# **CSI5126**. Algorithms in bioinformatics

## Pairwise Sequence **Alignment**

Marcel Turcotte

uOttawa

School of Electrical Engineering and Computer Science (EECS)
University of Ottawa

Version October 2, 2018

# Summary

We now exploring important **adaptations** of the pairwise sequence alignment problem to make it relevant to real-world **biology** problems.

**General objective**

- Select the appropriate pairwise alignment algorithm for a given problem.

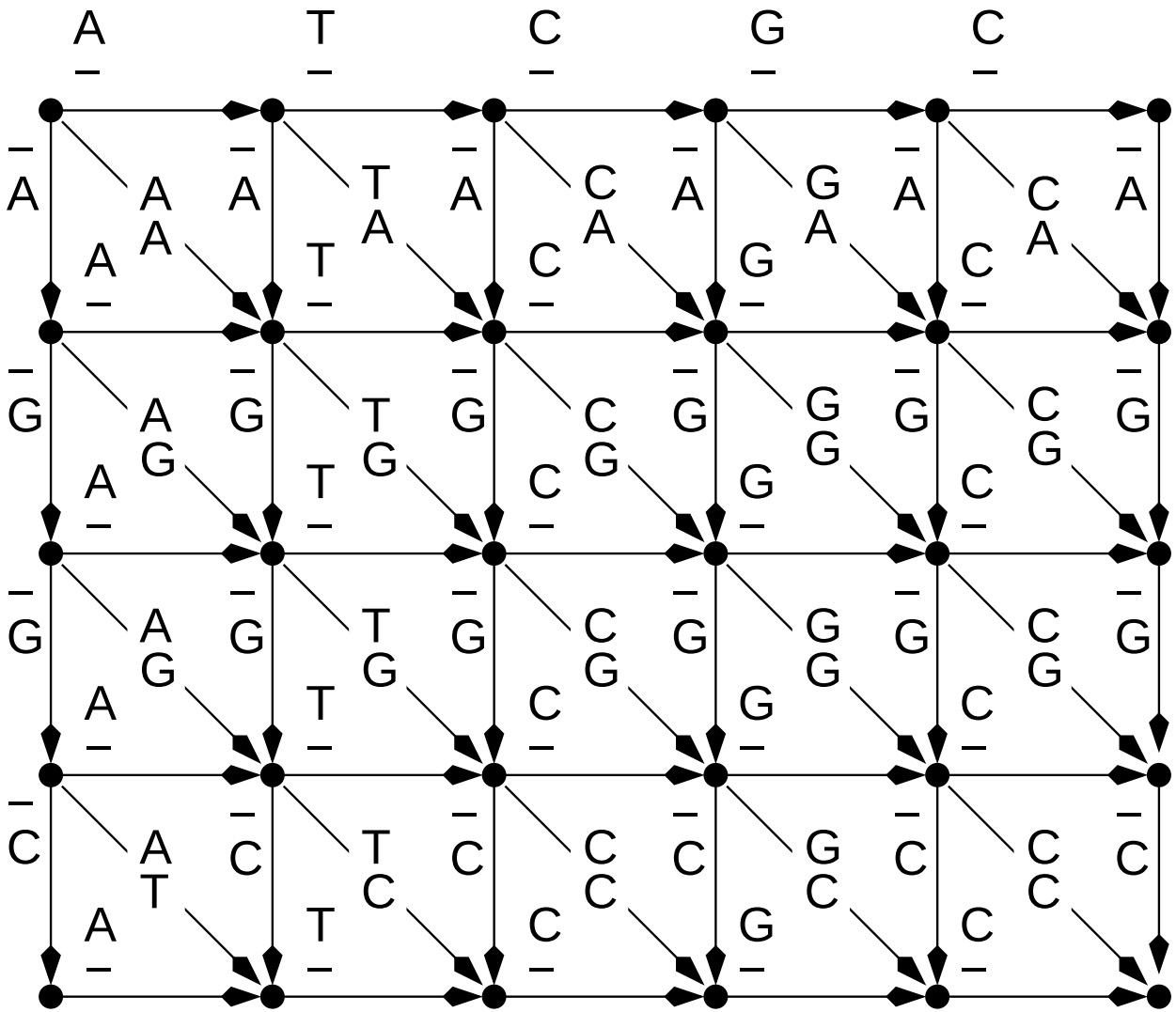
**Reading**

- Bernhard Haubold and Thomas Wiehe (2006). *Introduction to computational biology: an evolutionary approach.* Birkhäuser Basel. Pages 11-15, 30-33.

# Reading

- Bernhard Haubold and Thomas Wiehe (2006). *Introduction to computational biology: an evolutionary approach.* Birkhäuser Basel. Pages 11-15, 30-33.
- Wing-Kin Sung (2010) Algorithms in Bioinformatics: A Practical Introduction. Chapman & Hall/CRC. QH 324.2 .S86 2010 Chapter 2.
- Dan Gusfield (1997) *Algorithms on strings, trees, and sequences : computer science and computational biology*. Cambridge University Press. Chapters 10 and 11.
- Pavel A. Pevzner and Phillip Compeau (2018) *Bioinformatics Algorithms: An Active Learning Approach*. Active Learning Publishers. `http://bioinformaticsalgorithms.com` Chapter 5.

# Edit Graph


A T C G C
A A T A C A G A C A
A A T A C A G A C A
G A G T G C G G C G
A A G T G C G G C G
G A G T G C G G C G
A A G T G C G G C G
C A T T C C G C C C
A A T C C C G C C C

Edit Distance

min =

| | - | c | o | m | p | l | i | m | e | n | t | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| c | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| o | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| m | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| p | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| e | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| t | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| e | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| n | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |
| t | [ ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ][ | ] |

Edit Distance

min = 4

| | | - | c | o | m | p | l | i | m | e | n | t | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| | c | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | o | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | m | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | p | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | e | 5 | 4 | 3 | 2 | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 6 |
| | t | 6 | 5 | 4 | 3 | 2 | 2 | 2 | 3 | 4 | 4 | 4 | 5 |
| | e | 7 | 6 | 5 | 4 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 5 |
| | n | 8 | 7 | 6 | 5 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 5 |
| | t | 9 | 8 | 7 | 6 | 5 | 5 | 5 | 5 | 5 | 4 | 3 | 4 |

Edit Distance

min = 4

| | | - | c | o | m | p | l | i | m | e | n | t | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] | [ 8] | [ 9] | [ 10] | [ 11] |
| c | [ | 1] | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] | [ 8] | [ 9] | [ 10] |
| o | [ | 2] | [ 1] | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] | [ 8] | [ 9] |
| m | [ | 3] | [ 2] | [ 1] | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] | [ 8] |
| p | [ | 4] | [ 3] | [ 2] | [ 1] | { 0} | { 1} | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] |
| e | [ | 5] | [ 4] | [ 3] | [ 2] | [ 1] | [ 1] | { 2} | [ 3] | [ 3] | [ 4] | [ 5] | [ 6] |
| t | [ | 6] | [ 5] | [ 4] | [ 3] | [ 2] | [ 2] | [ 2] | { 3} | [ 4] | [ 4] | [ 4] | [ 5] |
| e | [ | 7] | [ 6] | [ 5] | [ 4] | [ 3] | [ 3] | [ 3] | [ 3] | { 3} | [ 4] | [ 5] | [ 5] |
| n | [ | 8] | [ 7] | [ 6] | [ 5] | [ 4] | [ 4] | [ 4] | [ 4] | [ 4] | { 3} | [ 4] | [ 5] |
| t | [ | 9] | [ 8] | [ 7] | [ 6] | [ 5] | [ 5] | [ 5] | [ 5] | [ 5] | [ 4] | { 3} | { 4} |

MMMMDSSMMMD

compliments

comp-etent-

Edit Distance

min = 4

| | | - | c | o | m | p | l | i | m | e | n | t | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] | [ 8] | [ 9] | [ 10] | [ 11] |
| c | [ | 1] | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] | [ 8] | [ 9] | [ 10] |
| o | [ | 2] | [ 1] | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] | [ 8] | [ 9] |
| m | [ | 3] | [ 2] | [ 1] | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] | [ 8] |
| p | [ | 4] | [ 3] | [ 2] | [ 1] | { 0} | [ 1] | [ 2] | [ 3] | [ 4] | [ 5] | [ 6] | [ 7] |
| e | [ | 5] | [ 4] | [ 3] | [ 2] | [ 1] | { 1} | [ 2] | [ 3] | [ 3] | [ 4] | [ 5] | [ 6] |
| t | [ | 6] | [ 5] | [ 4] | [ 3] | [ 2] | [ 2] | { 2} | { 3} | [ 4] | [ 4] | [ 4] | [ 5] |
| e | [ | 7] | [ 6] | [ 5] | [ 4] | [ 3] | [ 3] | [ 3] | [ 3] | { 3} | [ 4] | [ 5] | [ 5] |
| n | [ | 8] | [ 7] | [ 6] | [ 5] | [ 4] | [ 4] | [ 4] | [ 4] | [ 4] | { 3} | [ 4] | [ 5] |
| t | [ | 9] | [ 8] | [ 7] | [ 6] | [ 5] | [ 5] | [ 5] | [ 5] | [ 5] | [ 4] | { 3} | { 4} |

