Gotoh Scan Algorithm
for matching RNA sequences

By Hila Abukasis
& Shai Kerer
Contents

- What is RNA?
- Matching RNA
- Needleman-Wunsch Algorithm
- Global Alignment VS Local Alignment
- Smith-Waterman Algorithm
- Gotoh Scan Algorithm
  - Ideal Gap Penalty
  - Algorithm
- Summary
What is RNA?

A “copy” of a sub-sequence of the DNA. Carry information from DNA to the Ribosome – where it is translated to proteins.
Matching RNA
Motivation

It is believed that RNA is the most ancient genetic material.

Finding similarity between 2 RNA sequences can teach us about evolutionary relations.

Accurate RNA sequence alignment is an essential tool needed to understand basic biological and evolutionary processes.
Matching RNA

Given 2 RNA sequences (strings), we want to find the optimal alignment between them.

C A G C U G
% % $ $ $ $ %

G A C A A U A G U C

A A A A A C A U A C A A C A A C A G C
% % $ $ % % % ~ % $ % % % $ % %

C A A A G C A C A _ A U A A C U G C C C
The Needleman-Wunsch algorithm is an application of a best-path strategy (dynamic programming) used to find optimal sequence alignment.

Any partial sub-path that tends at a point along the true optimal path must itself be the optimal path leading up to that point.

Therefore the optimal path can be determined by incremental extension of the optimal sub-paths.

In a Needleman-Wunsch alignment, the optimal path must stretch from beginning to end in both sequences (hence the term ‘global alignment’).
Needleman-Wunsch Algorithm

- Given 2 RNA strings – A, B. We build a matrix as followed –
- Each alignment gets a score, which indicates of the compatibility of the 2 strings
- Where $S(A_i, B_j)$ is the score for matching single a char from the 2 strings
- Gap is a Penalty for entering a gap in the string.

\[ M(i,j) = \max \{ M(i-1,j-1) + S(A_i, B_j), \]
\[ M(i-1,j) + \text{gap}, \]
\[ M(i,j-1) + \text{gap} \} \]
Gap = -2 ; Mismatch = -3 ; Match = 2

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
<td>-14</td>
<td>-16</td>
<td>-18</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A_..</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>_A..</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>_A..</td>
<td>-3</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>_A..</td>
<td>-12</td>
<td>-6</td>
<td>-4</td>
<td>-4</td>
<td>-4</td>
<td>-4</td>
<td>-4</td>
<td>-4</td>
<td>-4</td>
</tr>
<tr>
<td>_A..</td>
<td>-10</td>
<td>-12</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Max { M[1, 0] + S(C, A) ; (−2 + −3) 
     M[1, 1] + Gap ; (2 + −2) 
     M[2, 0] + Gap } (−4 + −2)
- Trace Back - The optimal path is traced beginning from the lower right-hand corner
- Each step we go to the highest neighbor
- Horizontal and vertical movement is a gap, and diagonal movement is a match

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>C</td>
<td>-4</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
</tr>
<tr>
<td>G</td>
<td>-6</td>
<td>-2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td>-4</td>
<td>0</td>
<td>-1</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>-10</td>
<td>-6</td>
<td>-2</td>
<td>-3</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>T</td>
<td>-12</td>
<td>-8</td>
<td>-4</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>-14</td>
<td>-10</td>
<td>-6</td>
<td>-2</td>
<td>-4</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>A</td>
<td>-16</td>
<td>-12</td>
<td>-8</td>
<td>-4</td>
<td>-5</td>
<td>-2</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
Needleman-Wunsch - Result

A C T G _ A T T C A

|   |   |   |   |

A C _ G C A T _ C A

Score =

(AA) + (CC) + (T-) + (GG) + (-C) + (AA) + (TT) + (T-) + (CC) + (AA)

= 2 + 2 - 2 + 2 - 2 + 2 + 2 - 2 + 2 + 2

= 8
Global Alignment VS Local Alignment
Smith-Waterman Algorithm (1981)

- Modification of Needleman-Wunsch.
- Used to find Local Alignment

\[
M(i,j) = \text{MAX}\{M_{i-1,j-1} + S(A_i, B_j), M_{i-1,j} + \text{gap}, M_{i,j-1} + \text{gap}, 0\}
\]
Smith-Waterman - Example

