For written notes on this lecture, please read chapter 10 of *The Practical Bioinformatician*

CS2220: Introduction to Computational Biology

Lecture 7: Essence of Sequence Comparison

Limsoon Wong

24 February 2006

Plan

- **Dynamic Programming**
- **String Comparison**
- **Sequence Alignment**
  - Pairwise Alignment
    - **Needleman-Wunsch global alignment algorithm**
    - **Smith-Waterman local alignment algorithm**
  - Multiple Alignment
- **Popular tools**
  - FASTA, BLAST, Pattern Hunter
- **More advanced tools**
  - PHI BLAST, ISS, PSI-BLAST, SAM, ...

What is Dynamic Programming

The Knapsack Problem

Source: http://mat.gsia.cmu.edu/classes/dynamic/node6.html

- **The problem**
  - Each item that can go into the knapsack has a size and a benefit. The knapsack has a certain capacity. What should go into the knapsack so as to maximize the total benefit?
- **A dynamic programming solution**
  - Let w_j and b_j be weight and benefit for item *j*. Let $g(w)$ be max benefit that can be gained from a *w* pound knapsack. Then $g(w)$ relates to previously calculated *g* values as follows:

$$g(w) = \max_j \{ b_j + g(w - w_j) \}$$

An Example

Source: http://mat.gsia.cmu.edu/classes/dynamic/node6.html

- **Suppose the items are**

Item (j)	Weight (w_j)	Benefit(b_j)
1	2	65
2	3	80
3	1	30

- **Recall that**

$$g(w) = \max_j\{b_j + g(w - w_j)\}$$

- **To fill a *w* pound knapsack, we must end off by adding some item. If we add item *j*, we end up with a knapsack of size *w – wj* to fill …**

- **To illustrate:**
  - g(0) = 0
  - g(1) = 30, **item 3**
  - g(2) = max{65 + g(0) =65, 30 + g(1) = 60} = 65, **item 1**
  - g(3) = max{65 + g(1) = 95, 80 + g(0) = 80, 30 + g(2) = 95} = 95, **item 1/3**
  - g(4) = max{65 + g(2) = 130, 80 + g(1) = 110, 30 + g(3) = 125} = 130, **item 1**
  - g(5) = max{65 + g(3) = 160, 80 + g(2) = 145, 30 + g(4) = 160} = 160, **item 1/3**

$\Rightarrow$ **For knapsack of capacity 5, max benefit is 160, which is gained by adding 2 of item 1 and 1 of item 3**

Characteristics of Dynamic Programming

Source: http://mat.gsia.cmu.edu/classes/dynamic/node4.html

- The problem can be divided into *stages* with a *decision* required at each stage

Exercise: What is a stage in the Knapsack problem?

- Each stage has a number of *states* associated
- The decision at one stage transforms one state into a state in the next stage

Exercise: What is a state in the Knapsack problem?

- Given current state, the optimal decision for each remaining states does not depend on previous states or decisions

E.g., g(2) doesnt depends on g(3)

- There is a recursive relationship that identifies the optimal decision for stage *j*, given stage *j*+1 has already been solved
- The final stage must be solvable by itself

E.g., g(0) = 0

Sequence Alignment

NUS
National University
of Singapore

Motivations for Sequence Comparison

- DNA is blue print for living organisms

$\Rightarrow$ Evolution is related to changes in DNA

$\Rightarrow$ By comparing DNA sequences we can infer evolutionary relationships between the sequences w/o knowledge of the evolutionary events themselves

- Foundation for inferring function, active site, and key mutations

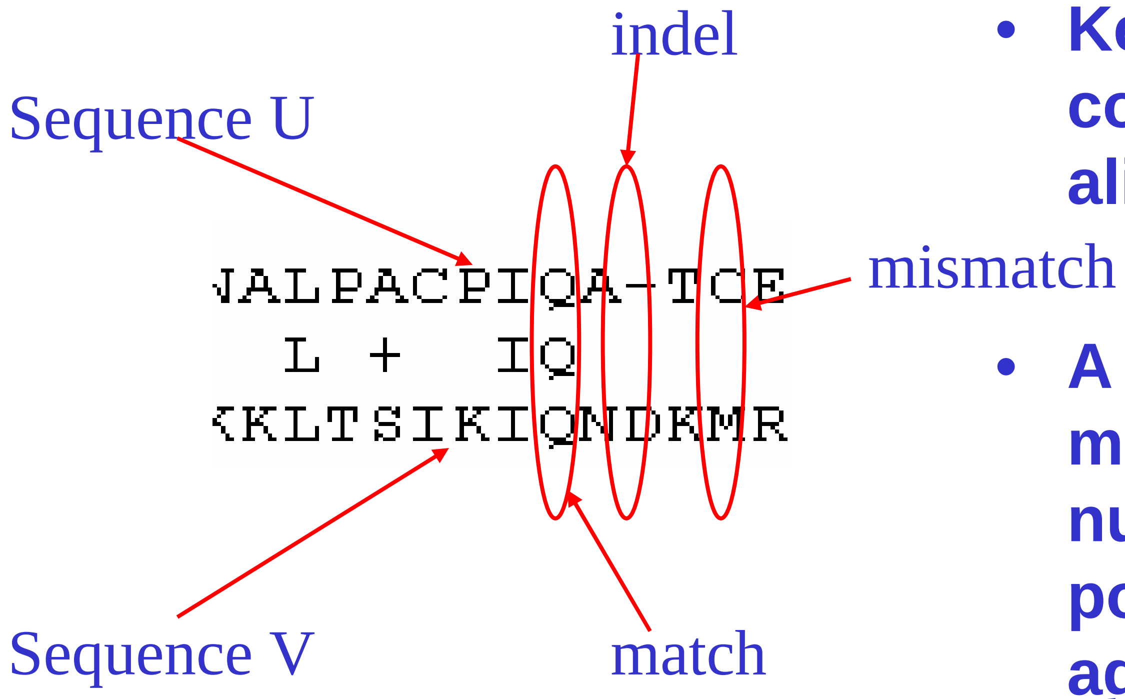
Earliest Research in Seq Comparison

Source: Ken Sung

- **Doolittle et al. (*Science*, July 1983) searched for platelet-derived growth factor (PDGF) in his own DB. He found that PDGF is similar to v-sis oncogene**

```
PDGF-2  1       SLGSLTIAEPAMIAECKTREEVFCICRRL?DR?? 34
p28sis 61 LARGKRSLGSLSVAEPAMIAECKTRTEVFEISRRLIDRTN 100
```

Sequence Alignment



- **Key aspect of seq comparison is seq alignment**
- **A seq alignment maximizes the number of positions that are in agreement in two sequences**

Sequence Alignment: Poor Example

- **Poor seq alignment shows few matched positions**

⇒ The two proteins are not likely to be homologous

Alignment by FASTA of the sequences of amicyanin and domain 1 of ascorbate oxidase

