Multiple Alignment

Outline

- Dynamic Programming in 3-D
- Progressive Alignment
- Profile Progressive Alignment (ClustalW)
- Scoring Multiple Alignments
- Entropy
- Sum of Pairs Alignment

Generalizing the Notion of Pairwise Alignment

- Up until now we have only tried to align two sequences to one another. What about more than two?
- 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

Outline - CHANGE

- Dynamic Programming in 3-D
- IN "Multiple Alignment: Greedy Approach" it is not clear which sequences are being merged. Before this slide create an extrra slide giving a geometrix interpretaion explaining that every 3-D multiple alignment has corresponding pairwise alignments
- Wrong tree in Step 2 (cont'd)
- Cryptic second sentence in ClustalW: Example" Next slide is also cryptic

Alignments = Paths in…

- 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

- Align the following 3 sequences:

ATGC, AATC,ATGC

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

-

Alignment Paths

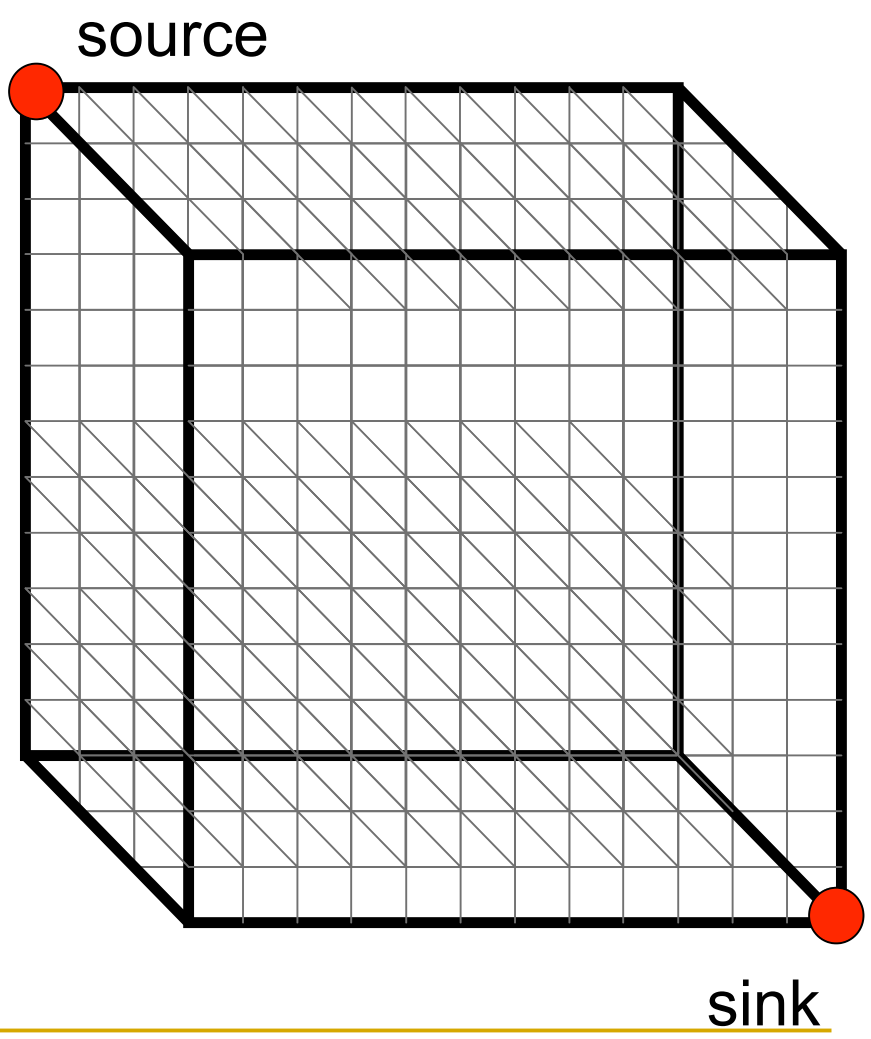
0	1	1	2	3	4	*x* coordinate
	A	--	T	G	C	
0	1	2	3	3	4	*y* coordinate
	A	A	T	--	C	
0	0	1	2	3	4	*z* coordinate
	--	A	T	G	C	

