Patterns, Profiles, and Multiple Alignments

Outline

- Profiles, Position Specific Scoring Matrices
- Profile Hidden Markov Models
- Alignment of Profiles
- Multiple Alignment Algorithms
  - Problem definition
  - Can we use Dynamic Programming to solve MSA?
  - Progressive Alignment
  - ClustalW
  - Scoring Multiple Alignments
    - Entropy
    - Sum of Pairs (SP) Score
- Sequence Pattern Discovery

Multiple Sequences

- Up to this point we have considered pairwise relationships between sequences
- Protein families contain multiple sequences
  - E.g. Globins
  - Conserved, important regions
- A common task is to find out whether a new sequence can be included in a protein family or not.
  - We can then assign a function and predict 3D structure

Profile Representation of Multiple Sequences

```
- A G G C T A T C A C C T G
T A G - C T A C C A - - - G
C A G - C T A C C A - - - G
C A G - C T A T C A C - G G
C A G - C T A T C G C - G G
```

A		1					1			.8				
C	.6				1			.4	1		.6	.2		
G			1	.2						.2			.4	1
T	.2					1		.6					.2	
-	.2			.8							.4	.8	.4	

Profile Representation of Multiple Sequences

```
-  A  G  G  C  T  A  T  C  A  C  C  T  G
T  A  G  -  C  T  A  C  C  A  -  -  -  G
C  A  G  -  C  T  A  C  C  A  -  -  -  G
C  A  G  -  C  T  A  T  C  A  C  -  G  G
C  A  G  -  C  T  A  T  C  G  C  -  G  G
```

A		1					1			.8				
C	.6				1			.4	1		.6	.2		
G			1	.2						.2			.4	1
T	.2					1		.6					.2	
-	.2			.8							.4	.8	.4	

Earlier, we were aligning a **sequence against a sequence**

Can we align a **sequence against a profile** in order to find out whether this sequence belongs to that protein family?

PSSM

- Position Specific Scoring Matrices
  - The profile representation of a protein family can be a considered as the coefficients of a scoring matrix which give amino acid preferences for each alignment position separately
- How to obtain PSSMs?
  - From a given multiple alignment of sequences
  - Database searches (where a set of database proteins are aligned to the query (i.e. reference) sequence)

How to obtain PSSMs?

- Suppose we have a multiple alignment of N sequences. To obtain the values of the PSSM matrix ($m_{u,b}$), we can use the technique we have used to align a sequence to a profile and use the frequencies of amino acids at each column as coefficients:

$$f_{u,b} = \frac{n_{u,b}}{N} \qquad\qquad m_{u,a} = \sum_{\substack{\text{residue} \\ \text{types } b}} f_{u,b} s_{a,b}$$

A better way to weigh residue preferences

- It is better to give preferred residues extra weight, because residue types rarely found are probably highly disfavored at that column
- Use logarithmic weighting:

$$m_{u,a} = \sum_{\substack{\text{residue} \\ \text{types } b}} \frac{\ln(1 - f'_{u,b})}{\ln(1/(N+1))} s_{a,b} \qquad f'_{u,b} = \frac{n_{u,b}}{N+1}$$

Using log-odds ratios

- We can also use a technique similar to the construction of the common scoring matrices

$$m_{u,a} = \log \frac{q_{u,a}}{p_a}$$

- If sufficient data is available:

$$q_{u,a} = f_{u,a}$$

If sufficient data is not available

- $q_{u,a}$ will cause problems
- Amino acids that do not occur in a column will cause a score of -∞
- A simple solution: assume at least one occurrence of each residue type

$$q_{u,a} = \frac{n_{u,a} + 1}{N + 20}$$

- These additional counts are called **pseudocounts**.

A better formula

- Incorporate background amino acid composition

$$q_{u,a} = \frac{n_{u,a} + \beta p_a}{N + \beta}$$

- β is the total number of pseudocounts in a column. Can be adjusted based on the amount of data available.

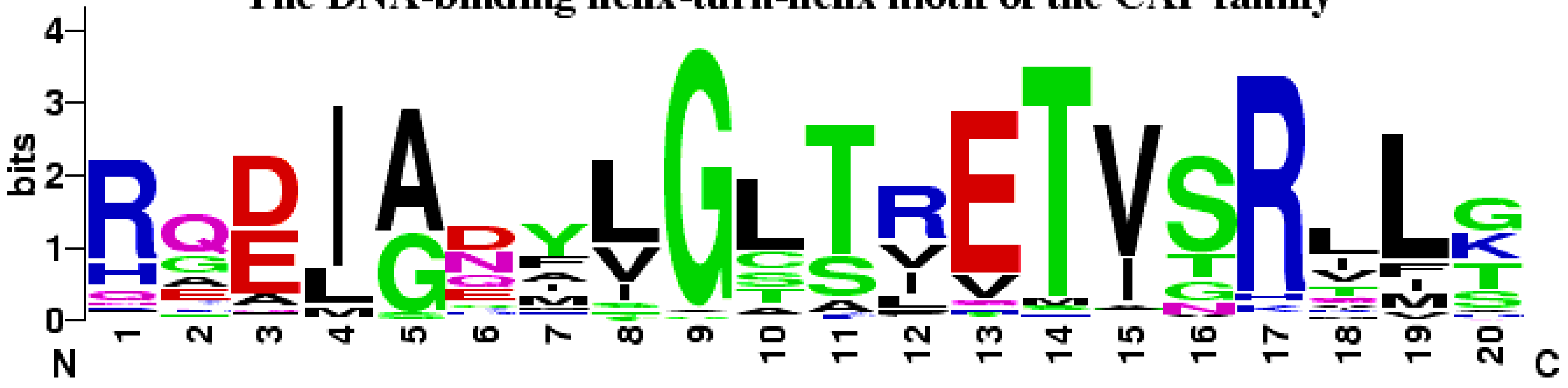
Viewing Profiles/PSSMs as logos

- Residue contribution at each position:

$$f_{u,a} I_u$$

$$I_u = \log_2 20 - H_u \qquad H_u = -\sum_a f_{u,a} \log_2 f_{u,a}$$

The DNA-binding helix-turn-helix motif of the CAP family



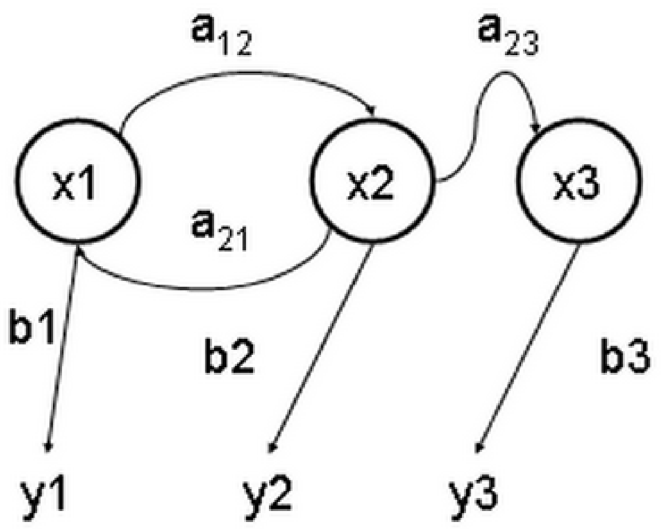
http://weblogo.berkeley.edu

Profile Hidden Markov Models

- Using HMMs to represent a profile and to align a sequence to a profile

Hidden Markov Model

- Hidden Markov models are probabilistic finite state machines. A hidden Markov model λ is determined by the following parameters:



  – A set of finite number of states: S_i, $1 \leq i \leq N$
  – The probability of starting in state S_i, π_i
  – The transition probability from state S_i to S_j, a_{ij}
  – The emission probability density of a symbol ω in state S_i

What is hidden?

- With a hidden Markov model, we usually model a temporal process whose output we can observe but do not know the actual underlying mathematical or physical model.
  - We try to model this process statistically.
- Here the states of the hidden Markov model are hidden to us. We assume that there is some underlying model (or some logic) that produces a set of output signals.

Problems associated with HMMs

- There are three typical questions one can ask regarding HMMs:
  - Given the parameters of the model, compute the probability of a particular output sequence. This problem is solved by the forward-backward procedure.
  - Given the parameters of the model, find the most likely sequence of hidden states that could have generated a given output sequence. This problem is solved by the Viterbi algorithm.
  - Given an output sequence or a set of such sequences, find the most likely set of state transition and output probabilities. In other words, train the parameters of the HMM given a dataset of output sequences. This problem is solved by the Baum-Welch algorithm.

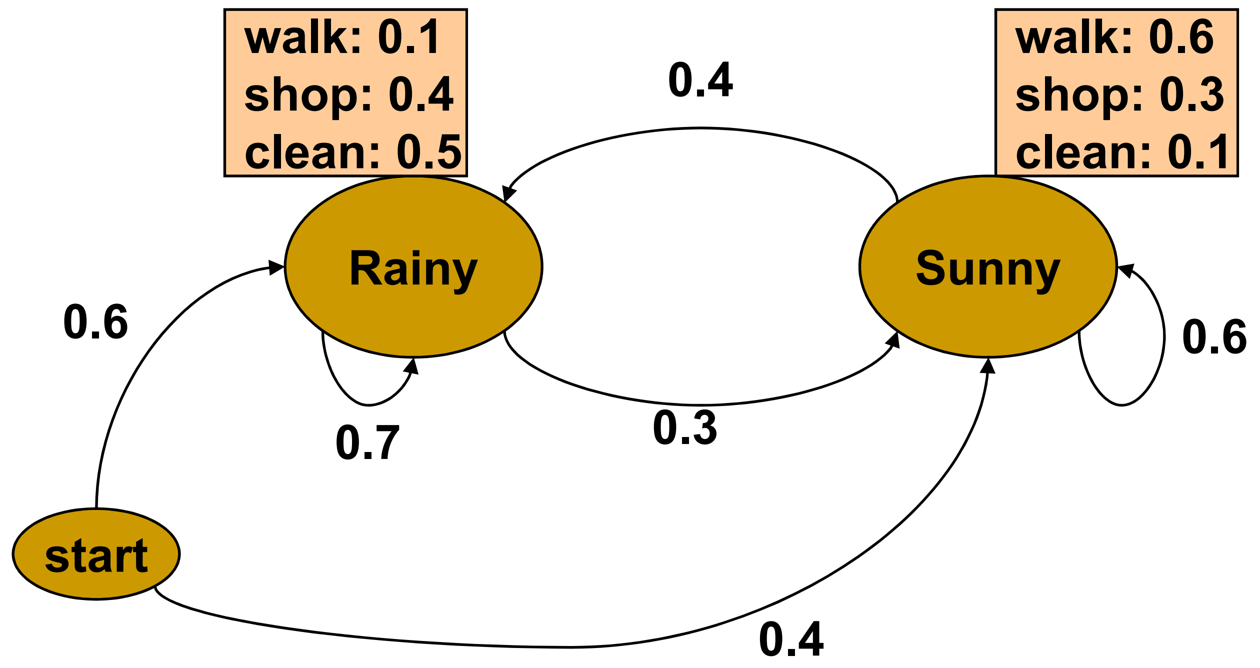
Example (from Wikipedia)

- You have a friend to whom you talk daily on the phone. Your friend is only interested in three activities: walking in the park, shopping, and cleaning his apartment. The choice of what to do is determined exclusively by the weather on a given day. You have no definite information about the weather where your friend lives, but you know general trends. Based on what he tells you he did each day, you try to guess what the weather must have been like.

- There are two states, "Rainy" and "Sunny", but you cannot observe them directly, that is, they are *hidden* from you. On each day, there is a certain chance that your friend will perform one of the following activities, depending on the weather: "walk", "shop", or "clean". Since your friend tells you about his activities, those are the *observations*.

Example (from Wikipedia)

- You know the general weather trends in the area, and what your friend likes to do on the average. In other words, the parameters of the HMM are known.

