Multiple Sequence Alignment
Sum-of-Pairs and Clustal-W

Ulf Leser
This Lecture

- **Multiple Sequence Alignment**
  - The problem
  - Theoretical approach: Sum-of-Pairs scores
  - Practical approach: Clustal-W
Multiple Sequence Alignment

• We now align multiple (k>2) sequences
  – Note: Also BLAST aligns only two sequences

• Why?
  – Imagine k sequences of the promoter region of genes, all regulated by the same transcription factor f. Which subsequence within the k sequences is recognized by f?
  – Imagine k sequences of proteins that bind to DNA. Which subsequence of the k sequences code for the part of the proteins that performs the binding?

• General
  – We want to know the common part(s) in k sequences
  – “common” does not mean identical
  – These parts can be anywhere within the sequences
Definition

- A multiple sequence alignment (MSA) of \( k \) Strings \( s_i, 1 \leq i \leq k \), is a table of \( k \) rows and \( l \) columns (\( \text{sum}(|s_i|) \geq l \geq \text{max}(|s_i|) \)) such that
  - Row \( i \) contains the sequence of \( s_i \) with an arbitrary number of blanks being inserted at arbitrary positions
  - Every symbol of every \( s_i \) stands in exactly one column
  - No column contains only blanks

\[
\begin{array}{cccccccccccc}
\text{AACGTGATTGAC} & \text{TCGAGTGT} & \text{CCTTTACAGT} & \text{GCCGTGT}\text{ACTAGT} & \text{TTCAGTGACGTGGTA} & \text{GGTGCAGACC} \\
\text{TTCAGTGGACGTGGTA} & \text{GGTGCAGACC} & \text{AACGTGATTGAC} & \text{TCGAGTGT} & \text{CCTTTACAGT} & \text{GCCGTGT}\text{ACTAGT} & \text{TTCAGTGACGTGGTA} & \text{GGTGCAGACC}
\end{array}
\]
Good MSA

- We are searching for **good (optimal) MSAs**
- Defining "optimal" here is not as simple as in the k=2 case
- Intuition
  - All sequences had a common ancestor and evolved by evolution
  - We want to assume as few evolutionary events as possible
  - Thus, we want few columns (~ few INSDELs)
  - Thus, we want homogeneous columns (~ few replacements)
This Lecture

- Multiple Sequence Alignment
  - The problem
  - Theoretical approach: Sum-of-Pairs scores
  - Practical approach: Clustal-W
What Should we Count?

• For two sequences
  – We scored each column using a scoring matrix
  – Find the alignment such that the total score is maximal

• But – how do we score a column with $5*T, 3*A, 1*_?$
  – We would need an exponentially large scoring matrix

• Alternative: Sum-of-Pairs Score
  – We only score the alignment of each pair of sequences
  – We aggregate over all pairs to score the MSA
  – We need a clever algorithm to find the MSA with the best score
Formally

• Definition
  – Let \( M \) be a MSA for the set \( S \) of sequences \( S = \{s_1, ..., s_k\} \)
  – The alignment of \( s_i \) with \( s_j \) induced by \( M \) is generated as follows
    • Remove from \( M \) all rows except \( i \) and \( j \)
    • Remove all columns that contain only blanks
  – The sum-of-pairs score (sop) of \( M \) is the sum of all pair-wise induced alignment scores
  – The optimal MSA for \( S \) wrt. sop is the MSA with the lowest sop-score over all possible MSAs for \( S \)
Example

- Given a MSA over \( k \) sequences of length \( l \) – how complex is it to compute its sop-score?
- How do we find the best MSA?
Analogy

- Think of the \( k=2 \) case
- Every alignment is a path through the matrix
- The three possible directions (down, right, down-right) conform to the three possible constellations in a column (XX, X_, _X)
- With growing paths, we align growing prefixes of both sequences
Analogy

- Assume $k=3$
- Think of a 3-dimensional cube with the three sequences giving the values in each dimension
- Now, we have paths aligning growing prefixes of three sequences
- Every column has seven possible constellations ($XXX, XX_, X_X, _XX, X__, _X_, __X$)
All Possible Steps