MMMMSSDMMMD

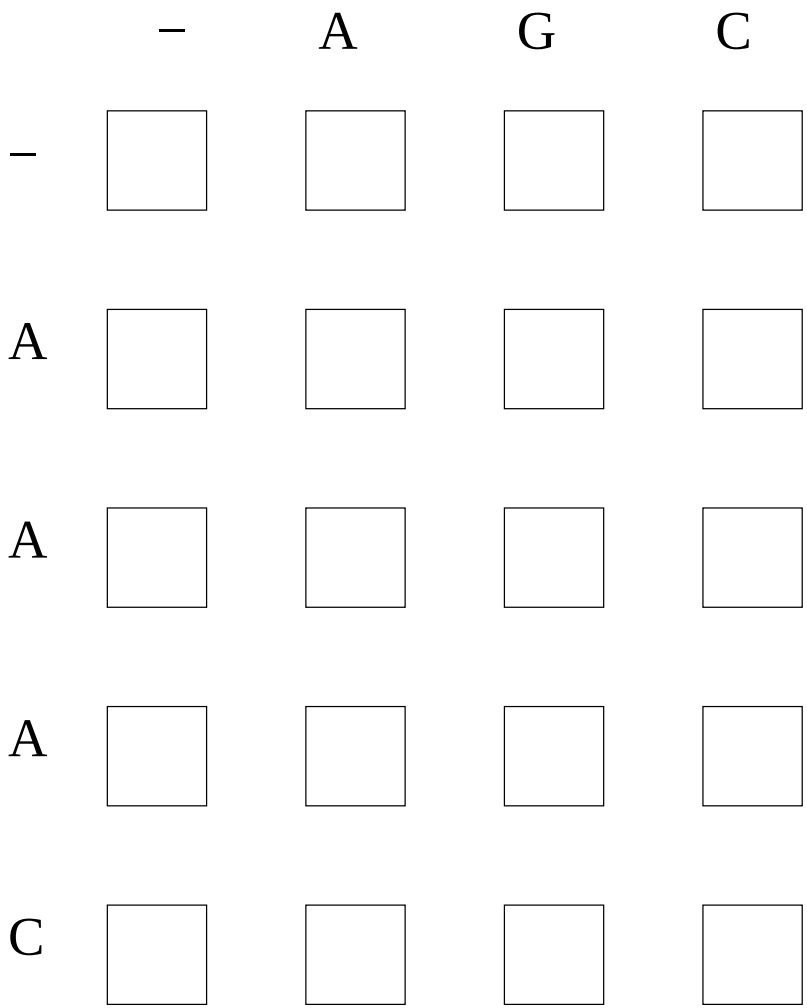
compliments

compet-ent-

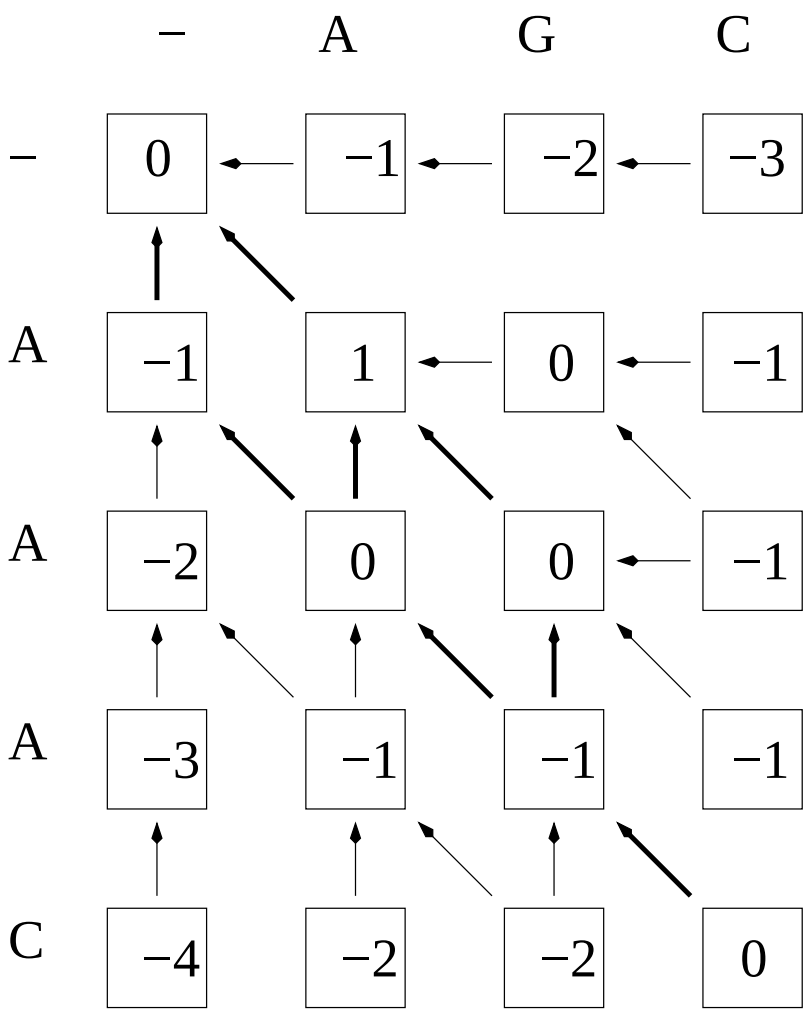
# Remarks

- The calculation of each cell necessitates only **three look-ups** (the algorithm does not reconstruct the partial alignments as we did as we did for the purpose of the example);
- **How many operations** are needed then?
- The order in which we visit the cells during the first pass is not important; as long as the value of the cells $(i-1, j-1), (i-1, j)$ and $(i, j-1)$ are known when calculating the value of the cell $(i, j)$.

# Sequence alignment



# Sequence alignment




⇒ **How many optimal alignments are there?**

# **Weighted** Edit Operations

- A first generalisation of the edit distance problem consists of **associating weights to the edit operations**: for instance, the cost of an insertion/deletion could be 1, the cost of a mismatch could be 2, and the cost of a match 0 (**useful weights will be derived in the next lecture**)

# **Weighted** Edit Operations

- A first generalisation of the edit distance problem consists of **associating weights to the edit operations**: for instance, the cost of an insertion/deletion could be 1, the cost of a mismatch could be 2, and the cost of a match 0 (**useful weights will be derived in the next lecture**)
- The same algorithm can be used only this time it finds the edit transcript/alignment which has the **minimum overall cost**.

# **Weighted** Edit Operations

- A first generalisation of the edit distance problem consists of **associating weights to the edit operations**: for instance, the cost of an insertion/deletion could be 1, the cost of a mismatch could be 2, and the cost of a match 0 (**useful weights will be derived in the next lecture**)
- The same algorithm can be used only this time it finds the edit transcript/alignment which has the **minimum overall cost**.
- The terms **weight** and **cost** are used interchangeably in the C.S. literature whilst **score** is most frequently used in the biological literature

# **Weighted** Edit Operations

- Can the weights be arbitrary?

```
A A A T A A        A A A T - A A
| | | x | |  vs   | | |     | |
A A A C A A        A A A - C A A
```

# **Weighted** Edit Operations

- Can the weights be arbitrary?

```
A A A T A A        A A A T - A A
| | | x | |  vs   | | |     | |
A A A C A A        A A A - C A A
```

- **No.** What is the **relationship** between the cost associated with a **substitution** and the cost associated with an **insertion**?

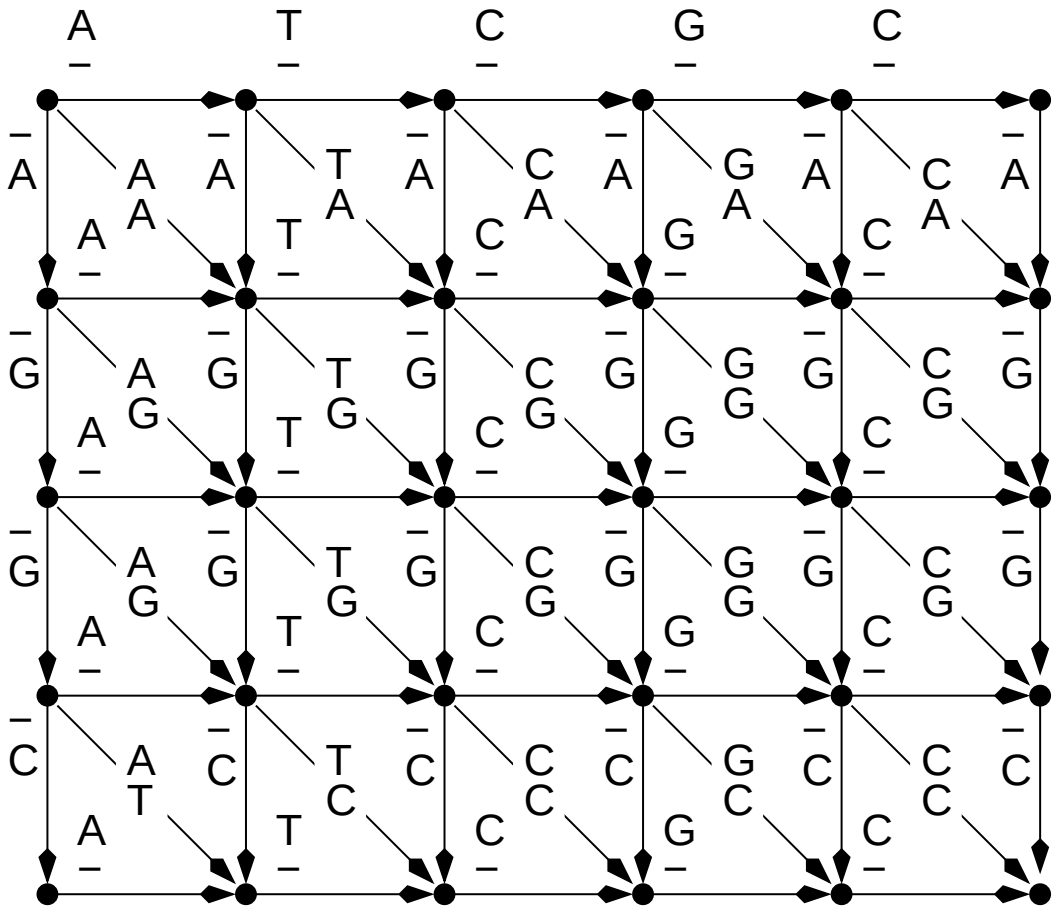
# **Weighted** Edit Operations

- Can the weights be arbitrary?

```
A A A T A A        A A A T - A A
| | | x | |  vs   | | |     | |
A A A C A A        A A A - C A A
```

- **No.** What is the **relationship** between the cost associated with a **substitution** and the cost associated with an **insertion**?
- For a substitution to be selected by the algorithm, its cost should be **less than twice the cost of an insertion**, otherwise the optimisation will favour two insertions, as above depicted.