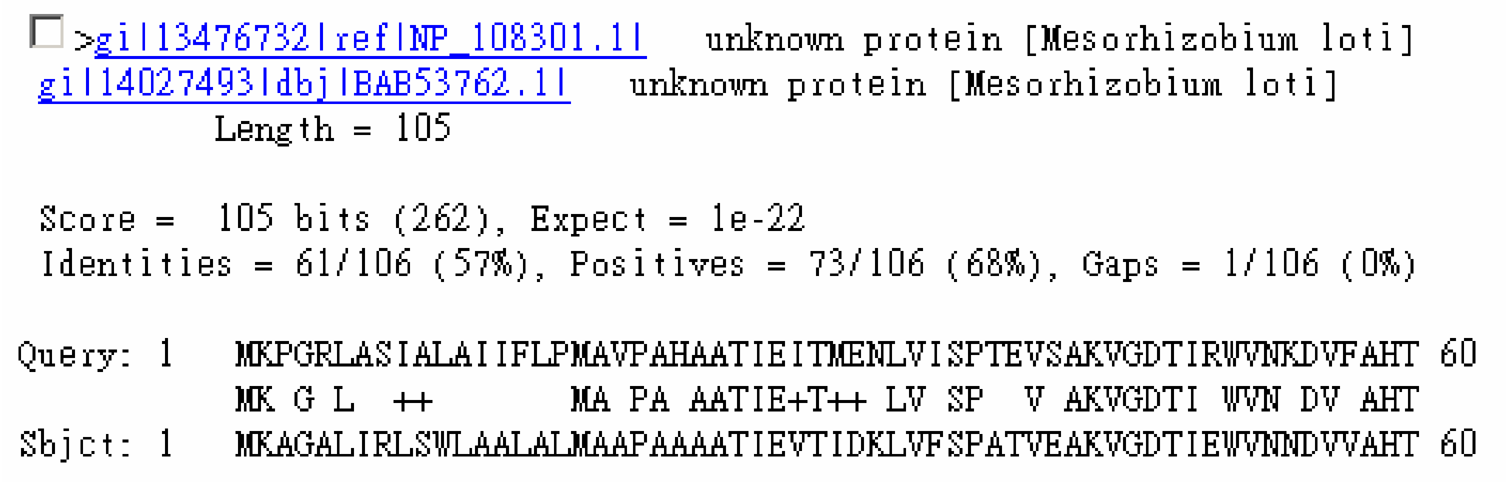
```
                          60        70        80        90       100
Amicyanin         MPHNVHFVAGVLGEAALKGPMMKKEQAYSLTFTEAGTYDYHCTPHPFMRGKVVVE
                                             :..:  . ::. ::
Ascorbate Oxidase ILQRGTPWADGTASISQCAINPGETFFYNFTVDNPGTFFYHGHLGMQRSAGLYGSLI
                       70        80        90       100       110       120
```

No obvious match between
Amicyanin and Ascorbate Oxidase

Sequence Alignment: Good Example

- **Good alignment usually has clusters of extensive matched positions**

⇒ **The two proteins are likely to be homologous**



```
☐ >gi|13476732|ref|NP_108301.1|   unknown protein [Mesorhizobium loti]
 gi|14027493|dbj|BAB53762.1|   unknown protein [Mesorhizobium loti]
          Length = 105

 Score =  105 bits (262), Expect = 1e-22
 Identities = 61/106 (57%), Positives = 73/106 (68%), Gaps = 1/106 (0%)

Query: 1    MKPGRLASIALAIIFLPMAVPAHAATIEITMENLVISPTEVSAKVGDTIRWVNKDVFAHT 60
            MK G L  ++       MA PA AATIE+T++ LV SP  V AKVGDTI WVN DV AHT
Sbjct: 1    MKAGALIRLSWLAALALMAAPAAAATIEVTIDKLVFSPATVEAKVGDTIEWVNNDVVAHT 60
```

good match between
Amicyanin and unknown M. loti protein

Alignment:
Simple-minded Probability & Score

Let p, q, r be respectively the probability of a match, a mismatch, and an indel. Then the probability of an alignment $A = (X, Y)$ is

$$prob(A) = p^m \cdot q^n \cdot r^h$$

where

$$\begin{array}{rcl} m & = & |\{i \mid x'_i = y'_i \neq -\}| \\ n & = & |\{i \mid x'_i \neq y'_i, x'_i \neq -, y'_i \neq -\}| \\ h & = & |\{i \mid x'_i = -, y'_i \neq -\} \cup \{i \mid x'_i \neq -, y'_i = -\}| \end{array}$$

- **Define score S(A) by simple log likelihood as**
  - S(A) = log(prob(A)) - [m log(s) + h log(s)], with log(p/s) = 1
- **Then S(A) = #matches - μ #mismatches - δ #indels**

Exercise: Derive μ and δ

Global Pairwise Alignment: Problem Definition

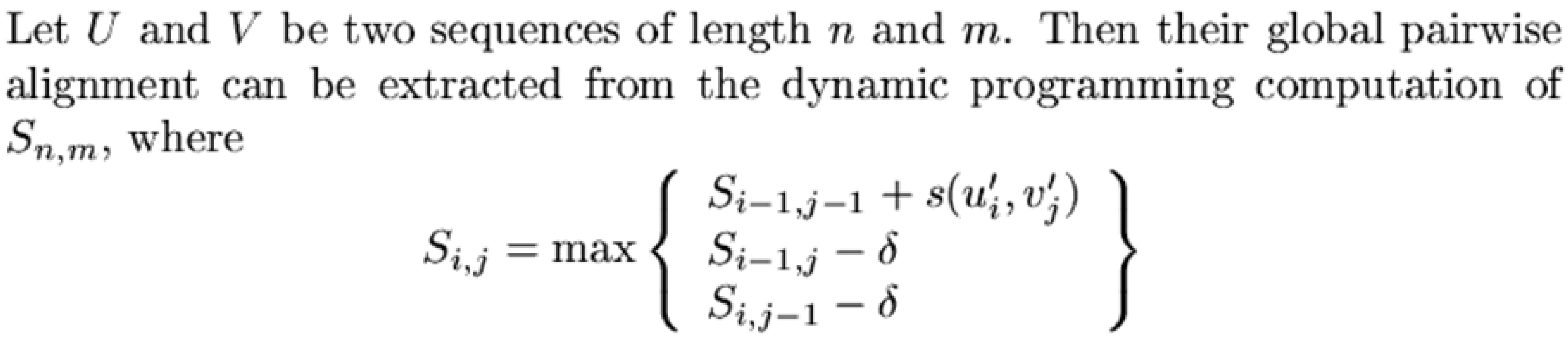
- **Given sequences *U* and *V* of lengths *n* and *m*, then number of possible alignments is given by**
  - $f(n, m) = f(n-1,m) + f(n-1,m-1) + f(n,m-1)$
  - $f(n,n) \sim (1 + \sqrt{2})^{2n+1} n^{-1/2}$

Exercise: Explain the recurrence above

- **The problem of finding a global pairwise alignment is to find an alignment *A* so that *S(A)* is max among exponential number of possible alternatives**

Global Pairwise Alignment: Dynamic Programming Solution

- **Define an indel-similarity matrix *s(.,.)*; e.g.,**
  - *$s(x,x) = 2$*
  - *$s(x,y) = -\mu$, if $x \neq y$*
- **Then**



Let U and V be two sequences of length n and m. Then their global pairwise alignment can be extracted from the dynamic programming computation of $S_{n,m}$, where

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(u'_i, v'_j) \\ S_{i-1,j} - \delta \\ S_{i,j-1} - \delta \end{array} \right\}$$

This is the basic idea of the Needleman-Wunsch algorithm

Needleman-Wunsch Algorithm (I)

Source: Ken Sung

- **Consider two strings S[1..n] and T[1..m]**
- **Let V(i, j) be score of opt alignment betw S[1..i] and T[1..j]**