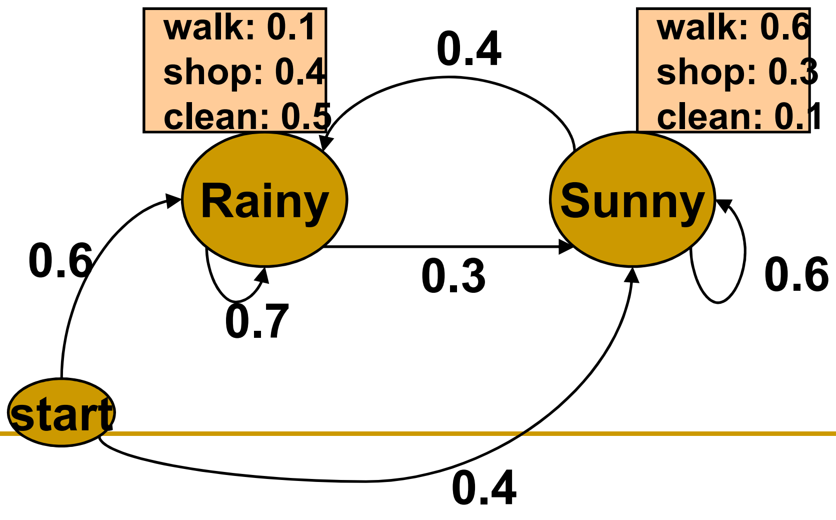


Example (from Wikipedia)

- Now, you talked to your friend for three days, and he told you that on the first day walked, the next day he shopped and the last day he cleaned.

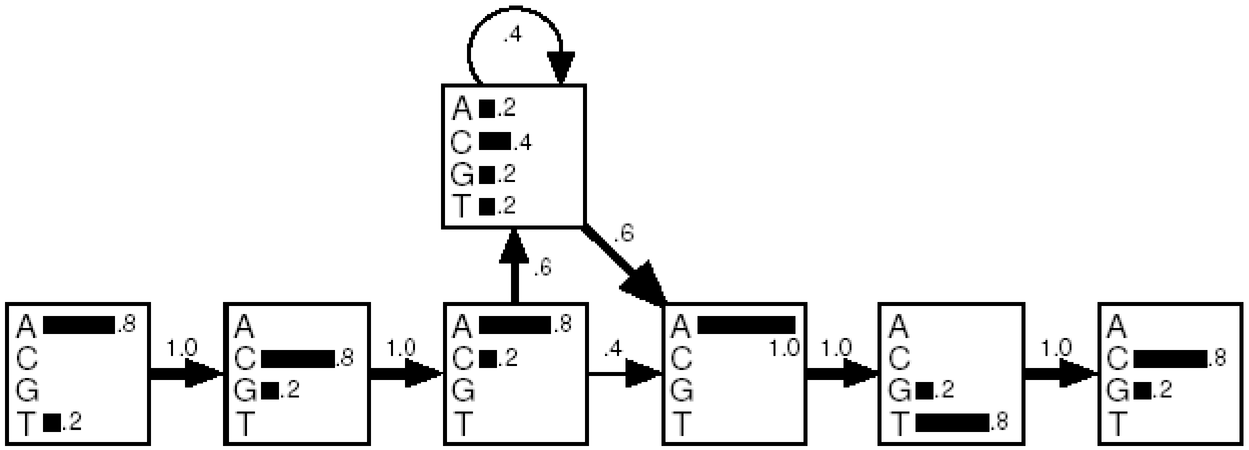
  What is the most likely sequence of days that could have produced this outcome? (Solved by the Viterbi algorithm)

  What is the overall probability of the observations? (Solved by forward-backward algorithm)



HMMs to represent a family of sequences

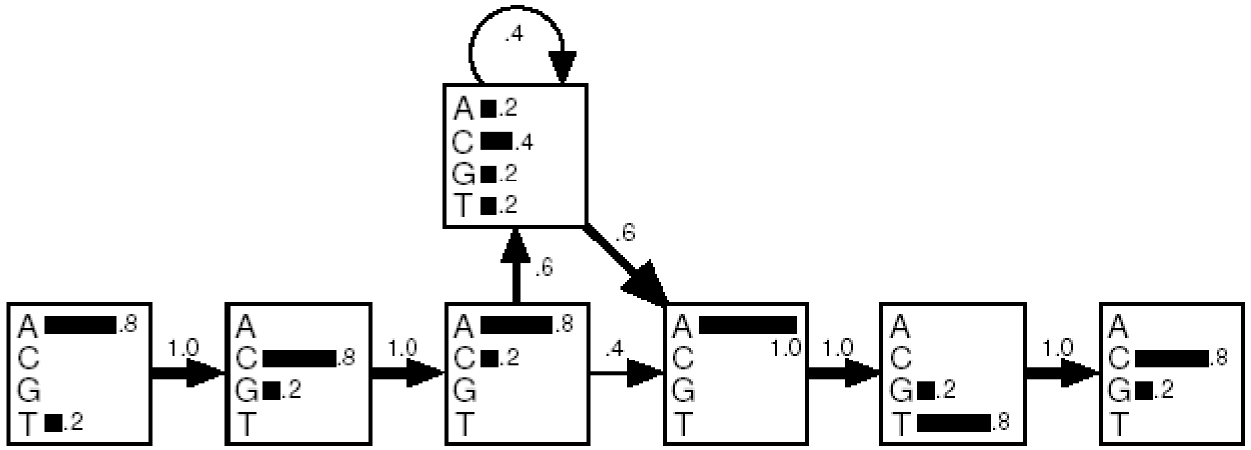
- Given a multiple alignment of sequences, we can use an HMM to model the sequences. Each column of the alignment may be represented by a hidden state that produced that column. Insertions and deletions can be represented by other states.