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

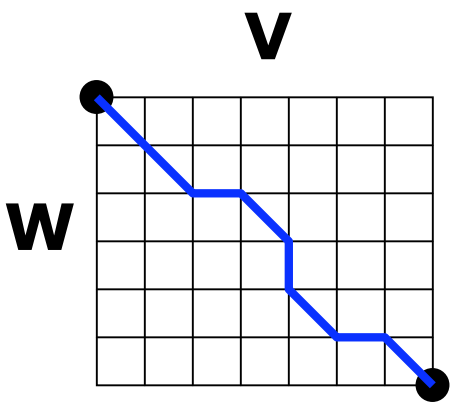
$(0,0,0)\rightarrow(1,1,0)\rightarrow(1,2,1)\rightarrow(2,3,2)\rightarrow(3,3,3)\rightarrow(4,4,4)$

Aligning Three Sequences

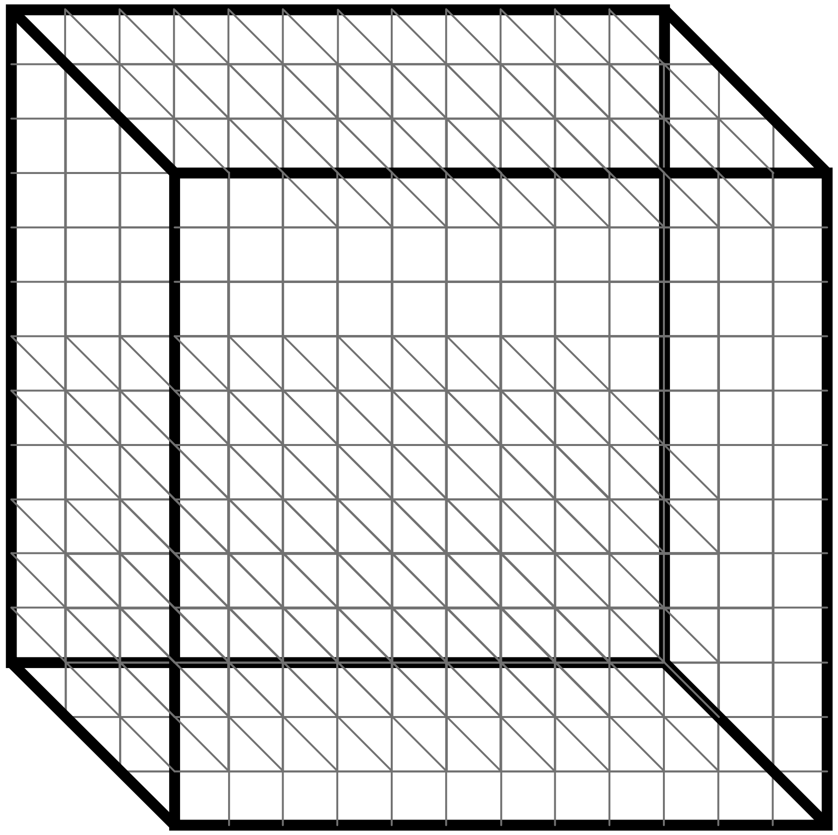
- Same strategy as aligning two sequences
- Use a 3-D “Manhattan Cube”, with each axis representing a sequence to align
- For global alignments, go from source to sink



