Pairwise Sequence Alignment

Dynamic Programming Algorithm

A popular method for identification of conservation patterns between two genes/proteins is

- **pairwise sequence alignment**

Outcome of pairwise sequence alignment:

- **identifying regions of similarity**
- **a score which measures similarity between sequences - quantify**

Some Definitions

Alignment – process of lining up two or more sequences allowing for mismatches

```
HEAGAWGHEE
    PAWHEAE
```

- for assessing the degree of similarity and the possibility of homology
- if the same letter occurs in both the sequences then this position has been **conserved** in evolution.
- if the letters **differ** it is assumed that the two derive from an **ancestral** letter (could be one of the two or neither)

Homology – having a common ancestral origin

- proteins with similar 3D-structures

Difference between similarity and homology:

- Similarity is simply a measure of expression how alike two sequences are
- Homology means there is an **evolutionary relationship** between two sequences - there are no degrees of homology.
- Extending this to individual residues they are ‘identical’ or ‘similar’ residues - similar implies that they **share certain physicochemical properties**
- Homology cannot be observed, it is only an **inference**

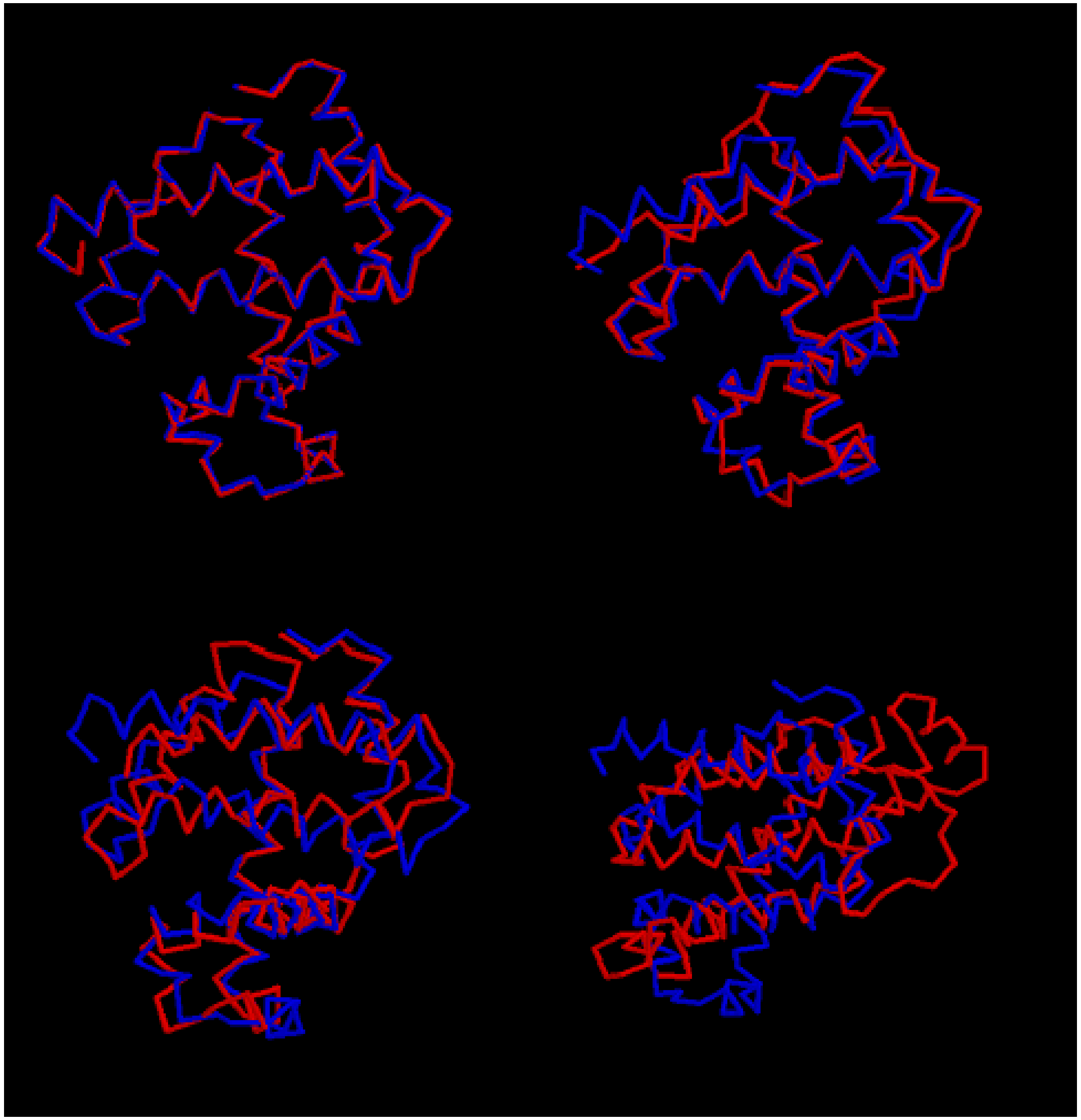
Difference between similarity and homology

Identical protein sequences result in identical 3-D structures - similar sequences may result in similar structures, and this is usually the case.

Converse is not true: identical 3-D structures do not necessarily indicate identical sequences. It is because of this that there is a distinction between “homology” and “similarity”.

There are examples of proteins in the databases that have nearly identical 3-D structures, and are therefore homologous, but do not exhibit significant (or detectable) sequence similarity

Sequence identity and rmsd of Sperm Whale myoglobin



myoglobin
pig

rmsd = 0.5 Å
id = 86%

haemoglobin
pig

rmsd = 1.5 Å
id = 28%

globin-3
P. piclitum

rmsd = 2.2 Å
id = 18%

phycocyanin
F. diplosiphon

rmsd = 3.3 Å
id = 8%

Comparison of Sequences

The main objective of sequence alignment is

- to identify regions of similarity, i.e., *conserved regions*

- to find out if the two sequences are related or not

- this would enable us to extrapolate knowledge of the known sequence to the unknown query sequence

Any other reasons for Sequence Comparison?

Comparison of Sequences

Any other reasons for Sequence Comparison?

- **Identifying species – as in the case of DNA barcoding**
- **Phylogenetic analysis – to find evolutionary relatedness between species**
- **Genome comparison between individuals in a population – for structural variation analysis**
- **Genome comparison between diseased (e.g., cancer) and normal cells – for identifying variations responsible for the disease**
- **Genome comparison between species – for understanding genome evolution**

Inferring Function from Similarity

Biology

Zoology

Geology

Thanatology ?

Spectrology

Botany

Linguistics

Physics

Chemistry

Inferring Function from Similarity

Biology – study of life

Zoology – study of animals

Geology – study of earth

Thanatology – study of death

Spectrology – study of visible light/radiative energy as function of its wavelength/frequency

Botany - study of plants

Linguistics - study of languages

Physics – study of matter & its motion thru space & time

Chemistry - study of matter & energy & interactions between them

Basic task

Most basic sequence analysis task is - to find out if the two sequences are related or not, i.e.,

– to decide whether the alignment is more likely to have occurred because they are related, or just by chance ?

The other point to consider is – to use DNA (gene) or protein sequences?

Example

For the sequences gctgaacg and ctataatc:

An uninformative alignment:

```
- - -  - - -  - - g c t g a a c g
c t a t a a t c  - - - - - - - - -
```

An alignment **without** gaps:

```
g c t g a a c g
c t  a t a a t c
```

An alignment **with** gaps:

```
g c t g a - a - - c g
-  - c t  - a t a a t c
```

An alternative alignment:
with gaps

```
g c t g  - a a  - c g
-  c t a t a a t c -
```

We also need to compute a score reflecting the quality of each alignment.

Key issues are:

- **what sort of alignment to be considered**
- **the scoring system to rank alignments**
- **the algorithm to find optimal scoring alignments**
- **statistical methods to evaluate the significance of scores**

Scoring scheme used is most crucial
- minor variations in scoring scheme may change the ranking of alignments, causing a different one to emerge as the best.

