approximation algorithm
6. Progressive alignment methods are widely used, but come with no theoretical bounds on alignment quality

History

- 1975 Sankoff
  Formulated MSA problem and gave dynamic programming solution
- 1988 Carrillo-Lipman
  Branch and Bound approach for MSA
- 1990 Feng-Doolittle
  Progressive alignment
- 1993 Gusfield
  Star alignment: 2-approximation algorithm
- 1994 Jiang and Wang
  MSA with SP-score is NP-hard
- 1994 Thompson-Higgins-Gibson: ClustalW
  Most popular multiple alignment program
- 2000 Notredam-Higgins-Heringa: T-coffee
  Use library of pairwise alignments
- 2004 Edgar: MUSCLE
  Refinement

Outline

- Scoring matrices
- Tree/star alignment
- Progressive alignment methods

Reading:

- Material based on Chapter 14.6 in book “Algorithms on Strings, Trees and Sequences” by Dan Gusfield
- Chapter 6.7 in Jones and Pevzner
- Lecture notes