Lecture 5, Multiple Alignment

Young-Rae Cho
Associate Professor
Department of Computer Science
Baylor University

Pairwise Alignment vs. Multiple Alignment

- Pairwise Alignment
  - Alignment of two sequences
  - Sometimes two sequences are functionally similar or have a common ancestor although they have weak sequence similarity

- Multiple Alignment
  - Alignment of more than two sequences
  - Finds invisible similarity in pairwise alignment
Alignment of 3 Sequences

- Alignment of 2 Sequences
  - Described in a 2-row representation
  - Best alignment is found in a 2-D grid by dynamic programming

- Alignment of 3 Sequences
  - Described in a 3-row representation
  - \( x = \text{"ATGTG"}, y = \text{"ACGTA"}, z = \text{"ATCTG"} \)
  - Best alignment is found in a 3-D grid by dynamic programming

Alignment in 3-D Grid

- 3-D Edit Graph
  - 3-D grid structure (cube) with diagonals in each cell

- Example
  - Path in 3-D grid:
  \[(0,0,0) \rightarrow (1,1,1) \rightarrow (2,1,2) \rightarrow (2,2,3) \rightarrow (3,3,3) \rightarrow (4,4,4) \rightarrow (5,4,5) \rightarrow (5,5,5)\]
3-D Grid Unit

- 2-D Grid Unit Cell
  - Maximum 3 edges in each unit of 2-D grid

- 3-D Grid Unit Cell
  - Maximum 7 edges in each unit of 3-D grid

\[(i, j, k)\]
\[(i-1, j, k)\]
\[(i, j-1, k)\]
\[(i, j, k-1)\]
\[(i-1, j-1, k)\]
\[(i-1, j, k-1)\]
\[(i, j-1, k-1)\]
\[(i-1, j-1, k-1)\]

Solving by Dynamic Programming

- Formula
  \[
  S_{i,j,k} = \max \begin{cases} 
  S_{i-1,j,k} + \delta(x_i, -, -) \\
  S_{i,j-1,k} + \delta(-, y_j, -) \\
  S_{i,j,k-1} + \delta(-, -, z_k) \\
  S_{i-1,j-1,k} + \delta(x_i, y_j, -) \\
  S_{i-1,j,k-1} + \delta(x_i, -, z_k) \\
  S_{i,j-1,k-1} + \delta(-, y_j, z_k) \\
  S_{i-1,j-1,k-1} + \delta(x_i, y_j, z_k) 
  \end{cases}
  \]

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

- Runtime ?
from 3-D Alignment to Multiple Alignment

- Alignment of \( k \) Sequences
  - Able to be solved by dynamic programming in \( k \)-D grid
  - Runtime?

- Conclusion
  - Dynamic programming for pairwise alignment can be extended to multiple alignment
  - However, computationally impractical
  - How can we solve this problem?

Heuristics of Multiple Alignment

- Background
  - Implementing pairwise alignment (2-D alignment) \( k \) times is better than implementing \( k \)-D multiple alignment once

- Heuristic Process
  1. Implementing all possible pairwise alignments
  2. Combining the most similar pair iteratively
Multiple Alignment Projection

- Multiple Alignment $\iff$ Pairwise Alignments

\[ x: AC - GCGG - C \]
\[ y: AC - GC - G AG \]
\[ z: GCCGC - G AG \]

- Projection

Pairwise Alignment to Multiple Alignment

- Pairwise Alignments $\rightarrow$ Multiple Alignment

\[ x: AC - GCGG - C \]
\[ y: ACGC - G AG \]

\[ x: AC - GCGG - C \]
\[ z: GCCGC - G AG \]

\[ y: AC - GC - G AG \]
\[ z: GCCGC - G AG \]

\[ y: AC - GC - G AG \]
\[ z: GCCGC - G AG \]

- Can we construct a multiple alignment that induces pairwise alignments?

- Conclusion
  - Can’t infer optimal multiple alignment from all optimal pairwise alignments
Greedy Approach (1)

- **Process**
  1. Choose the most similar pair of sequences
  2. Merge them into a new sequence
  3. Choose the most similar sequence to the new sequence
  4. Repeat (2) and (3) until choosing all sequences

- **Example**
  - Step 1
    - \( s_1: \) GATTCA
    - \( s_2: \) GTCTGA
    - \( s_3: \) GATATT
    - \( s_4: \) GTCA
    - \( s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_1 \)

Greedy Approach (2)

- **Example - continued**
  - Step 2
    - \( s_2, s_4 \rightarrow s_{2,4} \)
    - \( s_{2,4} \rightarrow s_{2,4} \)
  - Step 3
    - \( s_1: \) GATTCA
    - \( s_3: \) GATATT
    - \( s_{2,4}: \) GTCA

- **Features**
  - \( k \)-way alignment (alignment of \( k \) sequences) → Runtime?
  - Greedy algorithm → Not optimal multiple alignment
Progressive Alignment (1)

- **Features**
  - A variation of greedy algorithm (more intelligent strategy on each step)
  - Also called *hierarchical method*
  - Uses profiles to compare sequences
  - Gaps are permanent (“once a gap, always a gap”)
  - Works well for close sequences

- **Process**
  - **Stage 1**
    - Computes sequence identity of all possible pairs of sequences
      (identity = #match / sequence length)
    - Makes a similarity matrix

| V1  | V2  | V3  | V4  | ...
|-----|-----|-----|-----|-----
| -   | 0.17| 0.28| 0.59| 0.33| 0.62|

Progressive Alignment (2)

- **Process - continued**
  - **Stage 2**
    - Creates a guide tree using the similarity matrix
  - **Stage 3**
    - Applies a series of pairwise alignment following the guide tree
Application of Progressive Alignment