Profile HMMs from alignments

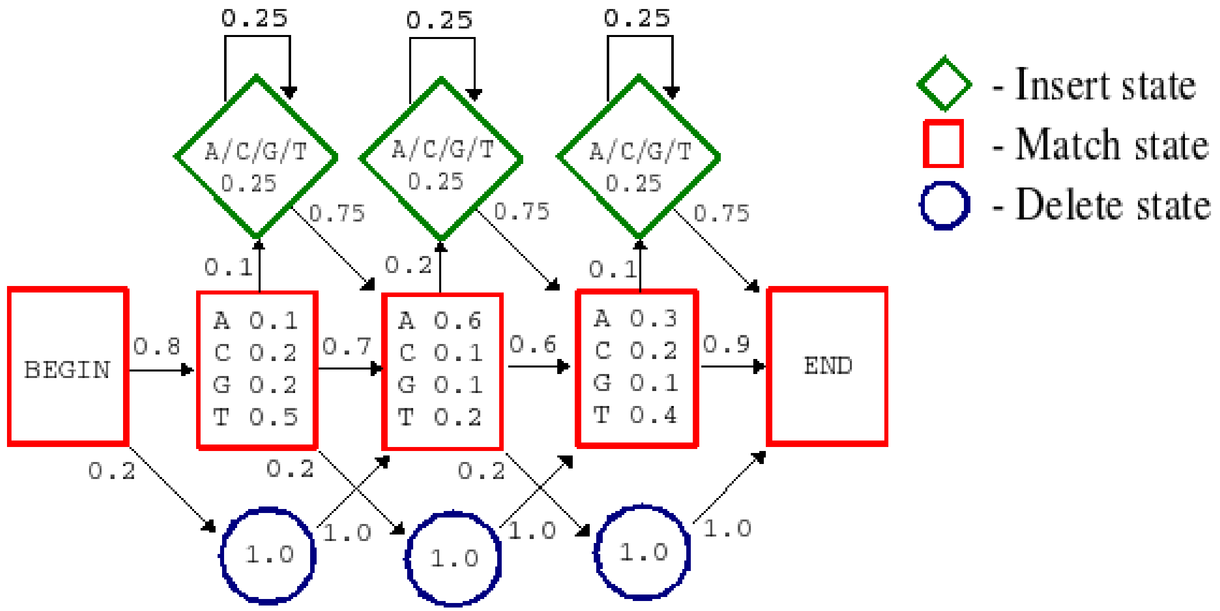
- A given multiple sequence alignment may be used to get the following HMM.

```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```



Profile HMMs

- The structure of the profile HMM is given by:

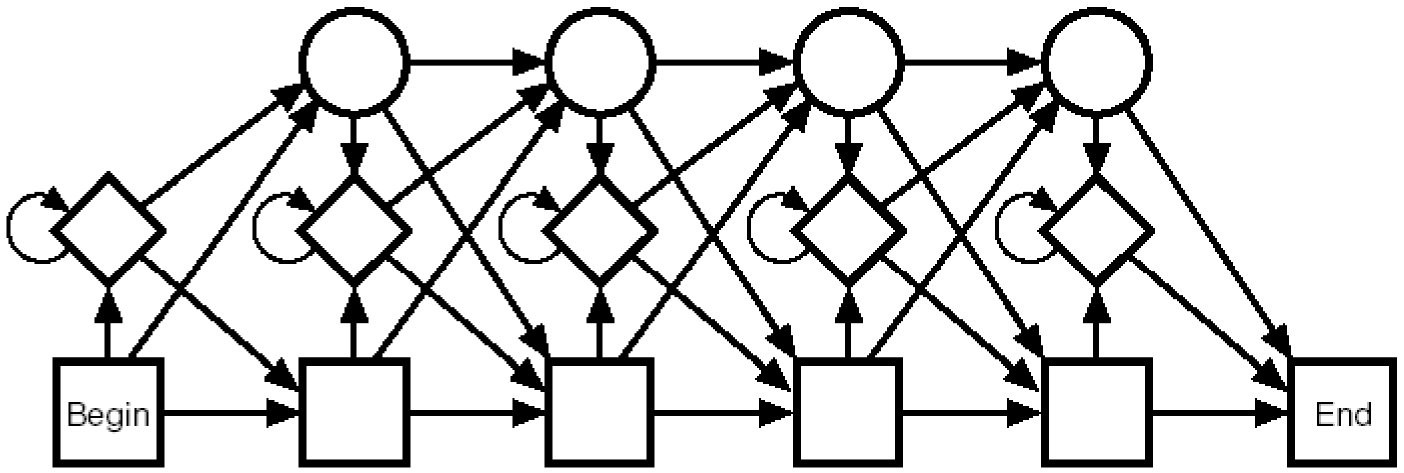


From: http://www.ifm.liu.se/bioinfo/assignments/hmm-simple.png

- There are match, insert, and delete states. Given a multiple sequence alignment we can easily determine the HMM parameters (no Baum-Welch needed in this case)

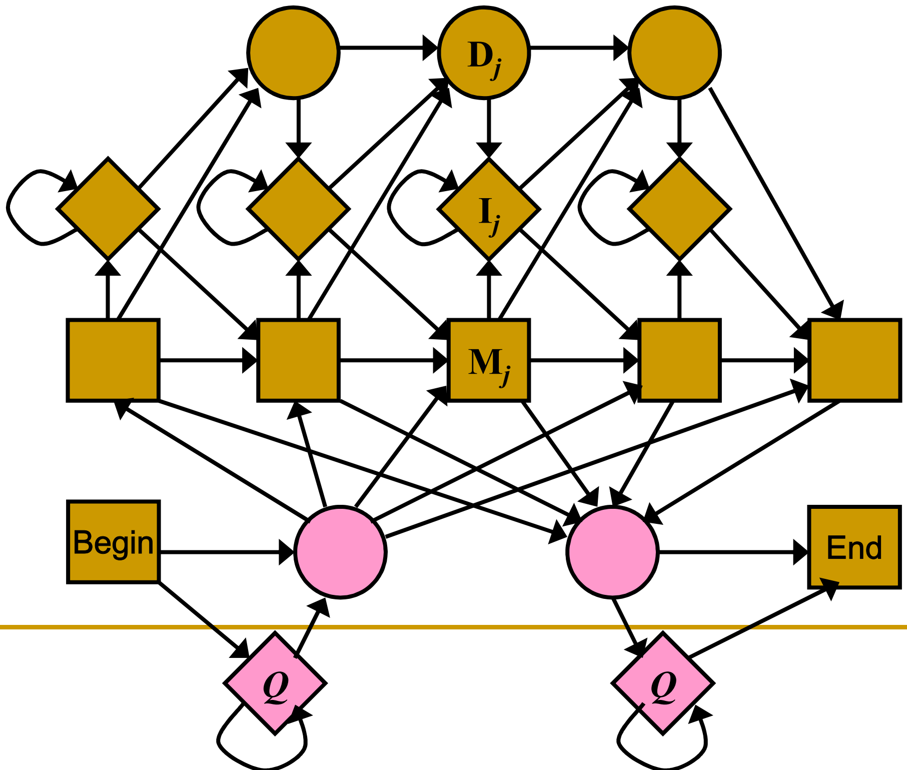
Profile HMMS

- The structure is of profile HMMs is usually fixed following some biological observations:



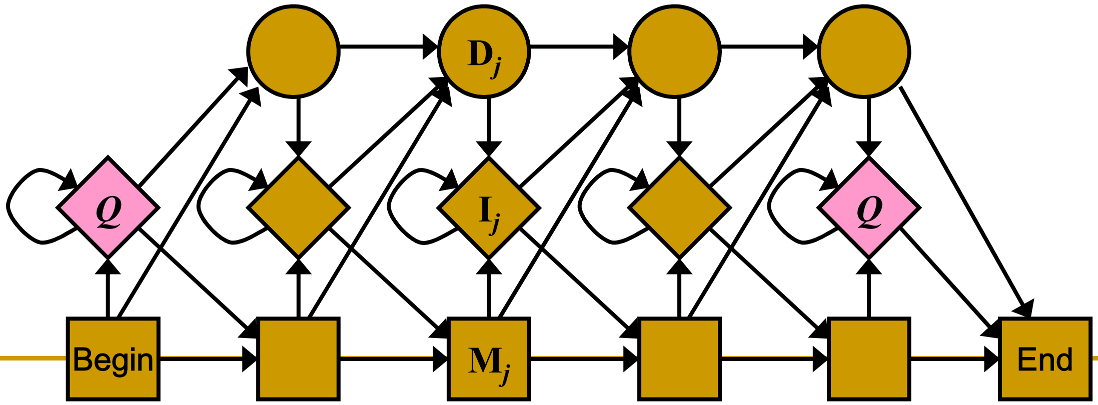
Variants for non-global alignments

- Local alignments (flanking model)
  - Emission prob. in flanking states use background values p_a.
  - Looping prob. close to 1, e.g. $(1-\eta)$ for some small η.



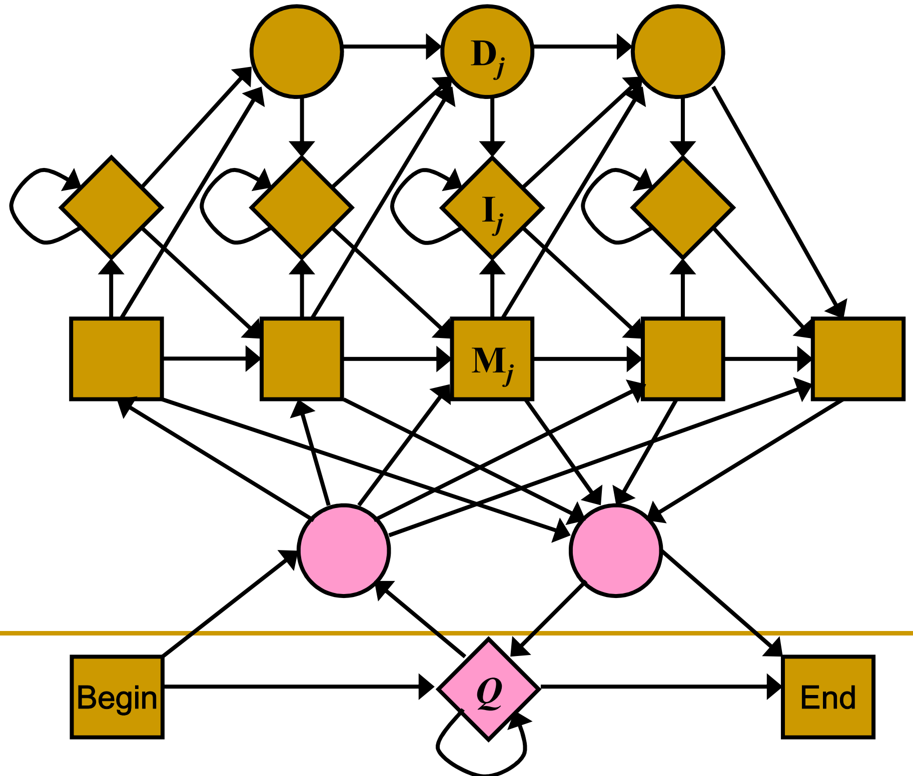
Variants for non-global alignments

- Overlap alignments
  - Only transitions to the first model state are allowed.
  - When expecting to find either present as a whole or absent
  - Transition to first delete state allows missing first residue



Variants for non-global alignments

- Repeat alignments
  - Transition from right flanking state back to random model
  - Can find multiple matching segments in query string



Determining the states of the HMM

- The structure is usually fixed and only the number of “match” states is to be determined

- An alignment column with no gaps can be considered as a “match” state.
- An alignment column with a majority of gaps can be considered an “insert” state.
- Usually a threshold on the gap proportion is determined to find the “match” states

Determining the transition probabilities

- The transition probabilities “from” a state always add up to 1 (except the “end” state which do not have any transitions from it).

$$\sum_{w} t(u,v) = 1$$

Determining the transition probabilities

- Count all the transitions on the given multiple alignment from an alignment position (or state) and use them in the following equation to find transition probabilities from a state

$$t(u,v) = \frac{m_{u,v}}{\sum_{w} m_{u,w}}$$

Determining the emission probabilities

- Emission probabilities in a match or insert state also adds up to 1.

$$e_{M_u}(a) = \frac{n_{M_u,a}}{\sum_b n_{M_u,b}}$$

$$e_{M_u}(a) = \frac{n_{M_u,a} + 1}{\sum_b n_{M_u,b} + 20}$$

Using pseudocounts

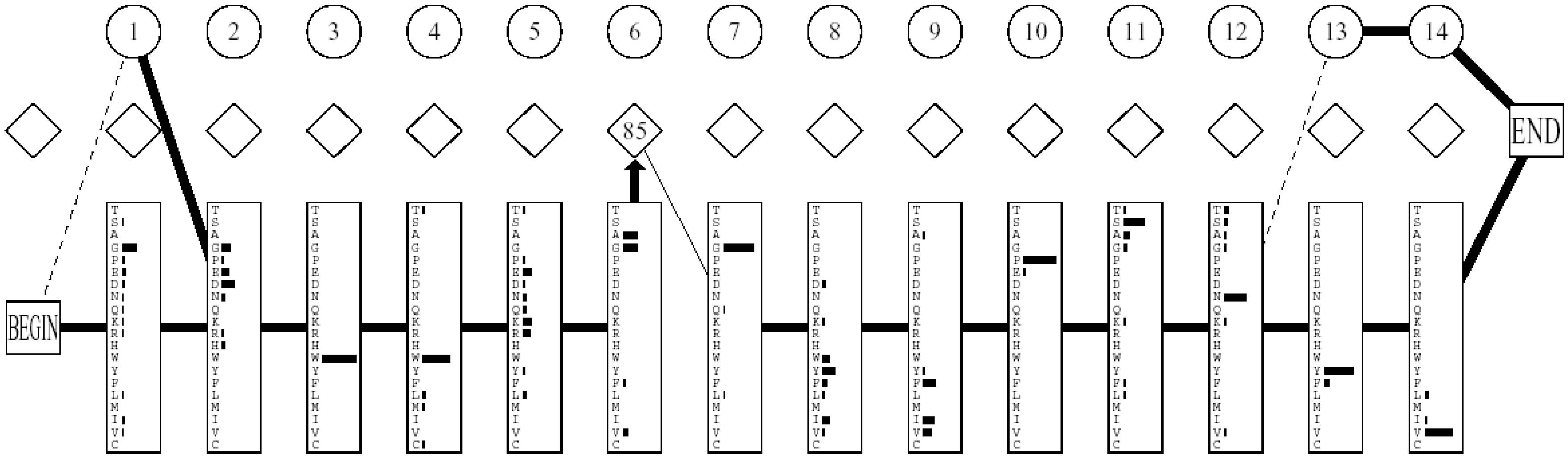
$$e_{I_u}(a) = p_a$$

Pseudocounts in HMMs

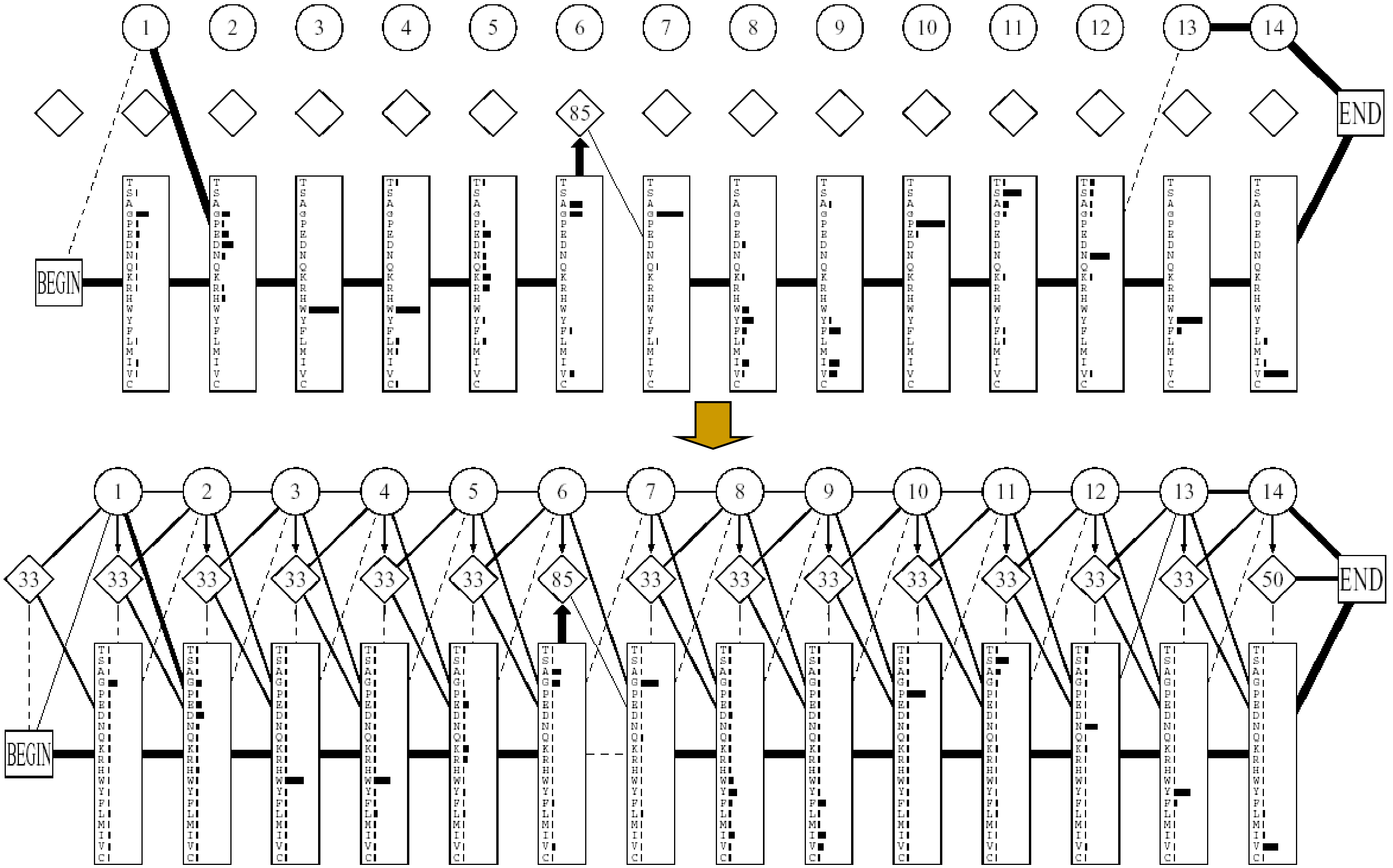
- If we do not observe a certain amino acid at a column of an MSA the emission probability is 0 for that amino acid.
    - This strategy will give 0 probability to such sequences. However, we do not want to do that, (i.e., we do not want to miss sequences that may be biologically important because of only one mismatch)
    - Solution: Add hypothetical occurrences of those amino acids into the columns. E.g, assume a background distribution for each column. The actual appearances of amino acids at a column update the background distribution.

Example
(without pseudocounts)