Complexity of the Problem

Consider three pairwise alignments, all to same region of human alpha globin protein sequence (HBA_Human):

- **human beta globin (HBB_Human)**
- **leghaemoglobin from yellow lupin (LGB2 LUPLU)**
- **nematode glutathione S-transferase (F11G11.2)**

Complexity of Problem

(a)

I=18, S=17, G=0

```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
        →  G+ +VK+HGKKV  A+++++AH+D++ +++++LS+LH  KL
HBB_HUMAN  GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL
```

both hydrophilic

(b)

I=8, S=17, G=4

```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
        →  ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

(c)

I=13, S=12, G=6

```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD----LHAHKL
        →  GS+ + G +   +D L  ++ H+ D+  A +AL D    ++AH+
F11G11.2   GSGYLVGDSLTFVDLL--VAQHTADLLAANAALLDEFPQFKAHQE
```

Complexity of the Problem

Challenge:

How to distinguish cases like (b) from those like (c)?

One needs to carefully choose the scoring system to evaluate alignments.

Even then it may not always be possible to distinguish true alignments from spurious ones.

e.g., it is extremely difficult to find significant similarity between lupin leghaemoglobin and human alpha globin (Alignment (b)) by pairwise alignments

The Scoring Model

When comparing two sequences one is looking for evidence that they have diverged from a common ancestor by a process of mutation or natural selection

Basic mutational processes are:

substitution – which change residues in a sequence

insertions and deletions – which add or remove residues, together referred as gaps or indels

Further apart two sequences are from each other, more frequent these changes are expected to occur.

The Scoring Model

Mutations potentially affect the function of the gene, which can either be beneficial, or lead to reduction in functionality and adaptability of the protein.

Natural selection comes into play – allowing mutations that are either evolutionarily advantageous or, occur in non-functional regions of the sequence

Natural selection has the effect of screening the mutations – some changes are seen more often than others

- results in a mosaic pattern of conserved & unconserved regions, the functional regions being conserved across evolutions

$\Rightarrow$ Some identities, Ala-Ala score lower than others, Trp-Trp

The Scoring Model

Total score of an alignment

- sum of the terms for each aligned pair of residues, plus terms for each gap

Identities and conservative substitutions
- contribute a positive score, while

non-conservative changes and gaps
- accumulate negative scores

Similarity-based scoring scheme

The Scoring Model

Additive scoring scheme corresponds to an assumption that - mutations at different sites in a sequence occurred independently (treating gap of arbitrary length as a single mutation)

- a reasonable approximation for DNA and protein sequences, though interactions between residues play a critical role in determining protein structure.

This assumption is inaccurate for structural RNAs where base pairing introduces long-range dependencies

Substitution Matrices

The scoring system is composed of a substitution matrix (4×4 for DNA, 20×20 for Proteins) and gap penalties

- **entries in substitution matrix reflect the likelihood of a substitution from one nucleic acid (amino acid) base to another during the course of evolution,**

e.g., how often is Ala replaced by Val, Gly, etc.

i.e., the frequency of Ala-Val substitution at a given evolutionary distance

The Scoring Model

What is the probability that the alignment observed is a true alignment and not by chance?

In probabilistic interpretation – this corresponds to the logarithm of the relative likelihood that the sequences are related, compared to being unrelated

- assign a probability to the alignment in each of the two cases and then consider the ratio of the two probabilities

Substitution Matrix

Random or unrelated model *R*

- assumes that symbol *a* occurs independently with some frequency q_a, hence the probability of two sequences is just the product of the probabilities of each nucleic/amino acid:

$$P(x, y \mid R) = \prod_i q_{x_i} \prod_j q_{y_j}$$

using the multiplication rule of probabilities

Substitution Matrix

Match model *M* - aligned pairs occur with a joint probability p_{ab}. Thus, the probability for the whole alignment:

$$P(x, y \mid M) = \prod_{i} p_{x_i y_i}$$

Odds-ratio - the ratio of these two likelihoods :

$$\frac{P(x, y \mid M)}{P(x, y \mid R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

Substitution Matrix

To have an **additive scoring system**, take logarithm of this ratio, to obtain **log-odds ratio:**

$$S = \sum_i s(x_i, y_i) \qquad s(a,b) = \log \frac{p_{ab}}{q_a q_b}$$

s(*a*,*b*) is the log likelihood ratio of the residue pair (*a*,*b*) occurring as an <u>aligned</u> pair, as opposed to an <u>unaligned</u> pair.

– these individual scores for each pair of residues are arranged in a matrix, called *scoring matrix* or *substitution matrix*

Gap Penalties

Gaps need to be penalized

The standard cost associated with a gap of length *g* is given either by a **linear score**

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

or, an **affine score**

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

$\gamma(g) = ?$
for $d = 8$, $e = 2$,
$g = 10$

d – gap-open penalty, *e* – gap-extension penalty;

$e < d$ - allows long insertions and deletions to be penalized less than they would be by the linear gap cost

Why?

Gap Penalties

An affine score assumes that consecutive deletions/insertions are a single mutation event as opposed to multiple insertions/deletions and hence should be penalized less.

Why not assign a long gap the penalty of a single gap?

Dynamic Programming Algorithm

Aim: Given a scoring system, to have an algorithm for finding an optimal alignment for a pair of sequences.

For two sequences of length n , there are

$$\begin{bmatrix} 2n \\ n \end{bmatrix} = \frac{(2n)!}{(n!\,)^2} \approx \frac{2^{2n}}{\sqrt{\pi n}}$$

possible global alignments!

Not computationally feasible

For a sequence of length 100, $2^{2n} \approx 10^{30}$, for n=1000, it is 10^{300}

Dynamic Programming Algorithm

Algorithm for finding optimal alignments given an additive alignment score is called the dynamic programming

The scoring scheme being introduced as a log-odds ratio

- *better* alignments will have *higher* score

So the aim is to *maximize* the score

When scores are assigned as *costs* or *edit distances*, the aim is to *minimize* the score

Dynamic programming algorithms apply to either case

- the differences are trivial changes of ‘min’ for ‘max’

Dynamic Programming Algorithm

Consider the following two amino acid sequences:

HEAGAWGHEE and PAWHEAE

To score the alignments, consider

Substitution scoring matrix: **BLOSUM50**

Gap penalty, **d = 8**

BLOSUM50 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**

Global Alignment

It’s end-to-end alignment of two sequences, allowing gaps, e.g., consider two sequences:

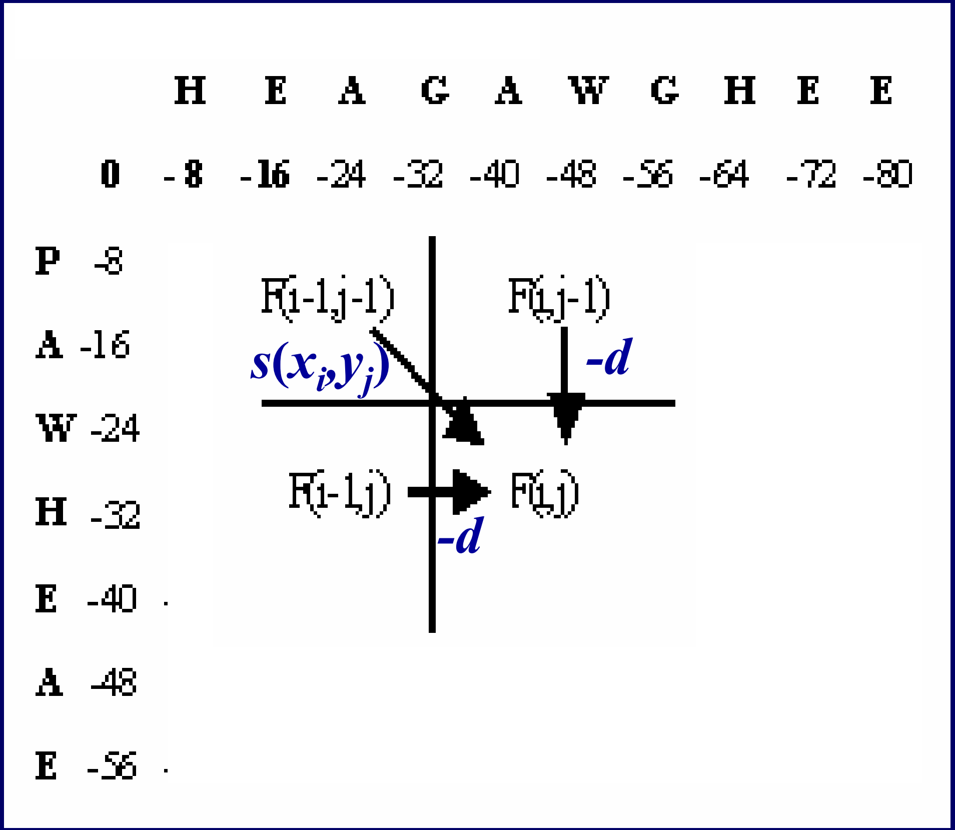
HEAGAWGHEE and PAWHEAE

Alignment:

```
HEAGAWGHE – E
– – P – AW – HEAE
```

is called a global alignment

The dynamic programming algorithm for obtaining global alignment of biological sequences is called the Needleman-Wunsch algorithm



Initialize: $F(0, 0)=0$

Boundary conditions:

$F(i, 0) = F(0, i) = -id,\ d = 8$

Score of the pair (P, H) = ?

If $F(i-1, j-1)$, $F(i-1, j)$ and $F(i, j-1)$ are known, it is possible to calculate $F(i, j)$

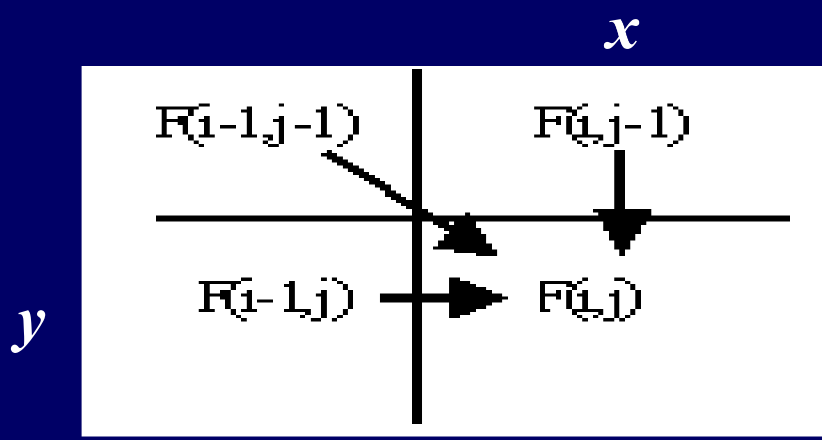
The idea is to build up an optimal alignment using previous solutions for optimal alignments of smaller subsequences

Global Alignment: Needleman-Wunsch Algorithm

$$\begin{array}{llll} \mathrm{I\ G\ A}\ x_i & \qquad \mathrm{A\ I\ G\ A}\ x_i & \qquad \mathrm{G\ A}\ x_i\ -\ - \\ \mathrm{L\ G\ V}\ y_j & \qquad \mathrm{G\ V}\ y_j\ -\ - & \qquad \mathrm{S\ L\ G\ V}\ y_j \end{array}$$

- to align a letter from the horizontal sequence, x_i, with a letter from the vertical sequence, y_j:

$$F(i, j) = F(i\text{-}1, j\text{-}1) + s(x_i, y_j)$$



Global Alignment: Needleman-Wunsch Algorithm

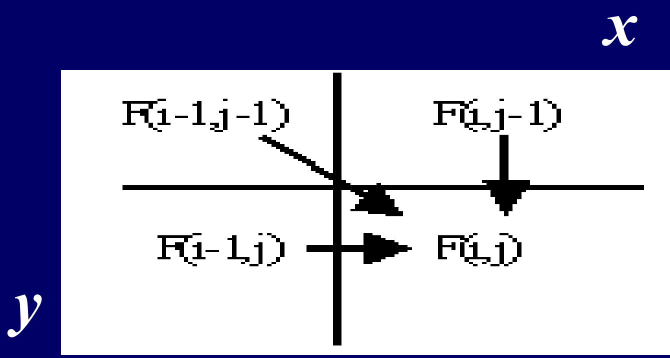
$$\begin{array}{llll} \mathtt{I\;G\;A}\;x_i & \qquad & \mathtt{A\;I\;G\;A}\;x_i & \qquad & \mathtt{G\;A}\;x_i\;-\;- \\ \mathtt{L\;G\;V}\;y_j & & \mathtt{G\;V}\;y_j\;-\;- & & \mathtt{S\;L\;G\;V}\;y_j \end{array}$$

- to align a letter from the horizontal sequence, x_i, against a gap in the vertical sequence; in which case

$$F(i, j) = F(i\text{-}1, j) - d$$

- to align a gap from the horizontal sequence against a letter in the vertical sequence, y_j, in which case

$$F(i, j) = F(i, j\text{-}1) - d$$



Global Alignment: Needleman-Wunsch Algorithm

The best score up to *(i, j)* will be the largest of these three options:

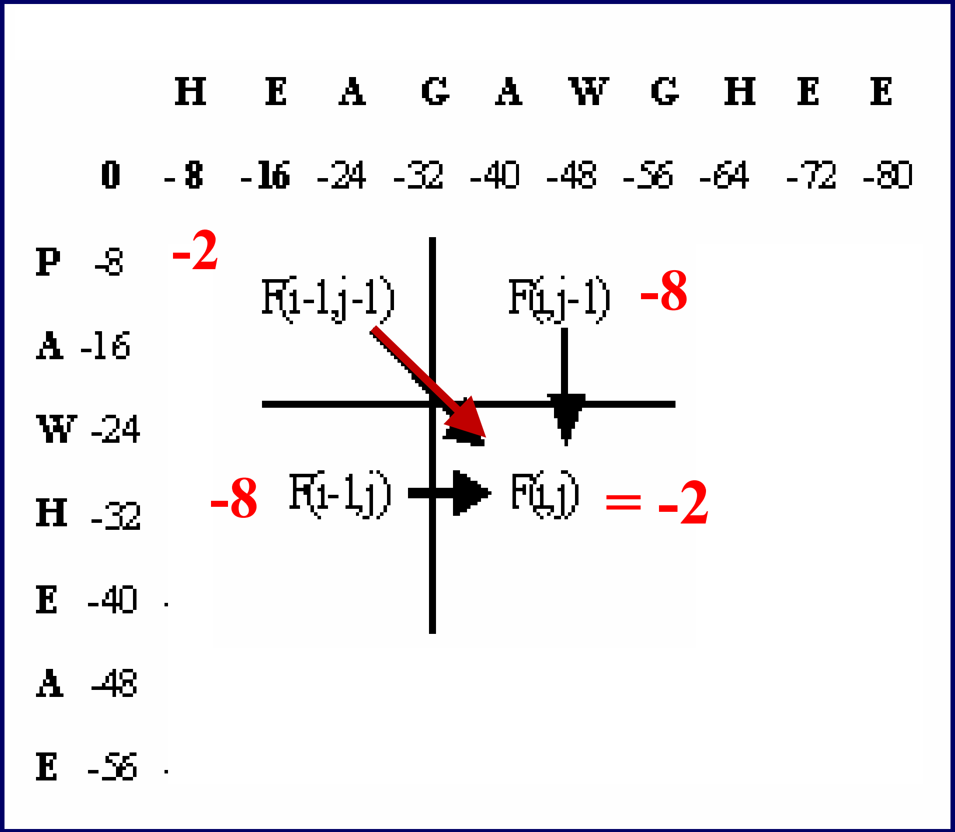
$$F(i,j) = max\begin{cases} F(i-1,j-1)+s(x_i,y_j) \\ F(i-1,j)-d \\ F(i,j-1)-d \end{cases}$$

This equation is applied repeatedly to fill in the matrix of *F*(*i*, *j*) values

A pointer is kept in each cell back to the cell from which its *F*(*i*, *j*) was derived.

Boundary conditions: *F*(*i*, 0) = - *id*, *F*(0, *j*) = - *jd*

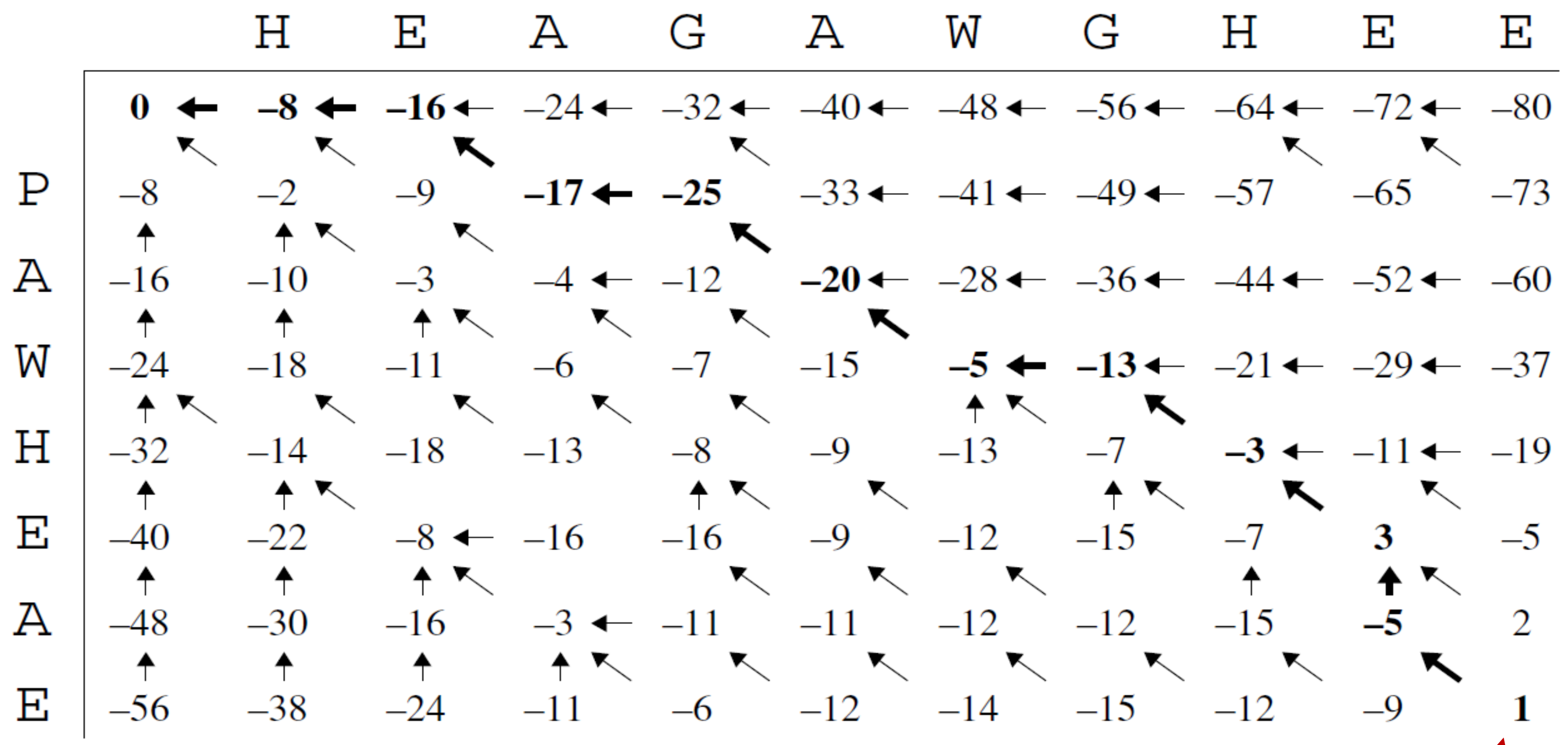
Score of pair (P, H) = -2



$F(i, j) = -2$, and the direction is diagonal

$F(i, 0) = F(0, j) = -8$

Global Alignment: Needleman-Wunsch Algorithm



		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
```