Gap = -1 ; Mismatch = -3 ; Match = 2

\[ M(i,j) = \max\{ M(i-1,j-1) + S(A_i, B_j), M(i-1,j) + \text{gap}, M(i,j-1) + \text{gap}, 0 \} \]

\[ M(i,j) = \max\{ (0+2), (0+(-1)), (0+(-1)), (0) \} \]
# Smith-Waterman - TraceBack

Gap = -1 ; Mismatch = -3 ; Match = 2

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

A_CA
AGCA

Score=

\[(AA)+(_G)+(CC)+(AA) = 2+(-1)+2+2=5\]
Run Time Complexity

- Needleman-Wunsch: $O(nm)$
- Smith-Waterman: $O(nm)$

Space Complexity

- Needleman-Wunsch: $O(nm)$
- Smith-Waterman: $O(nm)$
Space Improvement?

- Instead of $n \times m$ array, we can use 2 linear arrays of size $n$.

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$M(i,j) = \text{MAX} \{\begin{array}{l}
  M_{i-1,j-1} + S(A_i, B_j) \\
  M_{i-1,j} + \text{gap} \\
  M_{i,j-1} + \text{gap} \\
  0 \end{array}\}$

Problem?
Ideal Gap Penalty

- With the two previous algorithms the strategy is to add a fixed gap penalty when a gap occurs regardless what the alignment was.
- It is likely that if a particular character is gapped, the probability of the next one being gapped is higher, and hence should be penalized less.
Ideal Gap Penalty
Run Time Complexity

\[ M_{i,j} = \max \{ M_{i-1,j-1} + S(A_i, B_j), \]
\[ \max_k (M_{i-k,j} + \text{GAP}(k)), \]
\[ \max_k (M_{i,j-k} + \text{GAP}(k)), \]
\[ 0 \} \]

The algorithm using ideal gap penalty costs \( O(n^2 \times m) \) (assuming \( n > m \))

When evaluating the gap penalty we need to loop through all previous nucleotides to find the one that gives the maximum score.
Affine Gap Penalty

- The algorithm using ideal gap penalty costs $O(n^2 \times m)$, which is too expensive.
- In order to keep our $O(n \times m)$, we’ll use a “compromise”:
  - Gap Opening – Expensive
  - Gap Extension - Cheap
Ideal / Affine Gap – how to?

\[ M(i,j) = \text{MAX}\{ M_{i-1,j-1} + S(A_i, B_j), M_{i-1,j} + \text{gap}, M_{i,j-1} + \text{gap} \} \]
Affine Gap Penalty

**Solution - Affine Gap**

\[ D_{i,j} = \max_{1 \leq k \leq i} \left( M_{i-k,j} + \text{GAP}(k) \right) \]

\[ = \max \left\{ M_{i-1,j} + \text{GAP}(k) \right\} \]

\[ = \max \left\{ M_{i-1,j} + \text{GAP}(1) ; D_{i-1,j} + u \right\} \]

\[ \text{GAP}(k) = v + uk \]

**Problem**

\[ M_{i,j} = \max \{ M_{i-1,j-1} + S(A_i, B_j) \} \]

\[ \max_k (M_{i-k,j} + \text{GAP}(k)) \]

\[ \max_{k \leq i} (M_{i-k,j} + \text{GAP}(k)) \]

\[ = \max \left\{ M_{i-1,j-1} + S(A_i, B_j) \right\} \]

**Ideal Gap**

\[ \text{Ideal Gap} \]

Run Time Complexity
Semi Global Alignment

To Explain what is Semi-Global Alignment, we will use meaningful names for the 2 RNA sequences, instead A, B

- **Query** – The 1st String – A
- **DataBase** – The 2nd String – B
- **Semi-Global** – Match the **whole** query to **sub-sequence** of the **dataBase**.
Semi Global — How to?

