Pairwise Sequence Alignment

Jiří Kléma

Department of Computer Science,
Czech Technical University in Prague

Lecture based on Mark Craven's class at University of Wisconsin

http://cw.felk.cvut.cz/wiki/courses/b4m36bin/start

Overview

- Pairwise sequence alignment
  - the algorithmic task,
  - broader biological motivation
    - relationship among sequence similarity, homology and function,
    - types of DNA changes during evolution,
    - success stories in DNA sequence comparisons,
  - types of alignment, issues in sequence alignment,
- what is needed to score an alignment?
  - substitution costs, gap penalty,
- optimal solution
  - dynamic programming,
  - time and space complexity for different task modifications.

Pairwise alignment: task definition

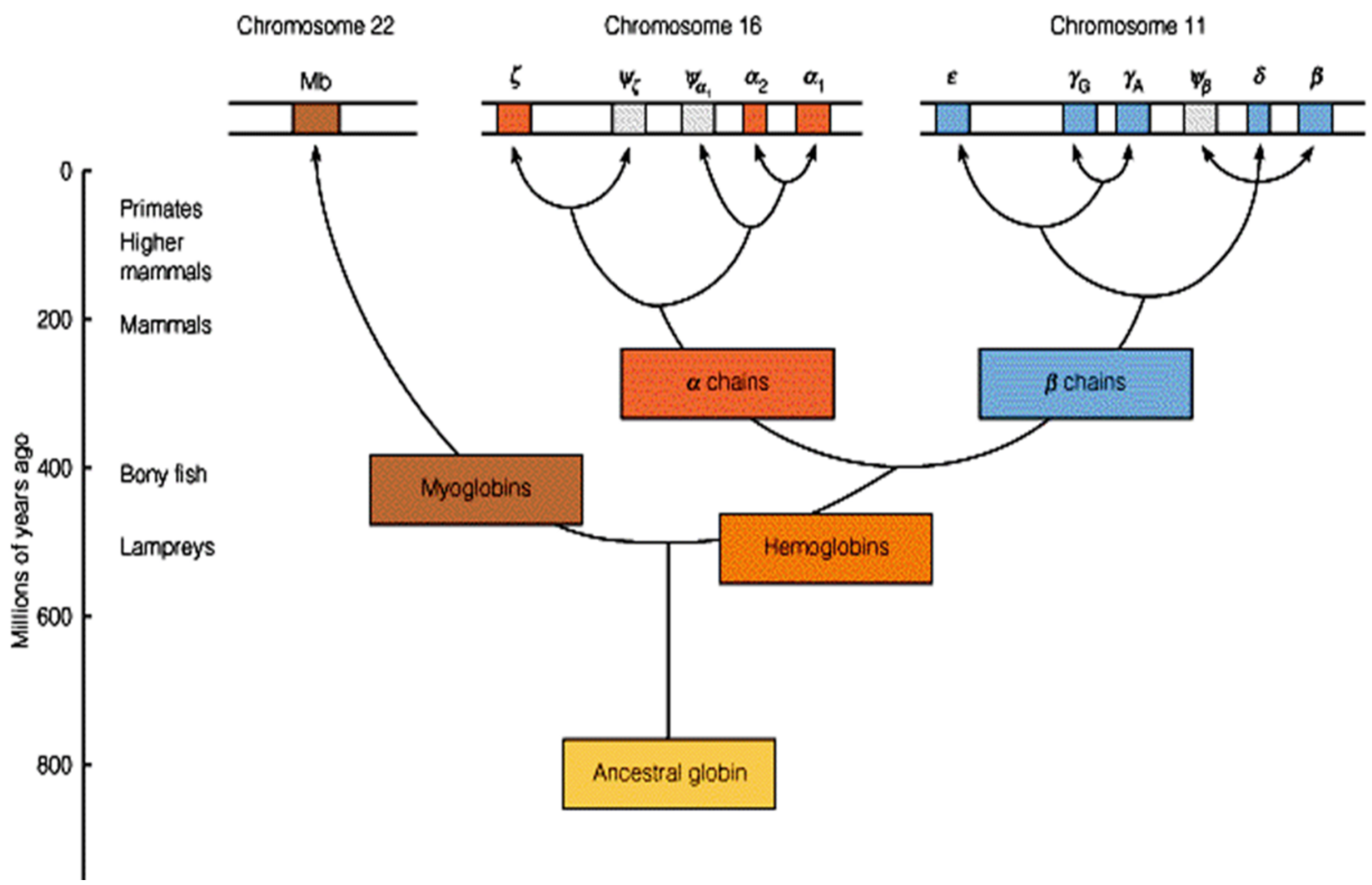
- Given
  - a pair of sequences (DNA or protein),
  - a method for scoring a candidate alignment,
- do
  - determine the correspondences between substrings in the sequences such that the similarity score is maximized,
- example

```
orig=align1    align2    align3
ACACT          ACACT     ACACT
AAT            A-A-T     AA--T
```