Score

Global Alignment: Needleman-Wunsch Algorithm

Final cell of the matrix, $F(n, m)$ gives the best score for global alignment of $x_{1\ldots n}$ to $y_{1\ldots m}$

To find the alignment itself, we must find the path of choices that led to this final value. The procedure for doing this is known as traceback.

It works by building the alignment in reverse, starting from the final cell, and following the pointers stored when building the matrix.

Global Alignment: Needleman-Wunsch Algorithm

Algorithmic complexity:

NW algorithm takes $O(nm)$ time & $O(nm)$ memory, where n & m are lengths of the two sequences

i.e., the computer time and memory storage required scales as the product of sequence lengths

Can this algorithm be used for comparing genomes?

Local Alignment

A common situation is where one is looking for the best alignment between subsequences of *x* and *y*.

This arises for e.g., when it is suspected that two protein sequences may share a common domain, or when comparing extended sections of genomic DNA.

It is also the most sensitive way to detect similarity when comparing two very highly diverged sequences sharing common evolutionary origin, or,

when one has no knowledge about divergence, as in database search.

Local Alignment

The highest scoring alignment of subsequences of x and y is called the best *local* alignment,

e.g., consider the sequences:

HEAGAWGHEE and PAWHEAE

```
Local Alignment:        AWGHE

                        AW- HE
```

Dynamic programming algorithm for finding optimal local alignment in biological sequences is the Smith-Waterman algorithm

Local Alignment: Smith-Waterman Algorithm

There are **two differences** for finding optimal local alignment, compared to global alignment

First, in each cell, an extra possibility is added, allowing $F(i, j)$ to take the **value 0** if all other options have value less than **0**:

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

Option **0** corresponds to starting a **new alignment** anywhere in the sequence.

Local Alignment: Smith-Waterman Algorithm

A consequence of 0 is that the boundary conditions are:

$$F(i, 0) = 0,\ F(0, j) = 0$$

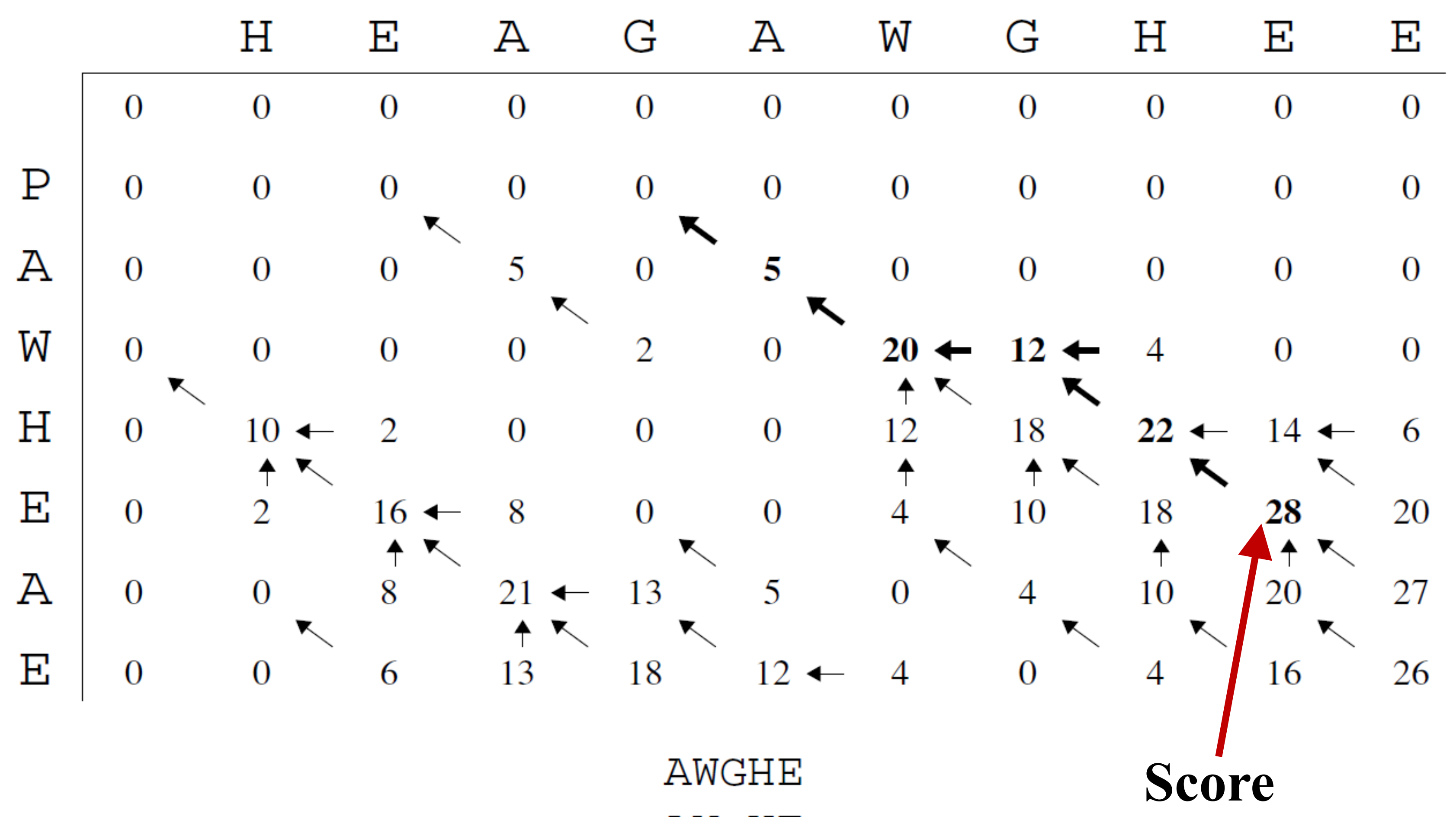
instead of $-id$ and $-jd$ as for the global alignment.

