

# Sequence Alignment

Linear space and  
Heuristic techniques

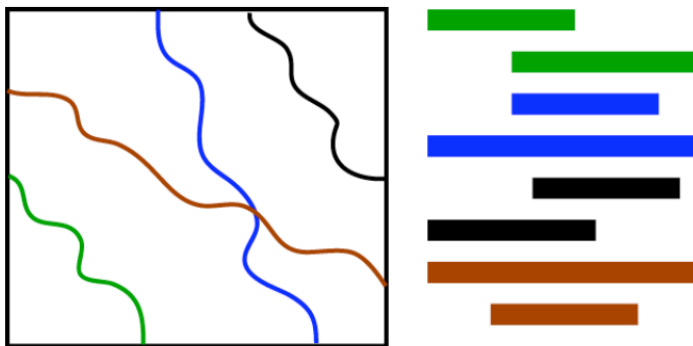
## Global Alignment (review)

- Input: Two strings, labeled  $s$  and  $t$ 
  - $|s|$  is  $n$
  - $|t|$  is  $m$
- Output: Two strings  $s_A$  and  $t_A$  such that:
  - $s_A$  and  $t_A$  are of equal length  $L$
  - Characters must be in same order, with “-” spacers as needed
  - If  $s_A[i] = '-'$ , then  $t_A[i] \neq '-'$
  - If  $t_A[i] = '-'$ , then  $s_A[i] \neq '-'$

## End-gap free alignment

- We often don't want to penalize gaps at the start or end of the alignment, especially when comparing short and long sequences
- Same as global alignment, except:
  - Initialize with zeros (free gaps at start)
  - Locate max in the last row/column (free gaps at end)
- Also called *semiglobal alignment*

## Example paths



## Practice

$$T[j, 0] = 0 \quad T[0, j] = 0$$

$$T[i, j] = \max \begin{cases} T[i-1, j-1] + \text{score}(s[i], t[j]) \\ T[i-1, j] + g \\ T[i, j-1] + g \end{cases}$$

Match = 5  
Mismatch = -2  
Gap = -3

S = AGCATTA  
T = ACATTTAG

Find the score of  
the best semiglobal  
alignment

	$\lambda$	C	T	C	G	C	A	G	C
$\lambda$	0	0	0	0	0	0	0	0	0
C	0	10	5	10	5	10	5	0	10
A	0	5	8	5	8	5	20	15	10
T	0	0	15	10	5	6	15	18	13
T	0	-2	10	13	8	3	10	13	16
C	0	10	5	20	15	18	13	8	23
A	0	5	8	15	18	13	28	23	18
G	0	0	3	10	25	20	23	38	33

+10 for match, -2 for mismatch, -5 for space (rowwise)

## Local Alignment

$T[i, j]$  = Score of optimally aligning a suffix of  $s$  with a suffix of  $t$ .

$$T[i, j] = \max \begin{cases} T[i-1, j-1] + \text{score}(s[i], t[j]) \\ T[i-1, j] + g \\ T[i, j-1] + g \\ 0 \end{cases}$$

Initialize top row and leftmost column to zero.

## Practice

$$T[j, 0] = 0 \quad T[0, j] = 0$$

$$T[i, j] = \max \begin{cases} T[i-1, j-1] + \text{score}(s[i], t[j]) \\ T[i-1, j] + g \\ T[i, j-1] + g \\ 0 \end{cases}$$

Match = 5  
Mismatch = -2  
Gap = -3

S = AGCATTA  
T = ACATTTAG

Find the score of  
the best local alignment

	$\lambda$	C	T	C	G	C	A	G	C
$\lambda$	0	0	0	0	0	0	0	0	0
C	0	1	0	1	0	1	0	0	1
A	0	0	0	0	0	0	2	0	0
T	0	0	1	0	0	0	0	1	0
T	0	0	1	0	0	0	0	0	0
C	0	1	0	2	0	1	0	0	1
A	0	0	0	0	1	0	2	0	0
C	0	1	0	1	0	2	0	1	1

+1 for a match, -1 for a mismatch, -5 for a gap

## Reducing space requirements

- $O(mn)$  tables are often the limiting factor in computing large alignments
- There is a linear space technique that only doubles the time required [Hirschberg77]

	$\lambda$	C	T	C	G	C	A	G	C
$\lambda$	0	0	0	0	0	0	0	0	0
C	0	10	5	10	5	10	5	0	10
A	0	5	8	5	8	5	20	15	10
T									
T									
C									
A									
G									

IDEA: We only need the *previous* row to calculate the *next*

## Global alignment in linear space

- The space required for the alignment table is  $O(mn)$ . For large strings, this is not good.
- The Hirschberg technique reduces this to  $O(\min(m,n))$ . We'll discuss this today.