2-D vs 3-D Alignment Grid

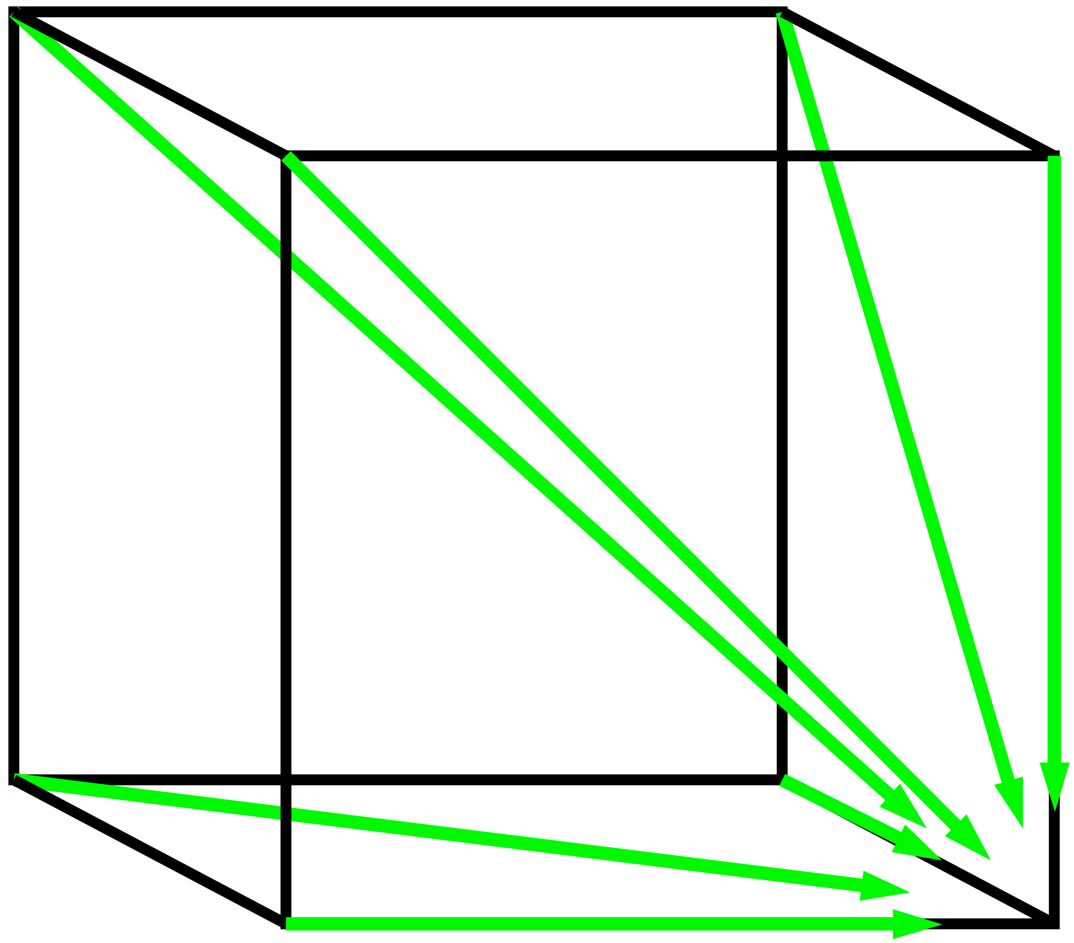


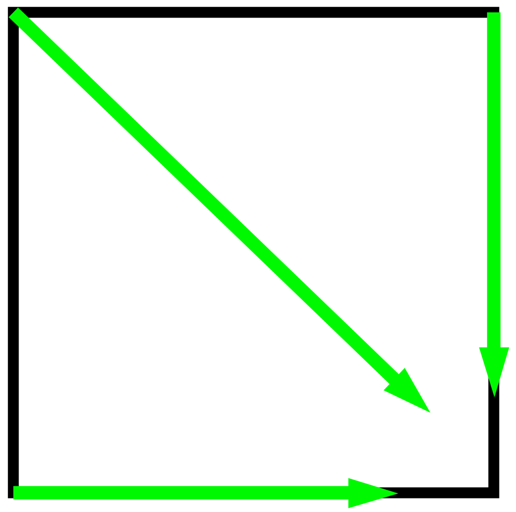
2-D edit graph



3-D?