```
G G W W R G d y . g g k k q L W F P S N Y V
I G W L N G y n e t t g e r G D F P G T Y V
P N W W E G q l . . n n r r G I F P S N Y V
D E W W Q A r r . . d e q i G I V P S K - -
G E W W K A q s . . t g q e G F I P F N F V
G D W W L A r s . . s g q t G Y I P S N Y V
G D W W D A e l . . k g r r G K V P S N Y L
- D W W E A r s l s s g h r G Y V P S N Y V
G D W W Y A r s l i t n s e G Y I P S T Y V
G E W W K A r s l a t r k e G Y I P S N Y V
G D W W L A r s l v t g r e G Y V P S N F V
G E W W K A k s l s s k r e G F I P S N Y V
G E W C E A q t . k n g q . G W V P S N Y I
S D W W R V v n l t t r q e G L I P L N F V
L P W W R A r d . k n g q e G Y I P S N Y I
R D W W E F r s k t v y t p G Y Y E S G Y V
E H W W K V k d . a l g n v G Y I P S N Y V
I H W W R V q d . r n g h e G Y V P S S Y L
K D W W K V e v . . n d r q G F V P A A Y V
V G W M P G l n e r t r q r G D F P G T Y V
P D W W E G e l . . n g q r G V F P A S Y V
E N W W N G e i . . g n r k G I F P A T Y V
E E W L E G e c . . k g k v G I F P K V F V
G G W W K G d y . g t r i q Q Y F P S N Y V
D G W W R G s y . . n g q v G W F P S N Y V
Q G W W R G e i . . y g r v G W F P A N Y V
G R W W K A r r . a n g e t G I I P S N Y V
G G W T Q G e l . k s g q k G W A P T N Y L
G D W W E A r s n . t g e n G Y I P S N Y V
N D W W T G r t . . n g k e G I F P A N Y V
```



Example (with pseudocounts)



Scoring a sequence against a profile HMM

- Given a profile HMM, any given path through the model will emit a sequence with an associated probability
- The path probability is the product of all transition and emission probabilities along the path.

Scoring a sequence against a profile HMM

- Given a query sequence using algorithms similar to DP for sequence alignment we can compute the most probable path that will emit that query sequence (Viterbi algorithm)
- The probability that the given query sequence is emitted by that HMM is given by the sum of all probabilities over all possible paths (forward/backward algorithms).

Viterbi algorithm

- It is a dynamic programming algorithm. It is based on the following assumptions:
  - At any time the process we are modeling is in some state
  - We have finite number of states
  - Each state produces a single output
  - Computing the most likely hidden sequence up to a certain point *t* must depend only on the observed event at point *t*, and the most likely sequence at point $t - 1$.
- These assumptions are satisfied by a first order HMM.

Viterbi DP recurrence relations

$$v_{M_u}(x_i) = e_{M_u}(x_i) \max \begin{cases} v_{M_{u-1}}(x_{i-1}) t(M_{u-1}, M_u) \\ v_{I_{u-1}}(x_{i-1}) t(I_{u-1}, M_u) \\ v_{D_{u-1}}(x_{i-1}) t(D_{u-1}, M_u) \end{cases}$$

$$v_{I_u}(x_i) = p_{x_i} \max \begin{cases} v_{M_u}(x_{i-1}) t(M_u, I_u) \\ v_{I_u}(x_{i-1}) t(I_u, I_u) \end{cases}$$

$$v_{D_u}(x_i) = \max \begin{cases} v_{M_{u-1}}(x_i) t(M_{u-1}, D_u) \\ v_{D_{u-1}}(x_i) t(D_{u-1}, D_u) \end{cases}$$

Viterbi algorithm

- Using these equations we can fill in a partial scores table v of size

  length$_{\text{query}}$ x number of states in HMM
- Initialization:
  - set v_{Start}(“”) = 1, v_u(“”) = 0 for all states u
- However, multiplying many probabilities will lead to very small numbers for long sequences. Therefore use log of probabilities instead and convert products to summation

Viterbi DP recurrence relations

$$\log v_{M_u}(x_i) = \log e_{M_u}(x_i) + \max \begin{cases} \log v_{M_{u-1}}(x_{i-1}) + \log t(M_{u-1}, M_u) \\ \log v_{I_{u-1}}(x_{i-1}) + \log t(I_{u-1}, M_u) \\ \log v_{D_{u-1}}(x_{i-1}) + \log t(D_{u-1}, M_u) \end{cases}$$

$$\log v_{I_u}(x_i) = \log p_{x_i} + \max \begin{cases} \log v_{M_u}(x_{i-1}) + \log t(M_u, I_u) \\ \log v_{I_u}(x_{i-1}) + \log t(I_u, I_u) \end{cases}$$

$$\log v_{D_u}(x_i) = \max \begin{cases} \log v_{M_{u-1}}(x_i) + \log t(M_{u-1}, D_u) \\ \log v_{D_{u-1}}(x_i) + \log t(D_{u-1}, D_u) \end{cases}$$

Forward algorithm

- Forward and backward algorithms are used to compute the probability of the sequence being emitted from an HMM by summing up the probabilities over all possible paths.
- Modification on Viterbi DP equations:
  - Instead of using max, sum all options

Forward algorithm

$$
\begin{aligned}
f_{M_u}(x_i) = e_{M_u}(x_i)[&f_{M_{u-1}}(x_{i-1})t(M_{u-1},M_u) + \\
&+ f_{I_{u-1}}(x_{i-1})t(I_{u-1},M_u) + \\
&+ f_{D_{u-1}}(x_{i-1})t(D_{u-1},M_u)]
\end{aligned}
$$

$$f_{I_u}(x_i) = p_{x_i}[f_{M_u}(x_{i-1})t(M_u,I_u) + f_{I_u}(x_{i-1})t(I_u,I_u)]$$

$$f_{D_u}(x_i) = [f_{M_{u-1}}(x_i)t(M_{u-1},D_u) + f_{D_{u-1}}(x_i)t(D_{u-1},D_u)]$$

Searching a database

- Given the hidden Markov model for a protein family. We can evaluate all the sequences in a database in terms of “how likely” they could have been produced by this HMM model. This likelihood score can be used to find new protein sequences as candidate members for that protein family.
- We can use the Viterbi algorithm or the forward/backward algorithms to compute the probability.

Finding the HMM for a protein family

- Given a set of sequences, can we find the parameters of an HMM without performing a multiple sequence alignment first?
  - Yes. We can use the Baum-Welch algorithm
- However, we have to decide first
  - How many match states are there?

Baum-Welch Algorithm

- The Baum-Welch algorithm is an expectation-maximization (EM) algorithm. It can compute maximum likelihood estimates and posterior mode estimates for the parameters (transition and emission probabilities) of an HMM, when given only emissions as training data.

Available profile HMM tools

- SAM
- HMMER2
- and many others

Multiple alignment algorithms

- One of the most essential tools in molecular biology
  - Finding highly conserved subregions or embedded patterns of a set of biological sequences
    - Conserved regions usually are key functional regions, prime targets for drug developments
  - Estimation of evolutionary distance between sequences
  - Prediction of protein secondary/tertiary structure
- Practically useful methods only since 1987 (D. Sankoff)
  - Before 1987 they were constructed by hand
  - Dynamic programming is expensive

Multiple Sequence Alignment (MSA)

- What is multiple sequence alignment?
- Given *k* sequences:

```
VTISCTGSSSNIGAGNHVKWYQQLPG
VTISCTGTSSNIGSITVNWYQQLPG
LRLSCSSSGFIFSSYAMYWVRQAPG
LSLTCTVSGTSFDDYYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG
ATLVCLISDFYPGAVTVAWKADS
AALGCLVKDYFPEPVTVSWNSG
VSLTCLVKGFYPSDIAVEWESNG
```

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSSSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSSSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

Conserved residues

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSSSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

Conserved regions

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSSSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

Patterns? Positions 1 and 3 are hydrophobic residues

Multiple Sequence Alignment (MSA)

- An MSA of these sequences:

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSSSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

Conserved residues, regions, patterns

MSA Warnings

- MSA algorithms work under the assumption that they are aligning related sequences
- They will align ANYTHING they are given, even if unrelated
- If it just “looks wrong” it probably is

Generalizing the Notion of Pairwise Alignment

- Alignment of 2 sequences is represented as a 2-row matrix
- In a similar way, we represent alignment of 3 sequences as a 3-row matrix

A	T	_	G	C	G	_
A	_	C	G	T	_	A
A	T	C	A	C	_	A

- Score: more conserved columns, better alignment

Alignments = Paths in *k* dimensional grids

- Align 3 sequences: ATGC, AATC,ATGC

	A	--	T	G	C

	A	A	T	--	C

	--	A	T	G	C

Alignment Paths

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

x coordinate

	A	A	T	--	C

	--	A	T	G	C

Alignment Paths

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

x coordinate

y coordinate

	--	A	T	G	C

-

Alignment Paths

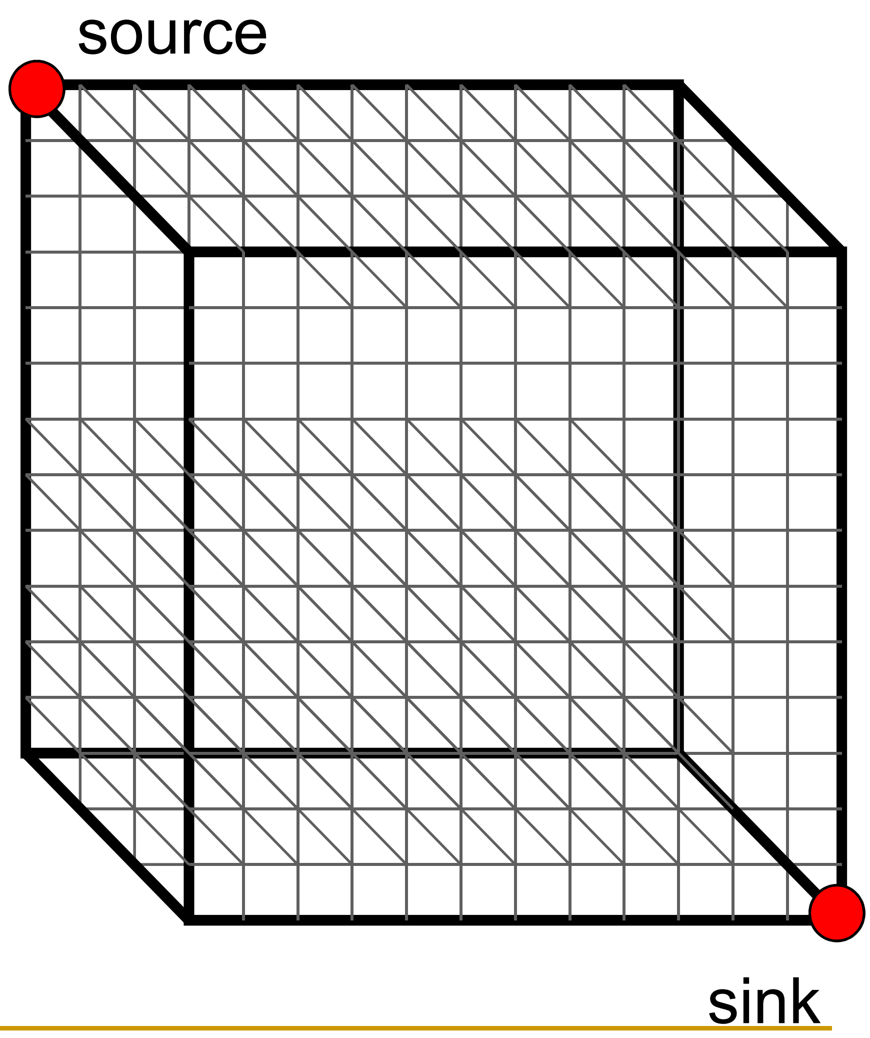
0	1	1	2	3	4	*x* coordinate
	A	--	T	G	C	
0	1	2	3	3	4	*y* coordinate
	A	A	T	--	C	
0	0	1	2	3	4	*z* coordinate
	--	A	T	G	C	

- Resulting path in *(x,y,z)* space:

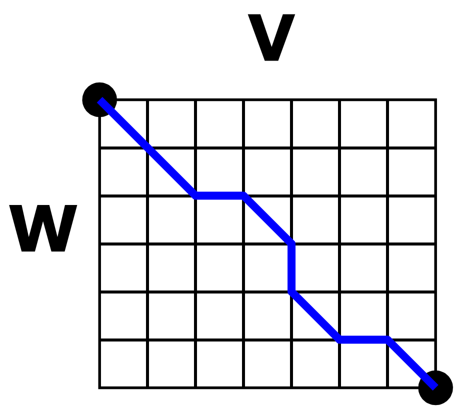
$(0,0,0)\rightarrow(1,1,0)\rightarrow(1,2,1)\rightarrow(2,3,2)\rightarrow(3,3,3)\rightarrow(4,4,4)$

Aligning Three Sequences

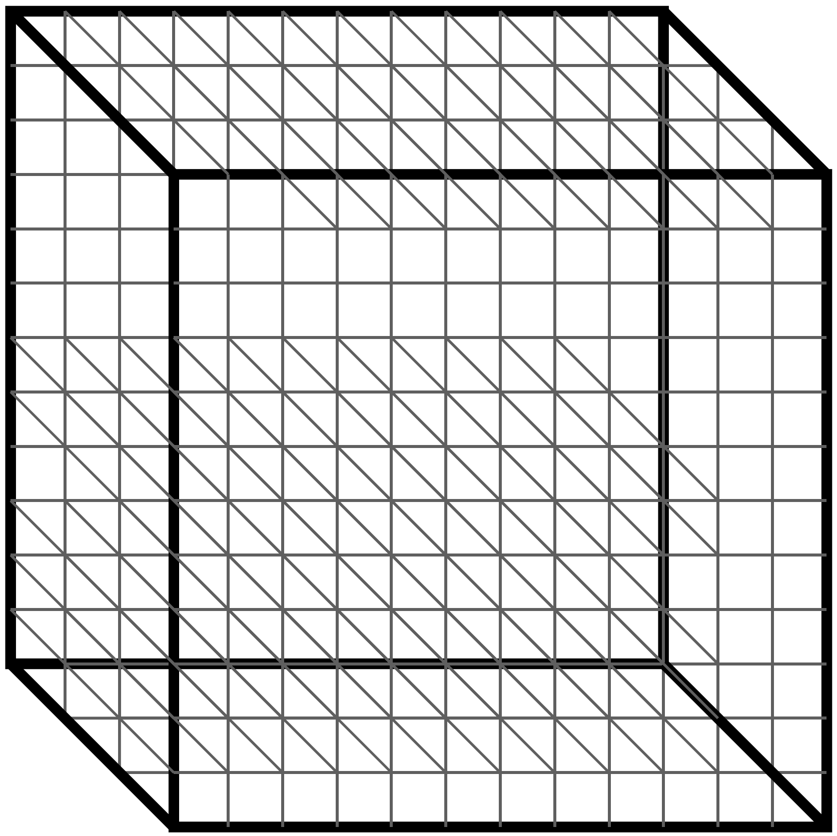
- Same strategy as aligning two sequences
- Use a 3-D matrix, with each axis representing a sequence to align
- For global alignments, go from source to sink



2-D vs 3-D Alignment Grid

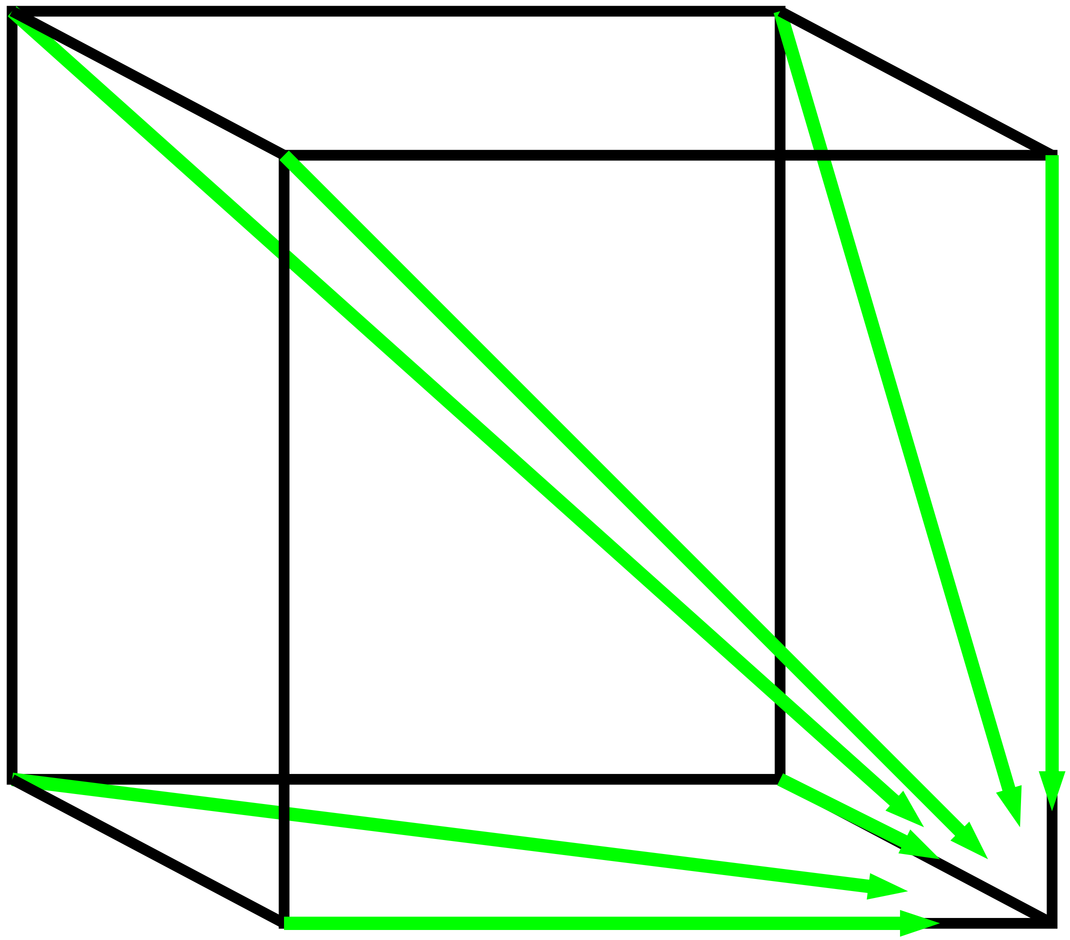


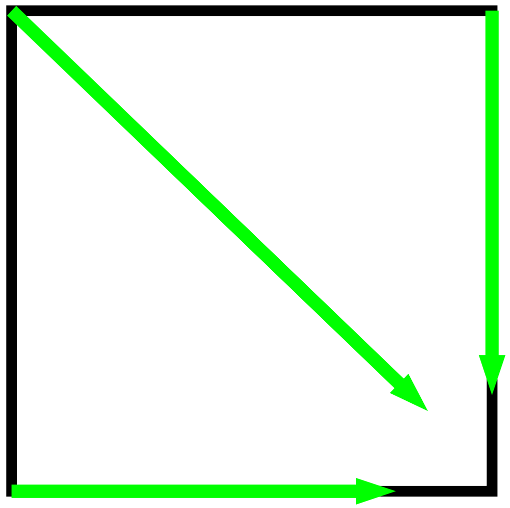
2-D alignment matrix



3-D alignment matrix

2-D cell versus 3-D Alignment Cell

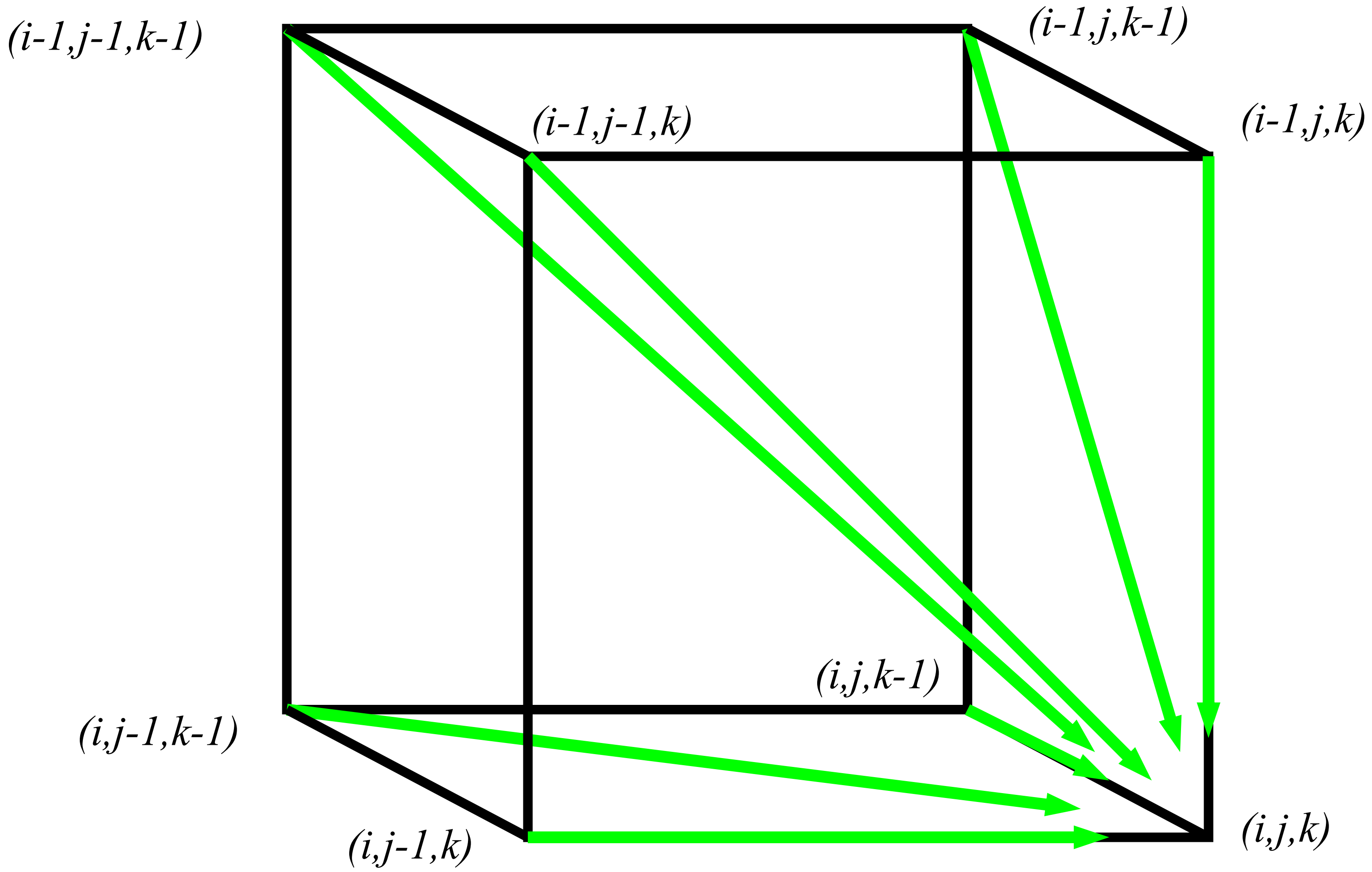




In **2-D**, 3 edges in each unit square

In **3-D**, 7 edges in each unit cube

Architecture of 3-D Alignment Cell



Multiple Alignment: Dynamic Programming

- $$s_{i,j,k} = \max \left\{ \begin{array}{ll}
\left. s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \right\} & \begin{array}{l}\text{cube diagonal:}\\ \text{no indels}\end{array} \\
\left. \begin{array}{l} s_{i-1,j-1,k} + \delta(v_i, w_j, \_) \\ s_{i-1,j,k-1} + \delta(v_i, \_, u_k) \\ s_{i,j-1,k-1} + \delta(\_, w_j, u_k) \end{array} \right\} & \begin{array}{l}\text{face diagonal:}\\ \text{one indel}\end{array} \\
\left. \begin{array}{l} s_{i-1,j,k} + \delta(v_i, \_, \_) \\ s_{i,j-1,k} + \delta(\_, w_j, \_) \\ s_{i,j,k-1} + \delta(\_, \_, u_k) \end{array} \right\} & \begin{array}{l}\text{edge diagonal:}\\ \text{two indels}\end{array}
\end{array} \right.$$

- $\delta(x, y, z)$ is an entry in the 3-D scoring matrix

Multiple Alignment: Running Time

- For 3 sequences of length $\boldsymbol{n}$, the run time is $7\boldsymbol{n}^3$; $O(\boldsymbol{n}^3)$

- For $\boldsymbol{k}$ sequences, build a $\boldsymbol{k}$-dimensional matrix, with run time $(2^k-1)(\boldsymbol{n}^k)$; $O(2^k\boldsymbol{n}^k)$

- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to $\boldsymbol{k}$ sequences but it is impractical due to exponential running time

Multiple Alignment Induces Pairwise Alignments

Every multiple alignment induces pairwise alignments

```
x:  AC-GCGG-C
y:  AC-GC-GAG
z:  GCCGC-GAG
```

Induces:

```
x: ACGCGG-C;   x: AC-GCGG-C;   y: AC-GCGAG
y: ACGC-GAC;   z: GCCGC-GAG;   z: GCCGCGAG
```

Reverse Problem: Constructing Multiple Alignment from Pairwise Alignments

Given 3 arbitrary pairwise alignments:

```
x: ACGCTGG-C;   x: AC-GCTGG-C;   y: AC-GC-GAG
y: ACGC--GAC;   z: GCCGCA-GAG;   z: GCCGCAGAG
```

can we construct a multiple alignment that induces them?

Reverse Problem: Constructing Multiple Alignment from Pairwise Alignments

Given 3 arbitrary pairwise alignments:

```
x: ACGCTGG-C;   x: AC-GCTGG-C;   y: AC-GC-GAG
y: ACGC--GAC;   z: GCCGCA-GAG;   z: GCCGCAGAG
```

can we construct a multiple alignment that induces them?