- ClustalW
  - Popular multiple alignment tool
  - Adopts the progressive multiple alignment

<table>
<thead>
<tr>
<th>Species</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOS_RAT</td>
<td>PEEMSVTS-LDLTGGLP-ESTPESEAFTLFLNDPEPK-PSLEPVKNIHMELKARPFDD</td>
</tr>
<tr>
<td>FOS_MOUSE</td>
<td>PEEMSVAAS-LDLTGGLP-ESTPESEAFTLFLNDPEPK-PSLEPVKNIHMELKARPFDD</td>
</tr>
<tr>
<td>FOS_CHICK</td>
<td>SEELAAATALDLG--------APSFAAEEAFALMLTEAPPAVPPKEPSG--SGLELKARPFDD</td>
</tr>
<tr>
<td>FOSB_MOUSE</td>
<td>PGPGPLAEVRLPG-----STSAKEDGFVMLLPPPPPP----------------------LPFQ</td>
</tr>
<tr>
<td>FOSB_HUMAN</td>
<td>PGPGPLAEVRLPG-----SAPAKEDGFVMLLPPPPPP----------------------LPFQ</td>
</tr>
</tbody>
</table>

Dots and stars show how well-conserved a column is

Scoring Schemes

- Number of Matches
  - Multiple longest common subsequence score
  - A column is a “match” if all the letters in the column are the same
    - AAA
    - AAG
    - AAT
    - ATC
  - Only good for very similar sequences

- Sum-of-Pair Scoring

- Entropy-Based Scoring
**Sum-of-Pair Scoring**

- **Sum-of-Pairs Scoring in Multiple Alignment**
  - Consider pairwise alignment of sequences, \( a_i \) and \( a_j \), imposed by a multiple alignment of \( k \) sequences
  - Denote the score of the pairwise alignment as \( S^*(a_i, a_j) \)
  - Sum up the pairwise scores for a multiple alignment:
    \[
    S(a_1, a_2, \cdots, a_k) = \sum_{i,j} S^*(a_i, a_j)
    \]

- **Example**
  - Aligning 4 sequences, \( a_1, a_2, a_3, \) and \( a_4 \), by
    \[
    S(a_1, a_2, a_3, a_4) = S^*(a_1, a_2) + S^*(a_1, a_3) + S^*(a_1, a_4) + S^*(a_2, a_3) + S^*(a_2, a_4) + S^*(a_3, a_4)
    \]

**Entropy-Based Scoring (1)**

- **Entropy in Information Theory**
  - A measure of the uncertainty associated with a random variable
  - \( H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \)

- **Entropy-Based Scoring in Multiple Alignment**
  - (1) Define frequencies for the occurrence of each letter on each column
  - (2) Compute entropy of each column
  - (3) Sum all entropies over all columns
Entropy-Based Scoring (2)

- **Example**
  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>AAG</td>
<td>AAT</td>
</tr>
<tr>
<td>ATC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  - **Frequency**
    - 1st column: $p(A) = 1$, $p(T) = p(G) = p(C) = 0$
    - 2nd column: $p(A) = 0.75$, $p(T) = 0.25$, $p(G) = p(C) = 0$
    - 3rd column: $p(A) = 0.25$, $p(T) = 0.25$, $p(C) = 0.25$, $p(G) = 0.25$

  - **Entropy**
    
    $$
    H_{AA} = 0 \\
    H_{AT} = \frac{3}{4} \log_4 \frac{3}{4} = -0.602 \\
    H_{AG} = \log_4 1 = 0.000 \\
    H_{AC} = \log_4 1 = 0.000 \\
    H_{TG} = \frac{1}{4} \log_4 \frac{1}{4} = -0.500 \\
    H_{TC} = \frac{1}{4} \log_4 \frac{1}{4} = -0.500 \\
    H_{GC} = \log_4 1 = 0.000 \\
    H_{GC} = \log_4 1 = 0.000
    $$

  - Entropy-based score in multiple alignment: $0 + 0.244 + 0.602$

Advanced Multiple Alignment

- **Background**
  
  - Progressive sequence alignment has loss of information
    -> not optimal even though very efficient
  - Multi-domain proteins evolve not only through point mutations
    but also through domain duplications and domain re-combinations
    -> Rearrangement might be meaningful for aligning multi-domain
    protein sequences

- **Examples**
  
  - Partial Order Multiple Sequence Alignment (PO-MSA)
  - A-Bruijn Alignment (ABA)
### Alignment as a Graph

Conventional pairwise alignment

A path of a sequence

Combining two paths

Combined directed acyclic graph (partial order)

---

### PO-MSA Algorithm

- **Partial Order Multiple Sequence Alignment (PO-MSA)**
  - Considers a set of sequences $S$ as a directed acyclic graph $G$ such that each sequence in $S$ is a path in $G$
    - (Each sequence is mapped into the graph.)
  - Focuses on ordering rather than positions

- **Algorithm**
  1. Construct a guide tree
  2. Apply progressive alignment following the guide tree
  3. Align two directed acyclic graphs (Partial Order Alignment) using dynamic programming algorithm at each step
Partial Order Alignment (1)

- **Schematic View**
  - (a) Conventional pairwise alignment
  - (b) PO-MSA

Partial Order Alignment (2)

- **Advantages**
  - **Scalability**
    - Linear increase of time complexity as the increment of predecessors
  - **Accuracy**
    - Homologous recombination for multi-domain protein sequences
      (The graph represents domain structure.)
Questions?

- Lecture Slides are found on the Course Website,
  web.ecs.baylor.edu/faculty/cho/3360