Multiple Alignment

Outline

1. Introduction to Multiple Alignment
2. Progressive Alignment
3. Scoring Multiple Alignments
4. Partial Order Alignment
5. A-Bruijn Approach to Multiple Alignment

Section 1: Introduction to Multiple Alignment

Multiple Sequence Alignment (MSA)

- Up until now we have only tried to align two sequences.

Multiple Sequence Alignment (MSA)

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

Multiple Sequence Alignment (MSA)

- Up until now we have only tried to align two sequences.
- What about aligning more than two sequences?
- A faint similarity between two sequences becomes significant if it is present in many other sequences.

Multiple Sequence Alignment (MSA)

- Up until now we have only tried to align two sequences.
- What about aligning more than two sequences?
- A faint similarity between two sequences becomes significant if it is present in many other sequences.
- Therefore multiple alignments can reveal subtle similarities that pairwise alignments do not reveal.

Generalizing Pairwise to Multiple 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
  - Example:

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

- Our scoring function should score alignments with conserved columns higher.

Alignments = Paths in 3-Space

- Say we have 3 sequences to align: ATGC, AATC, ATGC

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

Alignments = Paths in 3-Space

- Say we have 3 sequences to align: ATGC, AATC, ATGC

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

x coordinate

Alignments = Paths in 3-Space

- Say we have 3 sequences to align: 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	

Alignments = Paths in 3-Space

- Say we have 3 sequences to align: ATGC, AATC, ATGC

- Plotting the coordinates gives a path in 3-space:

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	

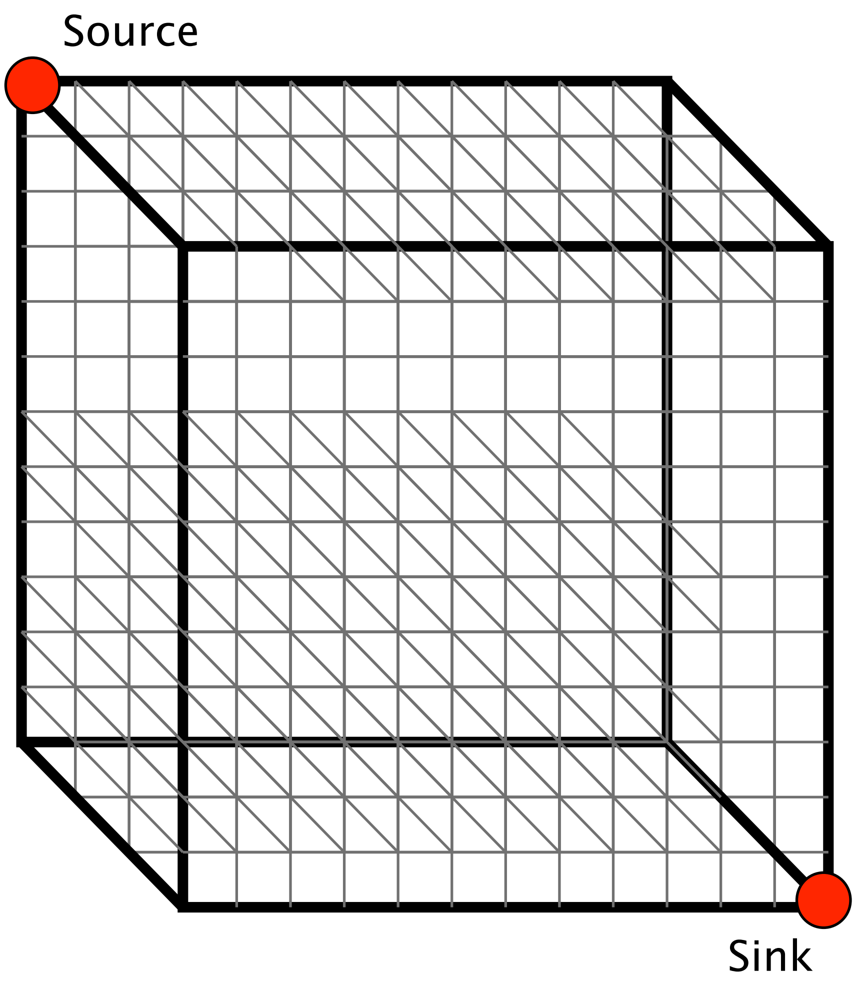
Alignments = Paths in 3-Space

- Say we have 3 sequences to align: ATGC, AATC, ATGC
- Plotting the coordinates gives a path in 3-space:
  - (0,0,0)→(1,1,0)→(1,2,1) →(2,3,2) →(3,3,3) →(4,4,4)

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	

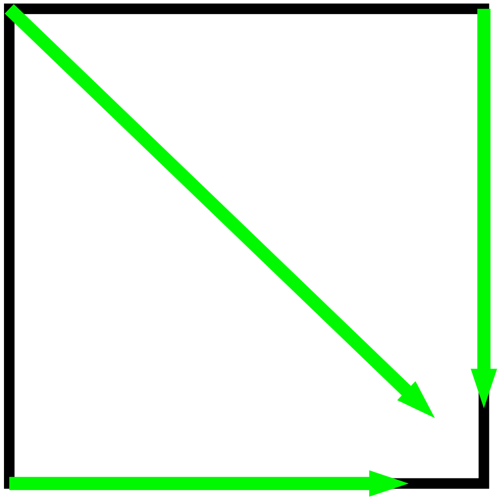
Alignments = Paths in 3-Space

- 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.