The role of homology in alignment

- **Homology**
  - similarity due to descent from a common ancestor,
  - orthologous sequences
    - sequences that differ because they are found in different species with a common ancestor (e.g. human α-globin and mouse α-globin),
  - paralogous sequences
    - sequences that differ because of a gene duplication event (e.g. human α-globin and human β-globin, various versions of both),
  - homologous sequences have similar structures, and frequently, they have **similar functions**,
- often we can infer homology from similarity
  - if two sequences share more similarity than expected by chance,
- thus we can sometimes infer structure/function from sequence similarity.

Homology example: evolution of the globins



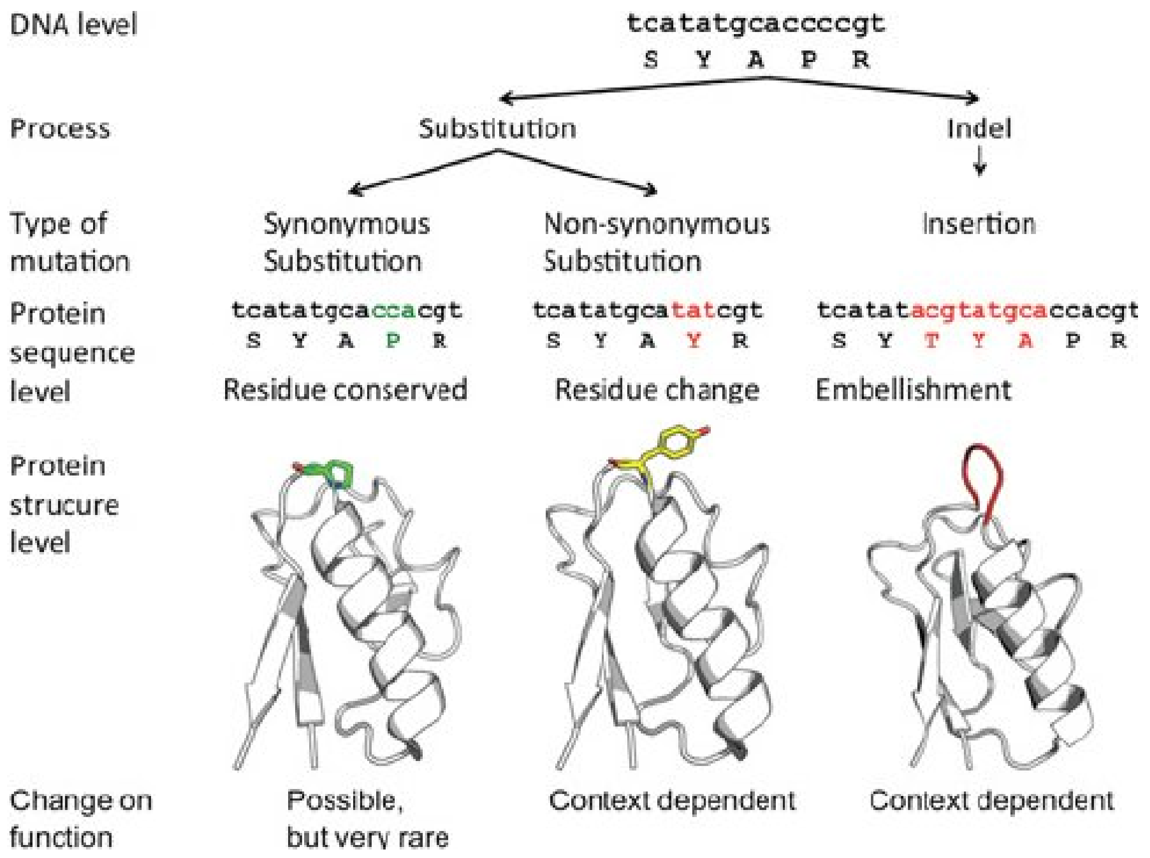
Marc Craven, BMI/CS 576, www.biostat.wisc.edu/bmi576.