- \( d(i-1,j-1,k-1) \)
- \( d(i,j-1,k-1) \)
- \( d(i,j,k-1) \)
- \( d(i,j,k) \)
- \( d(i-1,j,k) \)
- \( d(i-1,j-1,k) \)
- \( d(i-1,j,k-1) \)
- \( d(i-1,j,k-1) \)
Concrete Examples

- Best sop-score for $d(i,j-1,k)$ is known
- We want to compute $d(i,j,k)$
- This requires to align one symbol with two blanks (blank/blank does not count)
- $d(i,j,k) = d(i,j-1,k) + 2$

- Best sop-score for $d(i-1, j,k-1)$ is known
- We want to compute $d(i,j,k)$
- This requires aligning a blank with $s_1[i-1]$ and with $s_3[k-1]$ and to align $s_1[i-1]$ and $s_3[k-1]$
- $d(i,j,k) = d(i-1,j,k-1) + 2 + c_{ik}$
Dynamic Programming in three Dimensions

- We compute the best possible alignment $d(i,j,k)$ for every triple of prefixes (lengths $i,j,k$) using the following formula
  - We assume the usual edit costs: $I/D/R=+1$, $M=0$

\[
d(i,j,k) = \min \begin{cases} 
  d(i-1,j-1,k-1) + c_{ij} + c_{ik} + c_{jk} \\
  d(i-1,j-1,k) + c_{ij} + 2 \\
  d(i-1,j,k-1) + c_{ik} + 2 \\
  d(i,j-1,k-1) + c_{jk} + 2 \\
  d(i-1,j,k) + 2 \\
  d(i,j-1,k) + 2 \\
  d(i,j,k-1) + 2 
\end{cases}
\]

Three (mis)matches
One (mis)match, two ins

Let $c_{ij} = 0$, if $S_1(i) = S_2(j)$, else 1
Let $c_{ik} = 0$, if $S_1(i) = S_3(k)$, else 1
Let $c_{jk} = 0$, if $S_2(j) = S_3(k)$, else 1
Initialization

- Of course, we have $d(0,0,0)=0$
- Aligning in one dimension: $d(i,0,0)=2*i$
  - Same for $d(0,j,0), d(0,0,k)$
- Aligning in two dimensions: $d(i,j,0)= ...$
  - Let $d_{a,b}(i,j)$ be the alignment score for $S_a[1..i]$ with $S_b[1..j]$
  - $d(i, j, 0) = d_{1,2}(i, j) + (i+j)$
  - $d(i, 0, k) = d_{1,3}(i, k) + (i+k)$
  - $d(0, j, k) = d_{2,3}(j, k) + (j+k)$
Algorithm

initialize matrix d;
for i := 1 to |S_1|
    for j := 1 to |S_2|
        for k := 1 to |S_3|
            if (S_1(i) = S_2(j)) then c_{ij} := 0; else c_{ij} := 1;
            if (S_1(i) = S_3(k)) then c_{ik} := 0; else c_{ik} := 1;
            if (S_2(j) = S_3(k)) then c_{jk} := 0; else c_{jk} := 1;
            d_1 := d[i – 1,j – 1,k - 1] + c_{ij} + c_{ik} + c_{jk};
            d_2 := d[i – 1,j – 1,k] + c_{ij} + 2;
            d_3 := d[i – 1,j,k - 1] + c_{ik} + 2;
            d_4 := d[i,j – 1,k - 1] + c_{jk} + 2;
            d_5 := d[i – 1,j,k) + 2;
            d_6 := d[i,j – 1,k) + 2;
            d_7 := d[i,j,k - 1) + 2;
            d[i,j,k] := min(d_1, d_2, d_3, d_4, d_5, d_6, d_7);
        end for;
    end for;
end for;
Bad News: Complexity

• For 3 sequences of length $n$
  – There are $n^3$ cells in the cube
  – For each cell (top-left-front corner), we need to look at 7 corners
  – Together: $O(7 \times n^3)$ operations

• For $k$ sequences of length $n$
  – There are $n^k$ cell corners in the cube
  – For each corner, we need to look at $2^k - 1$ other corners
  – Together: $O(2^k \times n^k)$ operations
Bad News: Biological Meaningfulness

