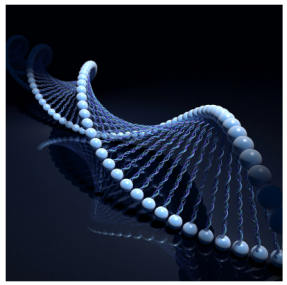
LGBIO2010: Pairwise alignment algorithms

Pierre Dupont



UCL – ICTEAM

Outline

Outline

Pairwise alignement

```
GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

Objective

Find a best way to align 2 sequences including matches, substitutions (a.k.a. mismatches) and possible gaps (insertions or deletions)

- Both sequences need not have the same length
- Scoring matrices (*e.g.* Unitary/BLAST or PAM/BLOSUM) define similarity scores between letters (*e.g.* nucleotides or amino acids)
  - The higher the score the more similar a given pair of letters
    $$s_{match} > s_{mismatch}$$
  - A gap penalty (negative score) is also defined
- Look for a pairwise alignment with maximal cumulative score (unlike DOT plots restricted to match positions!)

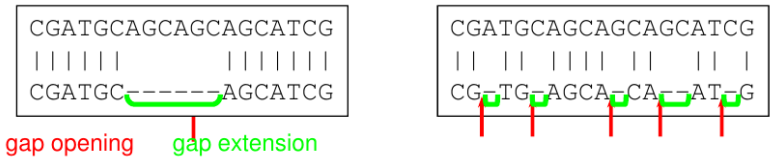
Scoring matrix

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

BLOSUM50

Illustration from Biological Sequence Analysis (© Cambridge University Press 1998)

Gap penalties



Linear gap penalty

$$\gamma(g) = -dg$$

g: number of consecutive gaps

d: gap penalty (*e.g.* 8)

Affine gap penalty

$$\gamma(g) = -d - e(g-1)$$

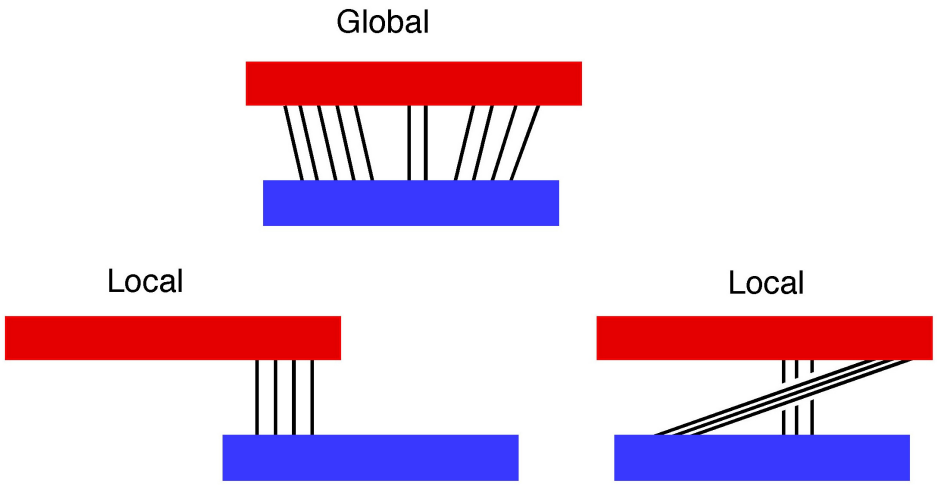
g: number of consecutive gaps

d: gap open penalty (*e.g.* 12)

e: gap extension penalty (*e.g.* 2)

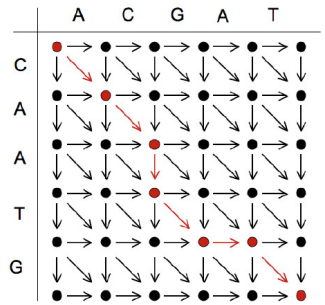
- Affine is more relevant from a biological viewpoint but more complex to compute with
- Affine reduces to linear whenever $e = d$

Global versus local alignement



Global Alignment

Input: a sequence x of length n and a sequence y of length m
Input: an appropriate scoring matrix S and a gap penalty $\gamma(g)$
Output: an alignment between x and y with maximal cumulative score



AC-GAT

CAAT-G

Example:

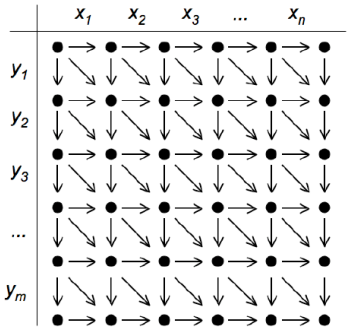
$$s(x_i, y_j) = \begin{cases} 1 & \text{if } x_i = y_j \\ 0 & \text{if } x_i \neq y_j \end{cases}$$

$$\gamma(g) = -8g$$

$$s(x, y) = 0 + 0 - 8 + 0 - 8 + 0 = -16$$

- How many such alignments exist?