DNA sequence edits

- gene scale (short DNA sequences)
  - substitutions: **ACGA** $\rightarrow$ **AGGA**,
  - insertions: **ACGA** $\rightarrow$ **ACCGGAGA**,
  - deletions: **ACGGAGA** $\rightarrow$ **AGA**,
- genome scale (long DNA sequences)
  - transpositions: **ACGGAGA** $\rightarrow$ **AAGCGGA**,
  - inversions: **ACGGAGA** $\rightarrow$ **ACTCCGA**,
- in this lecture we will focus on the case of short sequences.

Insertions/deletions and protein structure

- Why is it that two "similar" sequences may have large insertions/deletions?
  - some insertions and deletions may not significantly affect the structure of a protein.



Studer et al.: Residue mutations and their impact on protein structure and function, 2013.

Issues in sequence alignment

- the sequences typically differ in length,
- there may be only a relatively small region in the sequences that matches,
- some amino acid pairs are more substitutable than others,
- variable length regions may have been inserted/deleted from the common ancestral sequence,
- different **types of alignment** could be considered
  - global: find best match of both sequences in their entirety,
  - local: find best subsequence match,
  - semi-global: find best match without penalizing gaps on the ends of the alignment.

Scoring an alignment: what is needed?

- Substitution matrix S
  - $s(a,b)$ indicates score of aligning character a with character b,
  - either DNA substitution matrices
    - DNA is less conserved than protein sequences,
    - less effective to compare coding regions at nucleotide level,
  - or amino acid substitution matrices
    - PAM (Point Accepted Mutation),
    - BLOSUM (BLOcks SUbstitution Matrix),
- gap penalty function w
  - $w(g)$ indicates cost of a gap of length g,
  - the simplest case is when a linear gap function is used
    - $w(g) = -g \times d$, where d is a constant,
    - we will start by considering this case, the simplest alignment algorithms.

Substitution score

- Two aligned sequences: $X = x_1 \dots x_n$, $Y = y_1 \dots y_n$,
- null hypothesis = random model R
  - X and Y are unrelated (not homologous), independent residues

$$P(X,Y|R) = P(X|R)P(Y|R) = \prod_i p(x_i) \prod_i p(y_i)$$

- alternative hypothesis = match model M
  - each pair of aligned residues has a common (unknown) ancestor,
  - $p(x_i, y_i)$ = prob that x_i and y_i evolved from a common original residue