# What are the necessary changes to our framework?



# Operation-Weighted Edit Distance

**Base conditions**,

$$D(0,0) = 0$$

$$D(i,0) = i \times d, i \in 1..n$$

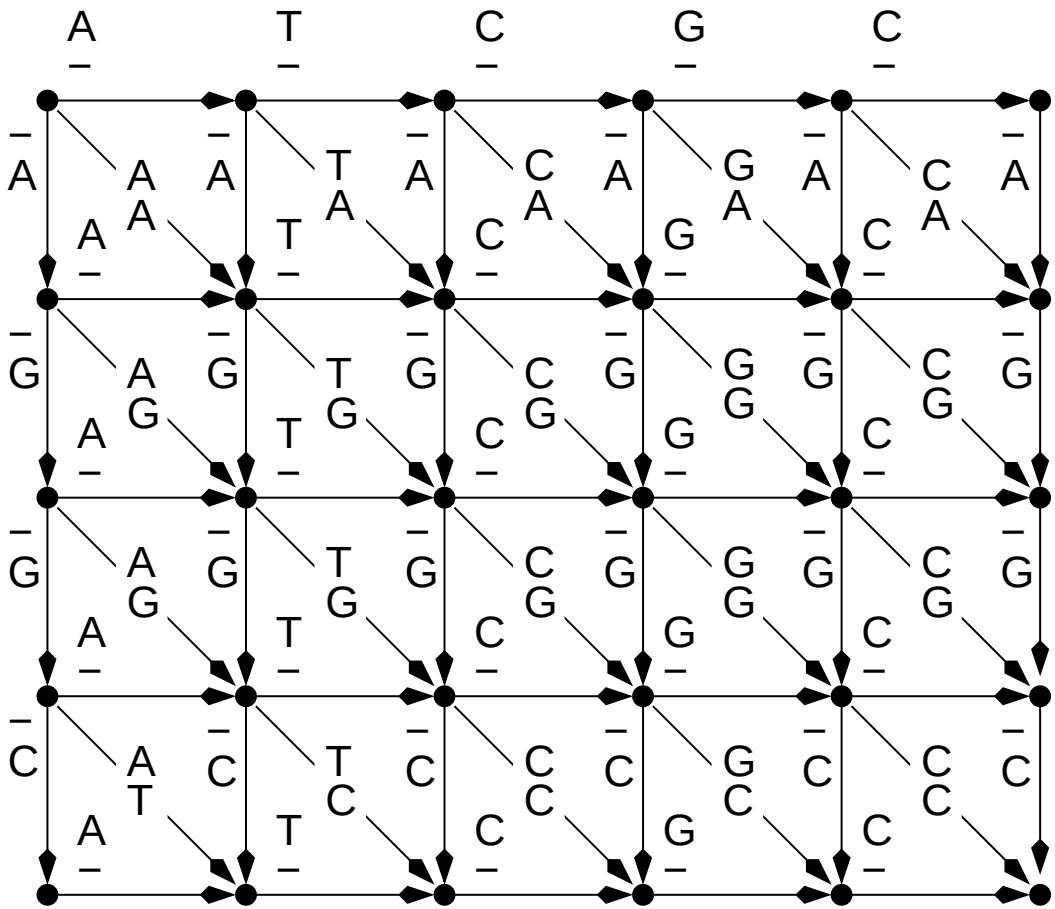
$$D(0,j) = j \times d, j \in 1..m$$

**General case**,

$$D(i,j) = \min \begin{cases} D(i-1,j) + d, \\ D(i,j-1) + d, \\ D(i-1,j-1) + m, \text{if } S_1(i) = S_2(j), \\ D(i-1,j-1) + s, \text{if } S_1(i) \neq S_2(j). \end{cases}$$

where $d$ represents the cost of a deletion, $m$ the cost of a match operation and $s$ the cost of a substitution.

# Alphabet-Weighted Edit Distance




**What are the necessary changes to our framework?**

# Alphabet-Weighted Edit Distance

**Base conditions**,

$$D(i,0) = \ i \times d, i \in 0..n$$

$$D(0,j) = \ j \times d, j \in 0..m$$

**General case**,

$$D(i,j) = \min \begin{cases} D(i-1,j) + d, \\ D(i,j-1) + d, \\ D(i-1,j-1) + s(S_1(i), S_2(j)). \end{cases}$$

where $d$ represents the **cost of a deletion** and $s(x,y)$ the **cost for substituting** $x$ **by** $y$, often represented as a substitution matrix:

| | A | G | T | C |
|---|---|---|---|---|
| A | 0.0 | 0.4 | 0.6 | 0.6 |
| G | 0.4 | 0.0 | 0.6 | 0.6 |
| T | 0.6 | 0.6 | 0.0 | 0.4 |
| C | 0.6 | 0.6 | 0.4 | 0.0 |

# Remarks

- To compare **protein** sequences, an alphabet weighted scoring scheme is always used
- There are well known schemes such as **PAM** and **BLOSUM**, more about in a next lecture

# BLOSUM50

| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | Z | X | * |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 5 | -2 | -1 | -2 | -1 | -1 | -1 | 0 | -2 | -1 | -2 | -1 | -1 | -3 | -1 | 1 | 0 | -3 | -2 | 0 | -2 | -1 | -1 | -5 |
| R | -2 | 7 | -1 | -2 | -4 | 1 | 0 | -3 | 0 | -4 | -3 | 3 | -2 | -3 | -3 | -1 | -1 | -3 | -1 | -3 | -1 | 0 | -1 | -5 |
| N | -1 | -1 | 7 | 2 | -2 | 0 | 0 | 0 | 1 | -3 | -4 | 0 | -2 | -4 | -2 | 1 | 0 | -4 | -2 | -3 | 4 | 0 | -1 | -5 |
| D | -2 | -2 | 2 | 8 | -4 | 0 | 2 | -1 | -1 | -4 | -4 | -1 | -4 | -5 | -1 | 0 | -1 | -5 | -3 | -4 | 5 | 1 | -1 | -5 |
| C | -1 | -4 | -2 | -4 | 13 | -3 | -3 | -3 | -3 | -2 | -2 | -3 | -2 | -2 | -4 | -1 | -1 | -5 | -3 | -1 | -3 | -3 | -2 | -5 |
| Q | -1 | 1 | 0 | 0 | -3 | 7 | 2 | -2 | 1 | -3 | -2 | 2 | 0 | -4 | -1 | 0 | -1 | -1 | -1 | -3 | 0 | 4 | -1 | -5 |
| E | -1 | 0 | 0 | 2 | -3 | 2 | 6 | -3 | 0 | -4 | -3 | 1 | -2 | -3 | -1 | -1 | -1 | -3 | -2 | -3 | 1 | 5 | -1 | -5 |
| G | 0 | -3 | 0 | -1 | -3 | -2 | -3 | 8 | -2 | -4 | -4 | -2 | -3 | -4 | -2 | 0 | -2 | -3 | -3 | -4 | -1 | -2 | -2 | -5 |
| H | -2 | 0 | 1 | -1 | -3 | 1 | 0 | -2 | 10 | -4 | -3 | 0 | -1 | -1 | -2 | -1 | -2 | -3 | 2 | -4 | 0 | 0 | -1 | -5 |
| I | -1 | -4 | -3 | -4 | -2 | -3 | -4 | -4 | -4 | 5 | 2 | -3 | 2 | 0 | -3 | -3 | -1 | -3 | -1 | 4 | -4 | -3 | -1 | -5 |
| L | -2 | -3 | -4 | -4 | -2 | -2 | -3 | -4 | -3 | 2 | 5 | -3 | 3 | 1 | -4 | -3 | -1 | -2 | -1 | 1 | -4 | -3 | -1 | -5 |
| K | -1 | 3 | 0 | -1 | -3 | 2 | 1 | -2 | 0 | -3 | -3 | 6 | -2 | -4 | -1 | 0 | -1 | -3 | -2 | -3 | 0 | 1 | -1 | -5 |
| M | -1 | -2 | -2 | -4 | -2 | 0 | -2 | -3 | -1 | 2 | 3 | -2 | 7 | 0 | -3 | -2 | -1 | -1 | 0 | 1 | -3 | -1 | -1 | -5 |
| F | -3 | -3 | -4 | -5 | -2 | -4 | -3 | -4 | -1 | 0 | 1 | -4 | 0 | 8 | -4 | -3 | -2 | 1 | 4 | -1 | -4 | -4 | -2 | -5 |
| P | -1 | -3 | -2 | -1 | -4 | -1 | -1 | -2 | -2 | -3 | -4 | -1 | -3 | -4 | 10 | -1 | -1 | -4 | -3 | -3 | -2 | -1 | -2 | -5 |
| S | 1 | -1 | 1 | 0 | -1 | 0 | -1 | 0 | -1 | -3 | -3 | 0 | -2 | -3 | -1 | 5 | 2 | -4 | -2 | -2 | 0 | 0 | -1 | -5 |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 2 | 5 | -3 | -2 | 0 | 0 | -1 | 0 | -5 |
| W | -3 | -3 | -4 | -5 | -5 | -1 | -3 | -3 | -3 | -3 | -2 | -3 | -1 | 1 | -4 | -4 | -3 | 15 | 2 | -3 | -5 | -2 | -3 | -5 |
| Y | -2 | -1 | -2 | -3 | -3 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | 0 | 4 | -3 | -2 | -2 | 2 | 8 | -1 | -3 | -2 | -1 | -5 |
| V | 0 | -3 | -3 | -4 | -1 | -3 | -3 | -4 | -4 | 4 | 1 | -3 | 1 | -1 | -3 | -2 | 0 | -3 | -1 | 5 | -4 | -3 | -1 | -5 |
| B | -2 | -1 | 4 | 5 | -3 | 0 | 1 | -1 | 0 | -4 | -4 | 0 | -3 | -4 | -2 | 0 | 0 | -5 | -3 | -4 | 5 | 2 | -1 | -5 |
| Z | -1 | 0 | 0 | 1 | -3 | 4 | 5 | -2 | 0 | -3 | -3 | 1 | -1 | -4 | -1 | 0 | -1 | -2 | -2 | -3 | 2 | 5 | -1 | -5 |
| X | -1 | -1 | -1 | -1 | -2 | -1 | -1 | -2 | -1 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | 0 | -3 | -1 | -1 | -1 | -1 | -1 | -5 |
| * | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | 1 |

$\Rightarrow$ Look at the costs, **can the matrix be used in our current framework**?