- How to compute an optimal alignment?

Number of gapped alignments



k diagonal, $0 \leq k \leq \min(n, m)$

$n - k$ horizontal

$m - k$ vertical

$n + m - k$ total

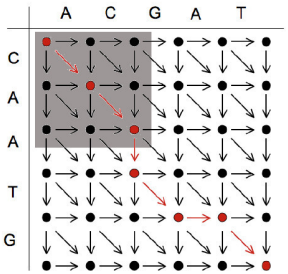
How many ways to combine k, $n - k$, and $m - k$ operations?

$$N = \sum_{k=0}^{\min(n,m)} \frac{(n+m-k)!}{k!(n-k)!(m-k)!} \approx \frac{4^n}{\sqrt{\pi n}}$$

Assuming $n \approx m$ (sound for a global alignment): $n = 300 \Rightarrow N \approx 1.10^{179}$

Outline

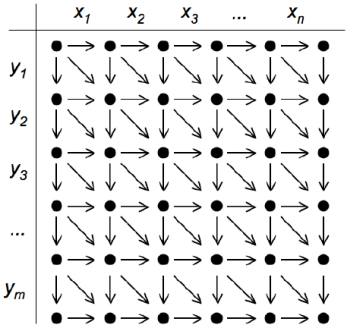
Dynamic programming



Key observations

1. we look for a maximal cumulative score
   - a sum of independent individual scores: $s(x_i, y_j)$ or gap penalties
2. an optimal global alignment between x and y is made of optimal alignments between subsequences (*e.g.* prefixes)
   - decomposition into optimal solutions to sub-problems
   - each sub-problem needs to be computed only once

Score of a partial alignment



- $F(i, j)$ cumulative score to align $x_1 \ldots x_i$ with $y_1 \ldots y_j$
- $F(0, 0) = 0$

Running example

Substitution scores with BLOSUM50 matrix

	H	E	A	G	A	W	G	H	E	E
P	−2	−1	−1	−2	−1	−4	−2	−2	−1	−1
A	−2	−1	**5**	0	**5**	−3	0	−2	−1	−1
W	−3	−3	−3	−3	−3	**15**	−3	−3	−3	−3
H	**10**	0	−2	−2	−2	−3	−2	**10**	0	0
E	0	**6**	−1	−3	−1	−3	−3	0	**6**	**6**
A	−2	−1	**5**	0	**5**	−3	0	−2	−1	−1
E	0	**6**	−1	−3	−1	−3	−3	0	**6**	**6**

Illustration from Biological Sequence Analysis (© Cambridge University Press 1998)

Alignment initialization

		H	E	A	G	A	W	G	H	E	E
	0										
P											
A											
W											
H											
E											
A											
E											

3 ways to extend a partial alignment

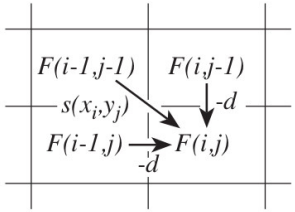
I G A x_i — L G V y_j

A I G A x_i — G V y_j — —

G A x_i — — — S L G V y_j

1. x_i aligned to y_j
2. x_i aligned to a gap
3. y_j aligned to a gap

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$



A backpointer stores which of the 3 possibilities is optimal (= the max)

Illustration from Biological Sequence Analysis (© Cambridge University Press 1998)

		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 ←	−41 ←	−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

- The table is filled from top to bottom and left to right (prefixes!)