- **Basis:**
  - $V(0, 0) = 0$
  - $V(0, j) = V(0, j-1) - \delta$
    - **Insert j times**
  - $V(i, 0) = V(i-1, 0) - \delta$
    - **Delete i times**

Needleman-Wunsch Algorithm (II)

Source: Ken Sung

- **Recurrence: For i>0, j>0**

$$V(i,j) = \max \begin{cases} V(i-1, j-1) + s(S[i], T[j]) & \textbf{Match / mismatch} \\ V(i-1, j) - \delta & \textbf{Delete} \\ V(i, j-1) - \delta & \textbf{Insert} \end{cases}$$

- **In the alignment, the last pair must be either match/mismatch, delete, insert**

```
xxx…xx        xxx…xx        xxx…x_
     |             |             |
xxx…yy        yyy…y_        yyy…yy
```

Match/mismatch Delete Insert

Example (I)

Source: Ken Sung

	_	A	G	C	A	T	G	C
_	0	–1	– 2	– 3	– 4	– 5	– 6	– 7
A	– 1							
C	– 2							
A	– 3							
A	– 4							
T	– 5							
C	– 6							
C	– 7							

Example (II)

Source: Ken Sung

	_	A	G	C	A	T	G	C
_	0	– 1	– 2	– 3	– 4	– 5	– 6	– 7
A	– 1	2	1	0	– 1	– 2	– 3	– 4
C	– 2	1	1	3	2			
A	– 3							
A	– 4							
T	– 5							
C	– 6							
C	– 7							

Exercise: Can you tell from these entries what
Are the values of s(A,G), s(A,C), s(A,A), etc.?

Example (III)

Source: Ken Sung

	_	A	G	C	A	T	G	C
_	0	–1	– 2	– 3	– 4	– 5	– 6	– 7
A	– 1	2	1	0	– 1	– 2	– 3	-4
C	– 2	1	1	3	2	1	0	-1
A	– 3	0	0	2	5	4	3	2
A	– 4	– 1	– 1	1	4	4	3	2
T	– 5	– 2	– 2	0	3	6	5	4
C	– 6	– 3	– 3	0	2	5	5	7
C	– 7	– 4	– 4	– 1	1	4	4	7

Pseudo Codes

Source: Ken Sung

```
Create the table V[0..n,0..m] and P[1..n,1..m];
V[0,0] = 0;
For j=1 to m, set V[0,j] := v[0,j – 1] – δ ;
For i=1 to n, set V[i,0] := V[i – 1,0] – δ ;
For j=1 to m {
   For i = 1 to n {
       set V[i,j] := V[i,j – 1] – δ ;
       set P[i,j] := (0, – 1);
       if V[i,j] < V[i – 1,j] – δ then
              set V[i,j] := V[i – 1,j] – δ ;
              set P[i,j] := (– 1, 0);
       if (V[i,j] < V[i – 1, j – 1] + s(S[i],T[j])) then
              set V[i,j] := V[i – 1, j – 1] + s(S[i],T[j]);
              set P[i,j] := (– 1, – 1);
   }
}
Backtracking P[n,m] to P[0,0] to find optimal alignment;
```

Analysis

Source: Ken Sung

- We need to fill in all entries in the n×m matrix
- Each entries can be computed in O(1) time

⇒ Time complexity = O(nm)

⇒ Space complexity = O(nm)

Problem on Speed

Source: Ken Sung

- **Aho, Hirschberg, Ullman 1976**
  - If we can only compare whether two symbols are equal or not, the string alignment problem can be solved in $\Omega(nm)$ time
- **Hirschberg 1978**
  - If symbols are ordered and can be compared, the string alignment problem can be solved in $\Omega(n \log n)$ time
- **Masek and Paterson 1980**
  - Based on Four-Russian's paradigm, the string alignment problem can be solved in O(nm/log2 n) time
- **Let d be the total number of inserts and deletes. Thus $0 \leq d \leq n+m$. If d is smaller than n+m, can we get a better algorithm? Yes!**

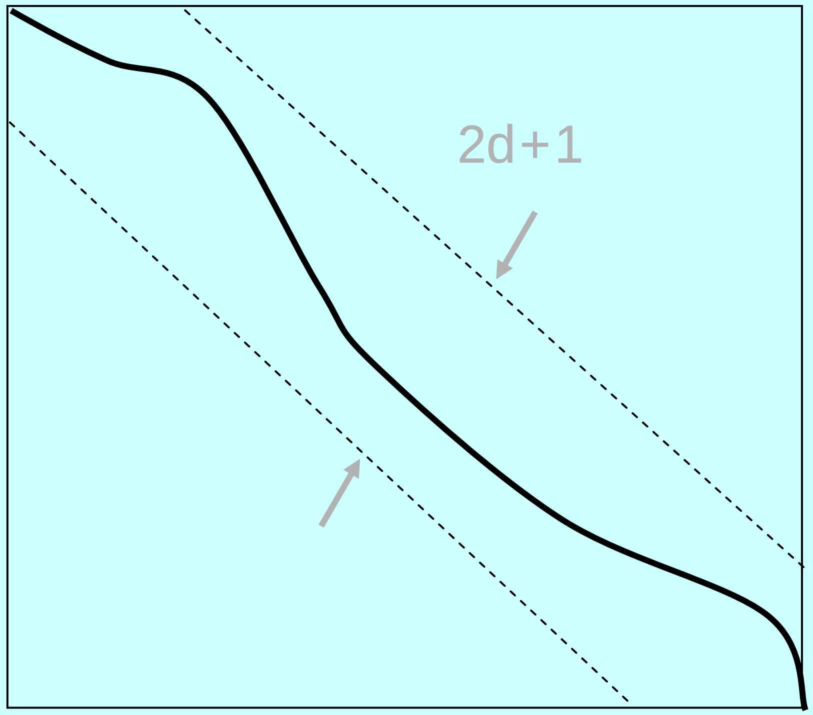
O(dn)-Time Algorithm

Source: Ken Sung

- **The alignment should be inside the 2d+1 band**

$\Rightarrow$ No need to fill-in the lower and upper triangle

$\Rightarrow$ Time complexity: O(dn)