$$P(X,Y|M) = \prod_i p(x_i, y_i)$$

- odds ratio = the strength of match

$$\frac{P(X,Y|M)}{P(X,Y|R)} = \prod_i \frac{p(x_i, y_i)}{p(x_i)p(y_i)}$$

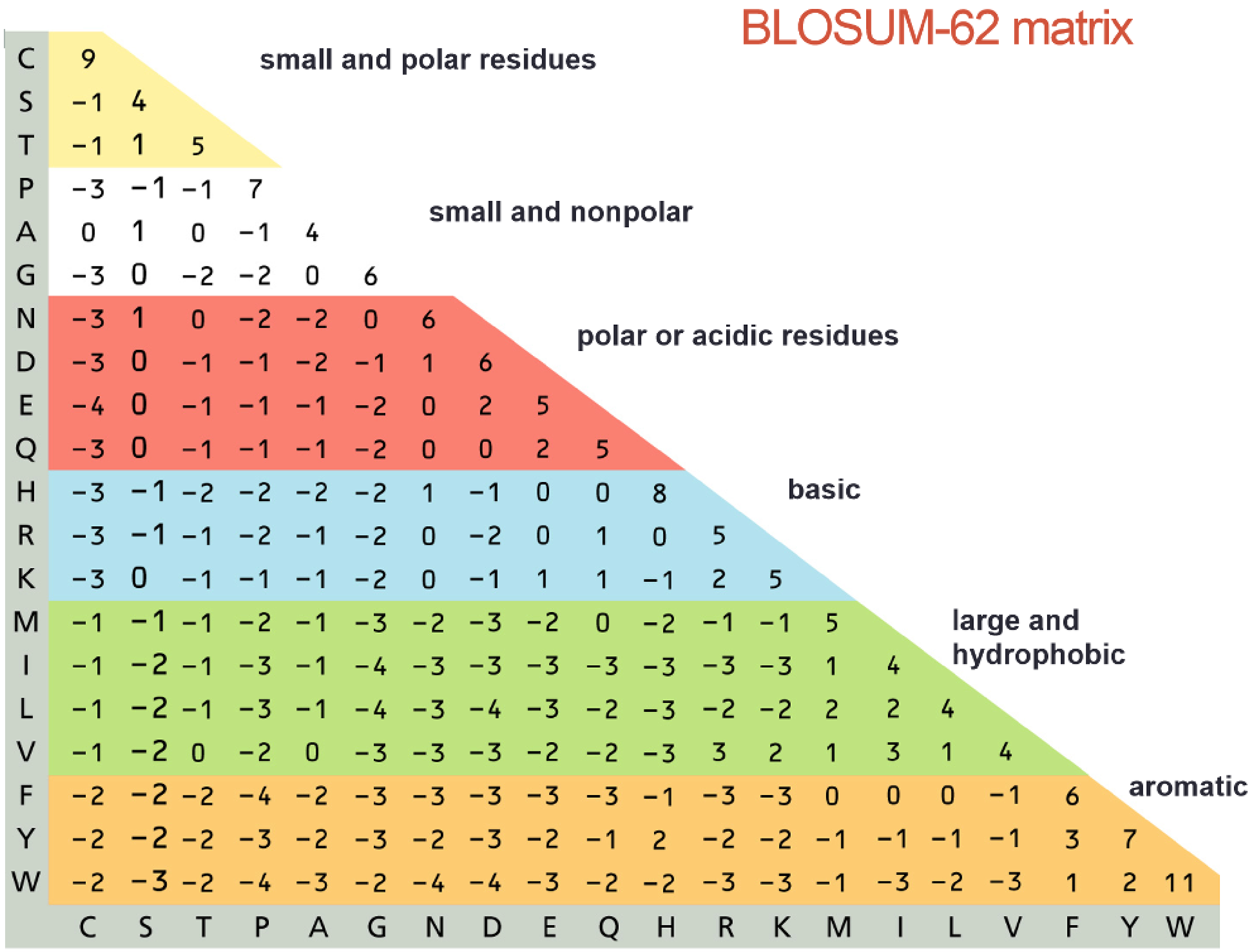
Substitution score

- Key issues
  - the score has to be additive,
  - how to guess the (joint) probabilities,
- log-odds ratio = an additive scoring scheme

$$log\frac{P(X,Y|M)}{P(X,Y|R)} = \sum_i log\frac{p(x_i,y_i)}{p(x_i)p(y_i)}$$

- BLOSUM matrices
  - guess the probs from BLOCKS database,
  - blocks are multiply aligned ungapped segments corresponding to the most highly conserved regions of proteins,
  - for each possible pair of amino acids the frequency $f(a_i,a_j)$ of common pairs in all columns is determined.

BLOSUM-62 substitution matrix

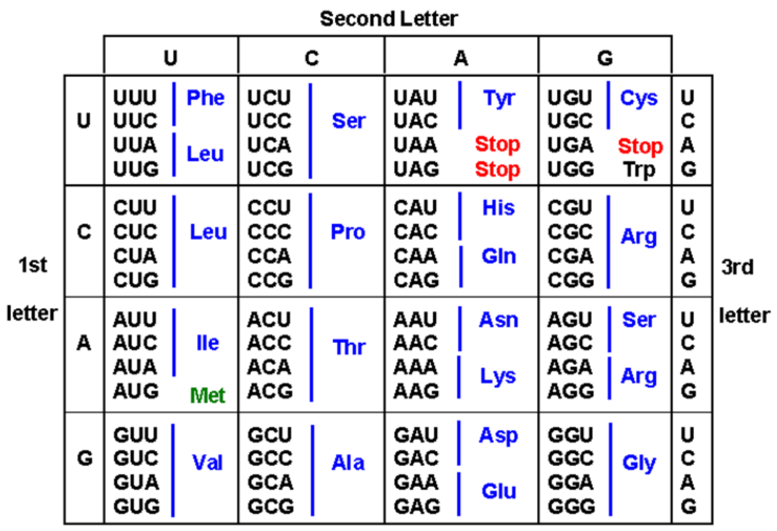

BLOSUM-62 matrix
small and polar residues
small and nonpolar
polar or acidic residues
basic
large and hydrophobic
aromatic