- $F(n,m)$ = the final alignment score
- The actual alignment is found following backpointers
- Computational complexity (with $n \approx m$) in $\mathcal{O}(n^2)$ instead of $\mathcal{O}(\frac{4^n}{\sqrt{n}})$

Illustration from Biological Sequence Analysis (© Cambridge University Press 1998)

Outline

Global versus local alignment

Global

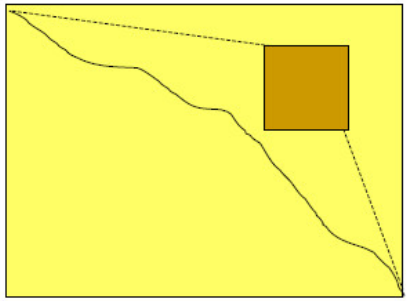
```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||     || | | | | ||||  |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

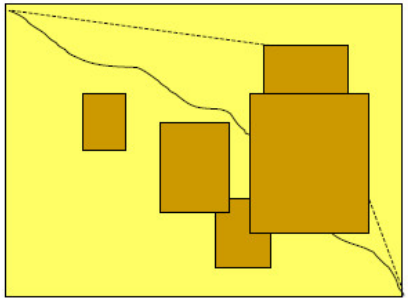
Local

```
                    tccCAGTTATGTCAGgggacacgagcatgcagagac
                       ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Look for conserved segments only

Naïve algorithm

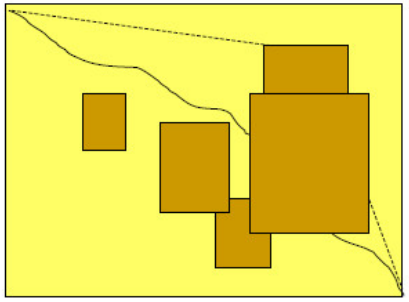




Look for a global alignement between subsequences

1. Apply NW from any starting point
2. For each starting point, look for the maximal ending score
3. Look for the maximum score overall

Computational complexity



1. Apply NW from any starting point
2. For each starting point, look for the maximal ending score
3. Look for the maximum score overall

- $\mathcal{O}(n^2)$ starting points (with $n \approx m$)
- Each NW takes $\mathcal{O}(n^2)$
- Globally $\mathcal{O}(n^4)$

Smith-Waterman

		H	E	A	G	A	W	G	H	E	E
	0										
P	0		0								
A							0				
W					0						
H									0		
E										0	
A		0									
E											

Key idea

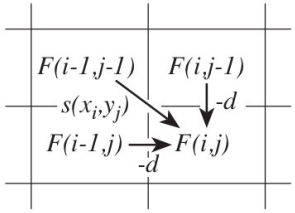
- reset to initial score = 0 from anywhere
- choose best between reset or pursuing current alignment

4 ways to extend a partial alignment

I G A x_i A I G A x_i G A x_i – –
L G V y_j G V y_j – – S L G V y_j

1. x_i aligned to y_j
2. x_i aligned to a gap
3. y_j aligned to a gap

$$F(i,j) = \max \begin{cases} \mathbf{0} \\ F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$



A backpointer stores which of the 3 last possibilities is optimal, otherwise 0

Smith-Waterman algorithm

		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

- End of the local alignment: position (i, j) such that $F(i, j)$ is maximal
- Start of the local alignment: follow backpointers till 0

Illustration from Biological Sequence Analysis (© Cambridge University Press 1998)

Smith-Waterman algorithm

		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

Requirements:

- scores $s(.,.)$ for strong mismatches (dissimilar residues) need to be < 0 (= worse than reset) otherwise long stretches of unrelated subsequences could be aligned
- score $s(.,.)$ for matching similar residues need to be > 0, otherwise 0 everywhere

Computational complexity (with $n \approx m$) in $\mathcal{O}(n^2)$

(same as Needleman-Wunsch, 11 years later!)