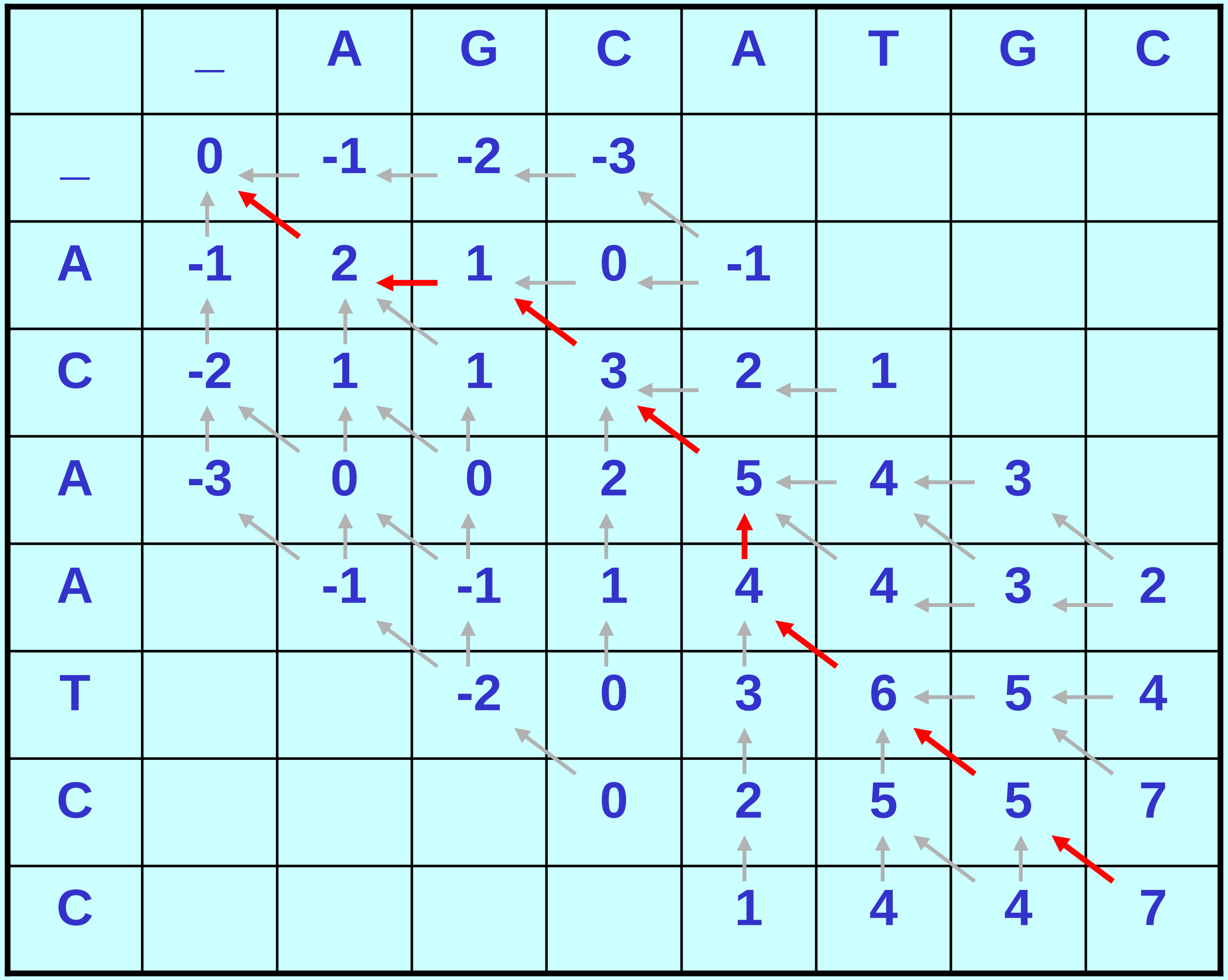


Example

- **d=3**

  A_CAATCC

  AGCA_TGC



	_	A	G	C	A	T	G	C
_	0	-1	-2	-3				
A	-1	2	1	0	-1			
C	-2	1	1	3	2	1		
A	-3	0	0	2	5	4	3	
A		-1	-1	1	4	4	3	2
T			-2	0	3	6	5	4
C				0	2	5	5	7
C					1	4	4	7

Global Pairwise Alignment: More Realistic Handling of Indels

- **In Nature, indels of several adjacent letters are not the sum of single indels, but the result of one event**
- **So reformulate as follows:**

Let $g(k)$ be the indel weight for an indel of k letters. Typically, $g(k) \leq k \cdot g(1)$. Let U and V be two sequences of length n and m. Then their global pairwise alignment can be extracted from the dynamic programming computation of $S_{n,m}$, where

$$S_{0,0} = 0, \quad S_{0,j} = -g(j), \quad S_{i,0} = -g(i)$$

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(u'_i, v'_j) \\ \max_{1 \leq k \leq j} \{S_{i,j-k} - g(k)\} \\ \max_{1 \leq k \leq i} \{S_{i-k,j} - g(k)\} \end{array} \right\}$$

Gap Penalty

Source: Ken Sung

- **$g(q): \mathbb{N} \rightarrow \Re$ is the penalty of a gap of length q**
- **Note g() is subadditive, i.e, $g(p+q) \leq g(p) + g(q)$**

- **If $g(k) = \alpha + \beta k$, the gap penalty is called affine**
  - A penalty (α) for initiating the gap
  - A penalty (β) for the length of the gap

N-W Algorithm w/ General Gap Penalty (I)

Source: Ken Sung

- **Global alignment of S[1..n] and T[1..m]:**
  - Denote V(i, j) be the score for global alignment between S[1..i] and T[1..j]
  - Base cases:
    - **V(0, 0) = 0**
    - **V(0, j) = g(j)**
    - **V(i, 0) = g(i)**

N-W Algorithm w/ General Gap Penalty (II)

Source: Ken Sung

- **Recurrence for i>0 and j>0,**

$$V(i,j) = \max\begin{cases} V(i-1, j-1) + \delta(S[i], T[j]) & \textbf{Match / mismatch} \\ \max\limits_{0 \le k \le j-1} \{V(i,k) + g(j-k)\} & \textbf{Insert T[k+1..j]} \\ \max\limits_{0 \le k \le i-1} \{V(k,j) + g(i-k)\} & \textbf{Delete S[k+1..i]} \end{cases}$$

Analysis

Source: Ken Sung

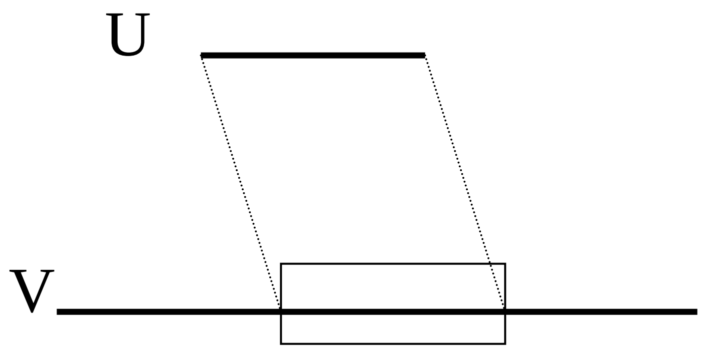
- We need to fill in all entries in the n×m table
- Each entry can be computed in O(max{n, m}) time

⇒ Time complexity = O(nm max{n, m})

⇒ Space complexity = O(nm)

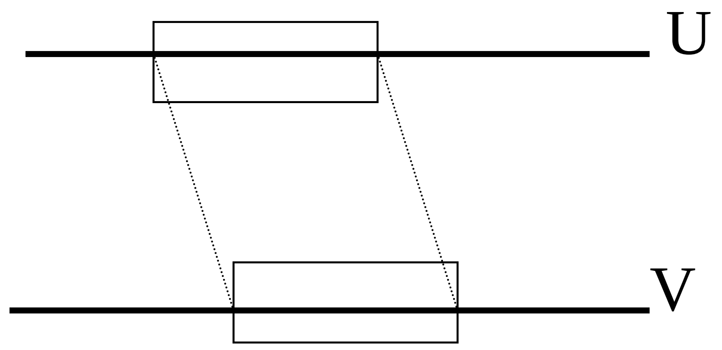
Variations of Pairwise Alignment

- **Fitting a "short'' seq to a "long'' seq**



- **Indels at beginning and end are not penalized**

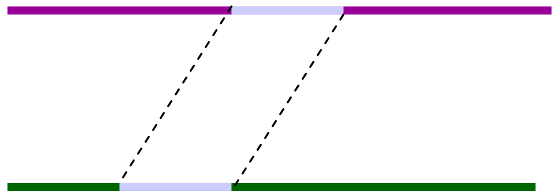
- **Find "local" alignment**



- **Find *i, j, k, l,* so that**
  - $S(A)$ is maximized,
  - A is alignment of $u_i \ldots u_j$ and $v_k \ldots v_l$

Local Alignment

Source: Ken Sung



- **Given two long DNAs, both of them contain the same gene or closely related gene**
  - Can we identify the gene?