## Assertion

- Before, we made the following observation:

$$A(n, m) = A(1..i, 1..j) + A(i..n, j..m)$$

What if the optimal alignment goes through  $A(i, j)$ ?  
Does this help?

## The trick

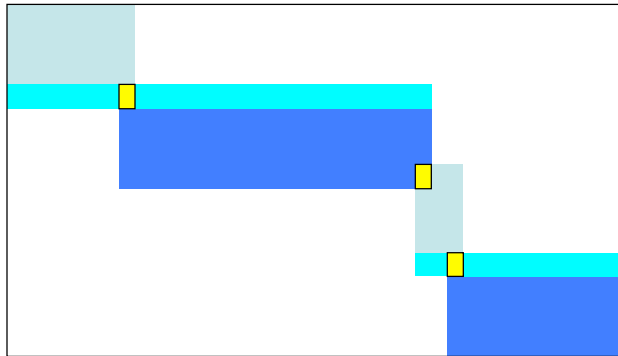
- Lemma:

$$A(n, m) = \max_{0 \leq k \leq m} \left\{ A\left(\frac{n}{2}, k\right) + A^r\left(\frac{n}{2}, m - k\right) \right\}$$

Where  $A^r$  is the reverse of strings  $s$  and  $t$

In other words,  $s[n]s[n-1] \dots s[1]$  aligned to  $t[m]t[n-1] \dots t[1]$

## Linear-space Alignments



$$mn + \frac{1}{2} mn + \frac{1}{4} mn + \frac{1}{8} mn + \frac{1}{16} mn + \dots = (2 mn) * 2$$

## Problem

- Prof. Ferdig in Biological Sciences has 5500 malaria genes. He would like to see if they match any sequences in a BIG database.
- Armed with sequence alignment, what do you want to do next?

## Reality

- We could in theory perform a global alignment of each malaria gene to the database.
- However, it turns out this will not be practical for most purposes.
- BLAST is one approach that will approximate the alignment, often orders of magnitude faster.

## The contenders

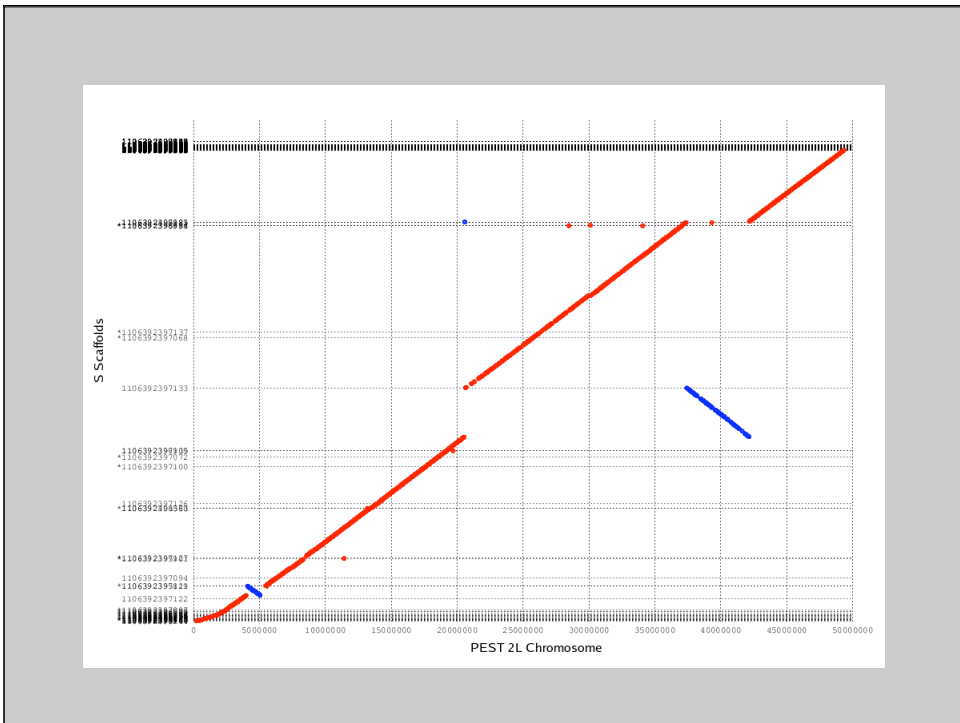
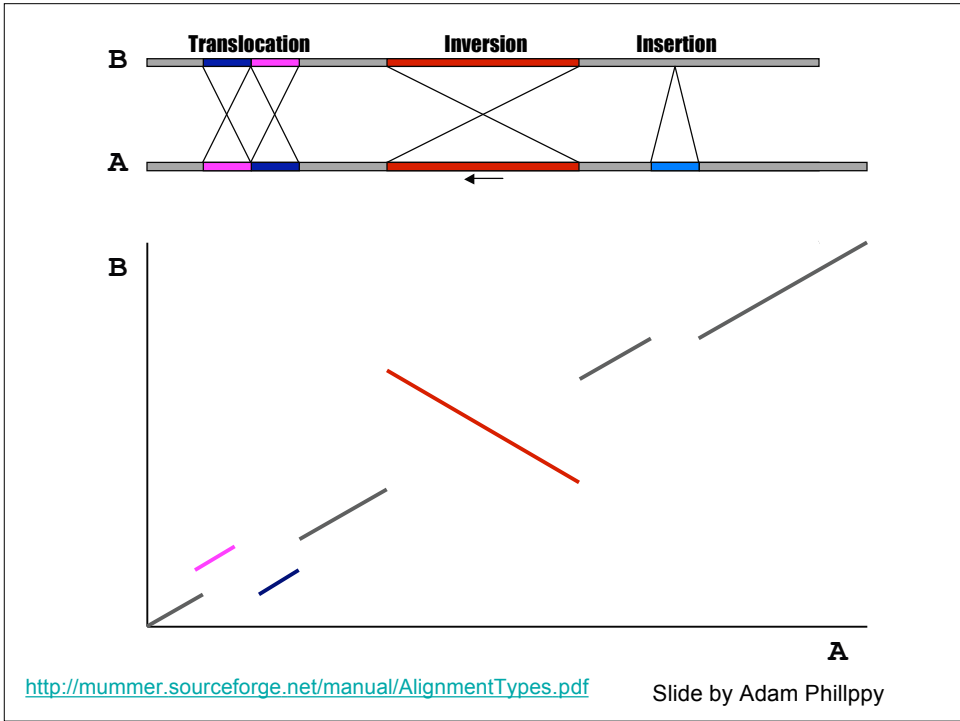
- FASTA - Fast-All
  - Pearson and Lipman (1988)
- BLAST
  - Altschul et al. (1990)