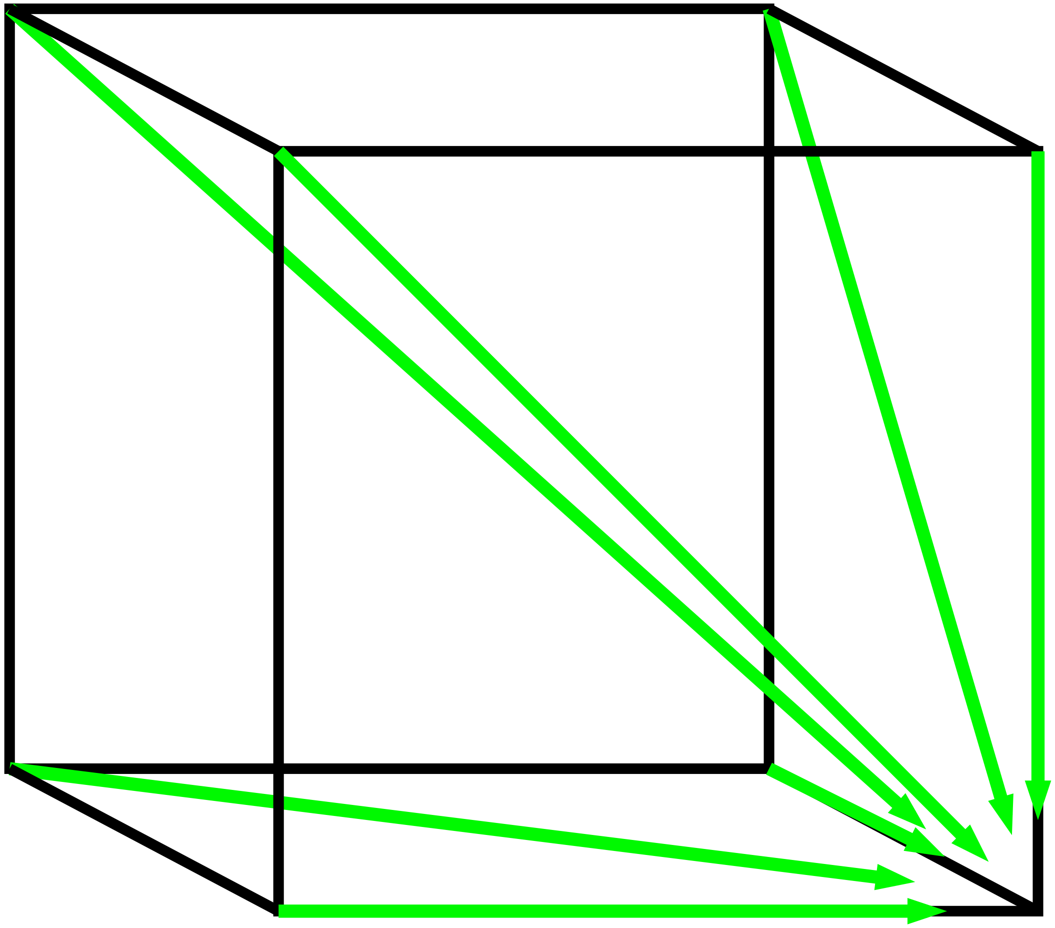


2-D Alignment Cell versus 3-D Alignment Cell

- In 2-D, 3 edges in each unit square
- In 3-D, 7 edges in each unit cube

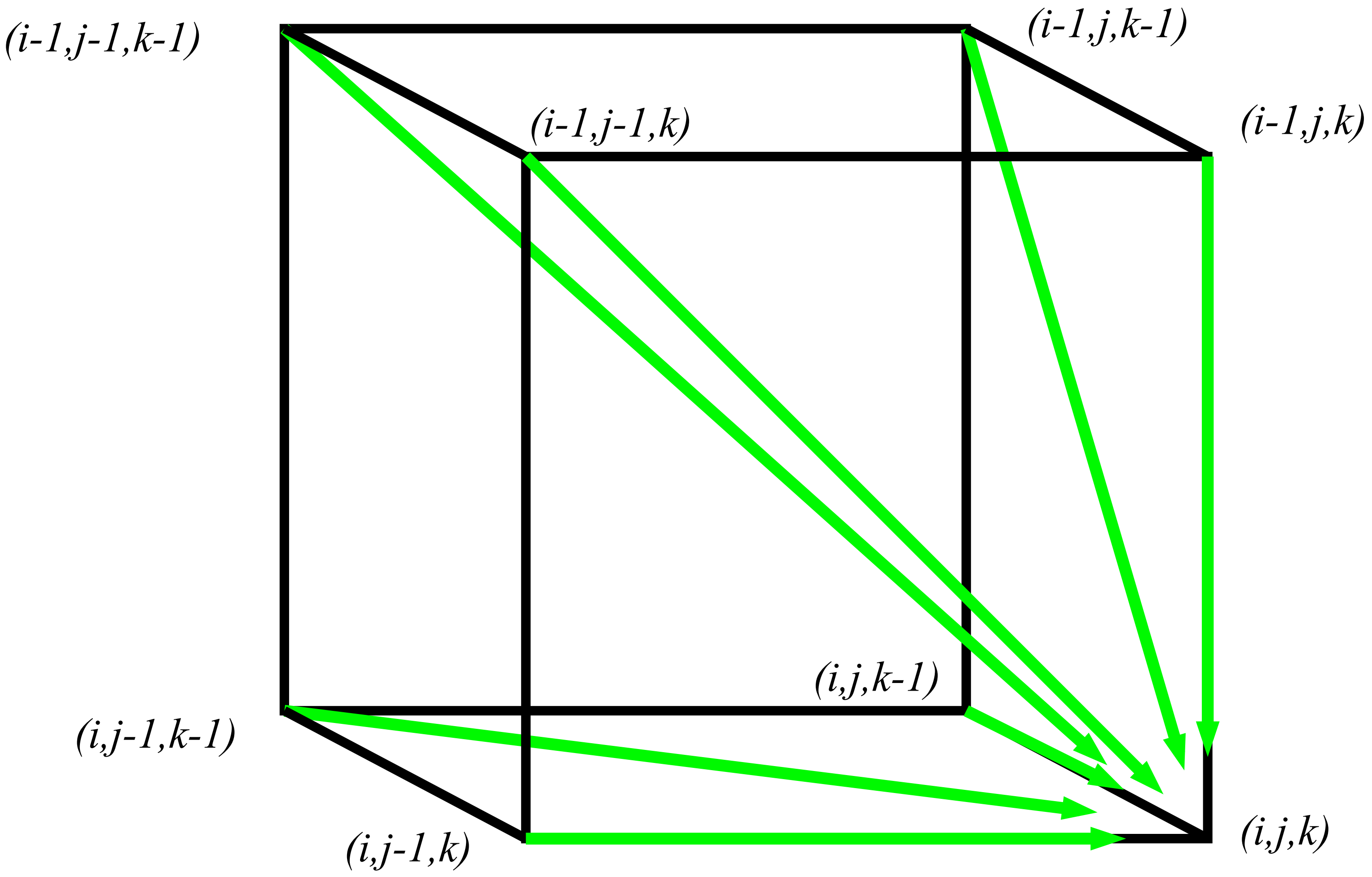


2-D Unit Square



3-D Unit Cube

Architecture of 3-D Alignment Cell



Multiple Alignment: Dynamic Programming

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

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

Multiple Alignment: Running Time

- For 3 sequences of length n, the run time is $7n^3 = O(n^3)$

- For generalization to k sequences, build a k-dimensional Manhattan graph:
  - There are n^k vertices in this graph.
  - Each vertex has $2^k - 1$ edges coming into it.
  - Therefore, run time $= (2^k - 1)(n^k) = O(2^k n^k)$

- **Conclusion**: The dynamic programming approach for alignment between two sequences is easily extended to k sequences but it is impractical due to a run time that is exponential in the number of sequences.

Multiple Alignment Induces Pairwise Alignments

- Every multiple alignment induces pairwise alignments
  - **Example**: The alignment

```
x:    A C - G C G G - C
y:    A C - G C - G A G
z:    G C C G C - G A G
```

induces the following three pairwise alignments:

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

Idea: Construct Multiple from Pairwise Alignments