# Similarity

- **Distance** and **similarity** are two related ("**opposed**") concepts.
- Intuitively, two sequences have "**high**" degree of similarity if their edit distance is "**low**"
- Whereas, two sequences have a "**low**" degree of similarity if their edit distance is "**high**"

# Similarity

- Let $\Sigma' = \Sigma \cup \{'-'\}$ denote the alphabet which includes the gap symbol, and $S'_1, S'_2$ denote strings obtained by inserting gap symbols into $S_1$ and $S_2$ so that both strings now have the same length, $l$, and let's call $S'_1, S'_2$ an alignment, $\mathcal{A}$, of $S_1, S_2$.
- The **value of an alignment** is

$$\sum_{i=1}^{l} s(S'_1(i), S'_2(i))$$

where $s(x, y)$ is the cost for matching $x$ against $y$ in the alignment $\mathcal{A}$.
- The **similarity** of two strings $S_1$ and $S_2$ is maximum value of the alignment.
- To distinguish similarity and distance, let's introduce a new index, $V(i, j)$, to denote the value of the optimal (maximal) alignment of $S_1[1..i]$ and $S_2[1..j]$, as well as a

# Similarity

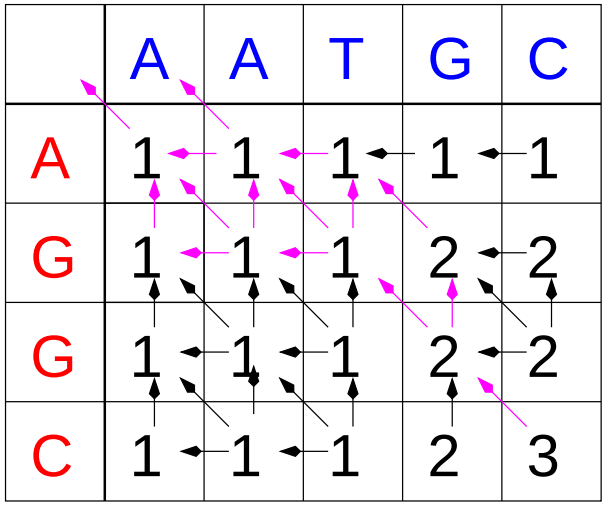
$$V(i,0) = \sum_{0 \leq k \leq i} s(S_1(k), ' - ')$$

$$V(0,j) = \sum_{0 \leq k \leq j} s(' - ', S_2(k))$$

$$V(i,j) = \max \begin{cases} V(i-1,j) + s(S_1(i), ' - '), \\ V(i,j-1) + s(' - ', S_2(j)), \\ V(i-1,j-1) + s(S_1'(i), S_2'(j)). \end{cases}$$

$\Rightarrow$ Similarity is **more often used** than edit distance in the context of **biological alignments**.

# A simple **Example** of Dynamic Programming




$\Rightarrow$ Deduce the scoring scheme for the maximum similarity alignment above.

# Remarks

It is common practice to use a scoring scheme such that the weight for a (**favourable**) match is **positive** and the weight for a **mismatch** is **negative**.

| | A | G | T | C | '-' |
|---|---|---|---|---|---|
| A | 2 | -1 | -2 | -2 | -2 |
| G | -1 | 2 | -2 | -2 | -2 |
| T | -2 | -2 | 1 | -1 | -1 |
| C | -2 | -2 | -1 | 1 | -1 |
| '-' | -2 | -2 | -1 | -1 | 0 |

# Needleman & Wunsch

$$V(i,0) = i \times d, i \in 0..n$$

$$V(0,j) = j \times d, i \in 0..m$$

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + s(S_1(i), S_2(j)), \\ V(i-1,j) + d, \\ V(i,j-1) + d. \end{cases}$$

where $d$ is the cost of a deletion and $d < 0$

$\Rightarrow$ Needleman & Wunsch (1970) *J. Mol. Biol.* **48**(3):443-453.

| | – | H | E | A | G | A | W | G | H | E | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| – | 0 | −8 | −16 | −24 | −32 | −40 | −48 | −56 | −64 | −72 | −80 |
| P | −8 | −2 | −9 | −17 | −25 | −33 | −42 | −49 | −57 | −65 | −73 |
| A | −16 | −10 | −3 | −4 | −12 | −20 | −28 | −36 | −44 | −52 | −60 |
| W | −24 | −18 | −11 | −6 | −7 | −15 | −5 | −13 | −21 | −29 | −37 |
| H | −32 | −14 | −18 | −13 | −8 | −9 | −13 | −7 | −3 | −11 | −19 |
| E | −40 | −22 | −8 | −16 | −16 | −9 | −12 | −15 | −7 | 3 | −5 |
| A | −48 | −30 | −16 | −3 | −11 | −11 | −12 | −12 | −15 | −5 | 2 |
| E | −56 | −38 | −24 | −11 | −6 | −12 | −14 | −15 | −12 | −9 | 1 |

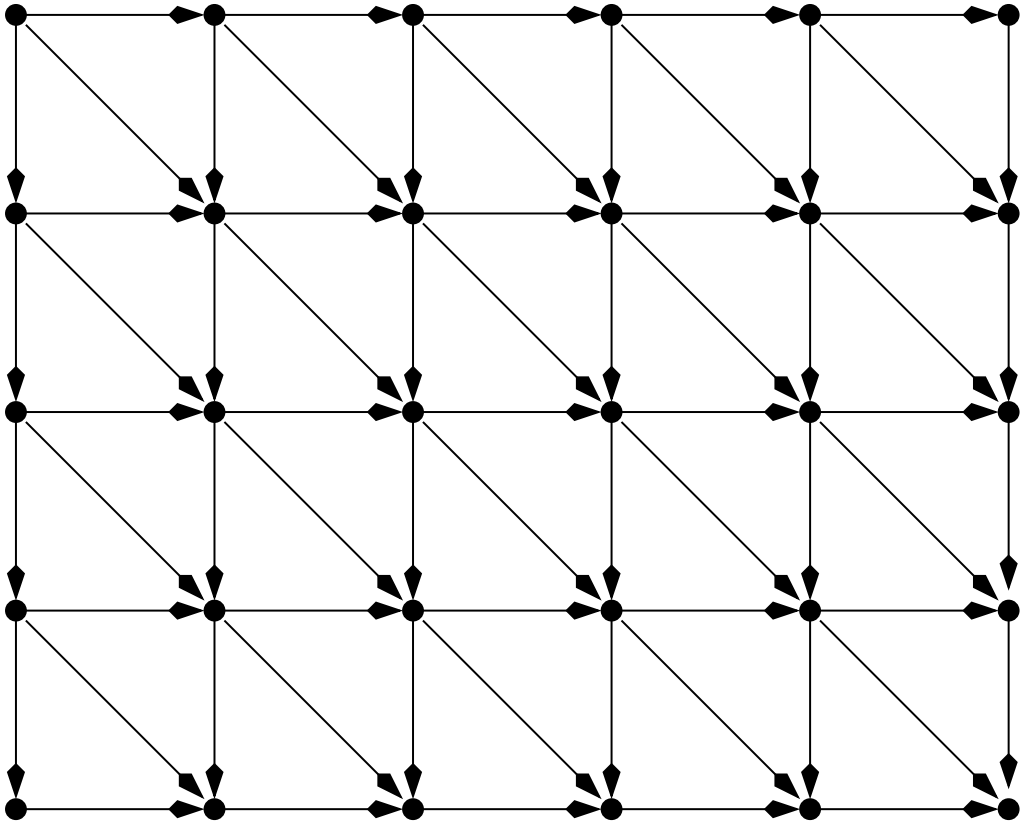
```
HEAGAWGHE-E
    || ||  |
--P-AW-HEAE
```

⇒ This alignment has been produced using the **Needleman & Wunsch** recurrence equation, the **BLOSUM50** matrix and what indel penalty cost?

# Free end-gaps (semi-global)

- It is common practice to **not penalize the gaps** at the **start** and the **end** of an alignment – internal insertions/deletions are penalized according to the same scheme as before.
- The end-gaps free alignments are considered to model more accurately the biological reality.

# What are the necessary changes to our framework?



# Free end-gaps (semi-global)

To achieve this result two modifications need to be made:

- The **initial conditions** have to be changed, $V(i,0) = V(0,j) = 0$ for all $i$ and $j$. This takes care of the indels at the start of the alignment;
- To take care of the **spaces at the end** of the alignment, instead of starting the traceback from $(n,m)$, it now starts from the cell $V(n,j)$ or $V(i,m)$ that has a maximum value for all $i$, $j$ (of course there could more than one place to start). This follows from the definition of $V(i,j)$.

# Semi-global

$$V(0,0) = 0$$

$$V(i,0) = 0, i = 1..m$$

$$V(0,j) = 0, j = 1..n$$

$$V(i,j) = \max \begin{cases} V(i-1,j) + s(S_1(i), '-'), \\ V(i,j-1) + s('-', S_2(j)), \\ V(i-1,j-1) + s(S'_1(i), S'_2(j)). \end{cases}$$

Solution is,

$$\max_{i=1..m, j=1..n} [V(m,n), V(i,n), V(m,j)]$$

$\Rightarrow$ Two modifications: initialisation, consider the last row/column to find the optimal value.

# Global

Indel = -3; Substitution score:

| | A | C | G | T |
|---|---|---|---|---|
| A | 1 | -5 | -1 | -5 |
| C | -5 | 1 | -5 | -1 |
| G | -1 | -5 | 1 | -5 |
| T | -5 | -1 | -5 | 1 |

Global
max = -17

| | - | A | A | C | A | C | G | T | G | T | C | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | { 0} | { -3} | { -6} | { -9} | [-12] | [-15] | [-18] | [-21] | [-24] | [-27] | [-30] | [-33] |
| A | [ -3] | [ 1] | [ -2] | [ -5] | { -8} | [-11] | [-14] | [-17] | [-20] | [-23] | [-26] | [-29] |
| C | [ -6] | [ -2] | [ -4] | [ -1] | [ -4] | { -7} | {-10} | {-13} | [-16] | [-19] | [-22] | [-25] |
| G | [ -9] | [ -5] | [ -3] | [ -4] | [ -2] | [ -5] | [ -6] | [ -9] | {-12} | {-15} | {-18} | [-21] |
| T | [-12] | [ -8] | [ -6] | [ -4] | [ -5] | [ -3] | [ -6] | [ -5] | [ -8] | [-11] | [-14] | {-17} |