Architecture of 3-D Alignment Grid

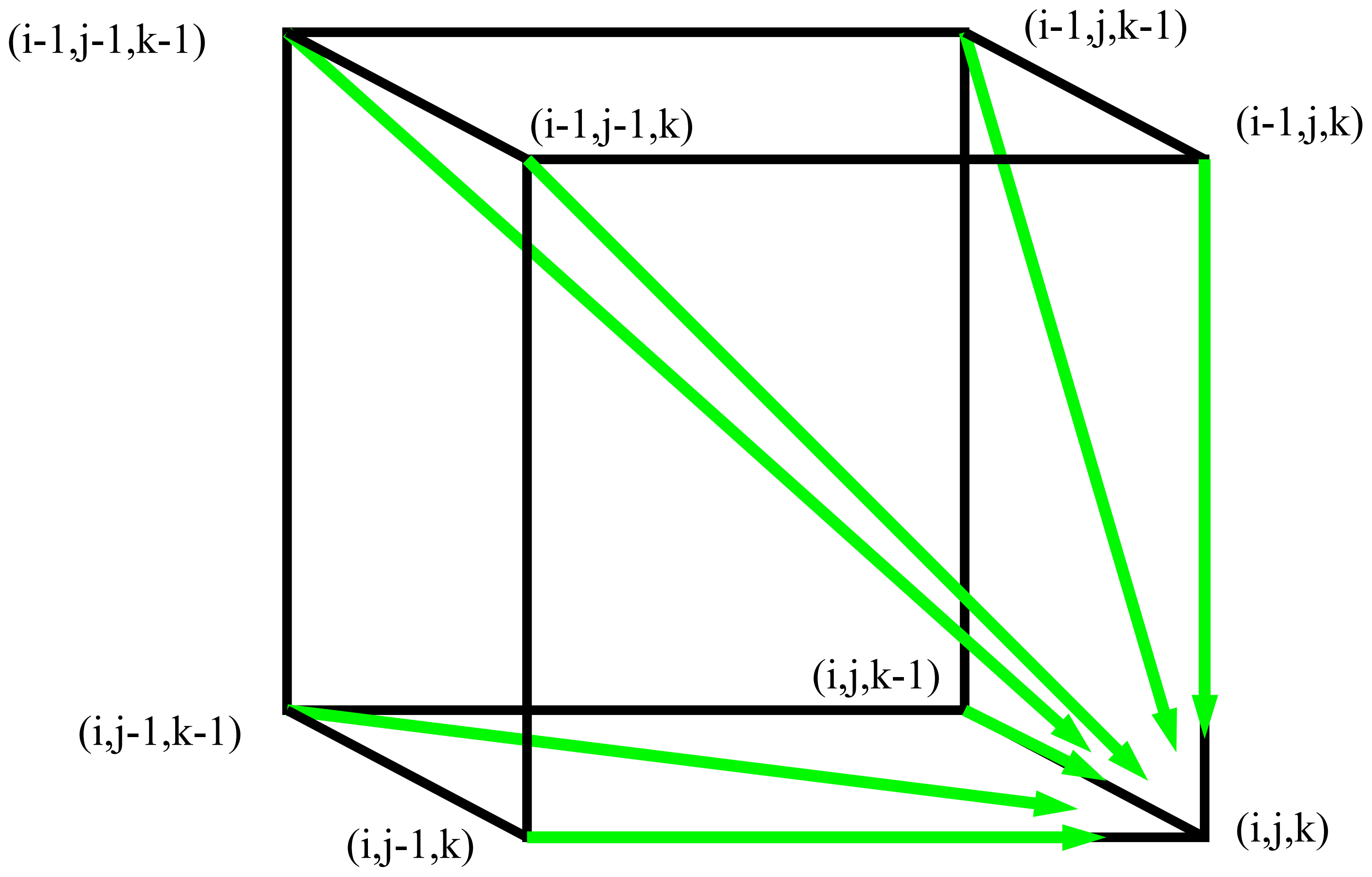




In **2-D**, 3 edges in each unit square

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

A Cell of 3-D Alignment Grid



Multiple Alignment: Dynamic Programming

- $$s_{i,j,k} = \max \begin{cases} s_{i-1,j-1,k-1} + \sigma(v_i, w_j, u_k) & \text{cube diagonal: no indels} \\ s_{i-1,j-1,k} + \sigma(v_i, w_j, \_) & \\ s_{i-1,j,k-1} + \sigma(v_i, \_, u_k) & \text{face diagonal: one indel} \\ s_{i,j-1,k-1} + \sigma(\_, w_j, u_k) & \\ s_{i-1,j,k} + \sigma(v_i, \_, \_) & \\ s_{i,j-1,k} + \sigma(\_, w_j, \_) & \text{edge diagonal: two indels} \\ s_{i,j,k-1} + \sigma(\_, \_, u_k) & \end{cases}$$