- Given *k* arbitrary pairwise alignments, can we construct a multiple alignment that induces them?

- **Example**: 3 sequence alignment
  - x = ACGCTGGC, y = ACGCGAC, z = GCCGCAGAG
  - Say we have optimal pairwise alignments as follows:

```
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?

Idea: Construct Multiple from Pairwise Alignments

- Given *k* arbitrary pairwise alignments, can we construct a multiple alignment that induces them?

- **Example**: 3 sequence alignment
  - x = ACGCTGGC, y = ACGCGAC, z = GCCGCAGAG
  - Say we have optimal pairwise alignments as follows:

```
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?
  - **Answer**: Not always!

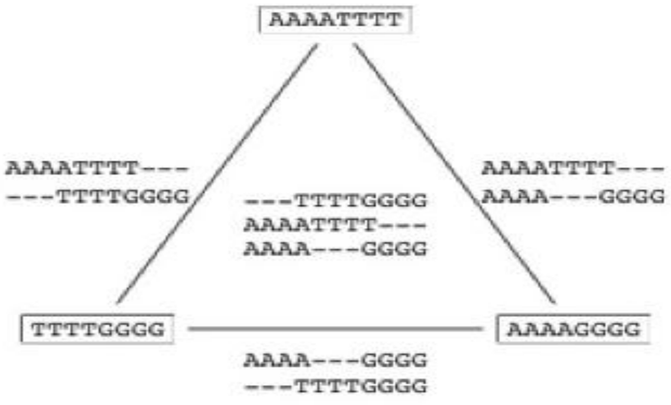
Idea: Construct Multiple from Pairwise Alignments

- From an optimal multiple alignment, we can infer pairwise alignments between all pairs of sequences, but they are not necessarily optimal.

- Likewise, it is difficult to infer a “good” multiple alignment from optimal pairwise alignments between all sequences.

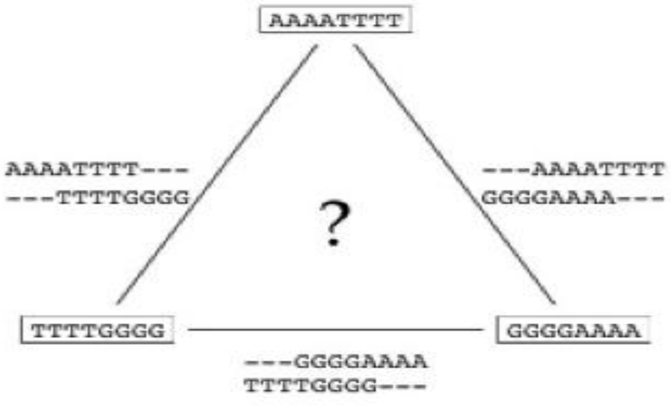
Idea: Construct Multiple from Pairwise Alignments

- **Example 1**: Can combine pairwise alignments into optimal multiple alignment.



(a) Compatible pairwise alignments

- **Example 2**: Can *not* combine pairwise alignments into optimal multiple alignment.