AACACGTGTCT
---AC--G--T

# Semi-global

```
Semi-global
max = 4
     -    A    A    C    A    C    G    T    G    T    C    T
- [  0][  0][  0][  0][  0][  0][  0][  0][  0][  0][  0][  0]
A [  0][  1][  1][ -2]{  1}[ -2][ -1][ -3][ -1][ -3][ -3][ -3]
C [  0][ -2][ -2][  2][ -1]{  2}[ -1][ -2][ -4][ -2][ -2][ -4]
G [  0][ -1][ -3][ -1][  1][ -1]{  3}[  0][ -1][ -4][ -5][ -7]
T [  0][ -3][ -6][ -4][ -2][  0][  0]{  4}[  1][  0][ -3][ -4]

AACACGTGTCT
---ACGT----
```

```
Global (extreme and non-realistic example)
max = -63
             A    A    A    A    A    A
    [  0][ -3][ -6][ -9][-12][-15][-18]
  C [ -3][ -5][ -8][-11][-14][-17][-20]
  C [ -6][ -8][-10][-13][-16][-19][-22]
  C [ -9][-11][-13][-15][-18][-21][-24]
  C [-12][-14][-16][-18][-20][-23][-26]
  C [-15][-17][-19][-21][-23][-25][-28]
  C [-18][-20][-22][-24][-26][-28][-30]
  C [-21][-23][-25][-27][-29][-31][-33]
  C [-24][-26][-28][-30][-32][-34][-36]
  C [-27][-29][-31][-33][-35][-37][-39]
  C {-30}[-32][-34][-36][-38][-40][-42]
  C [-33]{-35}[-37][-39][-41][-43][-45]
  C [-36][-38]{-40}[-42][-44][-46][-48]
  C [-39][-41][-43]{-45}[-47][-49][-51]
  C [-42][-44][-46][-48]{-50}[-52][-54]
  C [-45][-47][-49][-51][-53]{-55}[-57]
  C [-48][-50][-52][-54][-56][-58]{-60}
  C [-51][-53][-55][-57][-59][-61]{-63}

----------AAAAAA-
CCCCCCCCCCCCCCCCC
     (-63)
```

```
Semi-global
max = -3
             A     A     A     A     A     A
     [  0][  0][  0][  0][  0][  0][  0]
  C [  0][ -3][ -3][ -3][ -3][ -3][ -3]
  C [  0][ -3][ -6][ -6][ -6][ -6][ -6]
  C [  0][ -3][ -6][ -9][ -9][ -9][ -9]
  C [  0][ -3][ -6][ -9][-12][-12][-12]
  C [  0][ -3][ -6][ -9][-12][-15][-15]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]
  C [  0][ -3][ -6][ -9][-12][-15][-18]

AAAAAA-----------------
------CCCCCCCCCCCCCCCCC
      (0)
```

# **Local** Alignment

To apply a global alignment one has to **assume that the two strings can be aligned on their entire length**, which is the case when comparing proteins from the same family, for example the $\alpha$ chain of hemoglobin from the pig (*Sus scrofa*) and the trout (*Oncorhynchus mykiss*):

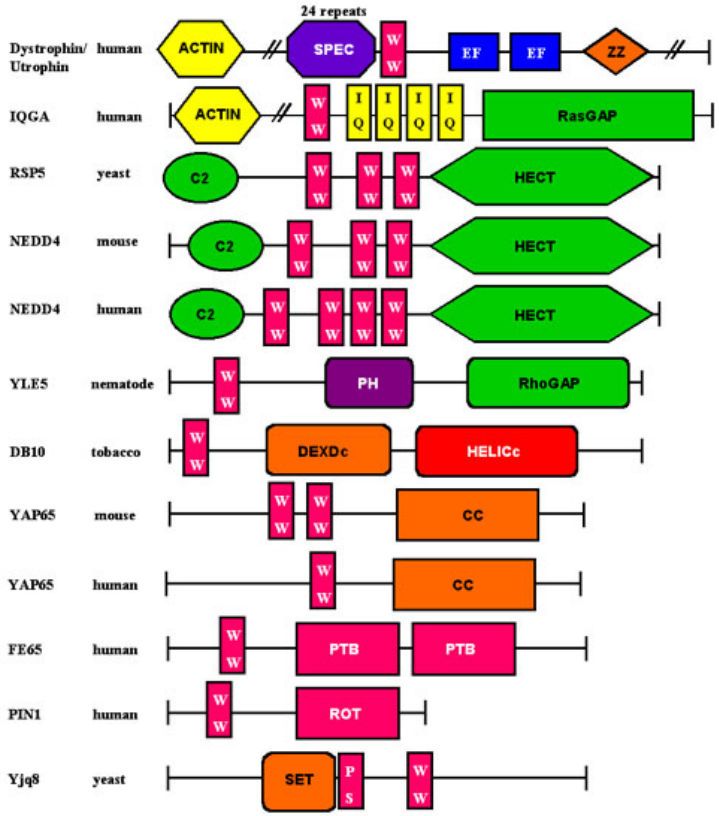
```
scoring matrix: BLOSUM50, gap penalties: -12/-2
60.6% identity; Global alignment score: 542

               10        20        30        40        50
Pig   VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQVKAHG
       :.: ::. ::: :::.:.: . ::::: ::. ..: ::::: :. .:: ::  :: ::
Trout SLTAKDKSVVKAFWGKISGKADVVGAEALGRMLTAYPQTKTYFSHWADLSPGSGPVKKHG
               10        20        30        40        50        60

     60        70        80        90       100       110
Pig   QKVADALTKAVGHLDDLPGALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHHPDDFNP
        .  :. :::: .::: :..:::::::: :::::: :::.::: .:::::: : :.:::
Trout GIIMGAIGKAVGLMDDLVGGMSALSDLHAFKLRVDPGNFKILSHNILVTLAIHFPSDFTP
               70        80        90       100       110       120

     120       130       140
Pig   SVHASLDKFLANVSTVLTSKYR
       :: ..:::::: ::.:.:.:::
Trout EVHIAVDKFLAAVSAALADKYR
               130       140
```

# **Local** Alignment

- In particular, the sequences being compared should be approximately the **same length**.
- However, sometimes we would like to compare the DNA sequence of a gene against an entire genome — looking for paralogous genes.

# **Local** Alignment

In the case of proteins, we are more and more appreciating their **modular architecture**: e.g. WW domain occurs many proteins.



Indel = -3; Substitution score:

| | A | C | G | T |
|---|---|---|---|---|
| A | 1 | -5 | -1 | -5 |
| C | -5 | 1 | -5 | -1 |
| G | -1 | -5 | 1 | -5 |
| T | -5 | -1 | -5 | 1 |

Global (Needleman-Wunsch)

max = -16

| | | A | A | C | C | T | A | T | A | G | C | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | { 0} | [ -3] | [ -6] | [ -9] | [-12] | [-15] | [-18] | [-21] | [-24] | [-27] | [-30] | [-33] |
| G | [ -3] | { -1} | { -4} | { -7} | [-10] | [-13] | [-16] | [-19] | [-22] | [-23] | [-26] | [-29] |
| C | [ -6] | [ -4] | [ -6] | [ -3] | { -6} | [ -9] | [-12] | [-15] | [-18] | [-21] | [-22] | [-25] |
| G | [ -9] | [ -7] | [ -5] | [ -6] | [ -8] | {-11} | [-10] | [-13] | [-16] | [-17] | [-20] | [-23] |
| A | [-12] | [ -8] | [ -6] | [ -9] | [-11] | [-13] | {-10} | [-13] | [-12] | [-15] | [-18] | [-21] |
| T | [-15] | [-11] | [ -9] | [ -7] | [-10] | [-10] | [-13] | { -9} | [-12] | [-15] | [-16] | [-17] |
| A | [-18] | [-14] | [-10] | [-10] | [-12] | [-13] | [ -9] | [-12] | { -8} | {-11} | {-14} | [-17] |
| T | [-21] | [-17] | [-13] | [-11] | [-11] | [-11] | [-12] | [ -8] | [-11] | [-13] | [-12] | {-13} |
| A | [-24] | [-20] | [-16] | [-14] | [-14] | [-14] | [-10] | [-11] | [ -7] | [-10] | [-13] | {-16} |

AACCTATAGCT-

G--CGATA--TA

Semi-global

max = 1

| | | A | A | C | C | T | A | T | A | G | C | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [ 0] | [ 0] | [ 0] | [ 0] | [ 0] | [ 0] | [ 0] | [ 0] | [ 0] | [ 0] | [ 0] | [ 0] |
| G | [ 0] | [-1] | [-1] | [-3] | [-3] | [-3] | [-1] | [-3] | [-1] | [ 1] | [-2] | [-3] |
| C | [ 0] | [-3] | [-4] | [ 0] | [-2] | [-4] | [-4] | [-2] | [-4] | [-2] | [ 2] | [-1] |
| G | [ 0] | [-1] | [-4] | [-3] | [-5] | [-7] | [-5] | [-5] | [-3] | [-3] | [-1] | [-3] |
| A | [ 0] | [ 1] | [ 0] | [-3] | [-6] | [-9] | [-6] | [-8] | [-4] | [-4] | [-4] | [-6] |
| T | [ 0] | [-2] | [-3] | [-1] | [-4] | [-5] | [-8] | [-5] | [-7] | [-7] | [-5] | [-3] |
| A | [ 0] | [ 1] | [-1] | [-4] | [-6] | [-8] | [-4] | [-7] | [-4] | [-7] | [-8] | [-6] |
| T | [ 0] | [-2] | [-4] | [-2] | [-5] | [-5] | [-7] | [-3] | [-6] | [-9] | [-8] | [-7] |
| A | [ 0] | [ 1] | [-1] | [-4] | [-7] | [-8] | [-4] | [-6] | [-2] | [-5] | [-8] | [-10] |

-------AACCTATAGCT

GCGATATA----------

(1)

# **Local** Alignment

- The previous slides are presenting examples where the global and semi-global alignment framework is not suited.

# **Local** Alignment

- The previous slides are presenting examples where the global and semi-global alignment framework is not suited.
- The **local alignment problem** consists in finding a pair of substrings $\alpha$ and $\beta$, of $S_1$ and $S_2$ respectively, whose optimal global alignment value is maximum over all possible pairs of substrings — denoted by $v^\star$.