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

Multiple Alignment: Running Time

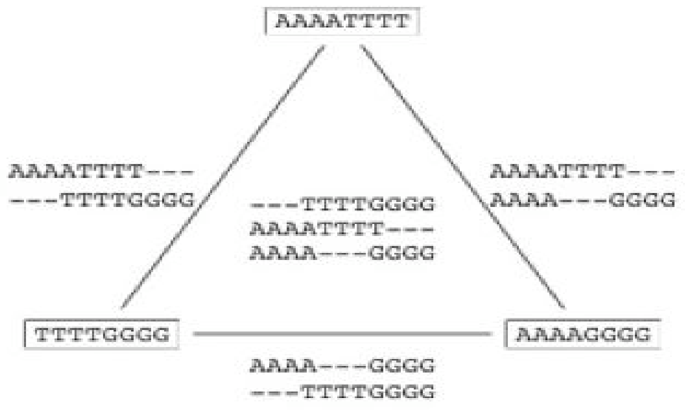
- For 3 sequences of length ***n***, the run time is $7\mathbf{n}^3$; $O(\mathbf{n}^3)$
- For ***k*** sequences, build a ***k***-dimensional Manhattan, with run time $(2^k-1)(\mathbf{n}^k)$; $O(2^k\mathbf{n}^k)$
- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to ***k*** sequences but it is impractical due to exponential running time

Inferring Multiple Alignment from Pairwise Alignments

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

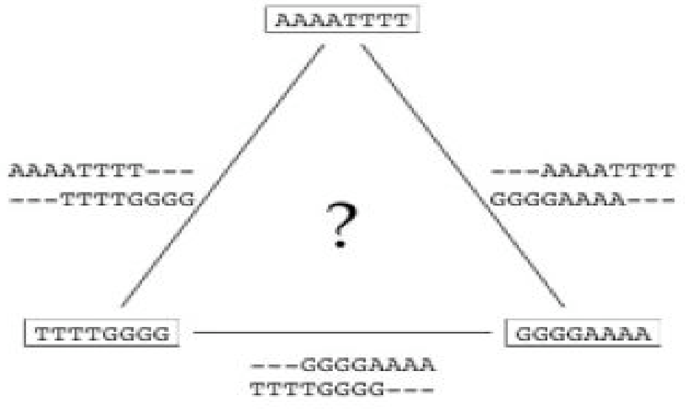
Combining Optimal Pairwise Alignments into Multiple Alignment