Outline

Semi-global alignment

		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0										
A	0										
W	0										
H	0										
E	0										
A	0										
E	0										

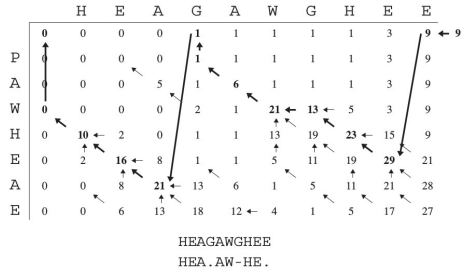
Do not penalize gaps at the beginning of either sequences

- Initialize $F(i,0) = F(0,j) = 0 \;\; ; \;\; 0 \leq i \leq n \;\; ; \;\; 0 \leq j \leq m$
- Compute the global recurrence

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

Local alignment with repeats

Look for several local matches of y into x



$$F(0,0) = 0$$

$$F(i,0) = \max \begin{cases} F(i-1,0) \\ F(i-1,j) - T \\ \forall j = 1, \ldots, m \end{cases}$$

$$F(i,j) = \max \begin{cases} \mathbf{F(i,0)} \\ F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

- Find local matches scoring higher than T (here $T = 20$)
- $F(n+1,0)$: total score of k matches $-kT$

$$9 = 49 - 40 = 21 + 28 - 2 * 20$$

Illustration from Biological Sequence Analysis (© Cambridge University Press 1998)

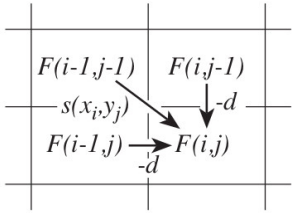
Gap penalties

So far: a linear gap penalty $\gamma(g) = -dg$

I G A x_i	A I G A x_i	G A x_i – –
L G V y_j	G V y_j – –	S L G V y_j

1. x_i aligned to y_j
2. x_i aligned to a gap
3. y_j aligned to a gap

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$



- an affine gap penalty $\gamma(g) = -d - e(g-1)$ is more relevant
- we need to distinguish whether we consider the first gap or subsequent gaps, either in x or y

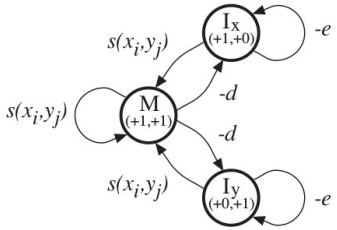
Affine gap penalty

Instead of a single state $F(i,j)$, one distinguishes 3 states

I G A x_i — L G V y_j

A I G A x_i — G V y_j — —

G A x_i — — — S L G V y_j



$$M(i,j) = \max \begin{cases} M(i-1,j-1) + s(x_i, y_j) \\ I_x(i-1,j-1) + s(x_i, y_j) \\ I_y(i-1,j-1) + s(x_i, y_j) \end{cases}$$

$$I_x(i,j) = \max \begin{cases} M(i-1,j) - d \\ I_x(i-1,j) - e \end{cases}$$

$$I_y(i,j) = \max \begin{cases} M(i,j-1) - d \\ I_y(i,j-1) - e \end{cases}$$

M state : x_i aligned to y_j
I_x state : x_i aligned to a gap
I_y state : y_j aligned to a gap

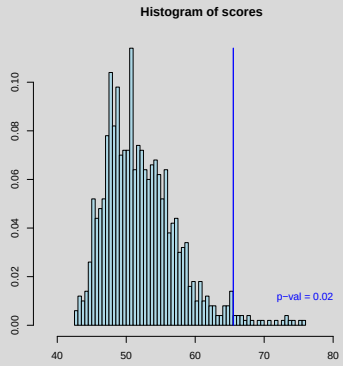