```
NOT ALWAYS
```

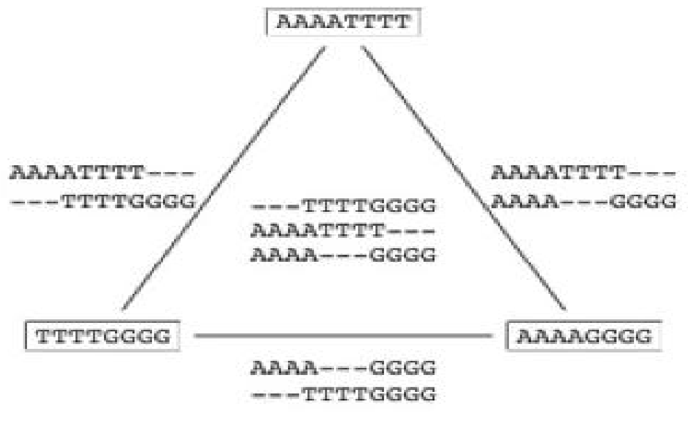
Pairwise alignments may be inconsistent

Inferring Multiple Alignment from Pairwise Alignments

- From an optimal multiple alignment, we can infer pairwise alignments between all pairs of sequences, but they are not necessarily optimal
- It is difficult to infer a “good” multiple alignment from optimal pairwise alignments between all sequences

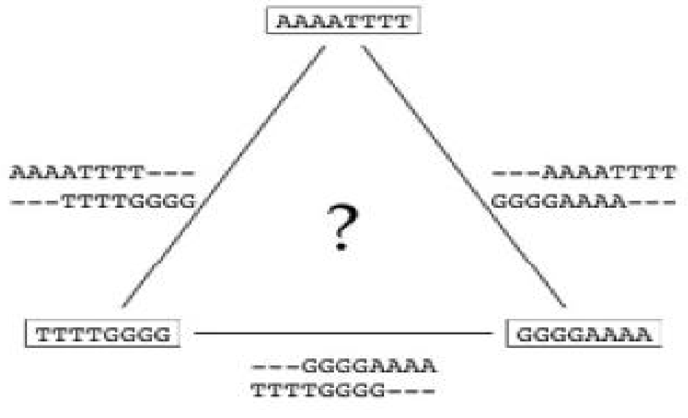
Combining Optimal Pairwise Alignments into Multiple Alignment

Can combine pairwise alignments into multiple alignment



(a) Compatible pairwise alignments

Can ***not*** combine pairwise alignments into multiple alignment



(b) Incompatible pairwise alignments

Consensus String of a Multiple Alignment

```
          -  A  G  G  C  T  A  T  C  A  C  C  T  G
          T  A  G  -  C  T  A  C  C  A  -  -  -  G
          C  A  G  -  C  T  A  C  C  A  -  -  -  G
          C  A  G  -  C  T  A  T  C  A  C  -  G  G
          C  A  G  -  C  T  A  T  C  G  C  -  G  G

          C  A  G  -  C  T  A  T  C  A  C  -  G  G
Consensus
String:   C  A  G  C  T  A  T  C  A  C  G  G
```

- The *consensus string* S_M derived from multiple alignment *M* is the concatenation of the consensus characters for each column of *M.*
  - The *consensus character* for column *i* is the character that minimizes the summed distance to it from all the characters in column *i.* (i.e., if match and mismatch scores are equal for all symbols, the majority symbol is the consensus character)

Profile Representation of Multiple Alignment

-	A	G	G	C	T	A	T	C	A	C	C	T	G
T	A	G	-	C	T	A	C	C	A	-	-	-	G
C	A	G	-	C	T	A	C	C	A	-	-	-	G
C	A	G	-	C	T	A	T	C	A	C	-	G	G
C	A	G	-	C	T	A	T	C	G	C	-	G	G

A		1					1			.8				
C	.6				1			.4	1		.6	.2		
G			1	.2						.2			.4	1
T	.2					1		.6					.2	
-	.2			.8							.4	.8	.4	

Profile Representation of Multiple Alignment

```
-  A  G  G  C  T  A  T  C  A  C  C  T  G
T  A  G  -  C  T  A  C  C  A  -  -  -  G
C  A  G  -  C  T  A  C  C  A  -  -  -  G
C  A  G  -  C  T  A  T  C  A  C  -  G  G
C  A  G  -  C  T  A  T  C  G  C  -  G  G
```

A		1					1			.8				
C	.6				1			.4	1		.6	.2		
G			1	.2						.2			.4	1
T	.2					1		.6					.2	
-	.2			.8							.4	.8	.4	

Earlier, we were aligning a **sequence against a sequence**

Can we align a **sequence against a profile?**

Can we align a **profile against a profile?**

Aligning alignments

- Given two alignments, can we align them?

```
x GGGCACTGCAT
y GGTTACGTC--     Alignment 1
z GGGAACTGCAG

w GGACGTACC--     Alignment 2
v GGACCT-----
```

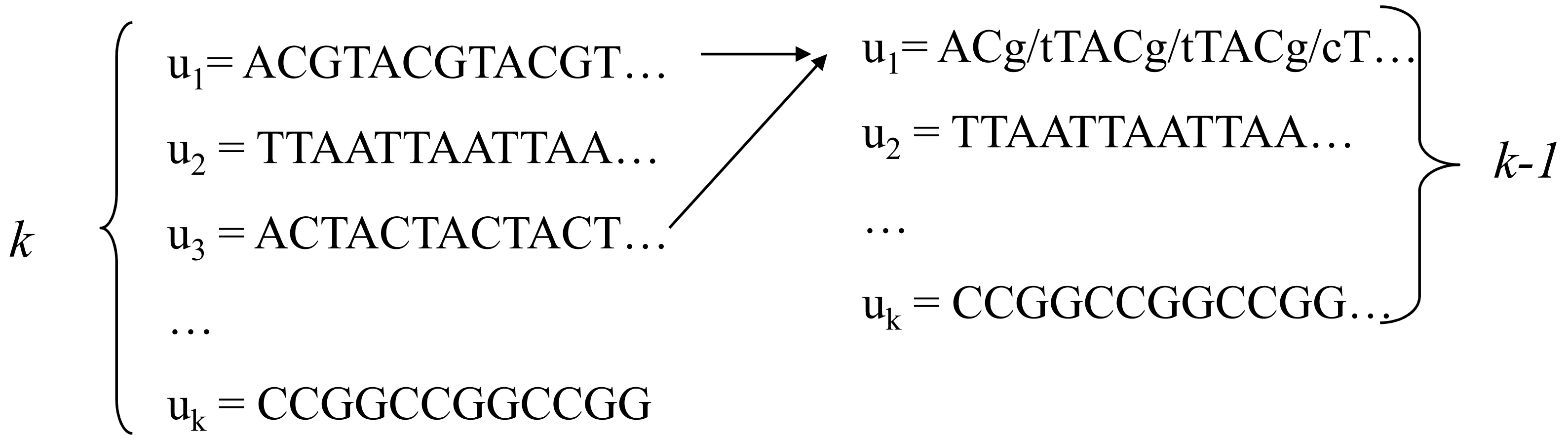
Aligning alignments

- Given two alignments, can we align them?
- Hint: use alignment of corresponding profiles

```
x GGGCACTGCAT
y GGTTACGTC--     Combined Alignment
z GGGAACTGCAG
w GGACGTACC--
v GGACCT-----
```

Multiple Alignment: Greedy Approach

- Choose most similar pair of strings and combine into a profile , thereby reducing alignment of *k* sequences to an alignment of of *k-1* sequences/profiles. **Repeat**
- This is a heuristic greedy method



Greedy Approach: Example

- Consider these 4 sequences

$s1$	GATTCA
$s2$	GTCTGA
$s3$	GATATT
$s4$	GTCAGC

Greedy Approach: Example (cont'd)

- There are $\binom{4}{2}$ = 6 possible alignments

```
s2  GTCTGA                      s1  GATTCA--
s4  GTCAGC (score = 2)          s4  G–T-CAGC(score = 0)

s1  GAT-TCA                     s2  G-TCTGA
s2  G-TCTGA (score = 1)         s3  GATAT-T (score = -1)

s1  GAT-TCA                     s3  GAT-ATT
s3  GATAT-T (score  = 1)        s4  G-TCAGC (score = -1)
```

Greedy Approach: Example (cont'd)

s_2 *and* s_4 are closest; combine:

$$\left.\begin{array}{ll} s2 & \text{GTCTGA} \\ s4 & \text{GTCAGC} \end{array}\right] \underset{\text{(profile)}}{s_{2,4}} \quad \text{GTCt/aGa/cA}$$

new set of 3 sequences:

s_1	GATTCA
s_3	GATATT
$s_{2,4}$	GTCt/aGa/c

Progressive Alignment

- *Progressive alignment* is a variation of greedy algorithm with a somewhat more intelligent strategy for choosing the order of alignments.
- Progressive alignment works well for close sequences, but deteriorates for distant sequences
  - Gaps in consensus string are permanent
  - Use profiles to compare sequences

Star alignment

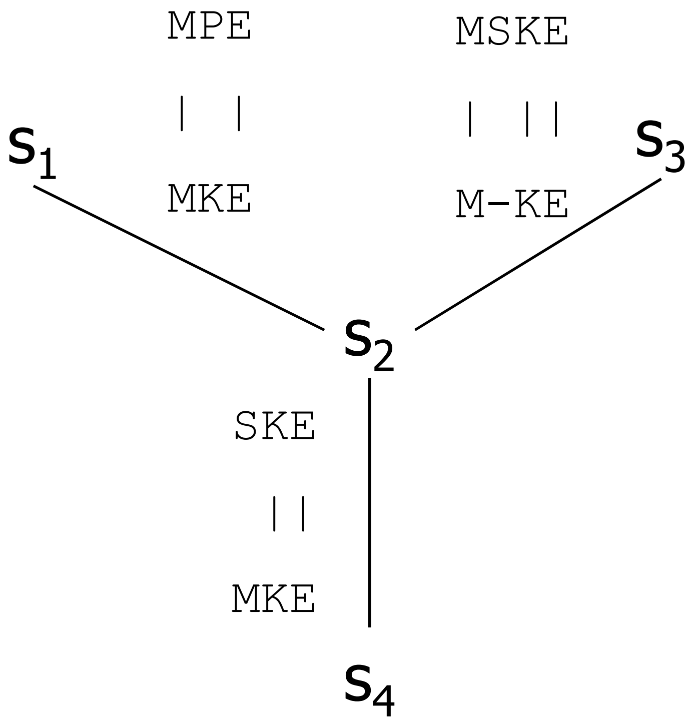
- Heuristic method for multiple sequence alignments
- Select a sequence *c* as the center of the star
- For each sequence x_1, …, x_k such that index $i \neq c$, perform a Needleman-Wunsch global alignment
- Aggregate alignments with the principle “once a gap, always a gap.”

Choosing a center

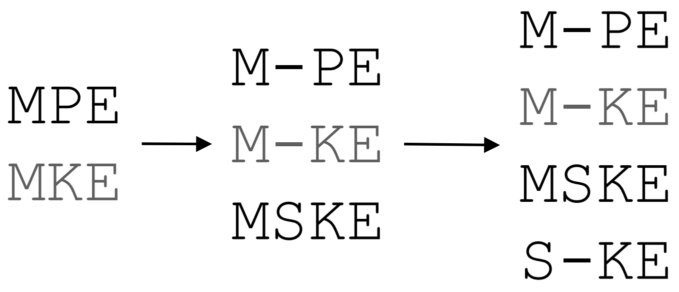
- Try them all and pick the one which is most similar to all of the sequences
- Let $S(x_i,x_j)$ be the optimal score between sequences x_i and x_j.
- Calculate all $O(k^2)$ alignments, and choose as x_c the sequence x_i that maximizes the following

$$\sum_{j \neq i} S(x_i,x_j)$$

Star alignment example



S_1: MPE
S_2: MKE
S_3: MSKE
S_4: SKE