Can combine pairwise alignments into multiple alignment



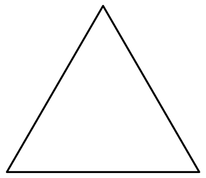
(a) Compatible pairwise alignments

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



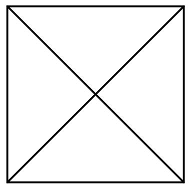
(b) Incompatible pairwise alignments

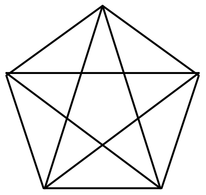
Inferring Pairwise Alignments



3 sequences, 3 comparisons

4 sequences, 6 comparisons

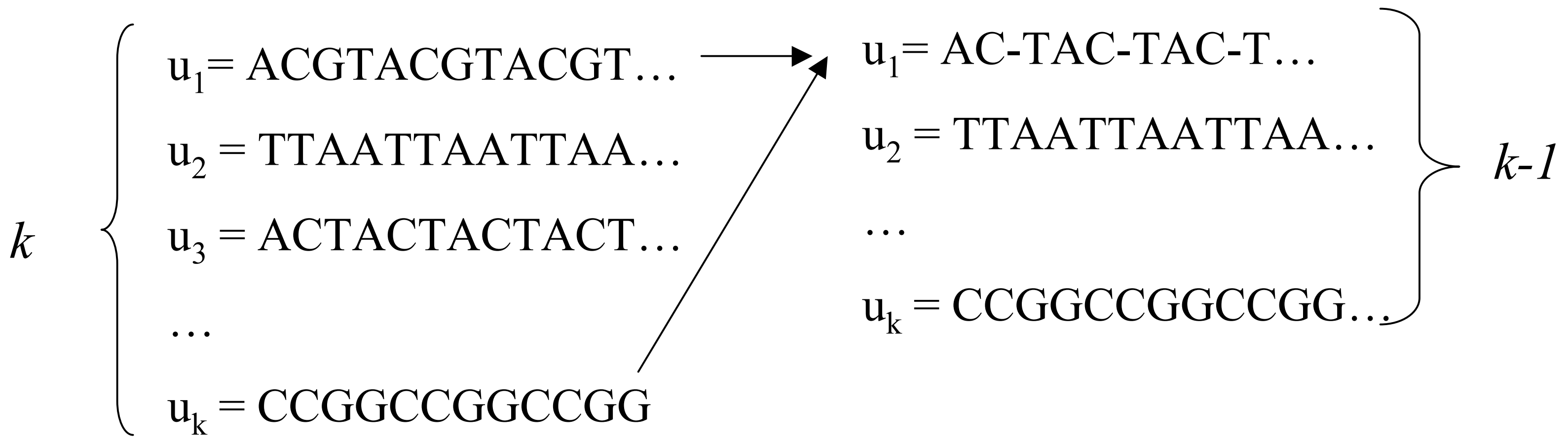




5 sequences, 10 comparisons

Multiple Alignment: Greedy Approach

- Choose most similar pair of strings and combine into a consensus, thereby reducing alignment of *k* sequences to an alignment of of *k-1* sequences. **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 **GTC**T**G**A
s4 **GTC**A**G**C (score = 2)

s1 **G**AT-**T**C**A**
s2 **G**-TCTG**A** (score = 1)

s1 **GAT**-**TCA**
s3 **GAT**AT-**T** (score = 1)

s1 **G**A**T**T**CA**--
s4 **G—T-CA**GC(score = 0)

s2 **G**-**T**CTGA
s3 **G**A**T**A**T**-T (score = -1)

s3 **G**AT-**A**TT
s4 **G**-TC**A**GC (score = -1)

Greedy Approach: Example (cont’d)

s2 and s4 are closest; combine:

s2	**GTC**T**G**A	s2,4 GTCTGA (consensus)
s4	**GTC**A**G**C	

There are many (4) alternative choices for the consensus, let’s assume we randomly choose one

new set becomes:

s1	GATTCA
s3	GATATT
s2,4	GTCTGA

Greedy Approach: Example (cont’d)

set is:

```
s1    GATTCA
s3    GATATT
s2,4  GTCTGA
```

scores are:

```
s1    GAT-TCA
s3    GATAT-T (score  = 1)

s1    GATTC--A
s2,4  G-T-CTGA (score  = 0)

s3    GATATT-
s2,4  G-TCTGA (score=-1)
```

Take best pair and form another consensus:

`s1,3 = GATATT` (arbitrarily break ties)

Greedy Approach: Example (cont’d)

new set is:

```
s1,3   GATATT
s2,4   GTCTGA
```

$\longrightarrow$

scores is:

```
s1,3   GATATT
s2,4   G-TCTGA  (score=-1)
```

Form consensus:

```
s1,3,2,4 = GATCTG
```

(arbitrarily break ties)