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	9																			
S	−1	4																		
T	−1	1	5																	
P	−3	−1	−1	7																
A	0	1	0	−1	4															
G	−3	0	−2	−2	0	6														
N	−3	1	0	−2	−2	0	6													
D	−3	0	−1	−1	−2	−1	1	6												
E	−4	0	−1	−1	−1	−2	0	2	5											
Q	−3	0	−1	−1	−1	−2	0	0	2	5										
H	−3	−1	−2	−2	−2	−2	1	−1	0	0	8									
R	−3	−1	−1	−2	−1	−2	0	−2	0	1	0	5								
K	−3	0	−1	−1	−1	−2	0	−1	1	1	−1	2	5							
M	−1	−1	−1	−2	−1	−3	−2	−3	−2	0	−2	−1	−1	5						
I	−1	−2	−1	−3	−1	−4	−3	−3	−3	−3	−3	−3	−3	1	4					
L	−1	−2	−1	−3	−1	−4	−3	−4	−3	−2	−3	−2	−2	2	2	4				
V	−1	−2	0	−2	0	−3	−3	−3	−2	−2	−3	3	2	1	3	1	4			
F	−2	−2	−2	−4	−2	−3	−3	−3	−3	−3	−1	−3	−3	0	0	0	−1	6		
Y	−2	−2	−2	−3	−2	−3	−2	−3	−2	−1	2	−2	−2	−1	−1	−1	−1	3	7	
W	−2	−3	−2	−4	−3	−2	−4	−4	−3	−2	−2	−3	−3	−1	−3	−2	−3	1	2	11

https://www.chegg.com/

Substitution score vs genetic code

- Substitution scores
  - derived empirically from an alignment,
  - reflect physico-chemical and other AA properties,
  - example of a high substitution score (2)
    - ∗ E – glutamic acid (Glu) and D – aspartic acid (Asp),
    - ∗ related genetic codes, similar properties and function
    - ∗ (acidic, polar, the most hydrophilic pair of AAs).



1st letter	Second Letter U		Second Letter C		Second Letter A		Second Letter G		3rd letter
U	UUU	Phe	UCU		UAU	Tyr	UGU	Cys	U
	UUC		UCC	Ser	UAC		UGC		C
	UUA	Leu	UCA		UAA	Stop	UGA	Stop	A
	UUG		UCG		UAG	Stop	UGG	Trp	G
C	CUU		CCU		CAU	His	CGU		U
	CUC	Leu	CCC	Pro	CAC		CGC	Arg	C
	CUA		CCA		CAA	Gln	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU		ACU		AAU	Asn	AGU	Ser	U
	AUC	Ile	ACC	Thr	AAC		AGC		C
	AUA		ACA		AAA	Lys	AGA	Arg	A
	AUG	Met	ACG		AAG		AGG		G
G	GUU		GCU		GAU	Asp	GGU		U
	GUC	Val	GCC	Ala	GAC		GGC	Gly	C
	GUA		GCA		GAA	Glu	GGA		A
	GUG		GCG		GAG		GGG		G

https://pediaa.com/how-to-find-amino-acid-sequence/

Scoring an alignment

- The score of an alignment is the sum of the scores for pairs of aligned characters plus the scores for gaps
- example: given the following alignment

V A H V – – – D – – D M P N A L S A L S D L H A H K L

A I Q L Q V T G V V V T D A T L K N L G S V H V S K G

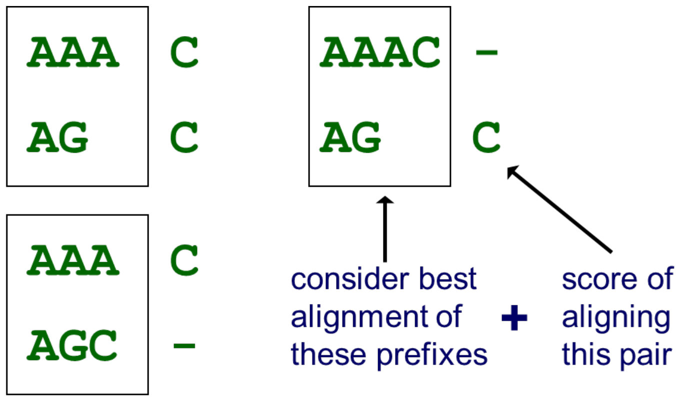
- we would score it by

$$s(V,A)+s(A,I)+s(H,Q)+s(V,L)–3d+s(D,G)–2d+\ldots,$$

- the number of possible alignments
  - can we find the highest scoring alignment by enumerating all possible alignments and picking the best?
  - no, there are $\binom{n+m}{n}$ alignments, for $n=m$ it is $\approx \frac{2^{2n}}{\sqrt{\pi n}}$

Pairwise alignment via dynamic programming

- dynamic programming
  - recursive decomposition of a complex problem into smaller subproblems,
  - gradually determine best alignment of all prefixes of the sequences,
  - guaranteed to find the optimal scoring alignment.
- the basic idea
  - consider last step in computing alignment of **AAAC** with **AGC**,
  - three possible options; in each a different pairing for end of alignment added to best alignment of previous characters.