```
MPE   →  M-PE   →  M-PE
MKE      M-KE      M-KE
         MSKE      MSKE
                   S-KE
```

Analysis

- Assuming all sequences have length n
- $O(k^2n^2)$ to calculate center
- Step i of iterative pairwise alignment takes $O((i \cdot n) \cdot n)$ time
  - two strings of length n and $i \cdot n$
- $O(k^2n^2)$ overall cost

ClustalW

- Most popular multiple alignment tool today
- ‘W’ stands for ‘weighted’ (different parts of alignment are weighted differently).
- Three-step process

  1.) Construct pairwise alignments

  2.) Build Guide Tree (by Neighbor Joining method)

  3.) Progressive Alignment guided by the tree

  - The sequences are aligned progressively according to the branching order in the guide tree

Step 1: Pairwise Alignment

- Aligns each sequence again each other giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & - & & & \\ v_2 & .17 & - & & \\ v_3 & .87 & .28 & - & \\ v_4 & .59 & .33 & .62 & - \end{array}$$

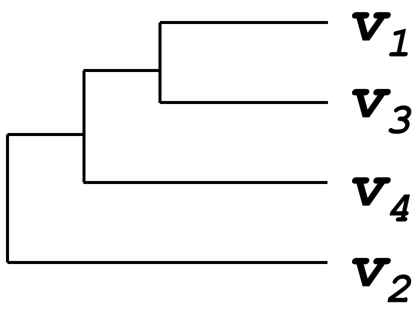
(.17 means 17 % identical)

Step 2: Guide Tree

- Create Guide Tree using the similarity matrix
  - ClustalW uses the neighbor-joining method
  - Guide tree roughly reflects evolutionary relations

Step 2: Guide Tree (cont'd)

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & - & & & \\ v_2 & .17 & - & & \\ v_3 & .87 & .28 & - & \\ v_4 & .59 & .33 & .62 & - \end{array}$$



Calculate:

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

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

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

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-----STSAKEDGFGWLLPPPPPPP-----------------LPFQ
FOSB_HUMAN      PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----------------LPFQ
                .   . :   ** .     :..  *:.*   *   . *                   **:
```

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

ClustalW: another example

S_1	`ALSK`
S_2	`TNSD`
S_3	`NASK`
S_4	`NTSD`

ClustalW example

S_1	ALSK
S_2	TNSD
S_3	NASK
S_4	NTSD

All pairwise alignments

	S_1	S_2	S_3	S_4
S_1	0	9	4	7
S_2		0	8	3
S_3			0	7
S_4				0

Distance Matrix

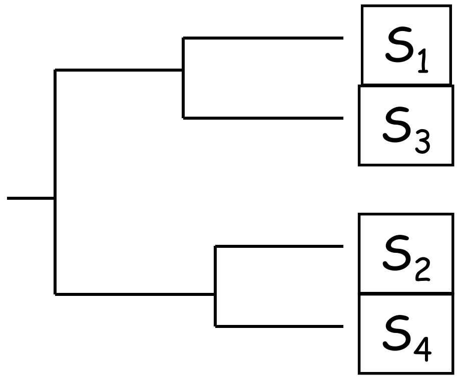
ClustalW example

S_1	ALSK
S_2	TNSD
S_3	NASK
S_4	NTSD

All pairwise alignments

	S_1	S_2	S_3	S_4
S_1	0	9	4	7
S_2		0	8	3
S_3			0	7
S_4				0

Neighbor Joining



Rooted Tree

Distance Matrix

ClustalW example

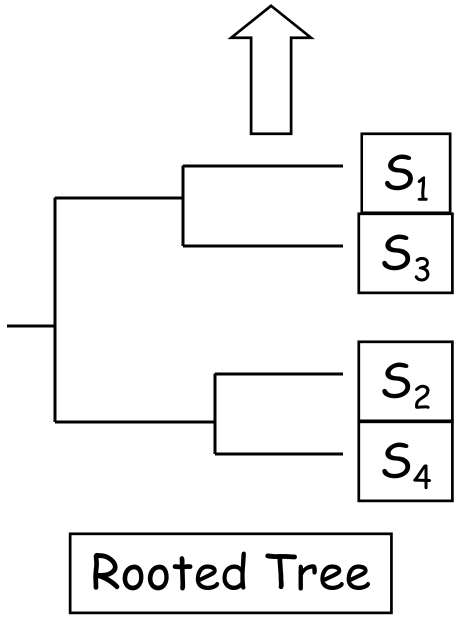
S_1	ALSK
S_2	TNSD
S_3	NASK
S_4	NTSD

All pairwise alignments

	S_1	S_2	S_3	S_4
S_1	0	9	4	7
S_2		0	8	3
S_3			0	7
S_4				0

Distance Matrix

Neighbor Joining



Multiple Alignment Steps

1. Align S_1 with S_3
2. Align S_2 with S_4
3. Align (S_1, S_3) with (S_2, S_4)

ClustalW example

S_1	ALSK
S_2	TNSD
S_3	NASK
S_4	NTSD

All pairwise alignments

```
-ALSK
-TNSD
NA-SK
NT-SD
```

Multiple Alignment

```
-ALSK
NA-SK

-TNSD
NT-SD
```

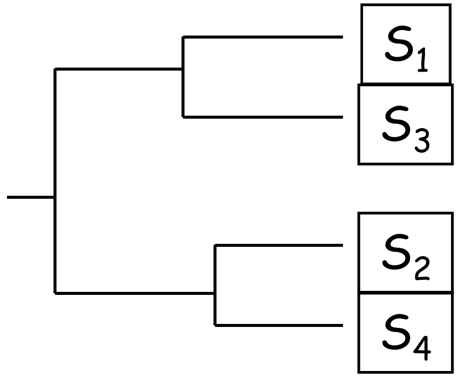
Multiple Alignment Steps

1. Align S_1 with S_3
2. Align S_2 with S_4
3. Align (S_1, S_3) with (S_2, S_4)

	S_1	S_2	S_3	S_4
S_1	0	9	4	7
S_2		0	8	3
S_3			0	7
S_4				0

Distance Matrix

Neighbor Joining



Rooted Tree

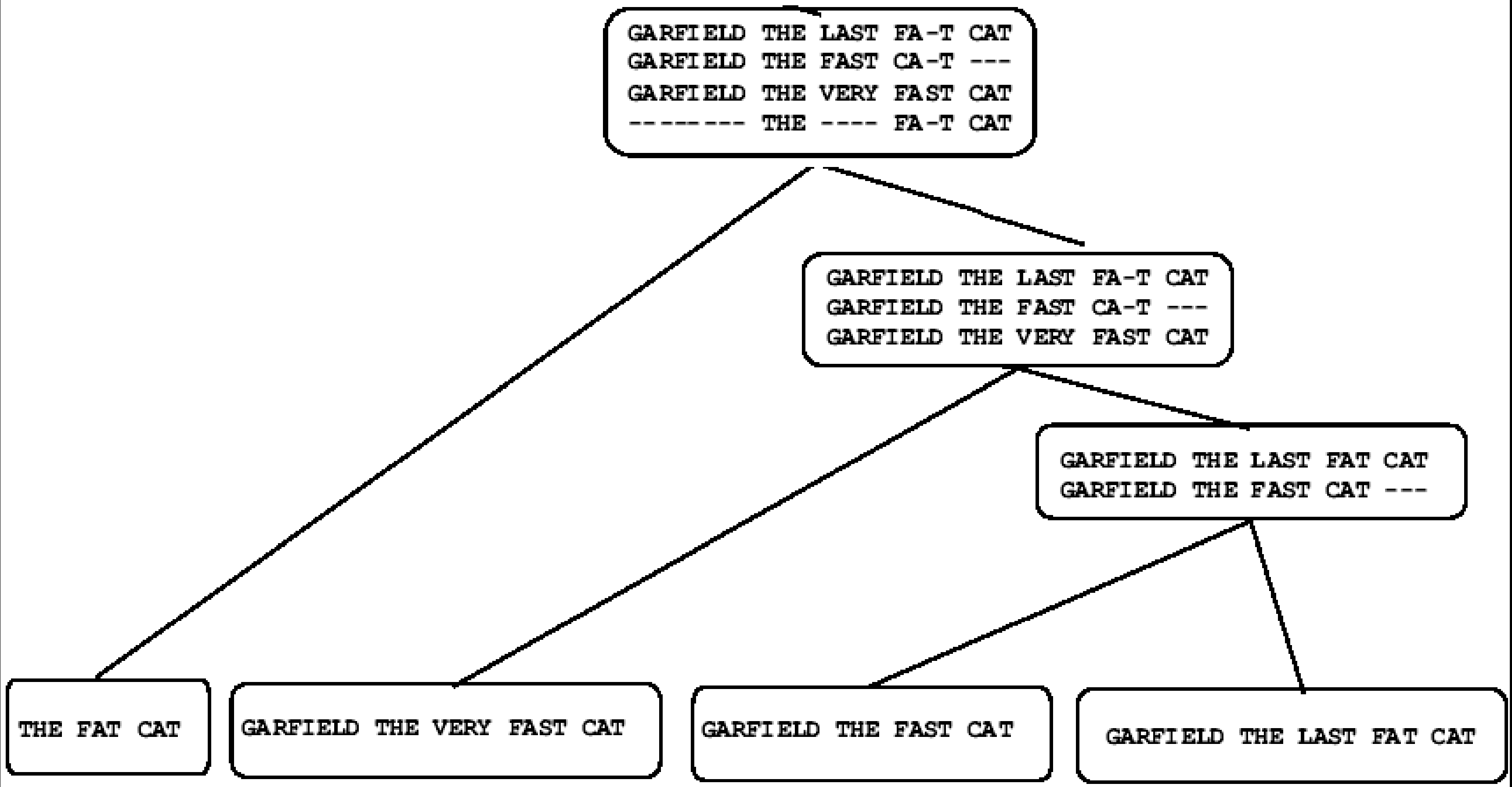
Other progressive approaches

- PILEUP
  - Similar to CLUSTALW
  - Uses UPGMA to produce tree

Problems with progressive alignments

- Depend on pairwise alignments
- If sequences are very distantly related, much higher likelihood of errors
- Care must be made in choosing scoring matrices and penalties

Figure 1. Limits of the progressive strategy.


GARFIELD THE LAST FA-T CAT
GARFIELD THE FAST CA-T ---
GARFIELD THE VERY FAST CAT
-------- THE ---- FA-T CAT
GARFIELD THE LAST FA-T CAT
GARFIELD THE FAST CA-T ---
GARFIELD THE VERY FAST CAT
GARFIELD THE LAST FAT CAT
GARFIELD THE FAST CAT ---
THE FAT CAT
GARFIELD THE VERY FAST CAT
GARFIELD THE FAST CAT
GARFIELD THE LAST FAT CAT

This example shows how a progressive alignment strategy can be misled. In the initial alignment of sequences 1 and 2, ClustalW has a choice between aligning CAT with CAT and making an internal gap or making a mismatch between C and F and having a terminal gap. Since terminal gaps are much cheaper than internals, the ClustalW scoring schemes prefers the former. In the next stage, when the extra sequence is added, it turns out that properly aligning the two CATs in the previous stage would have led to a better scoring sums-of-pairs multiple alignment.