Progressive Alignment

- *Progressive alignment* is a variation of greedy algorithm with a somewhat more intelligent strategy for choosing a consensus
- Progressive alignment works well for close sequences, but deteriorates for distant sequences
  - Gaps in consensus string are permanent
  - Simplified representation of the alignments
- Better solution? Use a profile to represent consensus

```
ATG-CAA
AT-CCA-
ACG-CTG
```

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

ClustalW

- Popular multiple alignment tool today
- Several heuristics to improve accuracy:
  - Sequences are weighted by relatedness
  - Scoring matrix can be chosen “on the fly”
  - Position-specific gap penalties

ClustalW (cont’d)

- Often used for protein alignment
- ‘W’ stands for ‘weighted’
  - Different parts of alignment are weighted.
  - Position/residue specific gap penalties.
- Three-step process

  1.) Pairwise alignment

  2.) Build Guide Tree

  3.) Progressive Alignment

Step 1: Pairwise Alignment

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

	S_1	S_2	S_3	S_4
S_1	-			
S_2	.17	-		
S_3	.87	.28	-	
S_4	.59	.33	.62	-

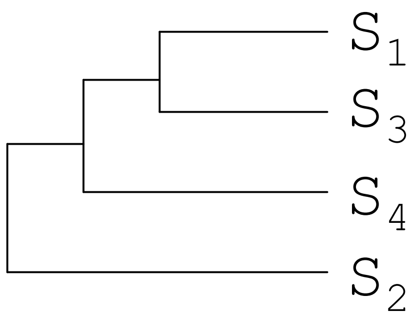
(.17 means 17 % identical)

Step 2: Guide Tree

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

Step 2: Guide Tree (cont’d)

$$
\begin{array}{c|cccc}
 & S_1 & S_2 & S_3 & S_4 \\
\hline
S_1 & - & & & \\
S_2 & .17 & - & & \\
S_3 & .87 & .28 & - & \\
S_4 & .59 & .43 & .62 & -
\end{array}
$$



Calculate:
s1,3 = consensus(s1, s3)
s1,3,4 = consensus((s1,3),s4)
s1,2,3,4 = consensus((s1,3,4),s2)