- **Local alignment problem: Given two strings S[1..n] and T[1..m], among all substrings of S and T, find substrings A of S and B of T whose global alignment has the highest score**

Brute-Force Solution

Source: Ken Sung

- **Algorithm:**
  - For every substring A of S, for every substring B of T, compute the global alignment of A and B
  - Return the pair (A, B) with the highest score

- **Time:**
  - There are n^2 choices of A and m^2 choices of B
  - Global alignment computable in O(nm) time
  - In total, time complexity = $O(n^3m^3)$
- **Can we do better?**

Some Background

Source: Ken Sung

- **X is a suffix of S[1..n] if X=S[k..n] for some k≥1**
- **X is a prefix of S[1..n] if X=S[1..k] for some k≤n**

- **E.g.**
  - Consider S[1..7] = ACCGATT
  - ACC is a prefix of S, GATT is a suffix of S
  - Empty string is both prefix and suffix of S

Dynamic Programming for Local Alignment Problem

Source: Ken Sung

- **Define V(i, j) be max score of global alignment of A and B over**
  - all suffixes A of S[1..i] and
  - all suffixes B of T[1..j]

- **Then, score of local alignment is**
  - $\max_{i,j} V(i\ ,j)$

Smith-Waterman Algorithm

Source: Ken Sung

- **Basis:**

  V(i, 0) = V(0, j) = 0

- **Recursion for i>0 and j>0:**

$$V(i,j) = \max\begin{cases} 0 & \textbf{Align empty strings} \\ V(i-1, j-1) + s(S[i], T[j]) & \textbf{Match / mismatch} \\ V(i-1, j) - \delta & \textbf{Delete} \\ V(i, j-1) - \delta & \textbf{Insert} \end{cases}$$

- Score for match = 2
- Score for insert, delete, mismatch = –1

Example (I)

Source: Ken Sung

	_	C	T	C	A	T	G	C
_	0	0	0	0	0	0	0	0
A	0							
C	0							
A	0							
A	0							
T	0							
C	0							
G	0							

- Score for match = 2
- Score for insert, delete, mismatch = –1

Example (II)

Source: Ken Sung

	_	C	T	C	A	T	G	C
_	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	0
C	0	2	1	2	1	1	0	2
A	0	0	1	1	4	3	2	1
A	0	0	0	0	3	3	2	1
T	0	0	2	1	2			
C								
G								

CAATCG

C_AT_G

Example (III)

Source: Ken Sung

	_	C	T	C	A	T	G	C
_	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	0
C	0	2	1	2	1	1	0	2
A	0	0	1	1	4	3	2	1
A	0	0	0	0	3	3	2	1
T	0	0	2	1	2	5	4	3
C	0	2	1	4	3	4	4	6
G	0	1	1	3	3	3	6	5

Analysis

Source: Ken Sung

- **Need to fill in all entries in the $n \times m$ matrix**
- **Each entries can be computed in O(1) time**
- **Finally, finding the entry with the max value**

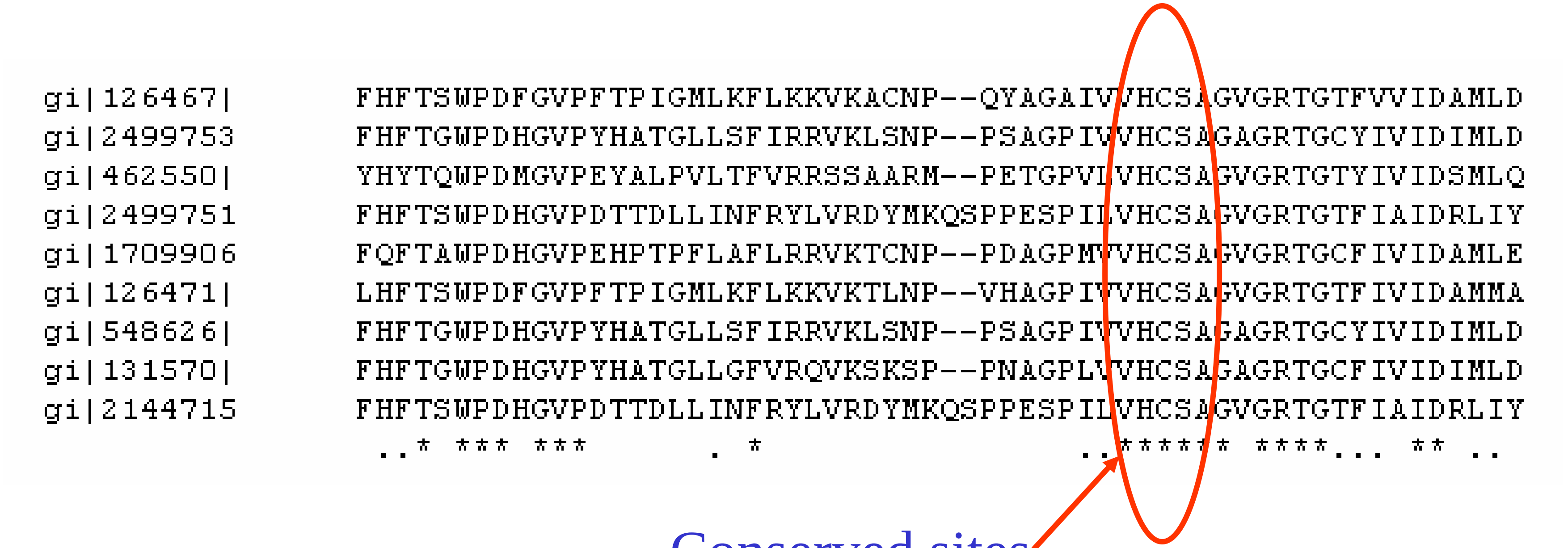
$\Rightarrow$ **Time complexity = ??**

$\Rightarrow$ **Space complexity = O(nm)**

Exercise: What is the time complexity?

Multiple Alignment: An Example

- **Multiple seq alignment maximizes number of positions in agreement across several seqs**
- **seqs belonging to same "family" usually have more conserved positions in a multiple seq alignment**



```
gi|126467|      FHFTSWPDFGVPFTPIGMLKFLKKVKACNP--QYAGAIVVHCSAGVGRTGTFVVIDAMLD
gi|2499753      FHFTGWPDHGVPYHATGLLSFIRRVKLSNP--PSAGPIVVHCSAGAGRTGCYIVIDIMLD
gi|462550|      YHYTQWPDMGVPEYALPVLTFVRRSSAARM--PETGPVLVHCSAGVGRTGTYIVIDSMLQ
gi|2499751      FHFTSWPDHGVPDTTDLLINFRYLVRDYMKQSPPESPILVHCSAGVGRTGTFIAIDRLIY
gi|1709906      FQFTAWPDHGVPEHPTPFLAFLRRVKTCNP--PDAGPMVVHCSAGVGRTGCFIVIDAMLE
gi|126471|      LHFTSWPDFGVPFTPIGMLKFLKKVKTLNP--VHAGPIVVHCSAGVGRTGTFIVIDAMMA
gi|548626|      FHFTGWPDHGVPYHATGLLSFIRRVKLSNP--PSAGPIVVHCSAGAGRTGCYIVIDIMLD
gi|131570|      FHFTGWPDHGVPYHATGLLGFVRQVKSKSP--PNAGPLVVHCSAGAGRTGCFIVIDIMLD
gi|2144715      FHFTSWPDHGVPDTTDLLINFRYLVRDYMKQSPPESPILVHCSAGVGRTGTFIAIDRLIY
                 ..* *** ***      . *                ..****** ****... ** ..
```

