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
  - This part can be anywhere within the sequences
Definition

• 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 ($l \geq \max(|s_i|)$), 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

\[
\begin{array}{c}
\text{AACGTGATTGAC} \\
\text{TCGAGTGGCTTTACAGT} \\
\text{GCCGTGCTAGTTCG} \\
\text{TTCAGTGGACGTGGTA} \\
\text{GGTGCAGACC}
\end{array}
\quad
\begin{array}{c}
\text{AACGTGATTGA} \\
\text{___TCGAGT__GCTTTACAGT_} \\
\text{GCCGTGCTAGT_____C_G_} \\
\text{TTCAGTGACGTGGTA_G__GTGCA_GAC___C____}
\end{array}
\]
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 score an entire MSA
  - We score the alignment of each pair of sequences in the usual way
  - 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 $k$ sequences $S = \{s_1, \ldots, s_k\}$
  – 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. to sop is the MSA with the lowest sop-score over all possible MSA 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

- $d(i-1,j-1,k-1)$
- $d(i,j-1,k-1)$
- $d(i,j,k-1)$
- $d(i,j-1,k)$
- $d(i-1,j,k)$
- $d(i-1,j-1,k)$
- $d(i-1,j,k-1)$
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

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

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
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 $s_3[k-1]$
- $d(i, j, k) = d(i-1, j, k-1) + 2 + c_{ik}$
Initialization

- Of course, we have $d(0,0,0)=0$
- Aligning in one dimension: $d(i,0,0)=2i$
  - 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

<table>
<thead>
<tr>
<th>initialize matrix d;</th>
</tr>
</thead>
<tbody>
<tr>
<td>for i := 1 to</td>
</tr>
<tr>
<td>for j := 1 to</td>
</tr>
<tr>
<td>for k := 1 to</td>
</tr>
<tr>
<td>if (S_1(i) = S_2(j)) then c_{ij} := 0; else c_{ij} := 1;</td>
</tr>
<tr>
<td>if (S_1(i) = S_3(k)) then c_{ik} := 0; else c_{ik} := 1;</td>
</tr>
<tr>
<td>if (S_2(j) = S_3(k)) then c_{jk} := 0; else c_{jk} := 1;</td>
</tr>
<tr>
<td>d_1 := d[i - 1, j - 1, k - 1] + c_{ij} + c_{ik} + c_{jk};</td>
</tr>
<tr>
<td>d_2 := d[i - 1, j - 1, k] + c_{ij} + 2;</td>
</tr>
<tr>
<td>d_3 := d[i - 1, j, k - 1] + c_{ik} + 2;</td>
</tr>
<tr>
<td>d_4 := d[i, j - 1, k - 1] + c_{jk} + 2;</td>
</tr>
<tr>
<td>d_5 := d[i - 1, j, k) + 2;</td>
</tr>
<tr>
<td>d_6 := d[i, j - 1, k) + 2;</td>
</tr>
<tr>
<td>d_7 := d[i, j, k - 1) + 2;</td>
</tr>
<tr>
<td>d[i, j, k] := min(d_1, d_2, d_3, d_4, d_5, d_6, d_7);</td>
</tr>
<tr>
<td>end for;</td>
</tr>
<tr>
<td>end for;</td>
</tr>
<tr>
<td>end for;</td>
</tr>
</tbody>
</table>
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: $O(7*n^3)$ computations

- **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: $O(2^k * n^k)$ computations
Bad News: Biological Meaningfulness

- Let’s take one step back
- What happened during evolution?

- Real number of events: 8
- sop-score: 2+3+6+6+2+...
  - Everything is counted multiple times

CT_TGC_A
GT_TGACAGT_TGGTTAGTATTTCTGTATTTTGAGTATTTTCA
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 with all its children
  – Aggregate all alignment scores
  – This gives the “real” number of evolutionary operations
• 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
    • Starting from single sequences
  – To escape the curse of dimensionality, compute MSA iteratively
    • “Progressive” MSA
    • Does not necessarily find the best solution
    • Needs a fast method to align two MSAs

• Works quite well in practice

• Many other, newer (better) proposals
  – DAlign, T-Coffee, HMMT, PRRT, MULTALIGN, ...
Step 1: Compute the Guide Tree

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

- Compute the guide tree using hierarchical clustering
  - Choose the smallest $M[i,j]$  
  - Let $s_i$ and $s_j$ form a 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 $k \neq ij$: $M'[ij,k] = (M[i,k] + M[j,k]) / 2$
    - Iterate until $M'$ has only one column / row
Sketch

\[
\begin{align*}
\text{ABCDEFG} & \quad \text{(B, D)} \rightarrow a \\
\text{ACEFGa} & \quad \text{(E, F)} \rightarrow b \\
\text{ACGab} & \quad \text{(A, b)} \rightarrow c \\
\text{CGac} & \quad \text{(C, G)} \rightarrow d \\
\text{acd} & \quad \text{(d, c)} \rightarrow e \\
\text{ae} & \quad \text{(a, e)} \rightarrow f
\end{align*}
\]
Example

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>B</td>
<td>37</td>
<td>61</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>13</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>B</td>
<td>53</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

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

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

- **Pair-wise alignment** in the order given by 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 this column

- **Example**

```
A   ...P...
B   ...G...
C   ...P...   Score of this column
D   ...A...
E   ...A...
F   ...Y...
```

$\text{Score of this column} = \frac{(2 \times s(P,A) + s(P,Y) + 2 \times s(G,A) + s(G,Y) + 2 \times s(P,A) + s(P,Y))}{9}$
Issues

- There is a lot to say about whether hierarchical clustering actually computes the “correct” tree
- Clustal-W actually uses a different, more accurate algorithm called “neighbor-joining”
- Clustal-W is fast \(O(k^2n^2+k^2\log(k))\)
  - For \(k\) sequences; plus cost for computing alignments, especially \(M\)
- 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 with progressive MSA

Source: Cedric Notredame, 2001
Further Reading

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