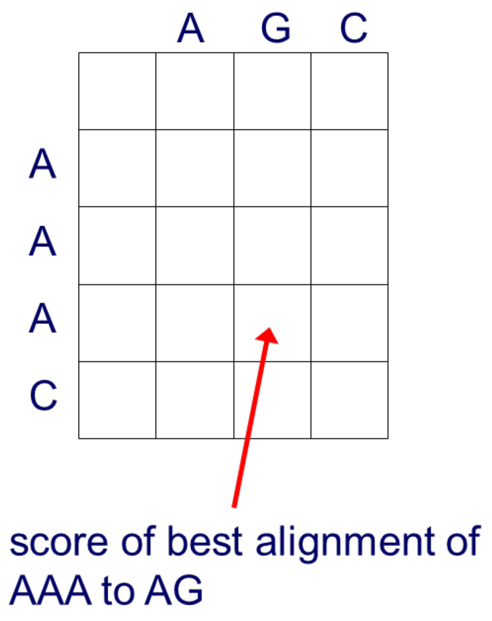

AAA C
AG C
AAAC -
AG C
AAA C
AGC -
consider best alignment of these prefixes
+
score of aligning this pair

Marc Craven, BMI/CS 576, www.biostat.wisc.edu/bmi576.

Dynamic programming: implementation for linear penalty

- First proposed by Needleman & Wunsch, Journal of Molecular Biology, 1970,
- given two unaligned sequences: $X = x_1 \dots x_n$, $Y = y_1 \dots y_m$,
- construct an (n+1) $\times$ (m+1) matrix F,
- $F(i,j) =$ score of the best alignment of X[1. . . i] with Y[1. . . j],

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

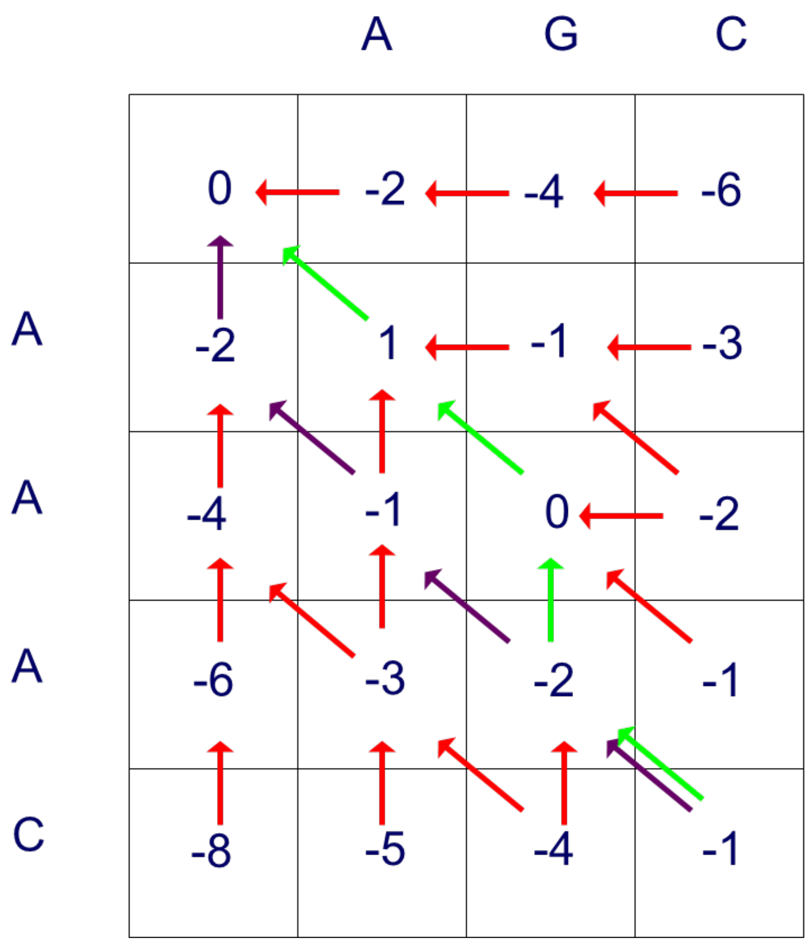


Marc Craven, BMI/CS 576, www.biostat.wisc.edu/bmi576.