Step 3: Progressive Alignment

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

```
Sample output:

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: Example

- Each sequence has a weight; groups of related sequences have lower weight
- Sum the score matrix entry for all pairs and weight each pair by the sequences’ weight

ClustalW: Example (cont’ d)

- Scoring alignments of sequences 1x2 and 3x4

1: peeksav**t**al

2: geekaav**l**al

3: egewgl**v**lhv

4: aaektk**i**rsa

Score:

w(1)*w(3)*M(t,v) +

w(1)*w(4)*M(t,i) +

w(2)*w(3)*M(l,v) +

w(2)*w(4)*M(l,i)

ClustalW: Scoring Alignments

- Distance between sequences determines which scoring matrix to use
  - 80 - 100% → Blosum80
  - 60-80% → Blosum60
  - 30-60% → Blosum45
  - 0-30% → Blosum30

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
  - $p_A = 1$ or $p_A = 0.75$, $p_T = 0.25$
- Compute entropy of each column

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

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:

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

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

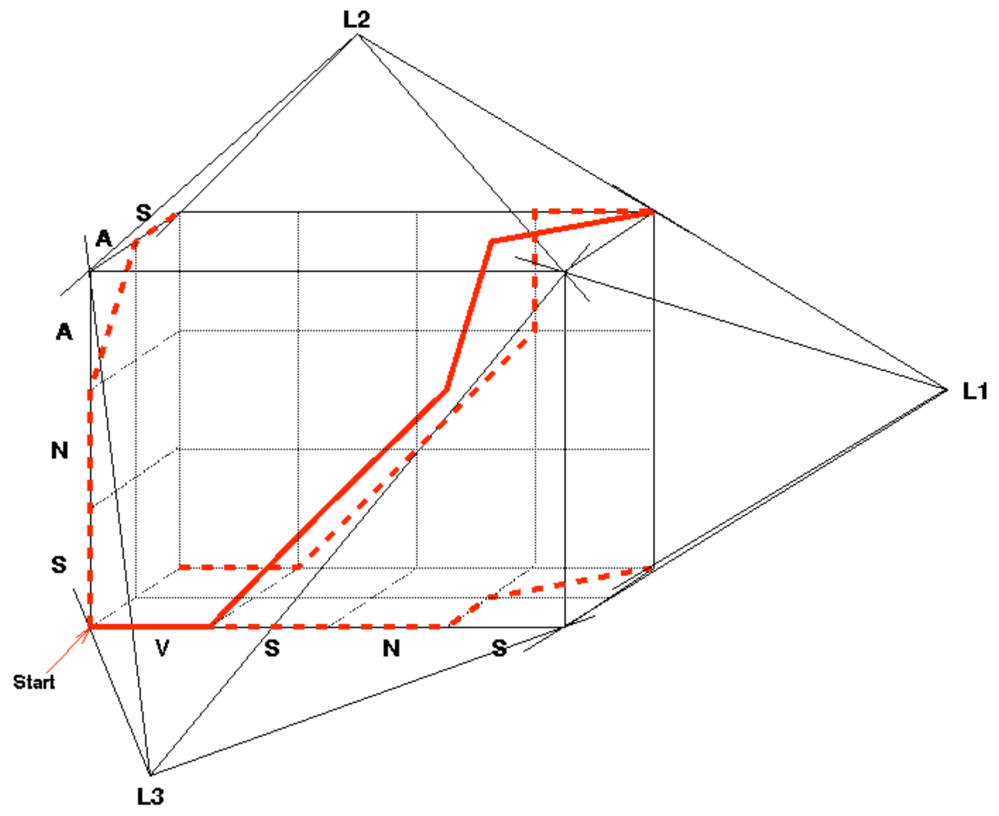
- Column 1 $= -[1*\log(1) + 0*\log 0 + 0*\log 0 + 0*\log 0]$
  $= 0$
- Column 2 $= -[(^1/_4)*\log(^1/_4) + (^3/_4)*\log(^3/_4) + 0*\log 0 + 0*\log 0]$
  $= -[\ (^1/_4)*(-2) + (^3/_4)*(-.415)\ ] = +0.811$
- Column 3 $= -[(^1/_4)*\log(^1/_4)+(^1/_4)*\log(^1/_4)+(^1/_4)*\log(^1/_4) +(^1/_4)*\log(^1/_4)]$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

Inferring Pairwise Alignments from Multiple Alignments

- From a multiple alignment, we can infer pairwise alignments between all sequences, but they are not necessarily optimal

- This is like projecting a 3-D multiple alignment path on to a 2-D face of the cube

Multiple Alignment Projections



All 3 Pairwise Projections of the Multiple Alignment

A 3-D alignment can be projected onto the 2-D plane to represent an alignment between a pair of sequences.

Sum of Pairs Score(SP-Score)

- Consider pairwise alignment of sequences

  a_i 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:

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

Computing SP-Score

Aligning 4 sequences: 6 pairwise alignments

Given a_1,a_2,a_3,a_4:

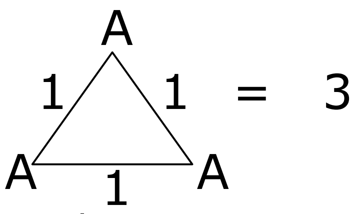
$$s(a_1 \ldots a_4) = \Sigma s^*(a_i,a_j) = s^*(a_1,a_2) + s^*(a_1,a_3) + s^*(a_1,a_4) + s^*(a_2,a_3) + s^*(a_2,a_4) + s^*(a_3,a_4)$$

SP-Score: Example

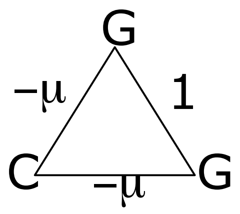
s_1 ATG-C-AAT
$\cdot$ A-G-CATAT
s_k ATCCCATTT

To calculate each column:

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



Column 1



Score = $1 - 2\mu$

Column 3