(b) Incompatible pairwise alignments

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

- In the past we were aligning a **sequence against a sequence**.

```
-  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

- In the past we were aligning a **sequence against a sequence**.
  - Can we align a **sequence against a profile?**

	-	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	

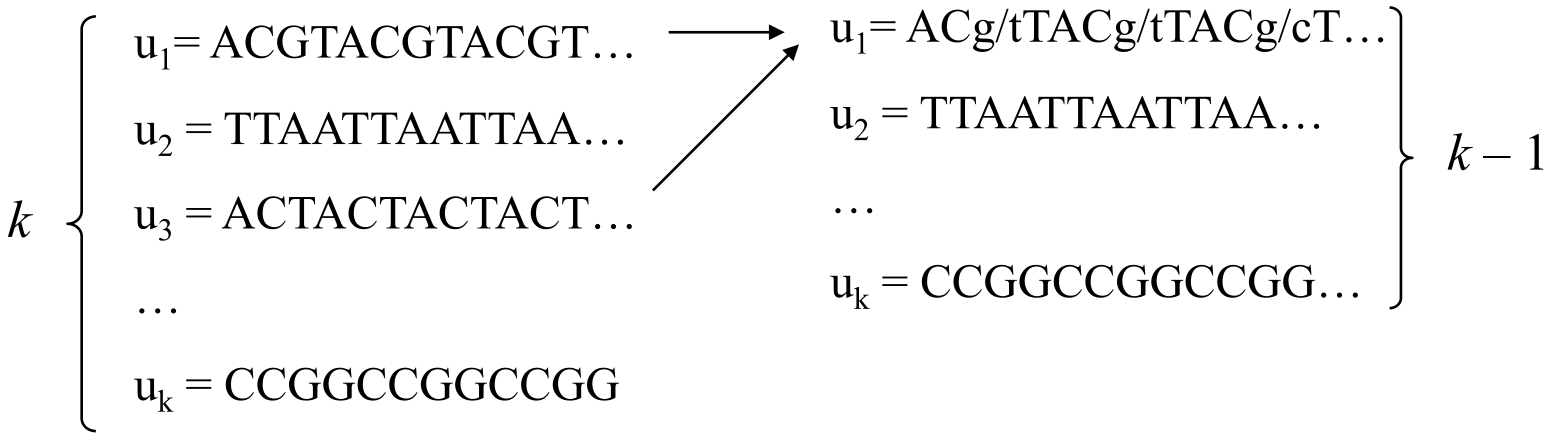
Profile Representation of Multiple Alignment

- In the past we were aligning a **sequence against a sequence.**
  - Can we align a **sequence against a profile?**
  - Can we align a **profile against a profile?**

	-	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	

Multiple Alignment: Greedy Approach

- Choose the most similar pair of strings and combine them into a profile, thereby reducing alignment of *k* sequences to an alignment of of *k* – 1 sequences/profiles.
- Then repeat.
- This is a **heuristic** (greedy) method.



k

u_1= ACGTACGTACGT…

u_2 = TTAATTAATTAA…

u_3 = ACTACTACTACT…

…

u_k = CCGGCCGGCCGG

u_1= ACg/tTACg/tTACg/cT…

u_2 = TTAATTAATTAA…

…

u_k = CCGGCCGGCCGG…

k – 1

Greedy Approach: Example

- Consider the 4 sequences: GATTCA, GTCTGA, GATATT, GTCAGC

Greedy Approach: Example

- Consider the 4 sequences: GATTCA, GTCTGA, GATATT, GTCAGC.
- There are $\binom{4}{2} = 6$ possible pairwise 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

- s_2 and s_4 are closest, so we consolidate these sequences into one by using the profile matrix:

$$\left.\begin{array}{ll} s2 & \mathrm{GTCTGA} \\ s4 & \mathrm{GTCAGC} \end{array}\right\} s_{2,4} = \mathrm{GTCt/aGa/cA}$$

- New set of 3 sequences to align:

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

- We can choose either of the nucleotides in question for $s_{2,\,4}$.

Section 2: Progressive Alignment

Progressive Alignment

- **Progressive alignment**: A variation of the greedy algorithm for multiple alignment 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.

ClustalW

- 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.
  3. Progressive alignment guided by the tree.

Step 1: Pairwise Alignment

- Aligns each sequence against each other, giving a similarity matrix.

- Similarity = exact matches / sequence length (percent identity).

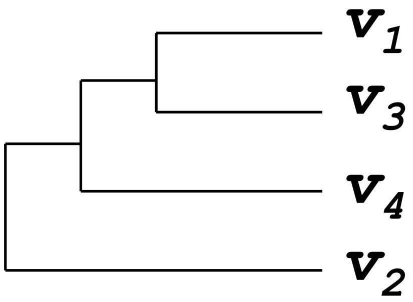
$$\begin{array}{c|cccc}
 & \boldsymbol{v_1} & \boldsymbol{v_2} & \boldsymbol{v_3} & \boldsymbol{v_4} \\
\hline
\boldsymbol{v_1} & - & & & \\
\boldsymbol{v_2} & .17 & - & & \\
\boldsymbol{v_3} & .87 & .28 & - & \\
\boldsymbol{v_4} & .59 & .33 & .62 & -
\end{array}$$

(.17 means 17 % identical)

Step 2: Guide Tree

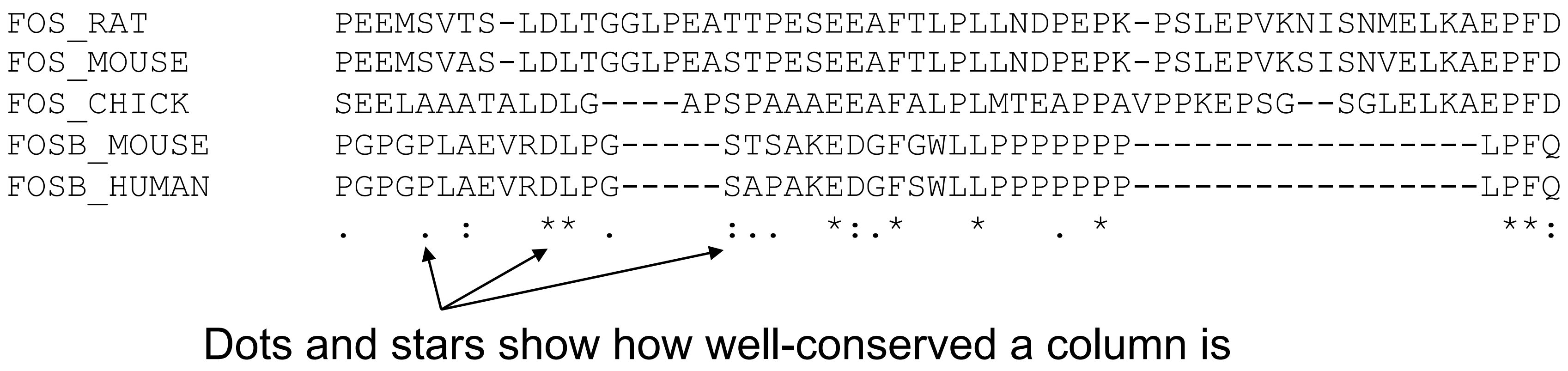
- Create guide tree using the similarity matrix.
- ClustalW uses the neighbor-joining method,
- Guide tree roughly reflects evolutionary relations.

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



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

Section 3: Scoring Multiple Alignments

Multiple Alignments: Scoring

- We will discuss three possible scoring systems:
  1. Number of matches (multiple longest common subsequence score)
  2. Entropy score
  3. Sum of pairs (SP-Score)

Score # 1: Multiple LCS Score

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

- **Example**: Only the first column in the following matching represents a “match:”