Iterative refinement in progressive alignment

Another problem of progressive alignment:

- Initial alignments are "frozen" even when new evidence comes

Example:

```
x:    GAAGTT
y:    GAC-TT  >   Frozen!

z:    GAACTG
w:    GTACTG  >   Now clear that correct y = GA-CTT
```

Evaluating multiple alignments

- Balibase benchmark (Thompson, 1999)
- De-facto standard for assessing the quality of a multiple alignment tool
- Manually refined multiple sequence alignments
- Quality measured by how good it matches the core blocks
- Another benchmark: SABmark benchmark
  - Based on protein structural families

Scoring multiple alignments

- Ideally, a scoring scheme should
  - Penalize variations in conserved positions higher
  - Relate sequences by a phylogenetic tree
    - Tree alignment
- Usually assume
  - Independence of columns
  - Quality computation
    - Entropy-based scoring
      - Compute the Shannon entropy of each column
    - Sum-of-pairs (SP) score

Multiple Alignments: Scoring

- Number of matches (multiple longest common subsequence score)
- Entropy score
- Sum of pairs (SP-Score)

Multiple LCS Score

- A column is a “match” if all the letters in the column are the same

```
AAA
AAA
AAT
ATC
```

- Only good for very similar sequences

Entropy

- Define frequencies for the occurrence of each letter in each column of multiple alignment
  - $p_A = 1, p_T=p_G=p_C=0$ (1st column)
  - $p_A = 0.75, p_T = 0.25, p_G=p_C=0$ (2nd column)
  - $p_A = 0.50, p_T = 0.25, p_C=0.25\ p_G=0$ (3rd column)
- Compute entropy of each column

$$-\sum_{X=A,T,G,C} p_X \log p_X$$

AAA
AAA
AAT
ATC

Entropy: Example

$$entropy\begin{pmatrix} A \\ A \\ A \\ A \end{pmatrix} = 0$$

Best case

Worst case

$$entropy\begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = -\sum \frac{1}{4}\log\frac{1}{4} = -4(\frac{1}{4} * -2) = 2$$

Multiple Alignment: Entropy Score

Entropy for a multiple alignment is the sum of entropies of its columns:

$$\sum_{\text{over all columns}} -\sum_{X=A,T,G,C} p_X \log p_X$$

Entropy of an Alignment: Example

column entropy:

$$-(p_A \log p_A + p_C \log p_C + p_G \log p_G + p_T \log p_T)$$

A	A	A
A	C	C
A	C	G
A	C	T

- Column 1 = $-[1*\log(1) + 0*\log 0 + 0*\log 0 + 0*\log 0]$
  $= 0$
- Column 2 = $-[(^1/_4)*\log(^1/_4) + (^3/_4)*\log(^3/_4) + 0*\log 0 + 0*\log 0]$
  $= -[\ (^1/_4)*(-2) + (^3/_4)*(-.415)\ ] = +0.811$
- Column 3 = $-[(^1/_4)*\log(^1/_4)+(^1/_4)*\log(^1/_4)+(^1/_4)*\log(^1/_4) +(^1/_4)*\log(^1/_4)]$
  $= 4* -[(^1/_4)*(-2)] = +2.0$
- Alignment Entropy = $0 + 0.811 + 2.0 = +2.811$

Multiple Alignment Induces Pairwise Alignments

Every multiple alignment induces pairwise alignments

```
x:  AC-GCGG-C
y:  AC-GC-GAG
z:  GCCGC-GAG
```

Induces:

```
x: ACGCGG-C;   x: AC-GCGG-C;   y: AC-GCGAG
y: ACGC-GAC;   z: GCCGC-GAG;   z: GCCGCGAG
```

Sum of Pairs (SP) Scoring

- SP scoring is the standard method for scoring multiple sequence alignments.
- Columns are scored by a ‘sum of pairs’ function using a substitution matrix (PAM or BLOSUM)
- Assumes statistical independence for the columns, does not use a phylogenetic tree.

Sum of Pairs Score(SP-Score)

- Consider pairwise alignment of sequences

  $$a_i \text{ and } a_j$$

  imposed by a multiple alignment of k sequences
- Denote the score of this suboptimal (not necessarily optimal) pairwise alignment as

  $$s^*(a_i, a_j)$$

- Sum up the pairwise scores for a multiple alignment:

  $$s(a_1,\ldots,a_k) = \Sigma_{i,j}\, s^*(a_i, a_j)$$

Computing SP-Score

Aligning 4 sequences: 6 pairwise alignments

Given a_1, a_2, a_3, a_4:

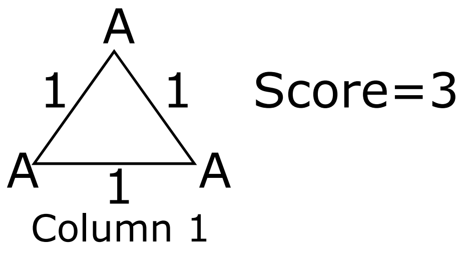
$$
\begin{aligned}
s(a_1 \ldots a_4) = \Sigma s^*(a_i, a_j) = {} & s^*(a_1, a_2) + s^*(a_1, a_3) \\
& + s^*(a_1, a_4) + s^*(a_2, a_3) \\
& + s^*(a_2, a_4) + s^*(a_3, a_4)
\end{aligned}
$$

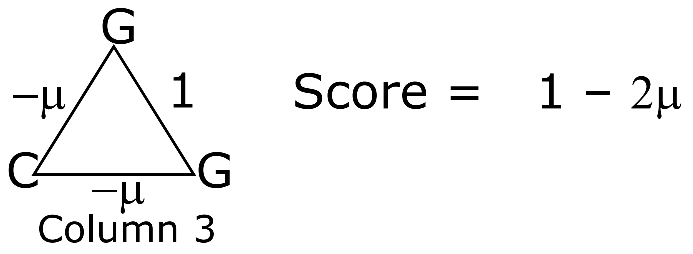
SP-Score: Example

a_1 ATG–C–AAT
. A–G–CATAT
a_k ATCCCATTT

$$S(a_1 \ldots a_k) = \sum_{i,j} S^*(a_i, a_j) \longleftarrow \binom{n}{2} \text{ Pairs of Sequences}$$

May also calculate the scores column by column:





Example

- Compute Sum of Pairs Score of the following multiple alignment with match = 3, mismatch = -1, S(X,-) = -1, S(-,-) = 0

```
X:  G  T  A  C  G
Y:  T  G  C  C  G
Z:  C  G  G  C  C
W:  C  G  G  A  C
   -2  6 -2  6  2
```

Sum of pairs = -2+6-2+6+2 = 10

Multiple alignment tools

- Clustal W (Thompson, 1994)
    - Most popular
- PRRP (Gotoh, 1993)
- HMMT (Eddy, 1995)
- DIALIGN (Morgenstern, 1998)
- T-Coffee (Notredame, 2000)
- MUSCLE (Edgar, 2004)
- Align-m (Walle, 2004)
- PROBCONS (Do, 2004)

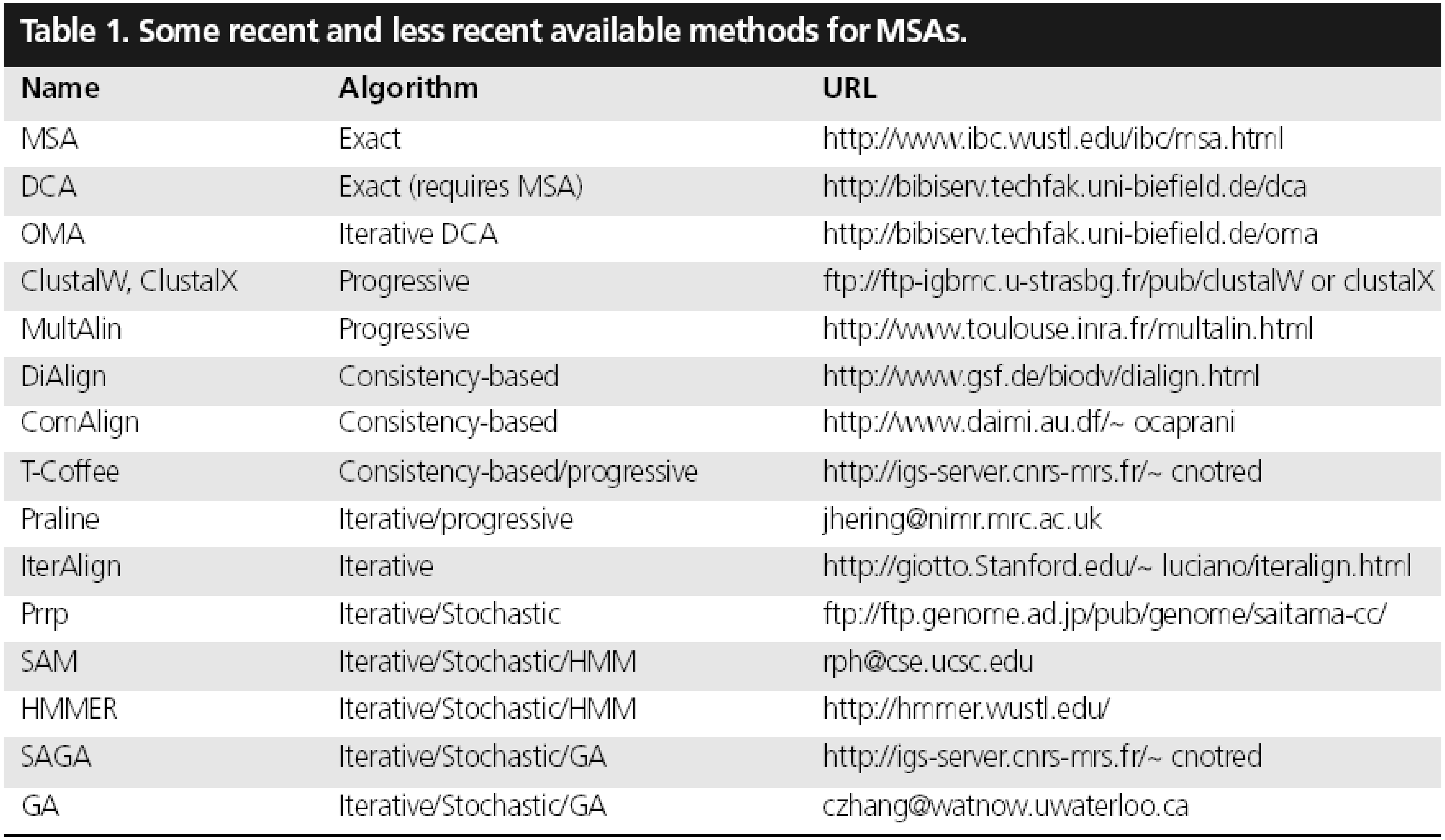


Table 1. Some recent and less recent available methods for MSAs.

Name	Algorithm	URL
MSA	Exact	http://www.ibc.wustl.edu/ibc/msa.html
DCA	Exact (requires MSA)	http://bibiserv.techfak.uni-biefield.de/dca
OMA	Iterative DCA	http://bibiserv.techfak.uni-biefield.de/oma
ClustalW, ClustalX	Progressive	ftp://ftp-igbmc.u-strasbg.fr/pub/clustalW or clustalX
MultAlin	Progressive	http://www.toulouse.inra.fr/multalin.html
DiAlign	Consistency-based	http://www.gsf.de/biodv/dialign.html
ComAlign	Consistency-based	http://www.daimi.au.df/~ ocaprani
T-Coffee	Consistency-based/progressive	http://igs-server.cnrs-mrs.fr/~ cnotred
Praline	Iterative/progressive	jhering@nimr.mrc.ac.uk
IterAlign	Iterative	http://giotto.Stanford.edu/~ luciano/iteralign.html
Prrp	Iterative/Stochastic	ftp://ftp.genome.ad.jp/pub/genome/saitama-cc/
SAM	Iterative/Stochastic/HMM	rph@cse.ucsc.edu
HMMER	Iterative/Stochastic/HMM	http://hmmer.wustl.edu/
SAGA	Iterative/Stochastic/GA	http://igs-server.cnrs-mrs.fr/~ cnotred
GA	Iterative/Stochastic/GA	czhang@watnow.uwaterloo.ca

from: C. Notredame, “Recent progresses in multiple alignment: a survey”, *Pharmacogenomics* (2002) 3(1)

Useful links

http://cnx.org/content/m11036/latest/

http://www.biokemi.uu.se/Utbildning/Exercises/ClustalX/index.shtm

http://bioinformatics.weizmann.ac.il/~pietro/Making_and_using_protein_MA/

http://homepage.usask.ca/~ctl271/857/paper1_overview.shtml

http://journal-ci.csse.monash.edu.au/ci/vol04/mulali/mulali.html

Sequence Pattern Discovery

- High conservation only for short stretches of sequence.
  - Statistical significance may not be high due to short length $\rightarrow$ harder to identify these regions
  - May be important for structure and function and can be used to identify diverged members of protein families.

Sequence Pattern Discovery

- From multiple sequence alignments
- By searching for possible patterns in the set of sequences

Pattern Discovery from MSA

- eMOTIF
  - Uses 20 groups of amino acids to denote amino acids that can be substituted by each other
  - For each column of the alignment determine which single group can cover the whole column
    - If cannot find a single group for all sequences, look for a single group for at least %30 of the sequences.
  - By examining the possible column combinations, identify patterns
    - Compute statistical significance by comparing to a background distribution

Pattern Discovery from MSA

- AACC
  - Amino Acid Class Covering
  - Uses an alternative set of groupings
  - Similar to progressive MSA techniques
    - Use the groupings to represent the intermediate alignments. If they cannot be grouped use the symbol X to denote the column is not conserved.
    - The final alignment shows the sequence patterns common to all sequences → can only find fully conserved patterns

Pattern Search in Unaligned Sequences

- Gibbs
  - Uses a probabilistic model to search for a pattern of length W in a set of N sequences.
  - Uses a model similar to PSSM construction
    - In addition the start of the pattern in each sequence is included in the model
  - Starts with a random pattern and iteratively refines the random patterns into an emergent pattern (about 100N iterations)
    - The starting positions are random first, the most likely starting positions are found at each step and the profile model is also modified accordingly
    - Another technique MEME is similar and uses the EM method.