Hint: Table initialization
Gotoh Scan
Semi-Global + Affine Gap
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>G</th>
<th>U</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>G</td>
<td>-2</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>A</td>
<td>-3</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>A</td>
<td>-4</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>U</td>
<td>-5</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>C</td>
<td>-6</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>A</td>
<td>-7</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>G</th>
<th>U</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>A</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>A</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>A</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>A</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
</tbody>
</table>

Gap Open = -2  Gap Extension = -1  
Match = 2  Mismatch = -3

\[
D_{ij} = \text{Max} \{ S_{i-1,j} + \text{go}, D_{i-1,j} + \text{ge} \}
\]

\[
F_{ij} = \text{Max} \{ S_{i,j-1} + \text{go}, F_{i,j-1} + \text{ge} \}
\]

\[
S_{ij} = \text{Max} \{ S_{i-1,j-1} + \text{score}(A_i,B_j), \ D_{i,j} , F_{i,j} \}
\]
Gap Open = -2  Gap Extension = -1
Match = 2  Mismatch = -3

\[ D[i][j] = \text{Max} \{ S_{i-1,j} + \text{go}, D_{i-1,j} + \text{ge} \} \]
\[ F[i][j] = \text{Max} \{ S_{i,j-1} + \text{go}, F_{i,j-1} + \text{ge} \} \]
\[ S[i][j] = \text{Max} \{ S_{i-1,j-1} + \text{score}(A_i, B_j), \]
\[ D_{i,j}, F_{i,j} \} \]
## Dynamic Programming for Sequence Alignment

### Parameters
- **Gap Open**: \(-2\)
- **Gap Extension**: \(-1\)
- **Match**: 2
- **Mismatch**: \(-3\)

### Score Matrices

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>U</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>G</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>A</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>A</td>
<td>-4</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
</tr>
<tr>
<td>U</td>
<td>-5</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>-4</td>
</tr>
<tr>
<td>C</td>
<td>-6</td>
<td>-4</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>-7</td>
<td>-5</td>
<td>-4</td>
<td>-4</td>
<td>-2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>U</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
</tr>
<tr>
<td>U</td>
<td>-5</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>-6</td>
<td>-4</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>-7</td>
<td>-4</td>
<td>-4</td>
<td>-4</td>
<td>-2</td>
</tr>
</tbody>
</table>

### formulas
- **D[i][j]** = \(\max\{S_{i-1,j} + \text{go}, D_{i-1,j} + \text{ge}\}\)
- **F[i][j]** = \(\max\{S_{i,j-1} + \text{go}, F_{i,j-1} + \text{ge}\}\)
- **S[i][j]** = \(\max\{S_{i-1,j-1} + \text{score}(A_i, B_j), D_{i-1,j-1}, F_{i,j-1}\}\)
The image contains a table representing a dynamic programming solution for sequence alignment, specifically an example of the Smith-Waterman algorithm for the sequences GAAUCAG and AGGUCA. The table shows the scores at each position, with options to choose between gap opening (go) and gap extension (ge). The final scores are calculated as follows:

- Match = 2
- Mismatch = -3
- Gap Open = -2
- Gap Extension = -1

The scores are calculated as follows:

\[
\text{D}[i][j] = \text{Max} \{S_{i-1,j} + \text{go}, D_{i-1,j} + \text{ge}\}
\]

\[
\text{F}[i][j] = \text{Max} \{S_{i,j-1} + \text{go}, F_{i,j-1} + \text{ge}\}
\]

\[
\text{S}[i][j] = \text{Max} \{S_{i-1,j-1} + \text{score}(A_i, B_j), D_{i,j}, F_{i,j}\}
\]

The score for the alignment is calculated as:

\[
\text{Score} = (GG) + (A_) + (AG) + (UU) + (CC) + (AA)
\]

\[
= 2 - 2 - 3 + 2 + 2 + 2 = 3 \neq 5
\]

The final score of 3 is less than the expected score of 5, indicating a mismatch in the alignment.
Score = (GG) + (A_) + (A_) + (UU) + (CC) + (AA) = 2 - 2 - 1 + 2 + 2 + 2 = 5 😊
Here Is A Thought

- Can we make it even more accurate regarding the Gap Penalty with the limits of $O(nm)$?
Summary

Thanks to Algorithms such as Gotoh Scan, we have more proof about our origin.