Second change is that now an alignment can end anywhere in the matrix

For this, look for the highest value of $F(i, j)$ in the whole matrix to start the traceback, instead of the (n, m) cell.

Traceback ends when a cell with value ‘0’ is encountered.

Local Alignment: 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
```

Tools - EMBOSS

Global Alignment:

- **needle** – uses Needleman-Wunsch global alignment algorithm (including gaps).

Local Alignment:

- **water** – uses Smith-Waterman algorithm (modified for speed enhancements) to calculate local alignment.
- **matcher** - compares two sequences looking for local sequence similarities using a rigorous algorithm, based on Bill Pearson's 'lalign'

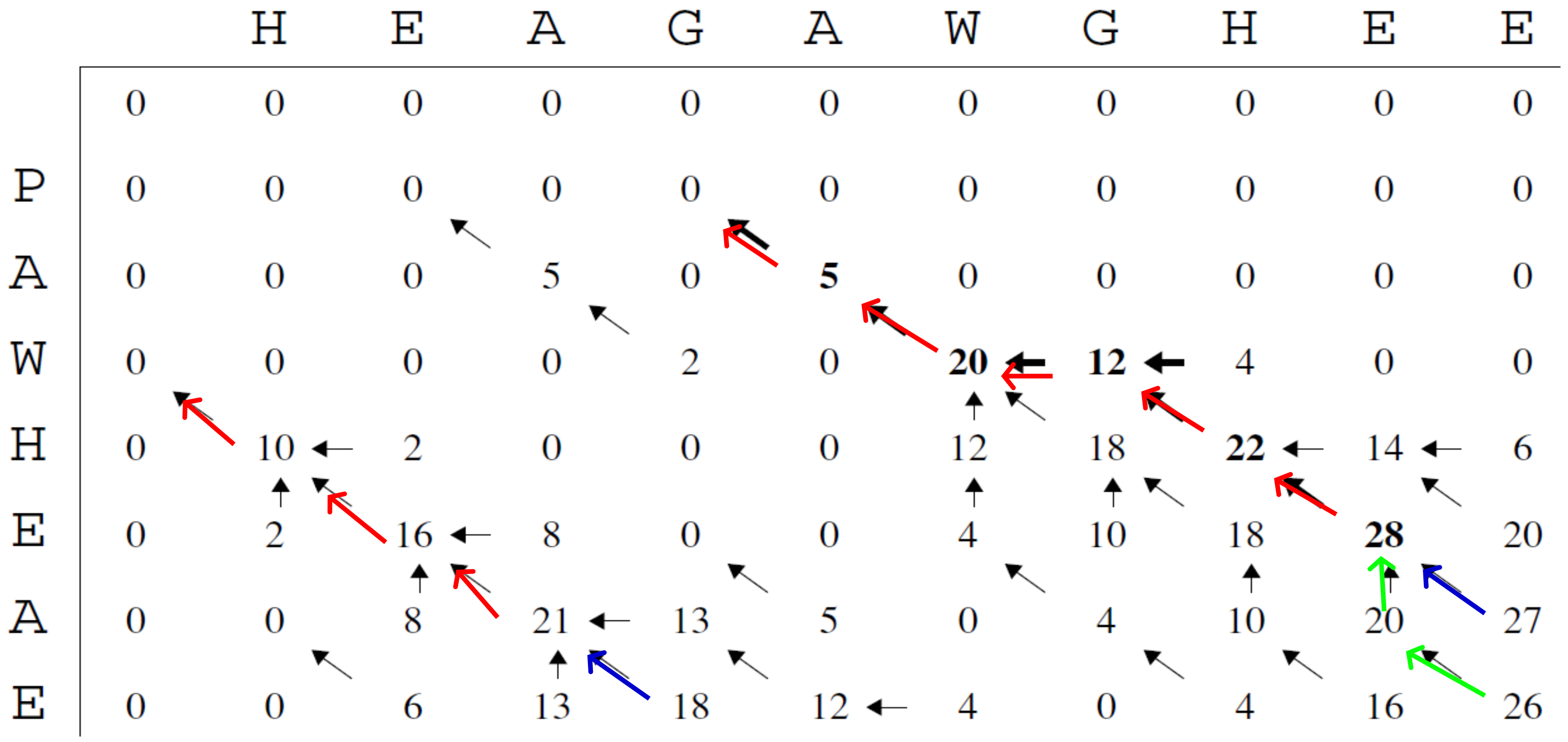
Suboptimal/Repeated Matches

Smith-Waterman algorithm - gives the best local match between two sequences

If one or both the sequences are long, it is possible to have many different local alignments with a significant score, and one may be interested in all of these

Example - many copies of a repeated domain or motif in a protein, multi-domain proteins, distantly related sequences can have more than one conserved regions

Local Alignment: 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

HEAG
HEAE