Conserved sites

Multiple Alignment: Naïve Approach

- **Let S(A) be the score of a multiple alignment A. The optimal multiple alignment A of sequences U_1, …, U_r can be extracted from the following dynamic programming computation of $S_{m1,\ldots,mr}$:**

$$S_{m_1,\ldots,m_r} = \max_{\epsilon_1 \in \{0,1\},\ldots,\epsilon_r \in \{0,1\}} \left\{ \begin{array}{l} S_{m_1-\epsilon_1,\ldots,m_r-\epsilon_r} + \\ s(\epsilon_1 \cdot u'_{1,m_1}, \ldots, \epsilon_r \cdot u'_{r,m_r}) \end{array} \right\}$$

where

$$\epsilon_i \cdot a = \begin{cases} a & \text{if } \epsilon_i = 1 \\ - & \text{if } \epsilon_i = 0 \end{cases}$$

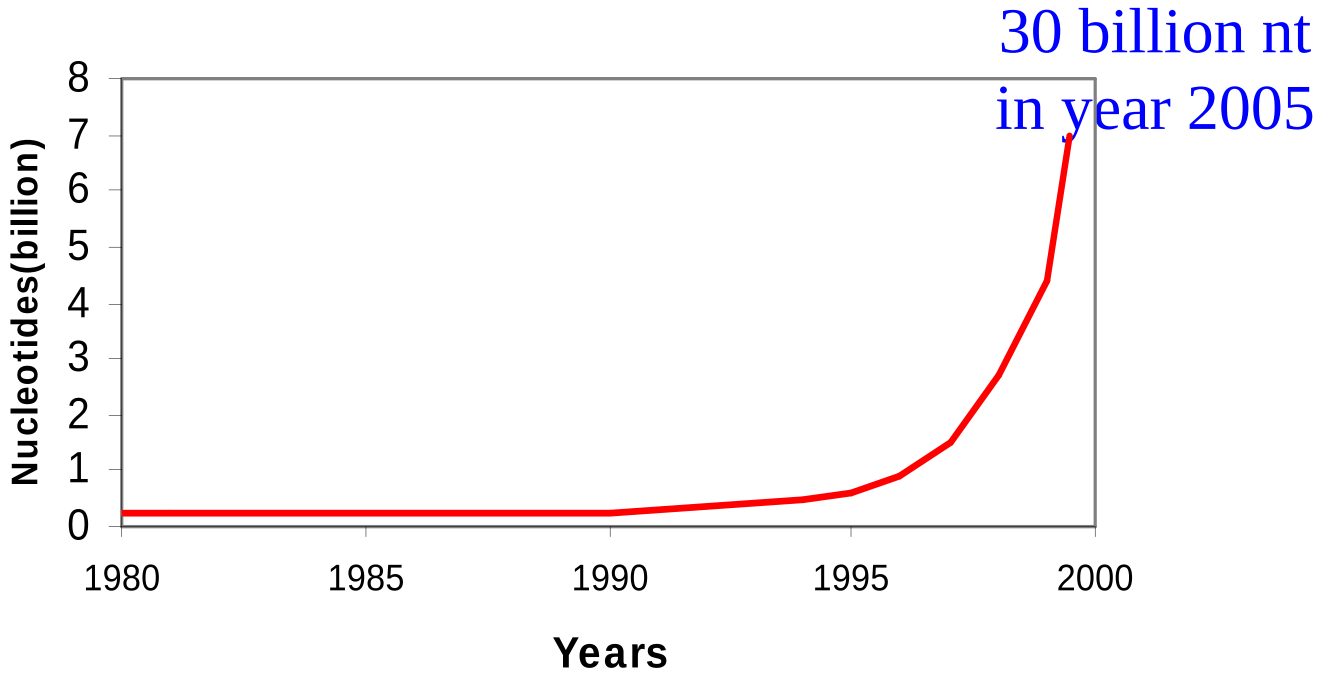
- **This requires $O(2^r)$ steps**

Exercise for the Brave:
Propose a practical approximation

Popular Tools for Sequence Comparison: FASTA, BLAST, Pattern Hunter

National University of Singapore

Scalability of Software



- **Increasing number of sequenced genomes: yeast, human, rice, mouse, fly, …**
- **S/w must be "linearly" scalable to large datasets**

Need Heuristics for Sequence Comparison

- Time complexity for optimal alignment is $O(n^2)$, where n is sequence length

$\Rightarrow$ Given current size of sequence databases, use of optimal algorithms is not practical for database search

- Heuristic techniques:
  - BLAST
  - FASTA
  - Pattern Hunter
  - MUMmer, ...

- Speed up:
  - 20 min (optimal alignment)
  - 2 min (FASTA)
  - 20 sec (BLAST)

Exercise: Describe MUMer

Basic Idea: Indexing & Filtering

- Good alignment includes short identical, or similar fragments

⇒ Break entire string into substrings, index the substrings

⇒ Search for matching short substrings and use as seed for further analysis

⇒ Extend to entire string find the most significant local alignment segment

BLAST in 3 Steps

Altschul et al, *JMB* 215:403-410, 1990

- **Similarity matching of words (3 aa's, 11 bases)**
  - No need identical words
- **If no words are similar, then no alignment**
  - Won't find matches for very short sequences
- **MSP: Highest scoring pair of segments of identical length. A segment pair is locally maximal if it cannot be improved by extending or shortening the segments**
- **Find alignments w/ optimal max segment pair (MSP) score**
- **Gaps not allowed**
- **Homologous seqs will contain a MSP w/ a high score; others will be filtered out**