Illustration from Biological Sequence Analysis (© Cambridge University Press 1998)

Outline

Statistical significance of the alignment score

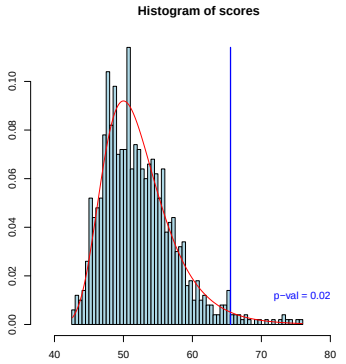
Procedure

1. Compute pairwise random alignment scores
   - between x and random sequences (*e.g.* 500 permutations of y)
   - between y and random sequences (*e.g.* 500 permutations of x)
2. Compute histogram of random alignment scores (normalized by length)
3. $p\text{-}val = 1-$ percentile of actual score between x and y



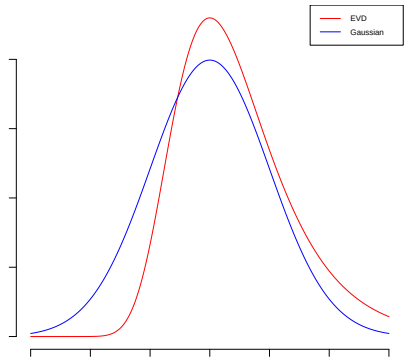
Computational complexity

- Sound procedure but requires to compute many (*e.g.* $1,000$) alignments
- Computational complexity of a single alignment: $\mathcal{O}(nm) \approx \mathcal{O}(n^2)$
- Actual distribution of alignment scores is known to follow an Extreme Value Distribution



Extreme Value Distribution

- Probability of a score larger than S
  $P(x > S) = 1 - e^{(-Kmn\, e^{-\lambda S})}$
  for some constants K and λ
- Alignment softwares include fitted values of K and λ for a wide range of substitution scoring matrices
- The lack of normality is due to non-independence between possible starting points of matches

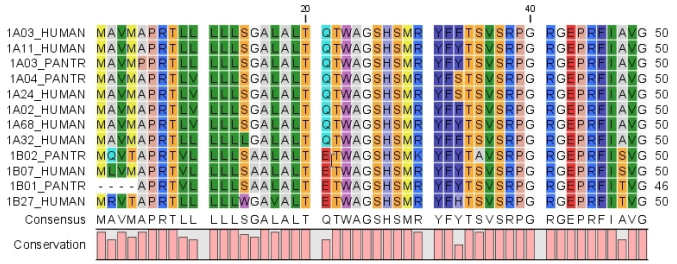


Concluding remarks

- All the above algorithms maximizing a score can be easily adapted when minimizing an edit distance or cost
- $\mathcal{O}(nm) \approx \mathcal{O}(n^2)$ is too much when aligning a query sequence to a large database of possibly homologous sequences
  - BLAST and FASTA are heuristic algorithms to speed-up such computation

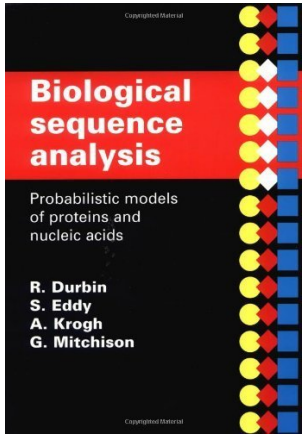
Concluding remarks (ctd.)

- Generalization to find a multiple alignment between k sequences



  - Dynamic programming scales as $\mathcal{O}(n^k)$ to find an optimal solution
  - CLUSTALW is an heuristic algorithm to speed-up such computation
- A further extension
  - summarize a multiple alignment into a probabilistic model (HMM)
  - compute an alignment between a new sequence and this model

Further Reading



Chapter 2: Pairwise alignment