```
A A A
A A A
A A T
A T C
```

- The Multiple LCS score is the total number of matches.
  - This score is good for very similar sequences.

Score # 2: Entropy Score

- Define frequencies p_x for the occurrence of each letter x in each column of the multiple alignment.

- Then, compute "entropy" of each column.

$$\text{Entropy of Column} = -\sum_{X=A,T,G,C} p_X \log p_X$$

- The entropy score is then given by the sum of the entropies of all the columns.

Entropy: Example

- For our sequences {AAA, AAA, AAT, ATC}:

Entropy: Example

- For our sequences {AAA, AAA, AAT, ATC}:
  - 1st Column: $p_A = 1$, $p_T = p_G = p_C = 0$

$$\text{Entropy} = -\left[1 \cdot \log(1) + 0 + 0 + 0\right] = 0$$

Entropy: Example

- For our sequences {AAA, AAA, AAT, ATC}:
  - 1st Column: $p_A = 1$, $p_T = p_G = p_C = 0$

    $$\text{Entropy} = -\left[1\cdot \log(1) + 0 + 0 + 0\right] = 0$$

  - 2nd Column: $p_A = 0.75$, $p_T = 0.25$, $p_G = p_C = 0$

    $$\text{Entropy} = -\left[0.75\cdot \log(0.75) + 0.25\cdot \log(0.25) + 0 + 0\right] = 0.56$$

Entropy: Example

- For our sequences {AAA, AAA, AAT, ATC}:
  - 1st Column: $p_A = 1$, $p_T = p_G = p_C = 0$

    $$\text{Entropy} = -\left[1 \cdot \log(1) + 0 + 0 + 0\right] = 0$$

  - 2nd Column: $p_A = 0.75$, $p_T = 0.25$, $p_G = p_C = 0$

    $$\text{Entropy} = -\left[0.75 \cdot \log(0.75) + 0.25 \cdot \log(0.25) + 0 + 0\right] = 0.56$$

  - 3rd Column: $p_A = 0.50$, $p_T = 0.25$, $p_C = 0.25$, $p_G = 0$

    $$\text{Entropy} = -\left[0.5 \cdot \log(0.5) + 2 \cdot 0.25 \cdot \log(0.25) + 0\right] = 1.04$$

Entropy: Example

- For our sequences {AAA, AAA, AAT, ATC}:
  - 1st Column: $p_A = 1$, $p_T = p_G = p_C = 0$

    $$\text{Entropy} = -\left[1 \cdot \log(1) + 0 + 0 + 0\right] = 0$$

  - 2nd Column: $p_A = 0.75$, $p_T = 0.25$, $p_G = p_C = 0$

    $$\text{Entropy} = -\left[0.75 \cdot \log(0.75) + 0.25 \cdot \log(0.25) + 0 + 0\right] = 0.56$$

  - 3rd Column: $p_A = 0.50$, $p_T = 0.25$, $p_C = 0.25$, $p_G = 0$

    $$\text{Entropy} = -\left[0.5 \cdot \log(0.5) + 2 \cdot 0.25 \cdot \log(0.25) + 0\right] = 1.04$$

- Entropy Score = 0 + 0.56 + 1.04 = 1.60

Entropy: Interpretation

- The more similar the members of a column, the lower the entropy score.
  - **Example**: Best and worst cases:

$$entropy\begin{pmatrix} A \\ A \\ A \\ A \end{pmatrix} = 0 \qquad entropy\begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = -\sum \frac{1}{4}\log\left(\frac{1}{4}\right) = -4\left(\frac{1}{4} * -2\right) = 2$$

- Therefore, if we are searching for the best multiple alignment, we will want to *minimize* the entropy score.

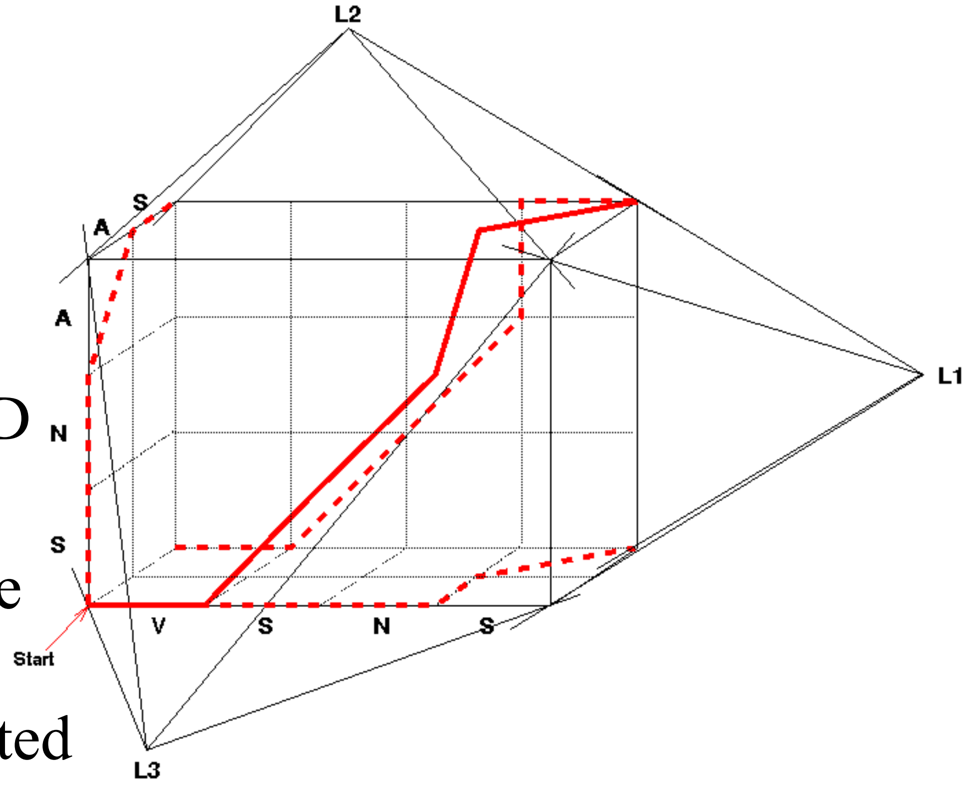
Inferring Pairwise from Multiple Alignments