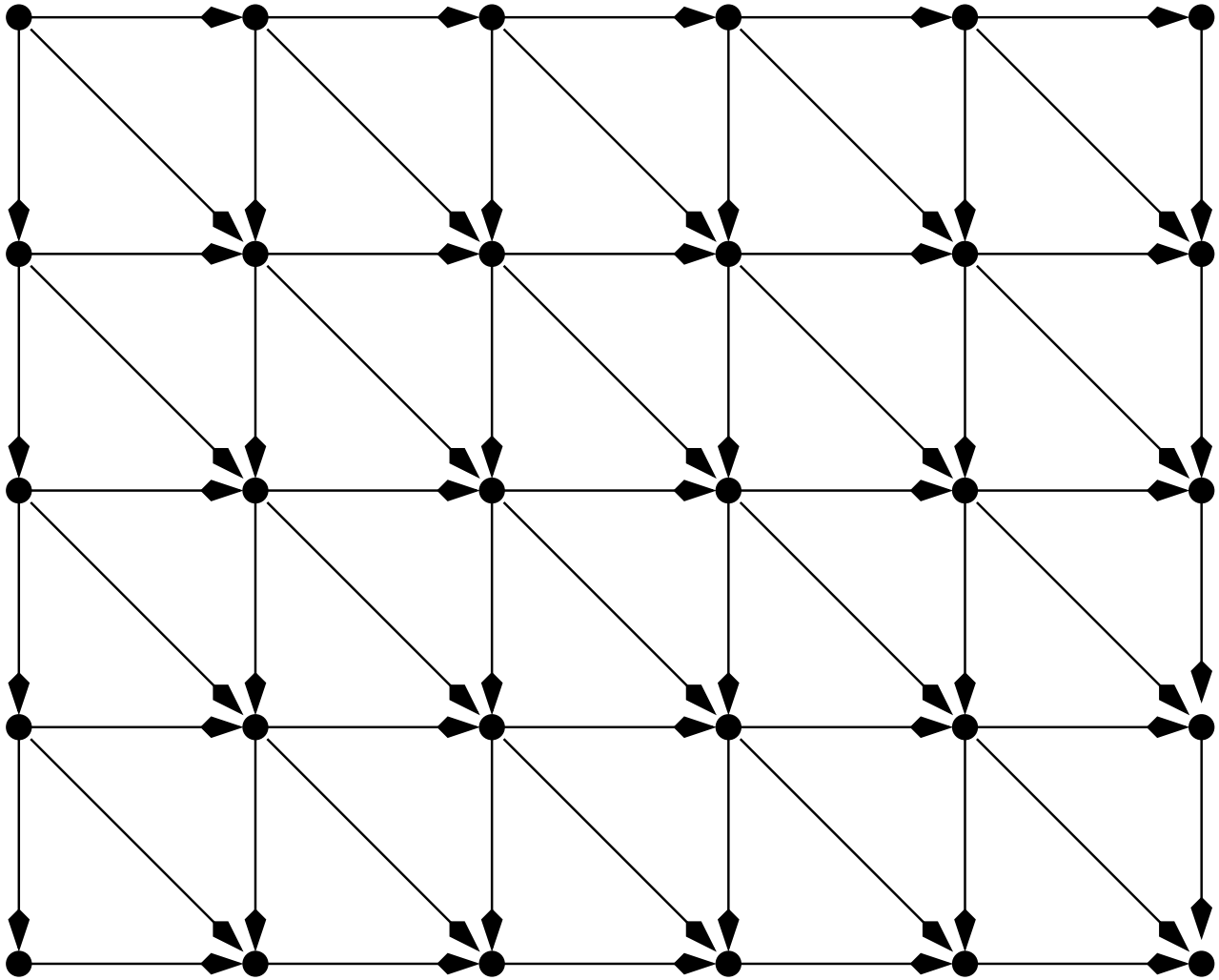
# **Local** Alignment

- The previous slides are presenting examples where the global and semi-global alignment framework is not suited.
- The **local alignment problem** consists in finding a pair of substrings $\alpha$ and $\beta$, of $S_1$ and $S_2$ respectively, whose optimal global alignment value is maximum over all possible pairs of substrings — denoted by $v^\star$.
- Given a string $S$ of length $n$, there are $\mathcal{O}(n^2)$ distinct substrings. Therefore, given two strings $S_1$, of length $n$, and $S_2$ of length $m$, there are $\mathcal{O}(n^2 m^2)$ possible pairs.

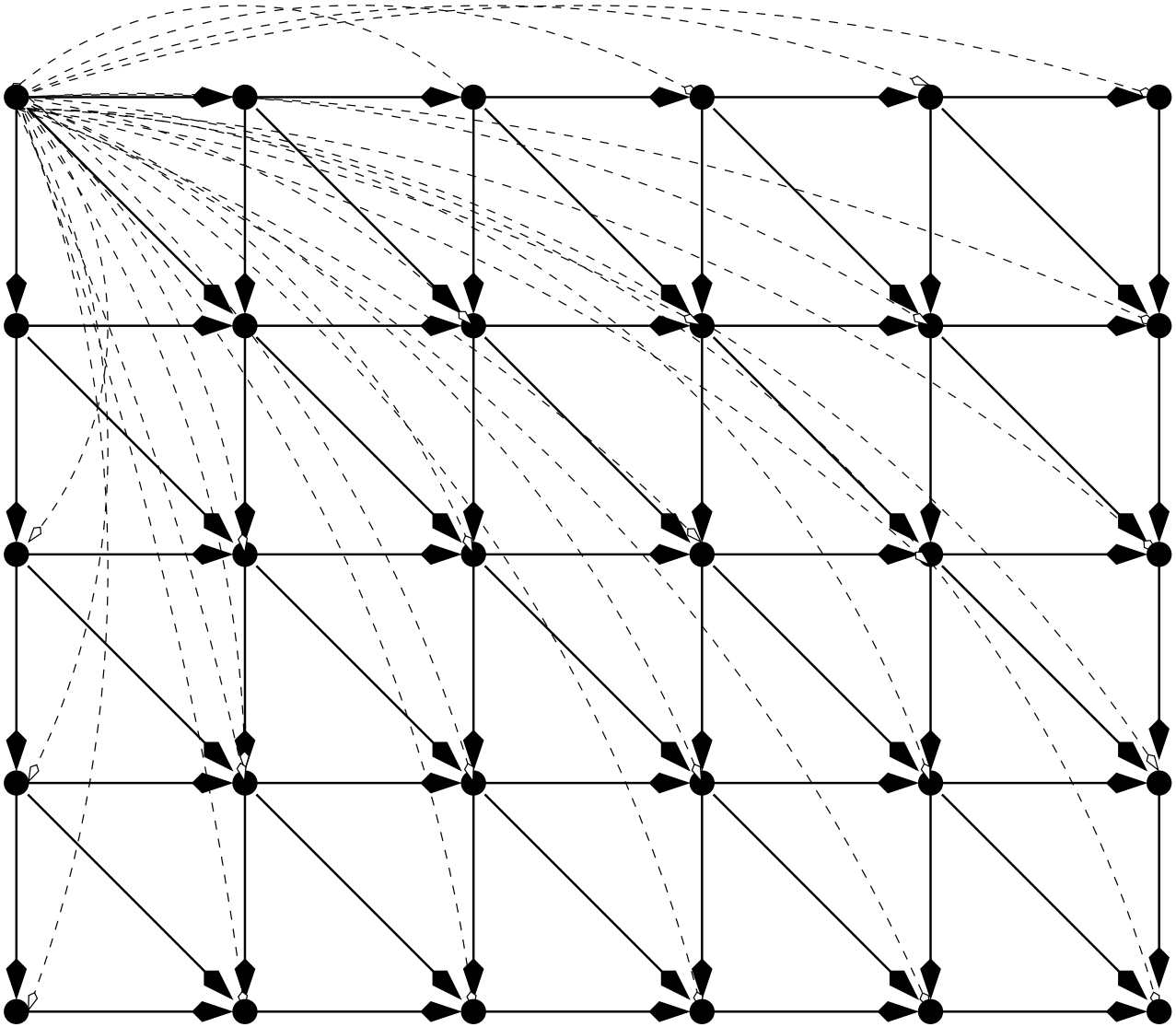
# **Local** Alignment

- The previous slides are presenting examples where the global and semi-global alignment framework is not suited.
- The **local alignment problem** consists in finding a pair of substrings $\alpha$ and $\beta$, of $S_1$ and $S_2$ respectively, whose optimal global alignment value is maximum over all possible pairs of substrings — denoted by $v^\star$.
- Given a string $S$ of length $n$, there are $\mathcal{O}(n^2)$ distinct substrings. Therefore, given two strings $S_1$, of length $n$, and $S_2$ of length $m$, there are $\mathcal{O}(n^2m^2)$ possible pairs.
- Finding the optimal global alignment of one pair takes $\mathcal{O}(mn)$, therefore, a naive approach to solve the local alignment problem would run in $\mathcal{O}(m^3n^3)$!

# What are the **necessary changes** to our framework?



# **Local** Alignment



- The **local alignment problem** consists in finding a pair of substrings $\alpha$ and $\beta$, of $S_1$ and $S_2$ respectively, whose optimal global alignment value is maximum over all possible pairs of substrings.

- The **local alignment problem** consists in finding a pair of substrings $\alpha$ and $\beta$, of $S_1$ and $S_2$ respectively, whose optimal global alignment value is maximum over all possible pairs of substrings.
- Effectively, **this represents a path in the edit graph from some $(i, j)$ to $(i', j')$ whose global alignment is maximum**; rather than a path from $(0, 0)$ to $(m, n)$.

- The **local alignment problem** consists in finding a pair of substrings $\alpha$ and $\beta$, of $S_1$ and $S_2$ respectively, whose optimal global alignment value is maximum over all possible pairs of substrings.
- Effectively, **this represents a path in the edit graph from some $(i, j)$ to $(i', j')$ whose global alignment is maximum**; rather than a path from $(0, 0)$ to $(m, n)$.
- The solution is surprisingly simple, it consists of **adding edges of weight** 0 from $(0, 0)$ to all the other nodes of the graph (and from all the nodes to $(m, n)$).

- The **local alignment problem** consists in finding a pair of substrings $\alpha$ and $\beta$, of $S_1$ and $S_2$ respectively, whose optimal global alignment value is maximum over all possible pairs of substrings.
- Effectively, **this represents a path in the edit graph from some** $(i, j)$ **to** $(i', j')$ **whose global alignment is maximum**; rather than a path from $(0, 0)$ to $(m, n)$.
- The solution is surprisingly simple, it consists of **adding edges of weight** 0 from $(0, 0)$ to all the other nodes of the graph (and from all the nodes to $(m, n)$).
- When computing the value of a cell $(i, j)$, this means there is one more path to consider, $(0, 0)$ to $(i, j)$, which always has a cost of 0.

# **Smith-Waterman** Algorithm

There are only **two differences** with respect to the Needleman-Wunsch algorithm:

1. An **extra term** is added to the recurrence, which allows to **reset** the alignment to zero when all other possibilities lead to a negative score, which also corresponds to starting a new alignment;
2. The alignment can now **stop anywhere**, therefore we need to search the grid for the maximum score and then follow the traceback pointers.