- Recall our motivation: Measure evolution

- Real number of events: 8
- sop-score: $2 + 3 + 6 + 6 + 2 + ...$
  - Single mutations are counted multiple times

```
GTTTCA
GTTGCA
GTTTTCA
GTATTTCA
CTTGCA
GTTGACA
GTTGTTA
GTATTTCT
GTATTTGA
CT_TGC_A
GT_TGACA
GT_TGTGA
GTATTTCT
GTATTTGA
```
This Lecture

- Multiple Sequence Alignment
  - The problem
  - Theoretical approach: Sum-of-Pairs scores
  - Practical approach: Clustal-W
Different Scoring Function

• If we knew the phylogenetic tree of the k sequences
  – Align every parent node pairwise with its children
  – Aggregate all alignment scores
  – This gives the “real” number of evolutionary operations
    • But not yet the best MSA
• But: Finding the true phylogenetic tree requires a MSA
  – Not covered in this lecture
• Use a heuristic: ClustalW
Clustal-W

- **Main idea**
  - Compute a “good enough” phylogeny – the guide tree
  - Use the guide tree to iteratively align small MSA to larger MSA
    - “Progressive” MSA
    - Starting from single sequences
    - Add more and more sequences and smaller MSA to ever bigger MSA
    - Needs a fast method to align two MSAs

- **Standard method for a long time**
- **Many newer (better) proposals**
  - DAlign, T-Coffee, HMMT, PRRT, MULTALIGN, ...
Step 1: Compute the Guide Tree

- Compute all pair-wise alignments and store in matrix M
  - \( M[i,j] = \text{sim}(s_i, s_j) \)

- Compute the guide tree using \textit{hierarchical clustering}
  - Choose the smallest \( M[i,j] \)
  - Let \( s_i \) and \( s_j \) form a \textit{new (next) branch} of the tree
  - Compute the distance from the ancestor of \( s_i \) and \( s_j \) to all other sequences as the average of the distances to \( s_i \) and \( s_j \)
    - Set \( M' = M \)
    - Delete rows and columns \( i \) and \( j \)
    - Add a new column and row \((ij)\)
    - For all \( k \neq ij: M'[ij,k] = (M[i,k] + M[j,k]) / 2 \)
  - \textit{Iterate until} \( M' \) has only one column / row
Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>17</td>
<td>59</td>
<td>59</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>37</td>
<td>61</td>
<td>53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>13</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>E</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>17</td>
<td>77</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>53</td>
<td>49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>E</th>
<th>CD</th>
<th>AB</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>31</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td></td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>
Example

Once a gap, always a gap
Step 2: Progressive MSA

- **Pair-wise alignment of MSAs** in the order of the guide tree
- **Aligning a MSA** $M_1$ **with a MSA** $M_2$
  - Use the usual (global) alignment algorithm
  - To score a column, compute the **average score over all pairs of symbols** in these columns

**Example**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>...</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>P...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>G...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>P...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>A...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>A...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>Y...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Score of this column:

\[
\frac{2 \cdot s(P,A) + s(P,Y) + 2 \cdot s(G,A) + s(G,Y) + 2 \cdot s(P,A) + s(P,Y)}{9}
\]
Issues

• There is a lot to say about whether hierarchical clustering actually computes the “correct” tree
• Clustal-W actually uses a different, more accurate phylogenetic algorithm called “neighbor-joining”
• Clustal-W is fast: \(O(k^2*n^2+k^2*\log(k))\)
  – For k sequences; plus cost for computing pairwise alignments
• Idea behind **progressive alignment**
  – Find strong signals (highly conserved blocks) first
  – Outliers are added last
  – Increases the chances that conserved blocks survive
  – Several improvements to this scheme are known
Problems of Progressive MSA

1: MAYFIELD THE LAST FAT RER
2: MAYFIELD THE FAST RAT
3: MAYLEENE IS A FAT RAT
4: MAYROONI THE LAST BIG RAT

Angelehnt: Cedric Notredame, 2001

Besser:

1: MAYFIELD THE LAST FAT RER
2: MAYFIELD THE FAST RAT ___
3: MAYLEENE IS _ _ A _ _ FAT RAT
4: MAYROONI THE LAST BIG RAT
Further Reading

- Merkl & Waack, chapter 13
- Böckenhauer & Bongartz, chapter 5.3