```
AWGHE
AW-HE
```

AWGHEE
AW - HEA

AWGHE - E
AW - HEAE

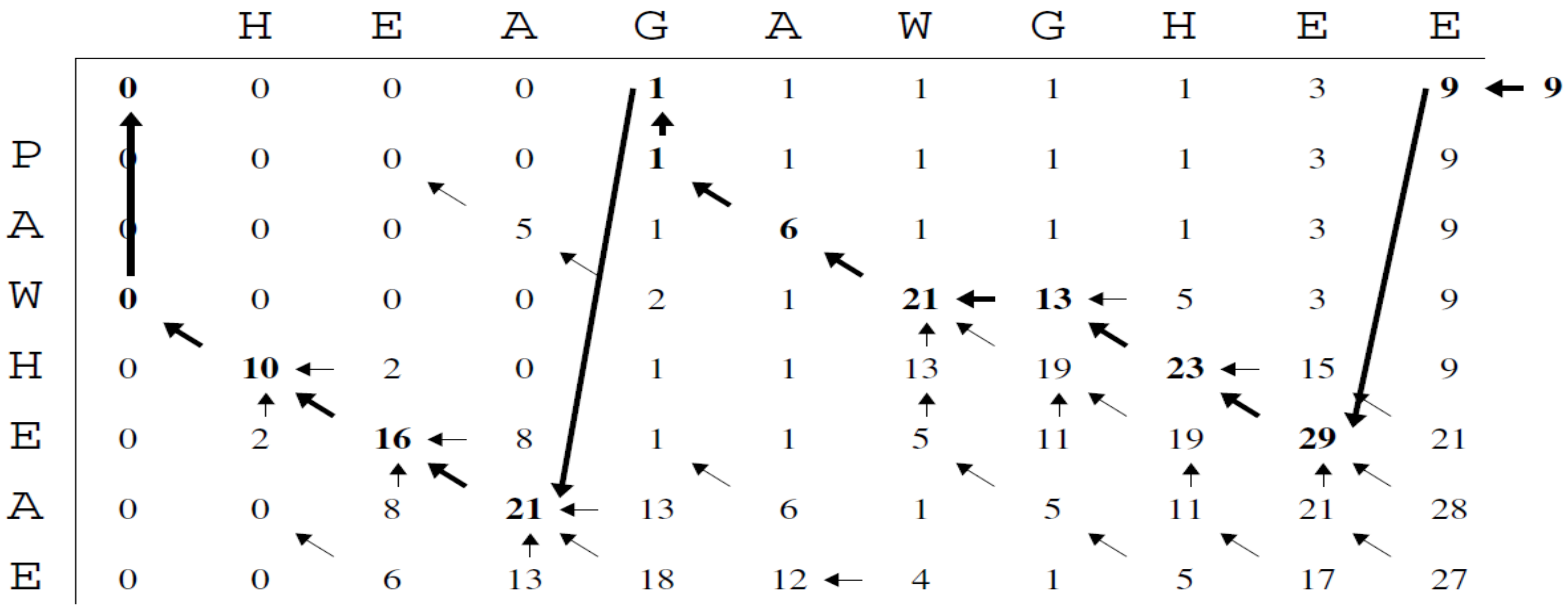
Suboptimal Matches

Suboptimal matches are obtained by taking a traceback not only from the maximum scoring cell, but from other high-scoring cells ($\geq T$)

All other conditions for local alignment hold true in this case.

Some of the high-scoring alignments may be overlapping – one may output only non-overlapping ones

Suboptimal Matches



		H	E	A	G	A	W	G	H	E	E	
	0	0	0	0	**1**	1	1	1	1	3	**9**	← **9**
P	0	0	0	0	**1**	1	1	1	1	3	9	
A	0	0	0	5	1	**6**	1	1	1	3	9	
W	**0**	0	0	0	2	1	**21** ←	**13** ←	5	3	9	
H	0	**10** ←	2	0	1	1	13	19	**23** ←	15	9	
E	0	2	**16** ←	8	1	1	5	11	19	**29**	21	
A	0	0	8	**21** ←	13	6	1	5	11	21	28	
E	0	0	6	13	18	12 ←	4	1	5	17	27	