# Smith & Waterman Algorithm

**Base conditions**,

$$v(i,0) = 0, i \in 0..n$$

$$v(0,j) = 0, j \in 0..m$$

**General case**,

$$v(i,j) = \max \begin{cases} 0, \\ v(i-1,j) - s(S_1(i),'-'), \\ v(i,j-1) - s('-',S_2(j)), \\ v(i-1,j-1) + s(S_1(i),S_2(j)). \end{cases}$$

**Solution**,

$$v^\star = \max[v(i,j) : i \leq n, j \leq m]$$

$\Rightarrow$ Smith & Waterman (1981) *J. Mol. Biol.* **147**:195-197.

Local (Smith-Waterman)

max =

| | | - | A | A | C | C | T | A | T | A | G | C | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | G | 0 | | | | | | | | | | | |
| | C | 0 | | | | | | | | | | | |
| | G | 0 | | | | | | | | | | | |
| | A | 0 | | | | | | | | | | | |
| | T | 0 | | | | | | | | | | | |
| | A | 0 | | | | | | | | | | | |
| | T | 0 | | | | | | | | | | | |
| | A | 0 | | | | | | | | | | | |

Local (Smith-Waterman)

max = 4

| | | A | A | C | C | T | A | T | A | G | C | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| C | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| G | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| A | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| T | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 1 |
| A | 0 | 1 | 1 | 0 | 0 | 0 | 2 | 0 | 3 | 0 | 0 | 0 |
| T | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 1 |
| A | 0 | 1 | 1 | 0 | 0 | 0 | 2 | 0 | 4 | 1 | 0 | 0 |

TATA

TATA

(4)

# Remarks

- To find the **optimum**, $v^\star$, necessitates finding the largest $v(i, j)$ for all $i, j$, this takes $\mathcal{O}(nm)$;
- The score for an **unfavorable local alignment** should be negative, scores derived as log likelihood ratio do meet this requirement (more later);
- Time/space **complexity**, $\mathcal{O}(nm)$.

| | – | H | E | A | G | A | W | G | H | E | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 0 | 5 | 0 | **5** | 0 | 0 | 0 | 0 | 0 |
| W | 0 | 0 | 0 | 0 | 2 | 0 | **20** | **12** | 4 | 0 | 0 |
| H | 0 | 10 | 2 | 0 | 0 | 0 | 12 | 18 | **22** | 14 | 6 |
| E | 0 | 2 | 16 | 8 | 0 | 0 | 4 | 10 | 18 | **28** | 20 |
| A | 0 | 0 | 8 | 21 | 13 | 5 | 0 | 4 | 10 | **20** | 27 |
| E | 0 | 0 | 6 | 13 | 18 | 12 | 4 | 0 | 4 | 16 | **26** |

AWGHE

AW-HE

$\Rightarrow$ BLOSUM50 substitution score was used.

| | – | H | E | A | G | A | W | G | H | E | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| – | **0** | **−8** | **−16** | −24 | −32 | −40 | −48 | −56 | −64 | −72 | −80 |
| P | −8 | −2 | −9 | **−17** | **−25** | −33 | −42 | −49 | −57 | −65 | −73 |
| A | −16 | −10 | −3 | −4 | −12 | **−20** | −28 | −36 | −44 | −52 | −60 |
| W | −24 | −18 | −11 | −6 | −7 | −15 | **−5** | **−13** | −21 | −29 | −37 |
| H | −32 | −14 | −18 | −13 | −8 | −9 | −13 | −7 | **−3** | −11 | −19 |
| E | −40 | −22 | −8 | −16 | −16 | −9 | −12 | −15 | −7 | **3** | −5 |
| A | −48 | −30 | −16 | −3 | −11 | −11 | −12 | −12 | −15 | **−5** | 2 |
| E | −56 | −38 | −24 | −11 | −6 | −12 | −14 | −15 | −12 | −9 | **1** |

```
HEAGAWGHE-E
    || ||  |
--P-AW-HEAE
```

$\Rightarrow$ Global alignment for the same input sequences.

# **Gap** Penalties

- More accurate models of **biological sequence alignments**.
- Let's call a **gap** a maximal, consecutive run of insertions (deletions) in a single string of an alignment.
- Often **a single mutational event** can delete or insert a run of consecutive nucleotides (unequal cross-over, DNA slippage, transposable elements (DNA repeats), translocaltion, etc.), in the alignment one would like to favor the clustering of insertions into gaps, instead of having them dispersed along the alignment.

# 3 popular gap scoring strategies

```
VLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTK
SLSAAQKDNVKSSWAKA---SAAWGTAGPEFFMALFDAHD
```

- Let $g$ denote the length of the gap, 3 in the above example, and $\gamma(g)$ the gap penalty term.
- Noticed that we **no** longer consider the positions **independent** one from another!

# 3 popular gap scoring strategies

```
...AAAAA...
...A---A...
```

- Under the **linear gap weight model**, the score for this alignment will be: the alignment score for the prefix $+s(A,A) + 3 \times d + s(A,A)+$ the alignment score for the suffix, where $d = -8$ would be a typical value. I.e. $\gamma(g) = g \times d$.

# 3 popular gap scoring strategies

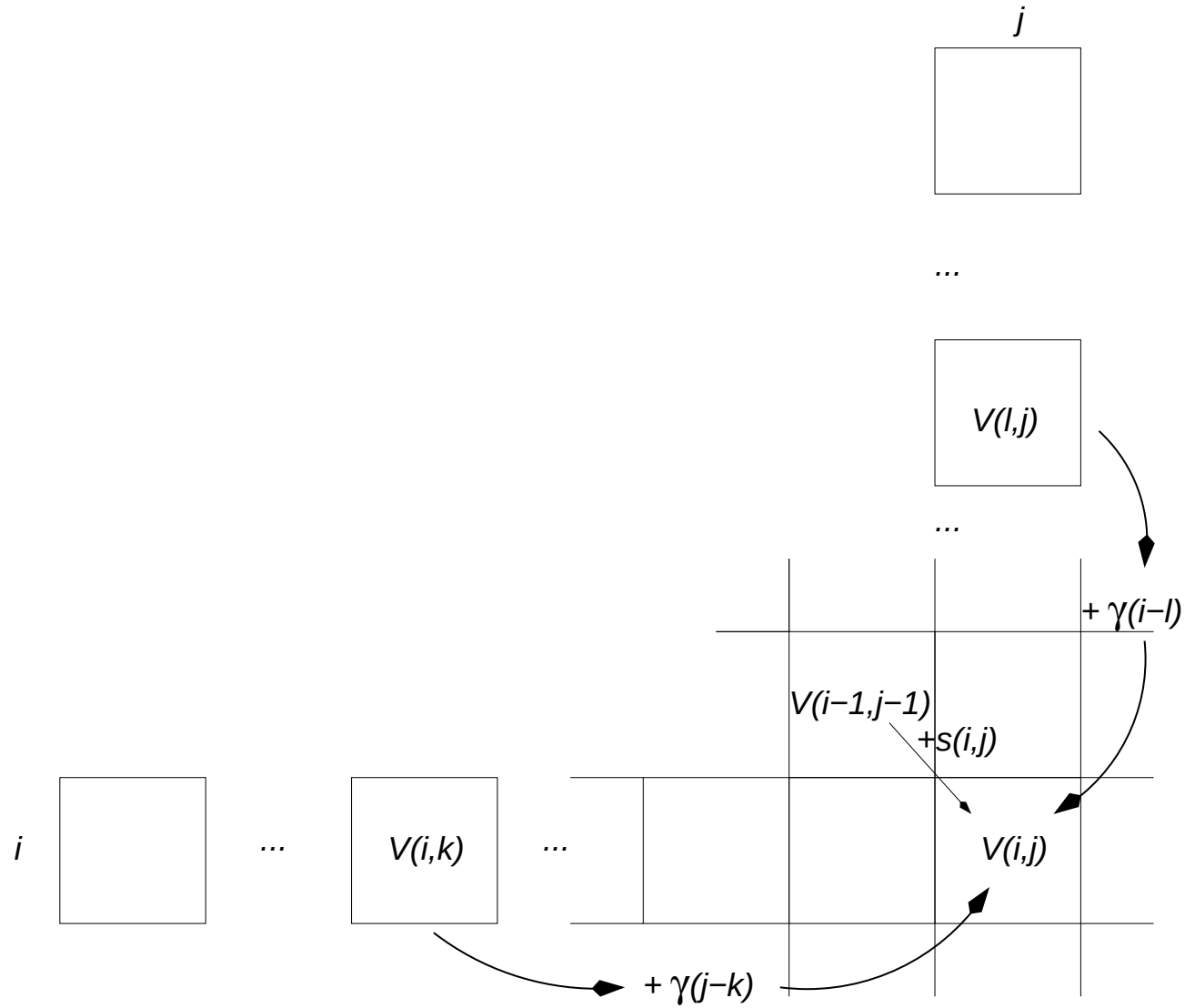
```
...AAAAA...
...A---A...
```

- Under the **affine gap weight model**, the score for this alignment will be: the alignment score for the prefix $+s(A,A) + d + 3 \times e + s(A,A)+$ the alignment score for the suffix, where $d$ is the gap opening (or initiation) cost, typical value is -12, and $e$ is the gap extension cost, typical value is -2. I.e. $\gamma(g) = d + g \times e$. The gap-extension, $e$, is usually smaller than the gap-opening, which has for effect to concentrate gaps in small islands. The affine gap weight model is the model which most implementations use.

# 3 popular gap scoring strategies

```
...AAAAA...
...A---A...
```

- The **general gap weight model** allows for any arbitrary function, such as $\gamma(g) = d + \ln g$.
- There is **no consensus** about the right model for gap weights at this point, it is still a matter of debates.
- Modeling gaps using an arbitrary function **raises the time complexity** of the algorithm to $\mathcal{O}(n^3)$, however, in the case of an affine function, we can lower this value to $\mathcal{O}(n^2)$ – which was the time complexity of the previous algorithms.


j
...
V(l,j)
...
+ γ(i−l)
V(i−1,j−1)
+s(i,j)
i
...
V(i,k)
...
V(i,j)
+ γ(j−k)

# Arbitrary Gap Weights

The general recurrence equation is modified to include $\gamma$, an arbitrary function which takes as input the length of the gap.