- **Recall**: Every multiple alignment induces pairwise alignments.
- From a multiple alignment, we can infer pairwise alignments between all sequences, but they are not necessarily optimal.
- We can view reducing multiple alignments to pairwise alignments as projecting a 3-D multiple alignment path onto a 2-D face of the cube
- Our third scoring function for MSA will be based off the projections.

Multiple Alignment Projections: Illustration

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

- **Example**: Figure at right.
  - Solid line: represents a 3-D alignment path.
  - Dashed lines: represent the three induced pairwise alignments that are projected onto the cube's faces.



Score 3: Sum of Pairs Score (SP-Score)

- Consider the pairwise alignment of sequences a_i and a_j implied from a multiple alignment of k sequences.

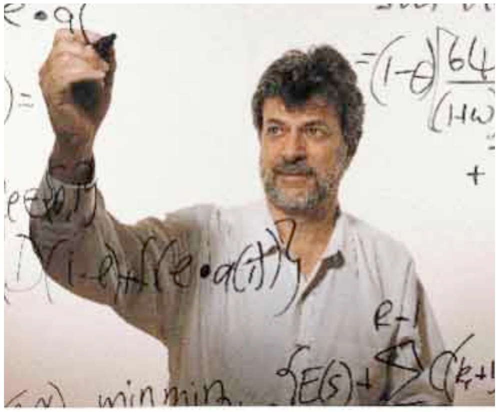
- Denote the score of this (not necessarily optimal) pairwise alignment as $s^*(a_i, a_j)$.

- **Sum of Pairs (SP) Score**: Obtained by summing the pairwise scores:

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

Multiple Alignment: History

- **1975**: Sankoff formulates multiple alignment problem and gives the dynamic programming solution.



David Sankoff

- **1988**: Carrillo and Lipman provide branch and bound approach for Multiple Alignment.

David Lipman

Multiple Alignment: History

- **1990**: Feng and Doolittle develop progressive alignment.

Russell Doolittle

- **1994**: Thompson, Higgins, and Gibson create ClustalW, which is the most popular multiple alignment program in the world.

Julie Thompson

Des Higgins

Toby Gibson

Multiple Alignment: History

- **1998**: Morgenstern et al. create DIALIGN, an algorithm for segment-based multiple alignment.

Burkhard Morgenstern

- **2000**: Notredame, Des Higgins, and Heringa develop T-Coffee, which aligns multiple sequences based off a library of pairwise alignments.

Cedric Notredame

Jaap Heringa

Multiple Alignment: History

- **2004**: Robert Edgar formulates MUSCLE, a faster and more efficient algorithm than ClustalW.

- **201X**: What is next?

Problems with Multiple Alignment

- Multidomain proteins evolve not only through point mutations but also through domain duplications and domain recombinations.

- Although Multiple Alignment is a 30 year old problem, there were no approaches for aligning *rearranged* sequences (i.e., multi-domain proteins with shuffled domains) prior to 2002.

- It is often impossible to align all protein sequences throughout their entire length.

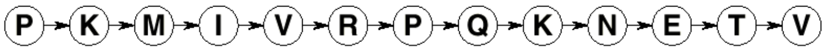
Section 4:
Partial Order Alignment

Alignment as a Graph

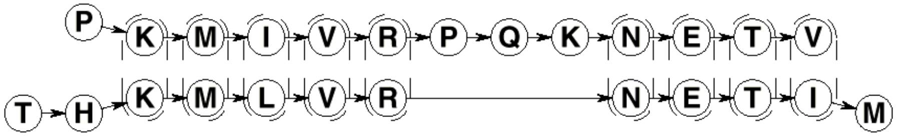
- Conventional Alignment