# BLAST

- Basic Local Alignment Search Tool
- Used to quickly compare a protein or DNA sequence to a database.
- Output:
  - <http://www.plasmodb.org/plasmo/showQuestion.do?questionFullName=UniversalQuestions.UnifiedBlast&target=SEQ>

## “There is no such thing as a free lunch”

- BLAST is fast and highly sensitive compared to competitors.
- Disadvantages:
  - Misses some homologous matches
  - Alignment is not guaranteed to be optimal



## Dirty details

- Four basic steps are performed in BLAST:
  - Compile a list of “interesting” words
  - Scan for these words, generating “hits”
  - Extend the hits into longer alignments
  - Determine if the longer hits are “good”

## Compiling words

- Given a word length  $l = 4$

ATGCTGTTTTGGGAATGTGTG

ATGC

TGCT

GCTG

CTGT

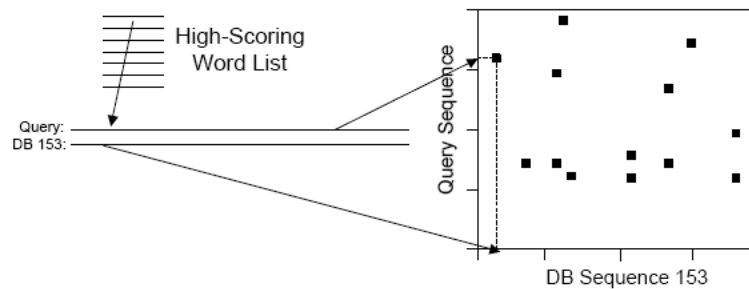
.....

TGTG

## Collecting hits

- Scan the first word against the entire database.
- For every matching word, record a diagonal in a special table
- Repeat for the entire query.

## Illustration



## Finding runs and extending them (old version)

- Extend alignments “greedily” off each end; stop when the score drops below a threshold
- Alignments are ungapped, so extension is straightforward.
- All hits whose score is greater than a minimum score  $S$  are displayed.

## Gapped BLAST

- Require two hits on the same “diagonal”
- Hits must be less than a specified distance away (antidiagonal difference). Alignment is explored in between matches (banded variant)
- Substantially reduces the number of extensions, and is a better heuristic in terms of biological value

## E-value

- So what? We found a bunch of probably useless alignments, right?
- There are sophisticated statistics at the core of BLAST that produce an “e-value”, which is the probability of observing an alignment by chance.
- Based on work of Altschul and Karlin, based on extreme value statistics.

## In-depth examples

- <http://magi.plantgenomics.iastate.edu/blast.do?blastGroup=SAMI>