DP algorithm sketch: global alignment

- initialize first row and column of F matrix
  - gradual alignment with gaps, knowledge of w is sufficient,
- fill in rest of matrix from top to bottom, left to right,
- for each $F(i,j)$, save pointer(s) to cell(s) that resulted in best score,
- $F(m,n)$ holds the optimal alignment score,
- trace pointers back from $F(m,n)$ to $F(0,0)$ to recover alignment,
- example:
  - suppose we choose the scoring scheme:
    - if $x_i = y_j$ then $s(i,j)$=1 otherwise $s(i,j)$=-1,
    - d (penalty for aligning with a gap) $= 2$.

Global alignment example



Marc Craven, BMI/CS 576, www.biostat.wisc.edu/bmi576.

highroad alignment

x:	A	A	A	C
y:	A	G	-	C

lowroad alignment

x:	A	A	A	C
y:	-	A	G	C

DP alignment: comments

- many optimal alignments may exist for a given pair of sequences
  - can use preference ordering over paths when doing traceback,
- works for either DNA or protein sequences, although the substitution matrices used differ
  - DNA examples here for the sake of simplicity,
- the exact algorithm (and computational complexity) depends on gap penalty function,
- computational complexity with linear gap penalty function
  - initialization: $\mathcal{O}(m)$, $\mathcal{O}(n)$ where sequence lengths are m and n,
  - filling in rest of matrix: $\mathcal{O}(mn)$,
  - traceback: $\mathcal{O}(m+n)$,
  - total when $m \approx n$: $\mathcal{O}(n^2)$,

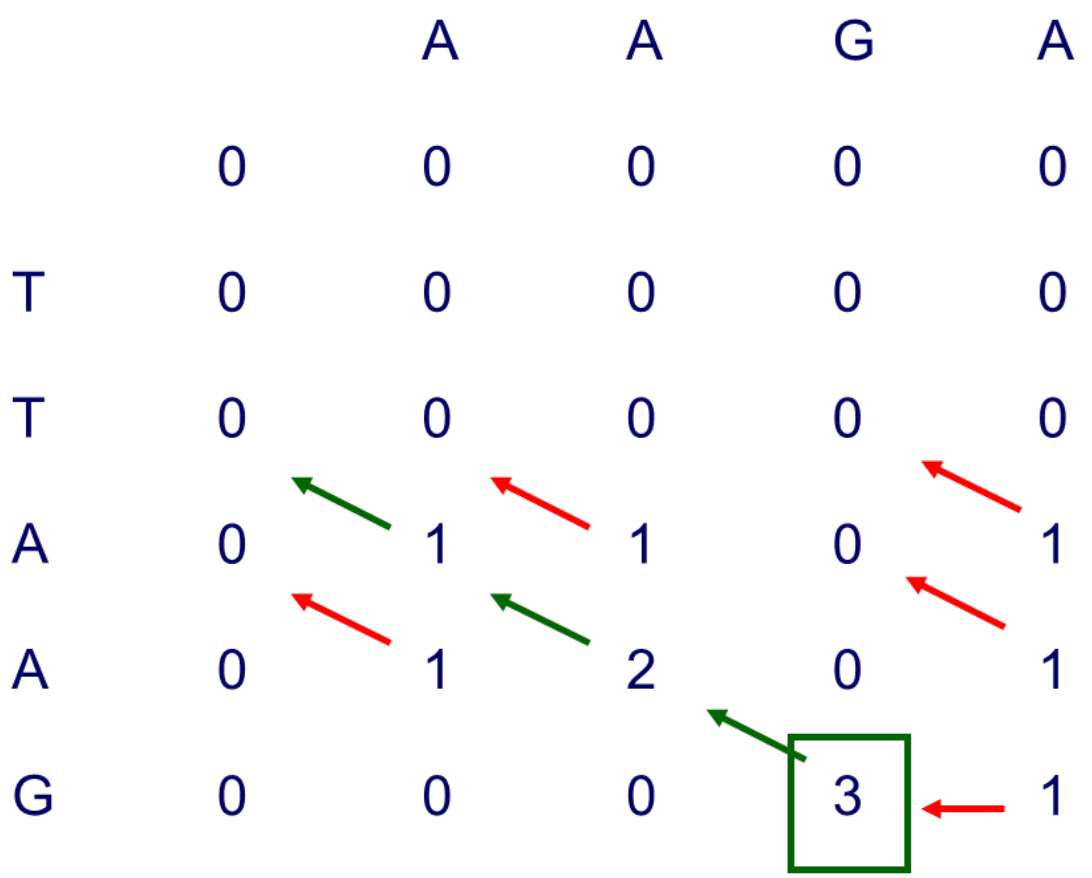
Local alignment

- so far we have discussed global alignment, where we are looking for best match between sequences from one end to the other,
- often we want the best match between subsequences of X and Y,
- motivation for local alignment
  - useful for comparing protein sequences that share a common motif (conserved pattern) or domain (independently folded unit) but differ elsewhere,
  - more sensitive when comparing highly diverged sequences,
  - useful for comparing protein sequences against genomic DNA sequences (long stretches of uncharacterized sequence).