```
HEAGAWGHEE
HEA.AW-HE.
```

T = 20

There are two separate match regions, with scores 1 and 8
Dots are used to indicate unmatched regions.

Suboptimal Matches

Initial Conditions: $F(0,0) = 0$

Boundary Conditions: $F(0,j) = 0$

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

- handles unmatched regions and ends of matches, only allowing matches to end when they have score at least T

Recurrence Relations:

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

- handles starts of matches and extensions

Suboptimal Matches

Total score of all the matches is obtained by adding an extra cell to the matrix, $F(n+1,0)$, using eq. (i)

This score will have T subtracted from each match; if there were no matches of score greater than T it will be **0** [repeated application of 1st option in (i)]

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

Suboptimal Matches

EMBOSS Programs:

- **matcher:** Rigorous Smith-Waterman alignment
- **supermatcher:** Finds a match of a large sequence against one or more sequences
- **wordmatch:** Finds all exact matches of a given size between two sequences

Overlap Matches

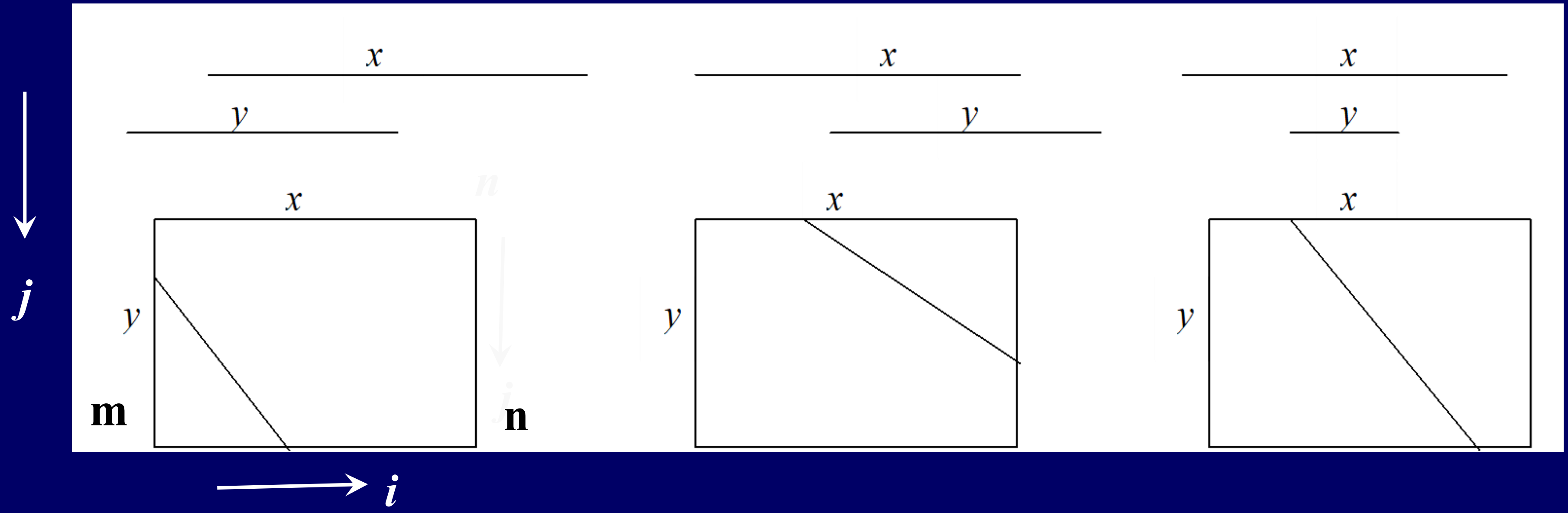
Another important situation commonly encountered with biological sequences is -

- one sequence contained in the other
- have common overlapping regions.

e.g., when comparing fragments of genomic DNA sequence to each other, or to large chromosomal sequences, in sequence assembly

Several different types of configurations can occur

Overlap Matches



Start: F(0, *j*), or F(*i*, 0)

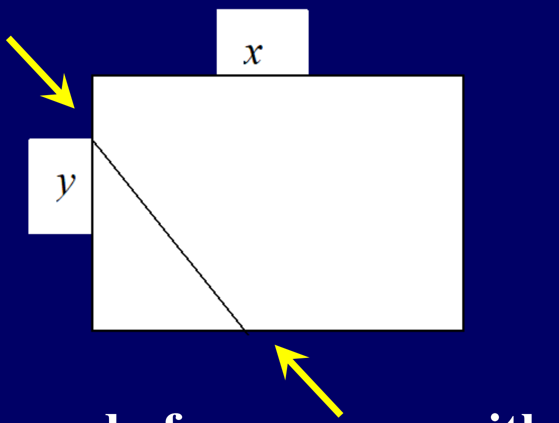
End: F(*i*, *m*), or F(*n*, *j*)

i: 1, … *n*, *j*: 1, … *m*

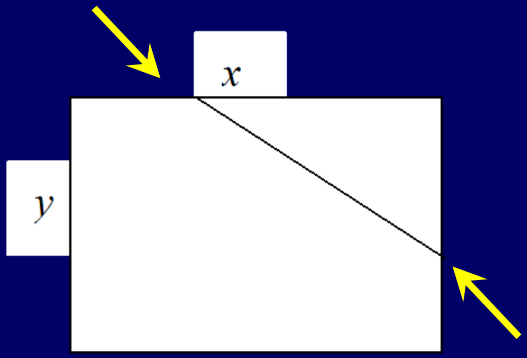
Overlap Matches

This is a special a case of global alignment that does not penalize **overhanging ends – also called semi-global alignment.**

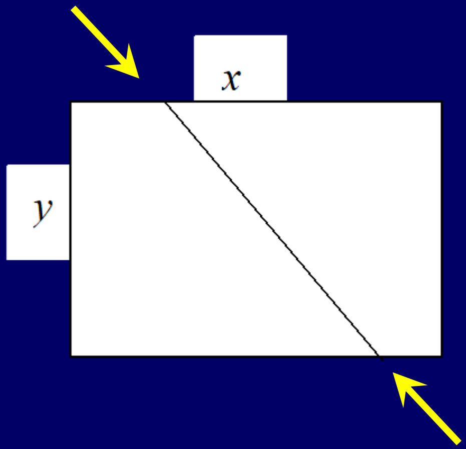
i.e., an algorithm for a match that may start on the **left / top border** of the matrix, and end on the **right / bottom border.**



end of sequence y with start of sequence x



start of y with end of x



y contained in x

Overlap Matches

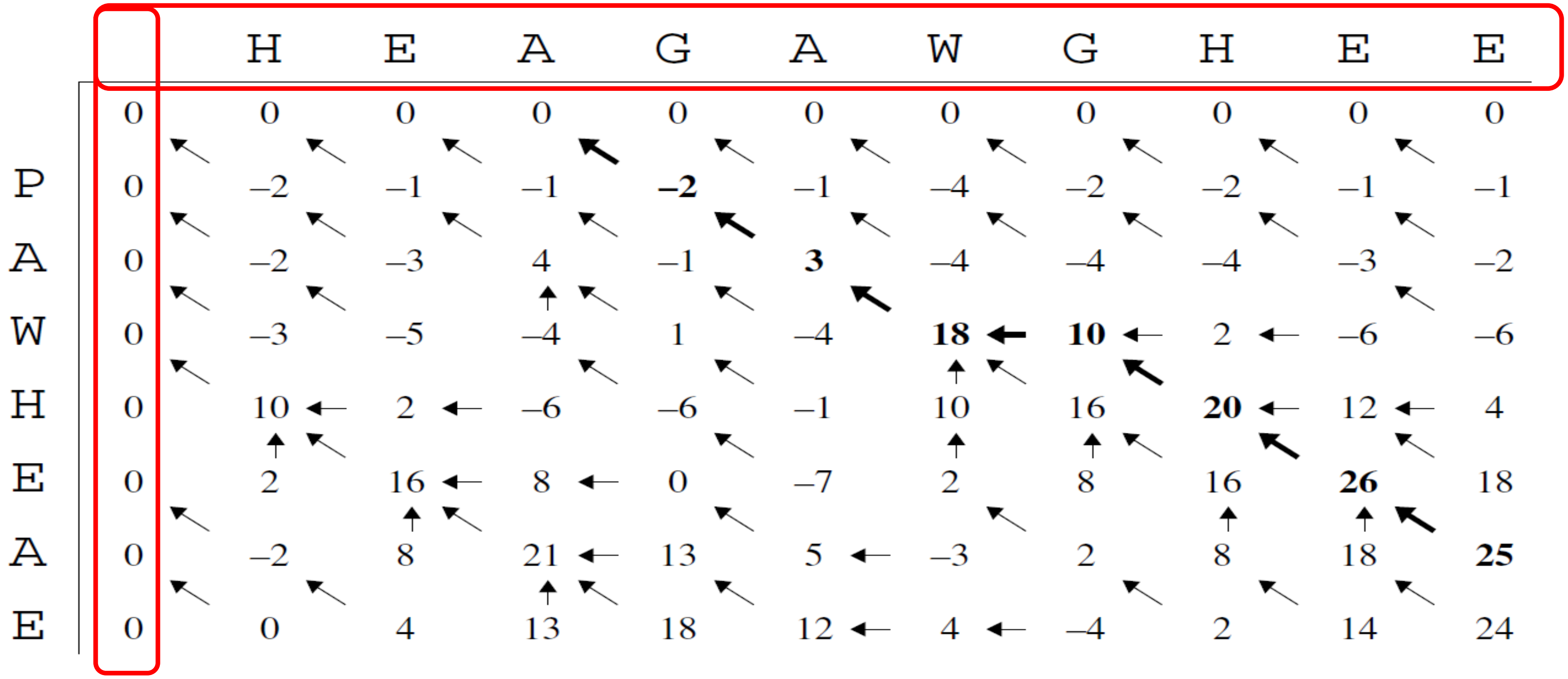
Initialization equations: (same as for <u>local</u> alignment)

$$F(0,0) = 0$$
$$F(i,0) = 0 \qquad i = 1,\ldots n$$
$$F(0,j) = 0 \qquad j = 1,\ldots m$$

Recurrence relations: (same as for <u>global</u> alignment)

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

Overlap Alignment



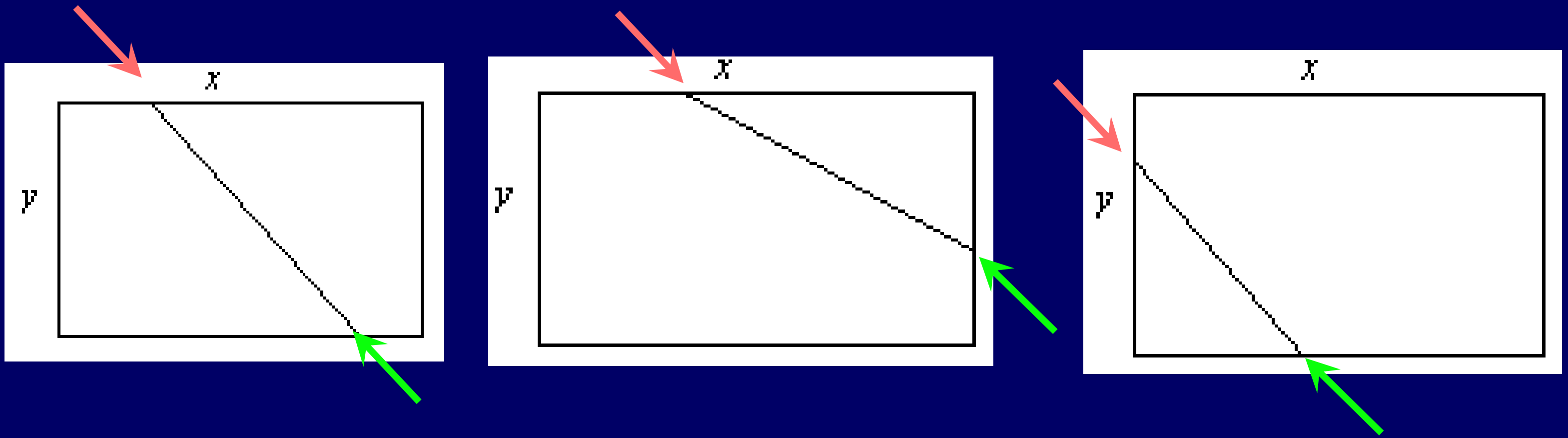
		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	–2	–1	–1	**–2**	–1	–4	–2	–2	–1	–1
A	0	–2	–3	4	–1	**3**	–4	–4	–4	–3	–2
W	0	–3	–5	–4	1	–4	**18**	**10**	2	–6	–6
H	0	10	2	–6	–6	–1	10	16	**20**	12	4
E	0	2	16	8	0	–7	2	8	16	**26**	18
A	0	–2	8	21	13	5	–3	2	8	18	**25**
E	0	0	4	13	18	12	4	–4	2	14	24