BLAST in 3 Steps

Altschul et al, *JMB* 215:403-410, 1990

Step 1

- **For the query, find the list of high scoring words of length w**

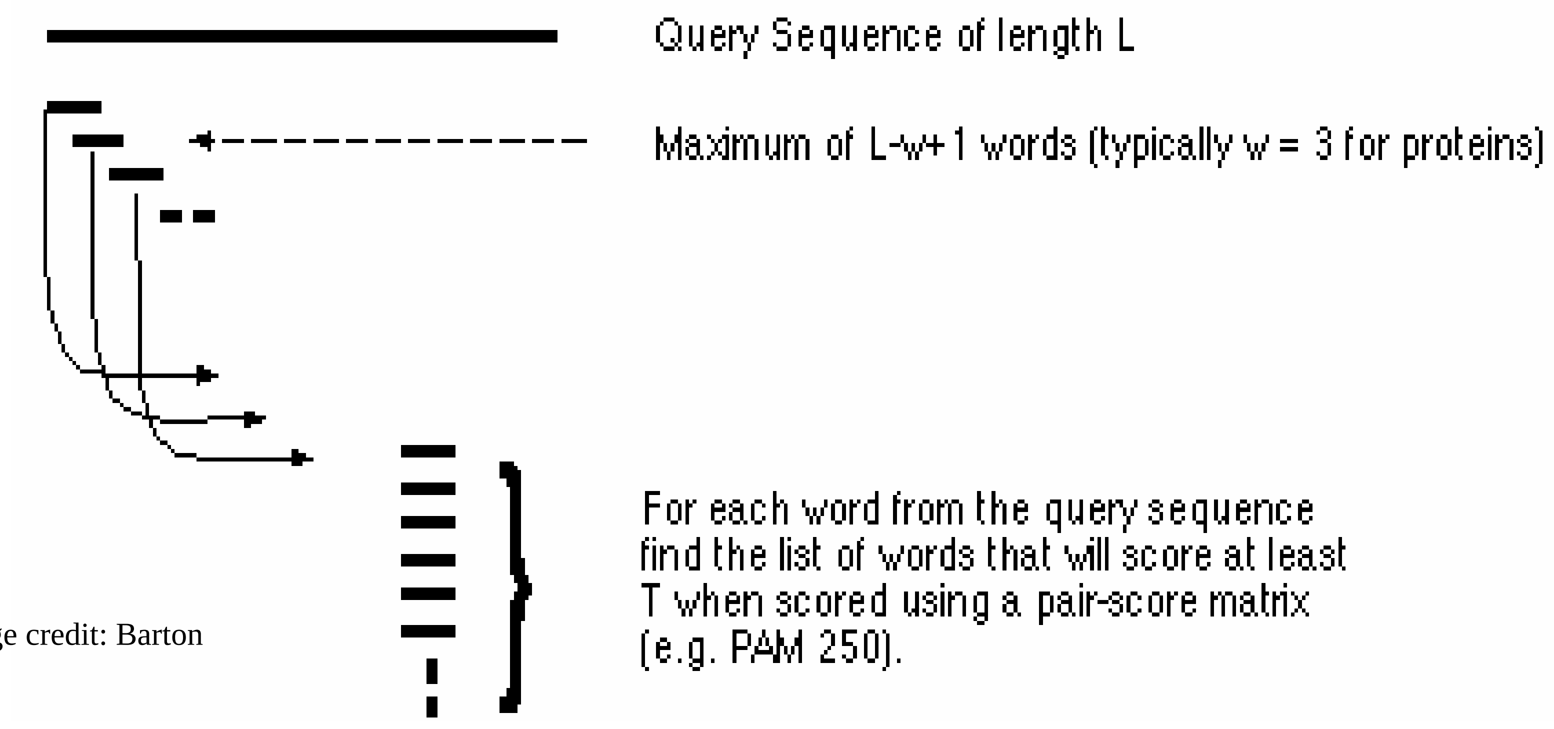

Query Sequence of length L
Maximum of L-w+1 words (typically w = 3 for proteins)
For each word from the query sequence find the list of words that will score at least T when scored using a pair-score matrix (e.g. PAM 250).

Image credit: Barton

BLAST in 3 Steps

Altschul et al, *JMB* 215:403-410, 1990

Step 2

- **Compare word list to db & find exact matches**

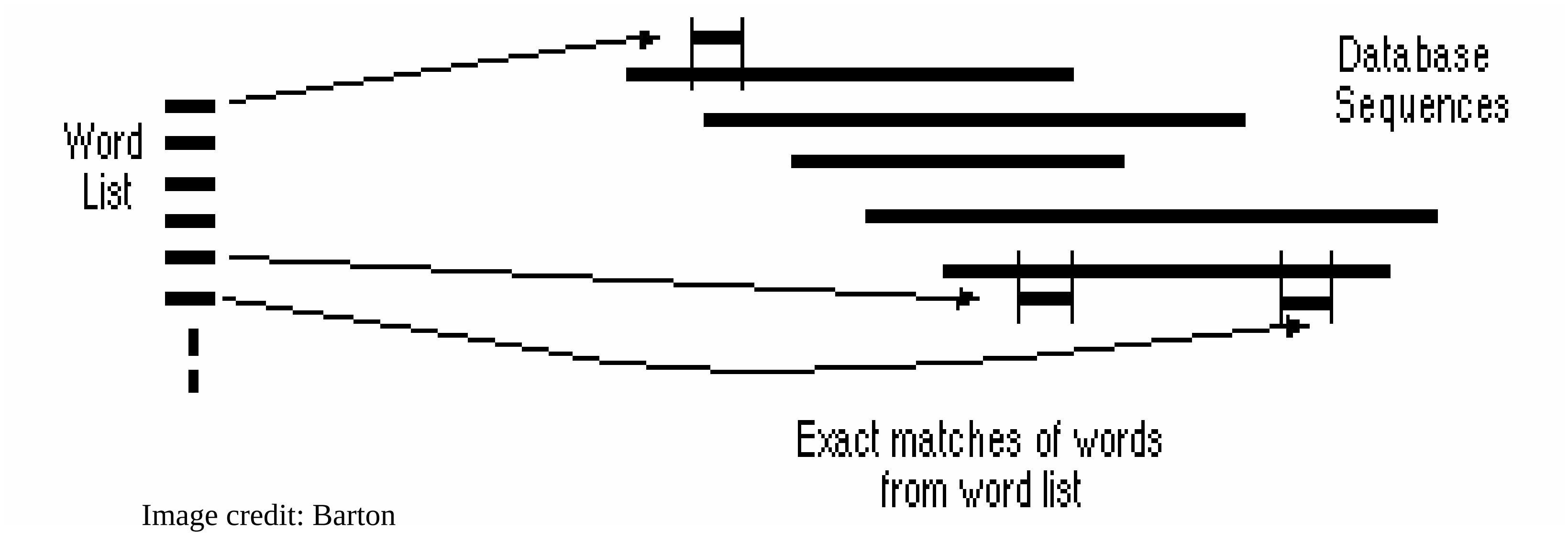


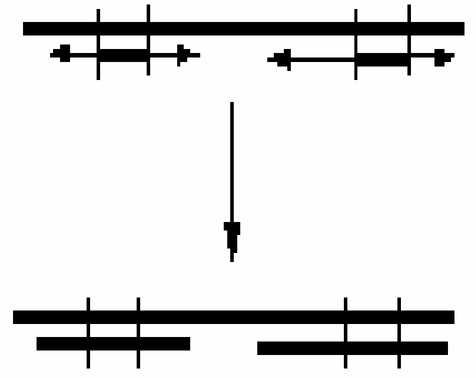
Image credit: Barton

BLAST in 3 Steps

Altschul et al, *JMB* 215:403-410, 1990

Step 3

- For each word match, extend alignment in both directions to find alignment that score greater than a threshold s



Maximal Segment Pairs (MSPs)

Image credit: Barton

Spaced Seeds

- **111010010100110111 is an example of a spaced seed model with**
  - 11 required matches (weight=11)
  - 7 “don’t care” positions

```
GAGTACTCAACACCAACATTAGTGGCAATGGAAAAT…
|| ||||||||| ||||| || |||||   ||||||
GAATACTCAACAGCAACACTAATGGCAGCAGAAAAT…
       111010010100110111
```

- **11111111111 is the BLAST seed model for comparing DNA seqs**

Observations on Spaced Seeds

- **Seed models w/ different shapes can detect different homologies**
  - the 3rd base in a codon “wobbles” so a seed like 110110110… should be more sensitive when matching coding regions

⇒ **Some models detect more homologies**
  - More sensitive homology search
  - PatternHunter I

⇒ **Use >1 seed models to hit more homologies**
  - Approaching 100% sensitive homology search
  - PatternHunter II

Exercise: Why does the 3rd base wobbles?