**Initialisation**,

$$V(i,0) = \gamma(i)$$

$$V(0,j) = \gamma(j)$$

**General recurrence**,

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + s(S_1(i), S_2(j)); \\ V(i,k) + \gamma(j-k), k = 0 \ldots j-1; \\ V(l,j) + \gamma(i-l), l = 0 \ldots i-1. \end{cases}$$

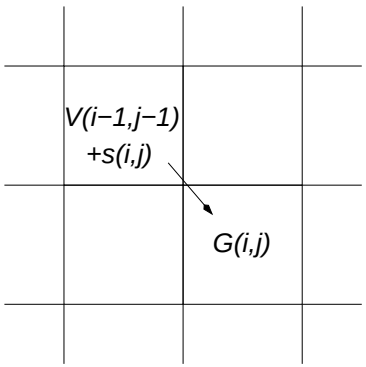
This increases the time complexity of the algorithm to $O(n^3)$, since we have to find the last non-gap position $k$ or $l$.

# **Affine** function

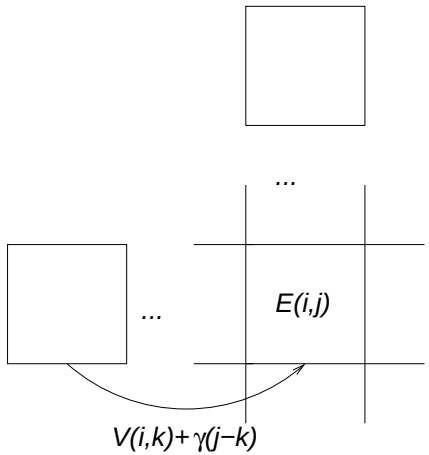
- **Gotoh** has proposed a dynamic programming approach that runs in $\mathcal{O}(n^2)$ time/space.
- **Opening** + **extension** costs:
  - $V(i,j) = V(l,j) + d + e \times (i - l)$

# Gotoh (affine function)

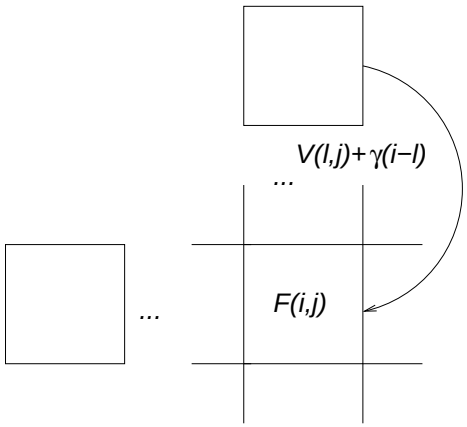
To develop the new recurrence equations, it will help to define three **new quantities**, $G$, $E$ and $F$, each keeping track of the last maximum score which was obtained as a result of a match or substitution of $S_1(i)$ and $S_2(j)$, an insertion into $S_1$ or an insertion into $S_2$, respectively.




(G)




(E)




(F)

$$\Rightarrow V(i,j) = \max[G(i,j), E(i,j), F(i,j)]$$

# **Gotoh** (affine function)

The general recurrence equation is modified to include $\gamma$, an arbitrary function which takes as input the gap length,

**Initialization**,

$$\begin{aligned} V(i,0) &= \gamma(i); \\ V(0,j) &= \gamma(j); \\ E(i,0) &= \gamma(i); \\ F(0,j) &= \gamma(j). \end{aligned}$$

**General case**,

$$\begin{aligned} V(i,j) &= \max[E(i,j), F(i,j), G(i,j)]; \\ E(i,j) &= \max_{0 \leq k \leq j-1}[V(i,k) + \gamma(j-k)]; \\ F(i,j) &= \max_{0 \leq l \leq i-1}[V(l,j) + \gamma(i-l)]; \\ G(i,j) &= V(i-1,j-1) + s(S_1(i), S_2(j)). \end{aligned}$$

This increases the time complexity of the algorithm to $O(n^3)$, since we have to find the last non-gap position $k$ or $l$.

# **Affine** Gap Weights Model (Gotoh)

- For the special case of the affine gap model, the time complexity, to calculate the optimal alignment, can be reduced to $\mathcal{O}(mn)$.
- They is idea is to observe that the cost for extending a gap varies by a constant amount, $e$, and therefore, it is not necessary to know the length of gap, but only the score of the alignment that is one position shorter.

# **Affine** Gap Weights Model (Gotoh)

**Initial conditions**,

$$V(i,0) = E(i,0) = d + i \times e$$

$$V(0,j) = F(0,j) = d + j \times e$$

**General case**,

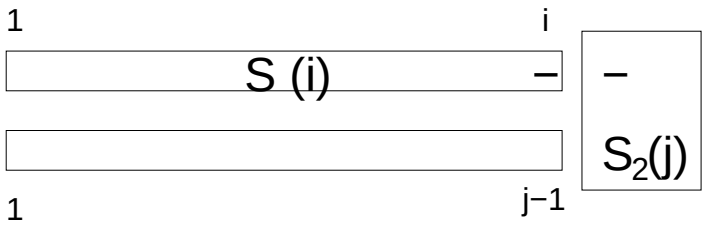
$$V(i,j) = \max[G(i,j), E(i,j), F(i,j)];$$

$$G(i,j) = V(i-1,j-1) + s(S_1(i), S_2(j));$$

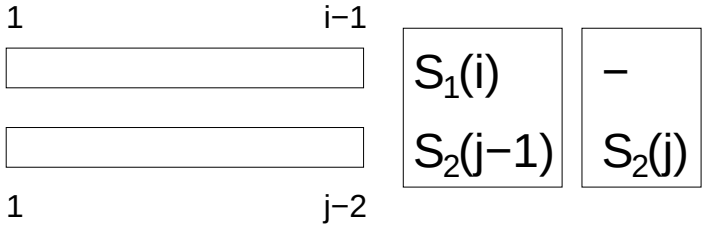
$$E(i,j) = \max[E(i,j-1) + e, V(i,j-1) + d + e];$$

$$F(i,j) = \max[F(i-1,j] + e, V(i-1,j) + d + e].$$

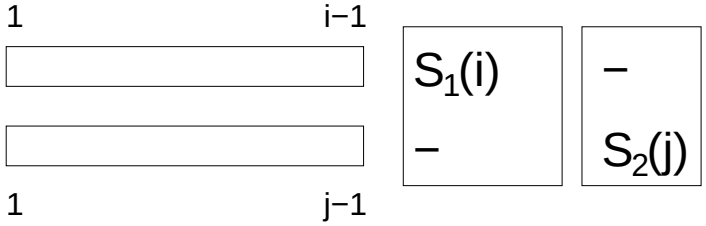
- Consider filling the $E(i,j)$ values.
- The first case consists of extending an alignment that is already ending with a dash symbol, $E(i,j) = E(i,j-1) + e$,




- For the second case, a new gap is created, i.e. the character to the left of the gap is $S_1(i)$. This can occur in two ways, either $S(i)$ is opposed to $S_2(j-1)$




or $S_1(i)$ is opposed to a dash,




- which means that the correct term to consider is $V(i,j-1) + d + e$ and not $G(i,j-1) + d + e$ (which takes into account only the first case).

# Summary

- Molecular sequences suffer **mutations** and therefore **change over time**.
- Organisms that have **diverged** only recently from a common ancestor will be more similar at the sequence level than organisms that have diverged further back in time.
- The **degree of similarity** between orthologous sequences, which perform the same function in two genomes, is "proportional" to time the organisms have actually diverged (not a linear relationship though).
- An **edit distance**, which represents the minimum number of edit operations that are necessary to transform one sequence into the other, is a more "realistic" metric to compare molecular sequences than $k$-mismatch, for instance.

# Summary

1. An **alignment** shows the **degree of similarity** (number of edit operations needed to transform one string into the other);
2. An **alignment** shows the regions of similarity or dis-similarity.

# Summary: Needleman & Wunsch (global) alignment

$$V(i,0) = i \times d, i \in 0..n$$

$$V(0,j) = j \times d, i \in 0..m$$

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + s(S_1(i), S_2(j)), \\ V(i-1,j) + d, \\ V(i,j-1) + d. \end{cases}$$

where $d$ is the cost of a deletion and $d < 0$

$\Rightarrow$ Needleman & Wunsch (1970) *J. Mol. Biol.* **48**(3):443-453.

# Summary: Semi-global alignment

$$V(0,0) = 0$$

$$V(i,0) = 0, i = 1..m$$

$$V(0,j) = 0, j = 1..n$$

$$V(i,j) = \max \begin{cases} V(i-1,j) + s(S_1(i), '-'), \\ V(i,j-1) + s('-', S_2(j)), \\ V(i-1,j-1) + s(S'_1(i), S'_2(j)). \end{cases}$$

Solution is,

$$\max_{i=1..m, j=1..n} [V(m,n), V(i,n), V(m,j)]$$

$\Rightarrow$ Two modifications: initialisation, consider the last row/column to find the optimal value.

# Summary: Smith & Waterman (local) alignment

**Base conditions**,

$$v(i,0) = 0, i \in 0..n$$

$$v(0,j) = 0, j \in 0..m$$

**General case**,

$$v(i,j) = \max \begin{cases} 0, \\ v(i-1,j) - s(S_1(i),'-'), \\ v(i,j-1) - s('-',S_2(j)), \\ v(i-1,j-1) + s(S_1(i),S_2(j)). \end{cases}$$

**Solution**,

$$v^\star = \max[v(i,j) : i \leq n, j \leq m]$$

$\Rightarrow$ Smith & Waterman (1981) *J. Mol. Biol.* **147**:195-197.

# Availability

Some of the implementations include:

- Align from the **FASTA** suite: fasta.bioch.virginia.edu
- and Needle from **EMBOSS**: www.emboss.org
- Bio**Java**, Bio**Perl**, Bio**Python**, etc.

# References


- Gusfield, D. (1997) *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge Press, pp. 215–224. (MRT General QA 76.9 .A43 G87 1997)
- Jones N.C. and Pevzner P.A. (2004) *An Introduction to Bioinformatics Algorithms*, MIT Press, pp. 147–178. (QH324.2 b.J66 2004)
- Durbin, R. *et al* (1998,2000) *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press. §2 (MRT General QP 620 .B576 1998)

# References

# Pensez-y!

L'impression de ces notes n'est probablement pas nécessaire!