```
. . P K M I V R P Q K N E T V .
T H . K M L V R . . . N E T I M
```

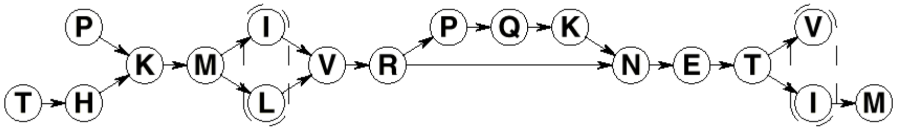
- Protein sequence as a path



- Two protein sequence paths

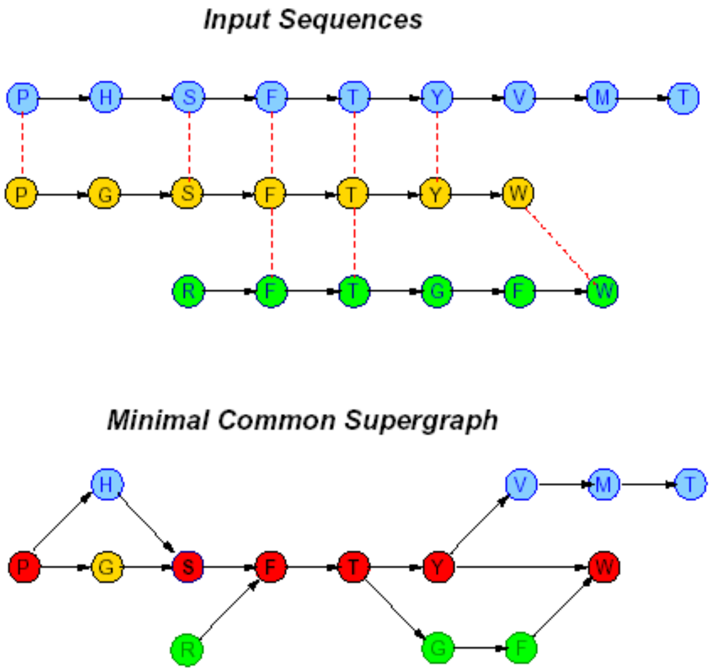


- Combination of two protein graphs into one graph



Representing Sequences as Paths in a Graph

- Each protein sequence is represented by a path.
- Dashed edges connect "equivalent" positions.
- Vertices with identical labels are fused.



Partial Order Multiple Alignment

- Two objectives:
  1. Find a graph that represents domain structure
  2. Find mapping of each sequence to this graph

- **Partial Order Alignment (POA)**: A graph such that every sequence in the given set is a path in G.

POA Algorithm

- Aligns sequences onto a directed acyclic graph (DAG)

- Outline:

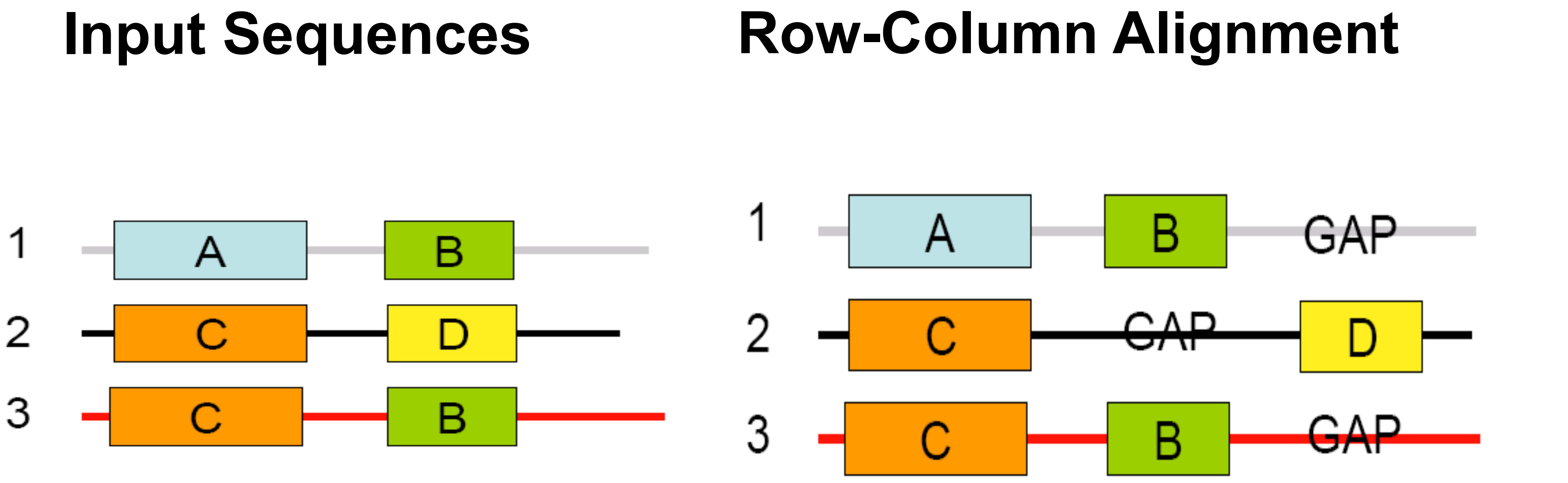
1. Guide Tree Construction
2. Progressive Alignment Following Guide Tree
3. Dynamic Programming Algorithm to align two POAs (POA-POA Alignment).
    - We learned how to align one sequence (*path*) against another sequence (*path*).
    - We need to develop an algorithm for aligning a directed *graph* against a directed *graph*.

Dynamic Programming for Aligning Two Graphs

- *S*(*n*, *o*) = optimal score for *n* = node in G, *o* = node in G'
    - Match/mismatch: Aligning two nodes with score s(*n*,*o*)
    - Deletion/insertion:
        - Omitting node *n* from the alignment with score Δ(*n*)
