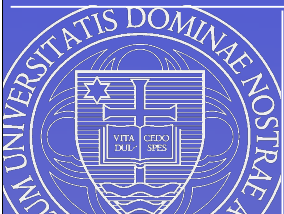


Local Alignment: The Algorithms

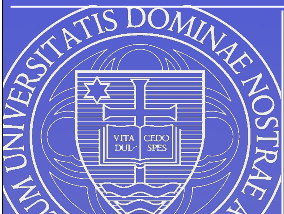
Tim Schoenharl
Computer Science and Engineering



University of Notre Dame
Department of Computer Science and Engineering

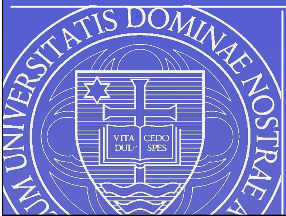
Local Sequence Alignment Algorithm

- Determine optimal local alignment of two sequences
- Potential alignments compared using a scoring model
 - Rank matches of residues, as well as substitutions and gaps
- Requires scoring matrix such as PAM or BLOSUM and value for gap penalty



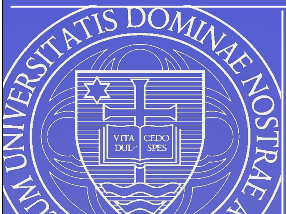
Scoring Matrices

- BLOSUM and PAM are commonly used
- PAM is derived from an explicit evolutionary model
- BLOSUM is derived from empirical observation
- Important to choose appropriate scoring matrix



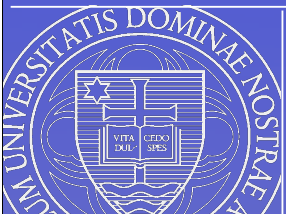
More on Scoring Matrices

- PAM - Percent Accepted Mutation
 - Dayhoff 1978
 - PAM1 calculated from comparisons of sequences with no more than 1% divergence [NCBI website]
 - Other PAM matrices calculated by matrix multiplication



More on Scoring Matrices

- BLOSUM - BLOck SUBstitution Matrix
 - Henikoff and Henikoff 1992
 - BLOSUM 62 calculated from comparisons of sequences with no less than 62% divergence [NCBI website]
 - All BLOSUM matrices calculated from observations



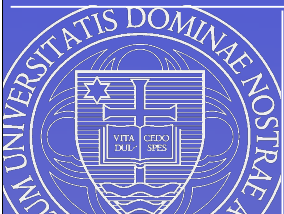
BLOSUM 50 Matrix

	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

Smith Waterman Algorithm

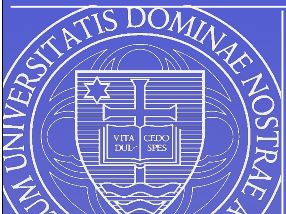
- Dynamic programming solution
- Finds best local alignment between two sequences
- Build table using following formula for each cell:

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



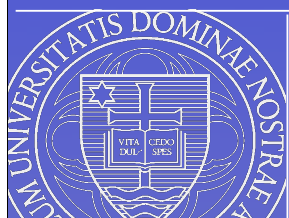
Example

- Taken from text handout
- Two sequences: HEAGAWGHEE
PAWHEAE
- Find best local alignment using BLOSUM50 scoring matrix and a gap penalty of 8



Building the Alignment Table

- We will use the dynamic programming approach: The best final solution is a result of the best sub-solution from the last stage
- Start at the upper left of the table, progress along the diagonal
- Can use recursion (not shown)



Building the Alignment Table

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

$F(2,2) = \max \left\{ \begin{array}{l} 0 \\ F(1,1) + s(E,A) = 0 - 1 = -1 \\ F(1,2) - 8 = -8 \\ F(2,1) - 8 = -8 \end{array} \right.$

Recovering the Local Alignment

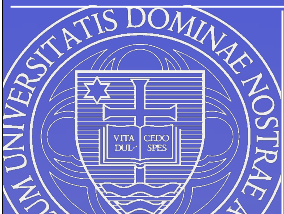
	H	E	A	G	A	W	G	H	E	E
P	0	0	0	0	0	0	0	0	0	0
A	0	0	5	0	5	0	0	0	0	0
W	0	0	0	2	0	2	1	4	0	0
H	1	2	0	0	0	1	1	2	1	6
E	2	1	8	0	0	4	1	1	2	2
A	0	8	2	1	5	0	4	1	2	2
E	0	6	1	1	1	4	0	4	1	2

A W G H E

A W - H E

Local Alignment

- The alignment table holds both the score for the current location and a back pointer
- The back pointer enables the local alignment to be extracted at the end of the algorithm
- Start with the highest value, follow pointers back until you reach a “0”



Variations

- Several variations of the Smith-Waterman algorithm exist
 - *k*-tuples: Initially find identical sequences of length *k*, then use these to build the table
 - Affine gap penalty: Split the gap penalty into gap-opening and gap-extension components
 - Several other domain specific optimizations