Local alignment DP algorithm

- first proposed by Smith and Waterman, Journal of Molecular Biology, 1981,
- changes wrt global aligment DP algorithm
  - F interpretation
    - $F(i, j)$ = score of the best alignment of **a suffix of** X[1. . . i] and **a suffix of** Y[1. . . j],
    - $F(i, j)$ cannot be negative (means skip the current prefices when the score is negative),
  - initialization
    - first row and first column initialized with 0's,
  - traceback
    - start from maximum value of $F(i, j)$, can be **anywhere** in matrix,
    - stop when we get to a cell with value 0.

Local alignment example



x: A A G
y: A A G

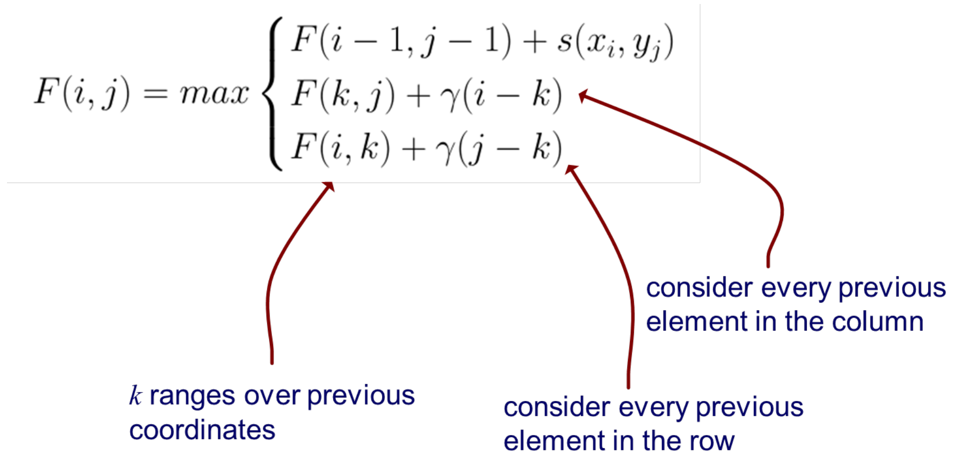
Marc Craven, BMI/CS 576, www.biostat.wisc.edu/bmi576.

Gap penalty functions

- a gap of length k is more probable than k gaps of length 1,
  - a gap may be due to a single mutational event that inserted/deleted a stretch of characters,
  - separated gaps are probably due to distinct mutational events,
  - a linear gap penalty function treats these cases the same,
- it is more common to use gap penalty functions involving two terms
  - a penalty d associated with opening a gap,
  - a smaller penalty e for extending the gap,
  - affine penalty: $w(g) = -d - (g-1)e$ for $g \geq 1$ otherwise 0,
  - convex penalty: $w(g) = -d - log(g)e$ for $g \geq 1$ otherwise 0.

Computational complexity and gap penalty functions

- assume two sequences of length n,
- DP time complexity depends on gap penalty function as follows
  - linear penalty: $\mathcal{O}(n^2)$,
  - affine penalty: $\mathcal{O}(n^2)$,
  - convex penalty: $\mathcal{O}(n^2 \log n)$,
  - general penalty: $\mathcal{O}(n^3)$.
- why the general case has time complexity $\mathcal{O}(n^3)$



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

consider every previous element in the column
k ranges over previous coordinates
consider every previous element in the row

Marc Craven, BMI/CS 576, www.biostat.wisc.edu/bmi576.

Pairwise alignment summary

- sequences must be aligned before similarity assessment,
- the number of possible alignments is exponential in the length of sequences being aligned,
- dynamic programming can find optimal-scoring alignments in polynomial time,
- the specifics of the DP depend on
  - local vs. global alignment,
  - gap penalty function,
- affine penalty functions are most commonly used,
- the alignment can be done in linear space.