        - Omitting node *o* from the alignment with score Δ(*o*)
    - Dynamic formula for *S*(*n*, *o*):

$$S(n,o) = \max_{p \to n, q \to o} \begin{cases} S(p,q) + s(n,o) \\ S(p,o) + \Delta(n) \\ S(n,q) + \Delta(o) \end{cases}$$

Row-Column Alignment



POA Advantages

- POA is more flexible: standard methods force sequences to align linearly.

- POA representation handles gaps more naturally and retains (and uses) all information in the MSA (unlike linear profiles).

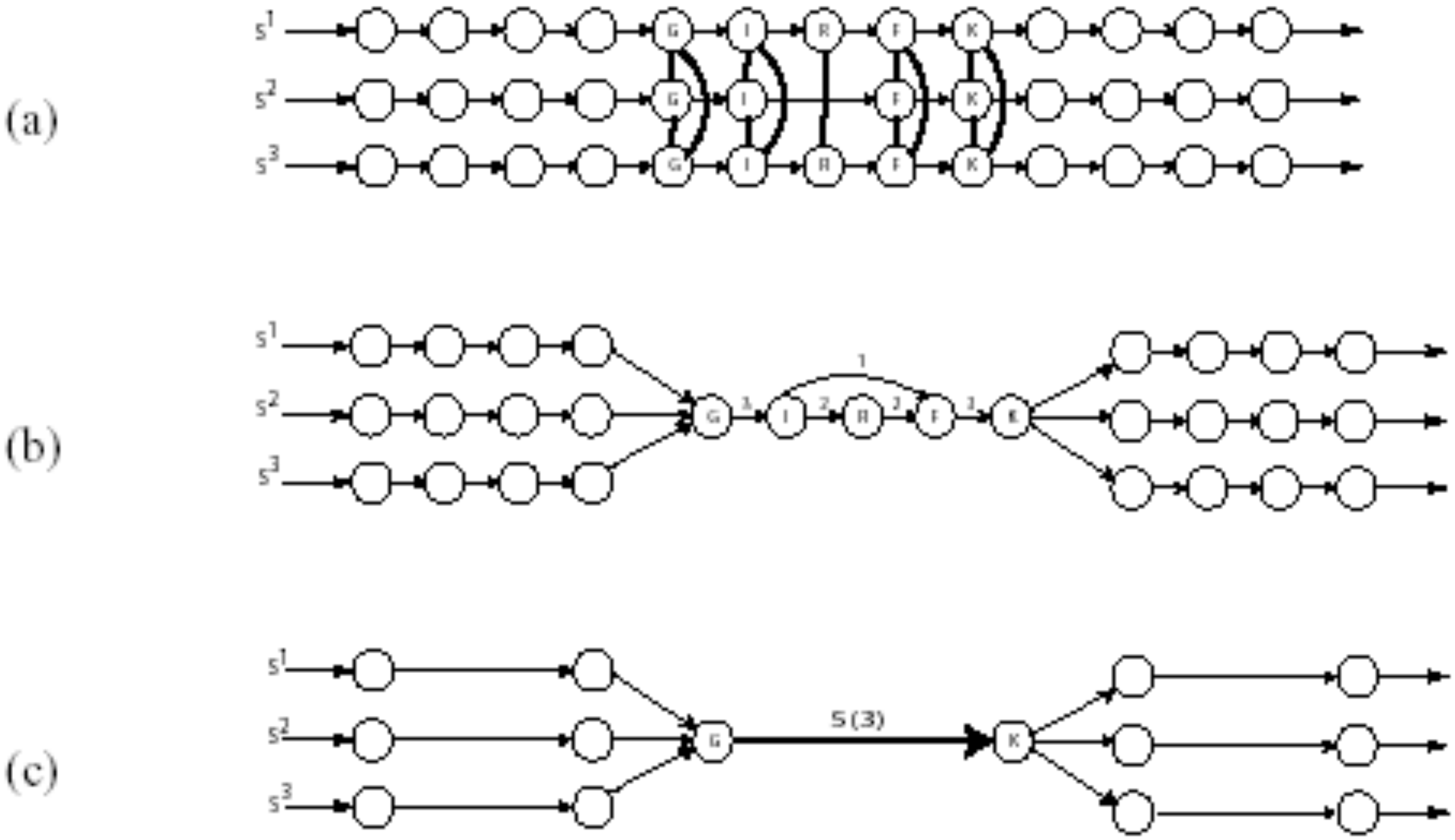
Section 5: A-Bruijn Approach to Multiple Alignment

A-Bruijn Alignment

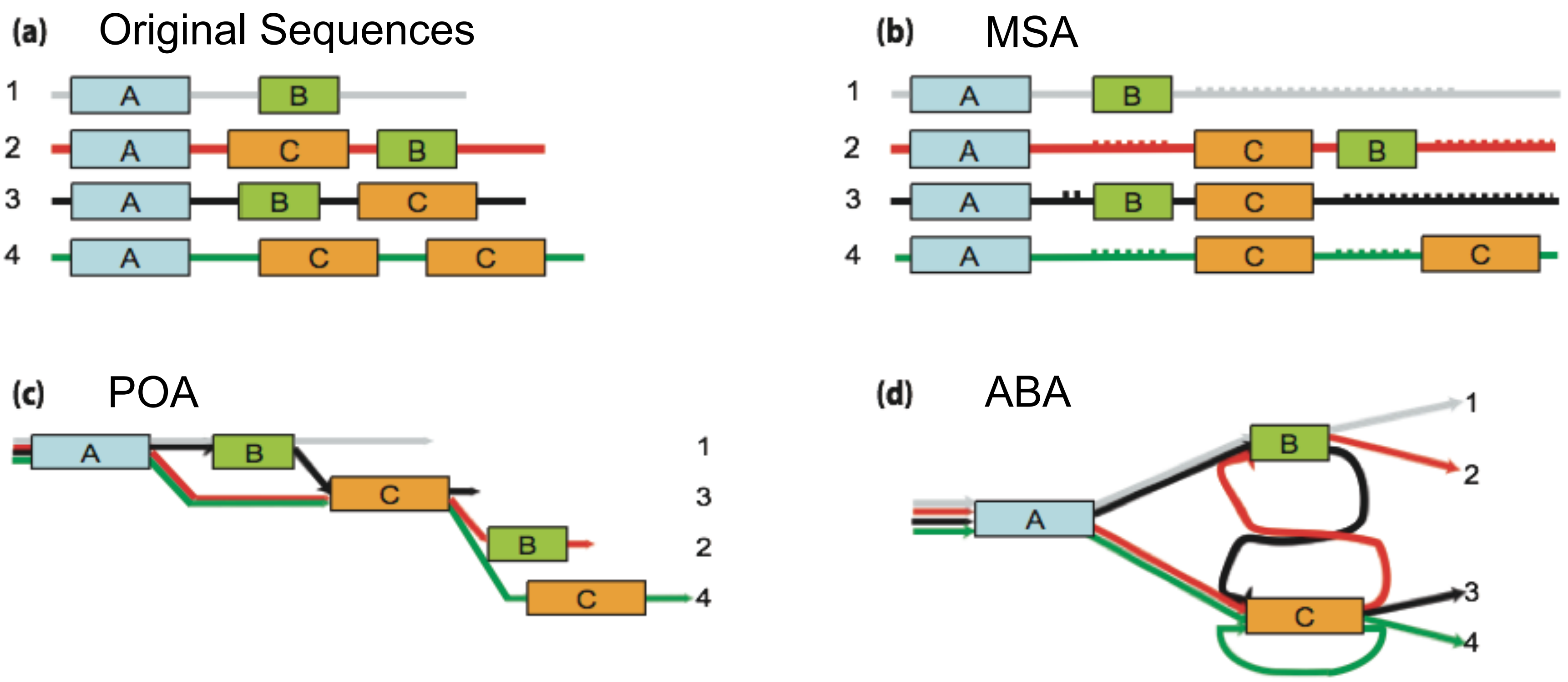
- **A-Bruijn Alignment (ABA)**: Represents alignment as directed graph that may contains cycles.

- This is in contrast to POA, which represents alignment as an acyclic directed graph.

ABA: How Is the Graph Created?



MSA vs. POA vs. ABA


(a) Original Sequences

(b) MSA

(c) POA

(d) ABA

Advantages of ABA

1. More flexible than POA: allows larger class of evolutionary relationships between aligned sequences
2. Can align proteins with shuffled and/or repeated domain structure
3. Can align proteins with domains present in multiple copies in some proteins
4. Handles:
   - Domains not present in all proteins.
   - Domains present in different orders in different proteins.

Credits

- *Chris Lee,* POA, *UCLA http://www.bioinformatics.ucla.edu/poa/Poa_Tutorial.html*