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

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
Pairwise Alignment to Multiple Alignment

- Pairwise Alignments → Multiple Alignment

x: \[\text{ACGC} \quad \text{G} \quad \text{C} \quad \text{G} \quad \text{G} \]
y: \[\text{ACG} \quad \text{C} \quad \text{G} \quad \text{A} \quad \text{G} \]

x: \[\text{AC} \quad \text{G} \quad \text{CG} \quad \text{G} \quad \text{C} \]
z: \[\text{GCG} \quad \text{C} \quad \text{G} \quad \text{A} \quad \text{G} \]

y: \[\text{AC} \quad \text{G} \quad \text{CG} \quad \text{A} \quad \text{G} \]
z: \[\text{GCG} \quad \text{C} \quad \text{GAG} \]

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

Multiple Alignment Projection

- Projection

- Conclusion
  - Can’t infer optimal multiple alignment from all optimal pairwise alignments
  - Example?
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
    - \( s_4 \): GTCAGC

  - **Step 2**
    - \( s_2 \) GTCTGA
    - \( s_4 \) GTCAGC

  - **Step 3**
    - \( s_1 \) GATTCA
    - \( s_2 \) GTCTGA
    - \( s_3 \) GATATT

  - **Step 4**
    - \( s_1 \) GATTCA
    - \( s_2 \) GTCTGA
    - \( s_4 \) GTCAGC

Greedy Approach (2)

- **Example - continued**
  - **Step 2**
    - \( s_2 \) GTCTGA
    - \( s_4 \) GTCAGC

  - **Step 3**
    - \( s_2, 4 \) GTCT/aGa/c
      - (called profile or consensus sequence)

- **Features**
  - \( k \)-way alignment (alignment of \( k \) sequences) \( \rightarrow \) Runtime?
  - Greedy algorithm \( \rightarrow \) 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

<table>
<thead>
<tr>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V2</td>
<td>.17</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V3</td>
<td>.87</td>
<td>.28</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V4</td>
<td>.59</td>
<td>.33</td>
<td>.62</td>
<td>-</td>
</tr>
</tbody>
</table>

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></th>
<th>FOS_RAT</th>
<th>FOS_MOUSE</th>
<th>FOS_CHICK</th>
<th>FOSB_MOUSE</th>
<th>FOSB_HUMAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PEEMSVTSLDLTGLPEATTPESEAEFTLPLINDPEPK-PSLEPVKNISNELKAEFPFD</td>
<td>PEEMSVAASDLTGLPEASTPESEAEFTLPLINDPEPK-PSLEPVKSNVELKAEFPFD</td>
<td>SEELAAATALDLG-APSFAQEAAFALMLTEAPPAYFPKEPSG-SELELKAEFPFD</td>
<td>PGPGPLAERVQDLPG-STSNAEKFQGMLLPPPPP--------------LPFQ</td>
<td>PGPGPLAERVQDLPG-SAPAKEDFQMLLLPPPPP--------------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:
    \[
    S(a_1, a_2, \ldots, a_k) = \sum_{i,j} S^*(a_i, a_j)
    \]

- 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
  - AAA
  - AAG
  - AAT
  - ATC
  - 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
    - $H(A) = 0$
    - $H(A) = H(T) = \frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} = 0.244$
    - $H(T) = H(C) = -\frac{1}{4} \log_2 \frac{1}{4} \times 4 = 0.602$
  - 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

A. Conventional pairwise alignment

- PKMIVRPQKNETV
- TKMLVR

B. A path of a sequence

- PK-M-I-V-R-P-Q-K-N-E-T-V

C. Combining two paths


D. 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

BAYLOR
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