```
GAWGHEE
PAW-HEA
```

Note the boundary conditions

Overlap Matches

F_{max} - the maximum value on the bottom border (i, m), $i = 1, \ldots, n$, or the right border (n, j), $j = 1, \ldots, m$.



Traceback starts from F_{max} and continues until the top $(i, 0)$ or left $(0, j)$ edge is reached.

Overlap Matches

EMBOSS programs:

est2genome: Align EST & genomic DNA sequences

merger: Merges two overlapping sequences

megamerger: Same as merger for large sequences

Tandem Repeats

Consider a situation where a repetitive sequence *y* is found in tandem copies not separated by gaps, then the recurrence relations for filling the path matrix are:

Initial & Boundary conditions:

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

$$F(i,0) = 0$$
$$F(0,j) = 0$$

$$F(i,1) = \max\begin{cases} F(i-1,0) + s(i,1) \\ F(i-1,m) + s(i,1) \leftarrow ? \\ F(i-1,1) - d \\ F(i,0) - d \end{cases}$$

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

Similar to global alignment

Allows a bypass of $-T$ penalty so the threshold applies only once to each tandem cluster of repeats, not once to each repeat

Tandem Repeats

EMBOSS:

etandem: Finds tandem repeats in nucleotide sequence

equicktandem: Finds tandem repeats up to a specified
size, the results can be used to run
etandem on the candidate repeat lengths

einverted: Finds DNA inverted repeats using DP

palindrome: Finds DNA inverted repeats

When comparing sequences we should ideally always consider

- what types of match we are looking for
- choose the appropriate algorithm accordingly
- choose an appropriate scoring matrix & gap penalties $\longleftarrow$

Alignment with Affine Gap Scores

Affine gap cost is given by

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

While using this gap cost we need to **keep track of multiple values for each pair of residue coefficients (i, j)** in place of single value $F(i, j)$

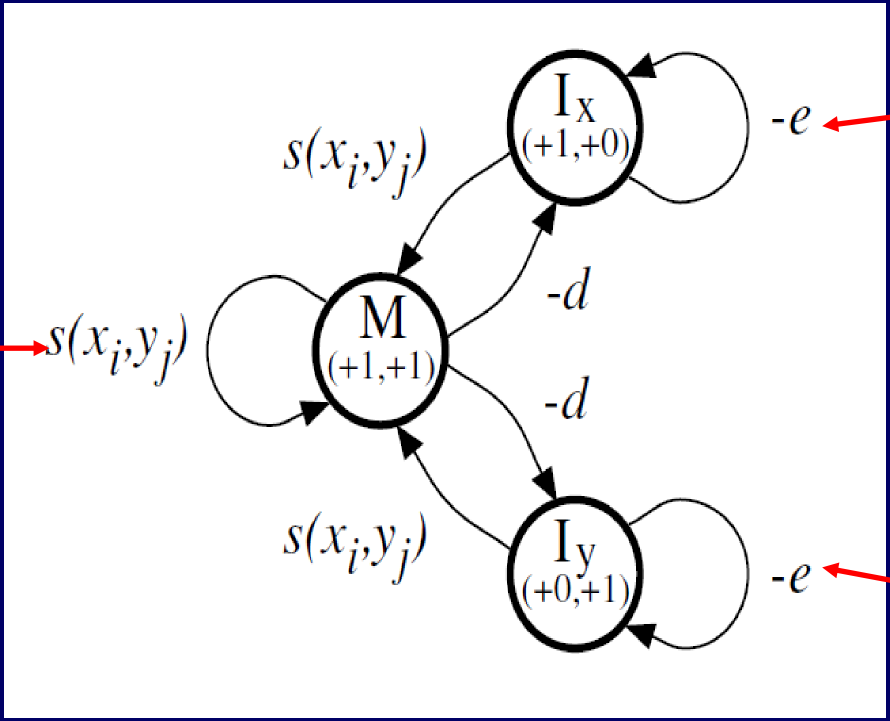
I G A x_i	A I G A x_i	G A x_i - -
L G V y_j	G V y_j - -	S L G V y_j

$M(i, j)$ - score up to (i, j), x_i aligned to y_j (L)

$I_x(i, j)$ - best score given that x_i aligned to a gap in y (C)

$I_y(i, j)$ - best score given that y_j aligned to a gap in x (R)

Alignment with Affine Gap Scores



x_i aligned to consecutive gaps in y

x_i aligned to y_j

y_j aligned to consecutive gaps in x

a finite state automaton model representing relationships between the three states used for affine gap alignment

Alignment with Affine Gap Scores

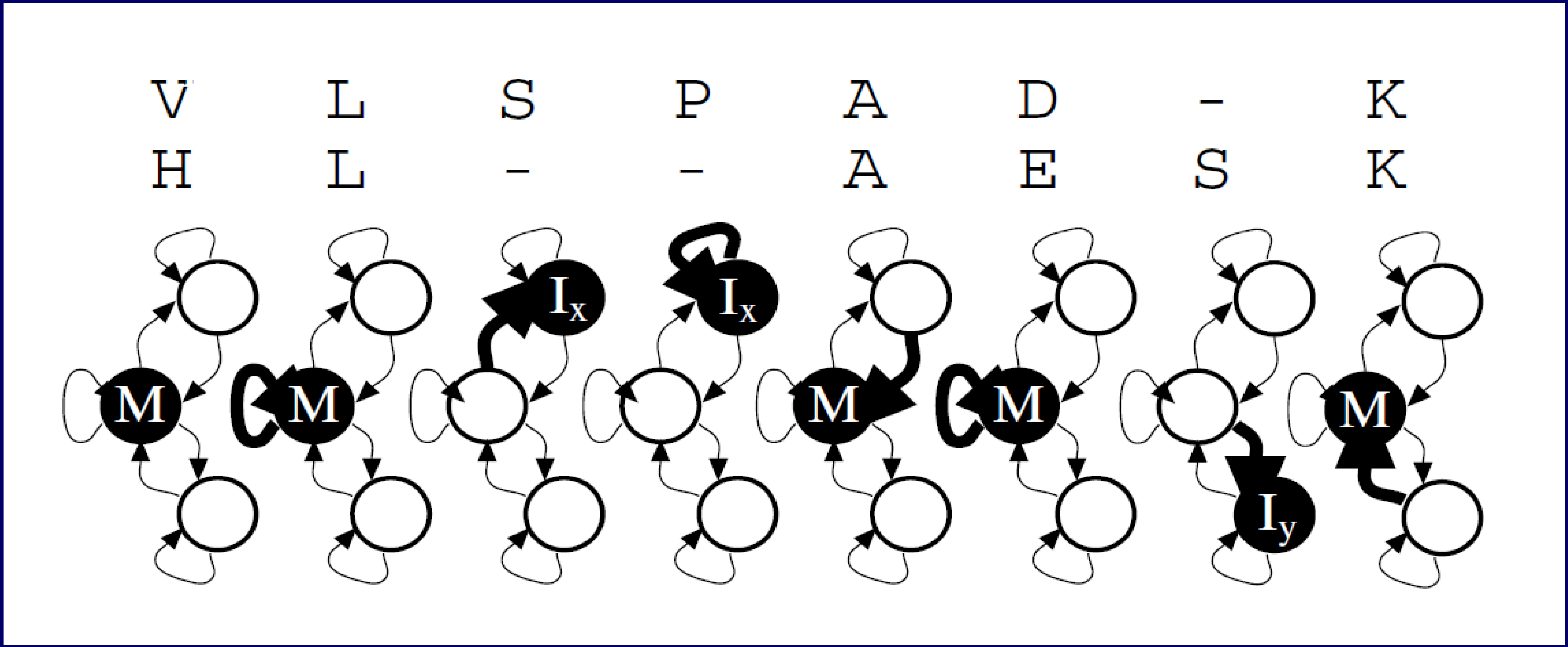
Recurrence relations (global alignment):

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

Assumption: a deletion will not be followed directly by an insertion (true for the optimal path if – d – e is less than the lowest mismatch score)

Affine gap versions provide the most sensitive sequence matching methods

State Assignments for an Alignment with Affine Gap Scores



Complexity of Dynamic Program

It is of order – O(nm), where n, m are the length of the two sequences.

Not feasible for comparing complete genomes or chromosomes ~ a few Mbs long

- Space complexity needs to be addressed

In database search, a query sequence of length n is searched a database of size ~ few Gbs

- Time complexity is an issue in this case

References

(1) Biological Sequence Analysis, R. Durbin,S. Eddy, A.Krogh, G.Mitchison, Cambridge University Press

Chap-2: Pairwise Alignment