PatternHunter I

Ma et al., *Bioinformatics* 18:440-445, 2002

- **BLAST's seed usually uses more than one hits to detect one homology**

⇒ **Wasteful**

- **Spaced seeds uses fewer hits to detect one homology**

⇒ **Efficient**

```
TTGACCTCACC?
|||||||||||?
TTGACCTCACC?
11111111111
 11111111111
```

1/4 chances to have 2nd hit next to the 1st hit

```
CAA?A??A?C??TA?TGG?
|||?|??|?|??||?|||?
CAA?A??A?C??TA?TGG?
111010010100110111
 111010010100110111
```

$1/4^6$ chances to have 2nd hit next to the 1st hit

PatternHunter I

Ma et al., *Bioinformatics* 18:440-445, 2002

Proposition. The expected number of hits of a weight-*W* length-*M* model within a length-*L* region of similarity *p* is *$(L - M + 1) * p^W$*

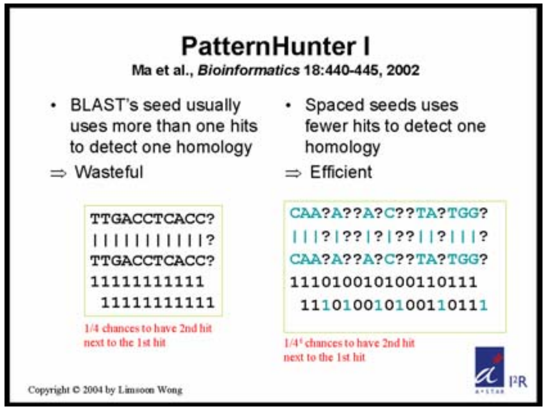
Proof.

For any fixed position, the prob of a hit is p^W.

There are $L - M + 1$ candidate positions.

The proposition follows.

Implication



PatternHunter I

Ma et al., *Bioinformatics* 18:440-445, 2002

- BLAST's seed usually uses more than one hits to detect one homology

$\Rightarrow$ Wasteful

```
TTGACCTCACC?
|||||||||||?
TTGACCTCACC?
11111111111
 11111111111
```

1/4 chances to have 2nd hit next to the 1st hit

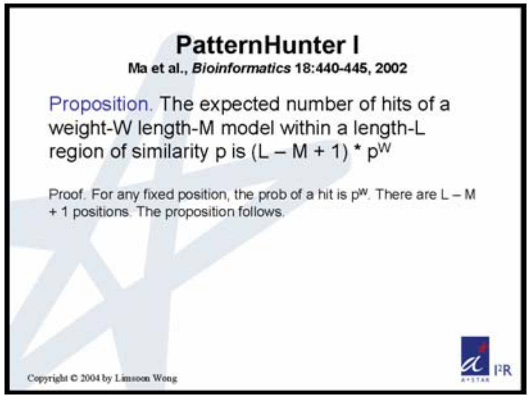
- Spaced seeds uses fewer hits to detect one homology

$\Rightarrow$ Efficient

```
CAA?A??A?C??TA?TGG?
|||?|??|?|??||?|||?
CAA?A??A?C??TA?TGG?
111010010100110111
 111010010100110111
```

$1/4^6$ chances to have 2nd hit next to the 1st hit

Copyright © 2004 by Limsoon Wong



PatternHunter I

Ma et al., *Bioinformatics* 18:440-445, 2002

Proposition. The expected number of hits of a weight-W length-M model within a length-L region of similarity p is $(L – M + 1) * p^W$

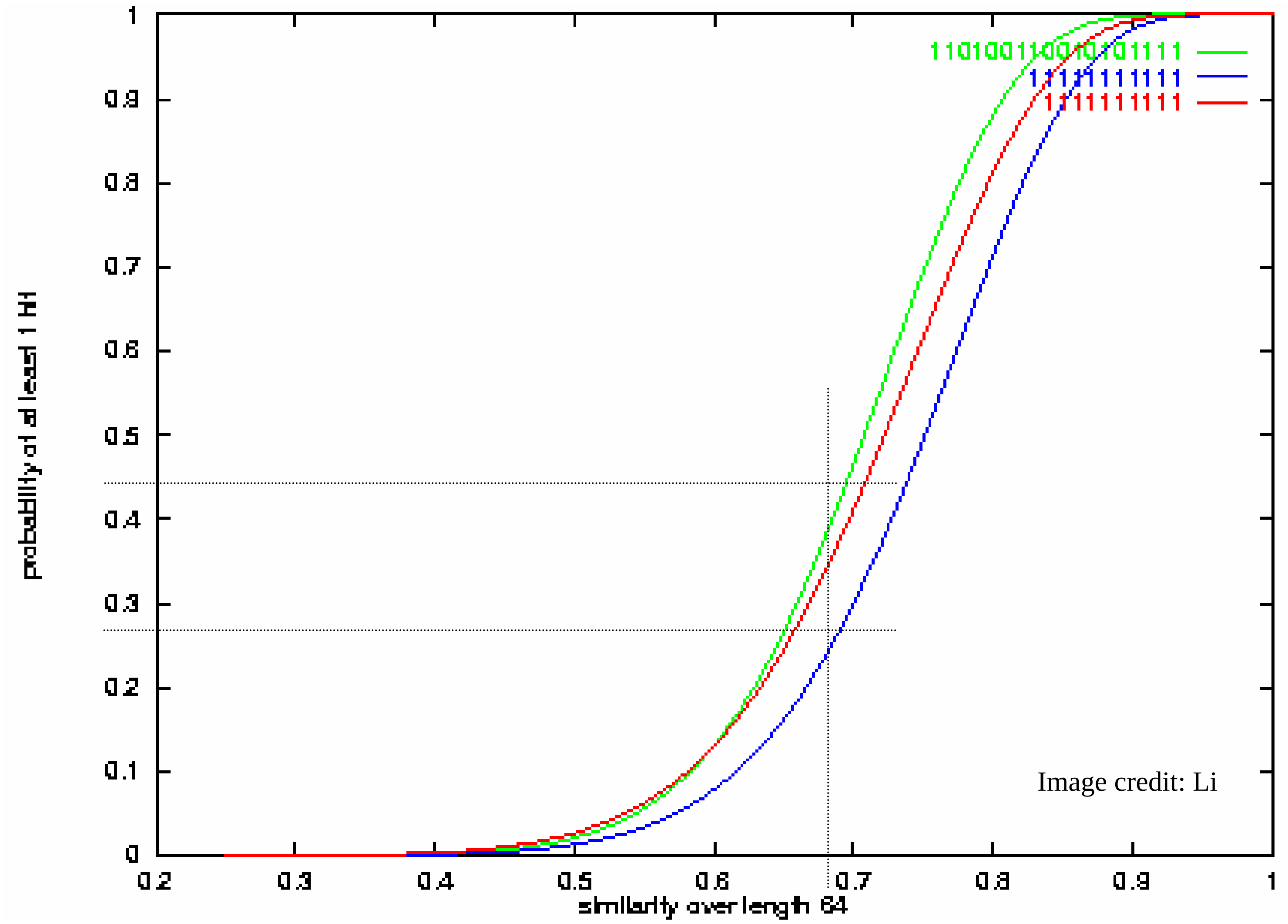
Proof. For any fixed position, the prob of a hit is p^W. There are L – M + 1 positions. The proposition follows.

Copyright © 2004 by Limsoon Wong

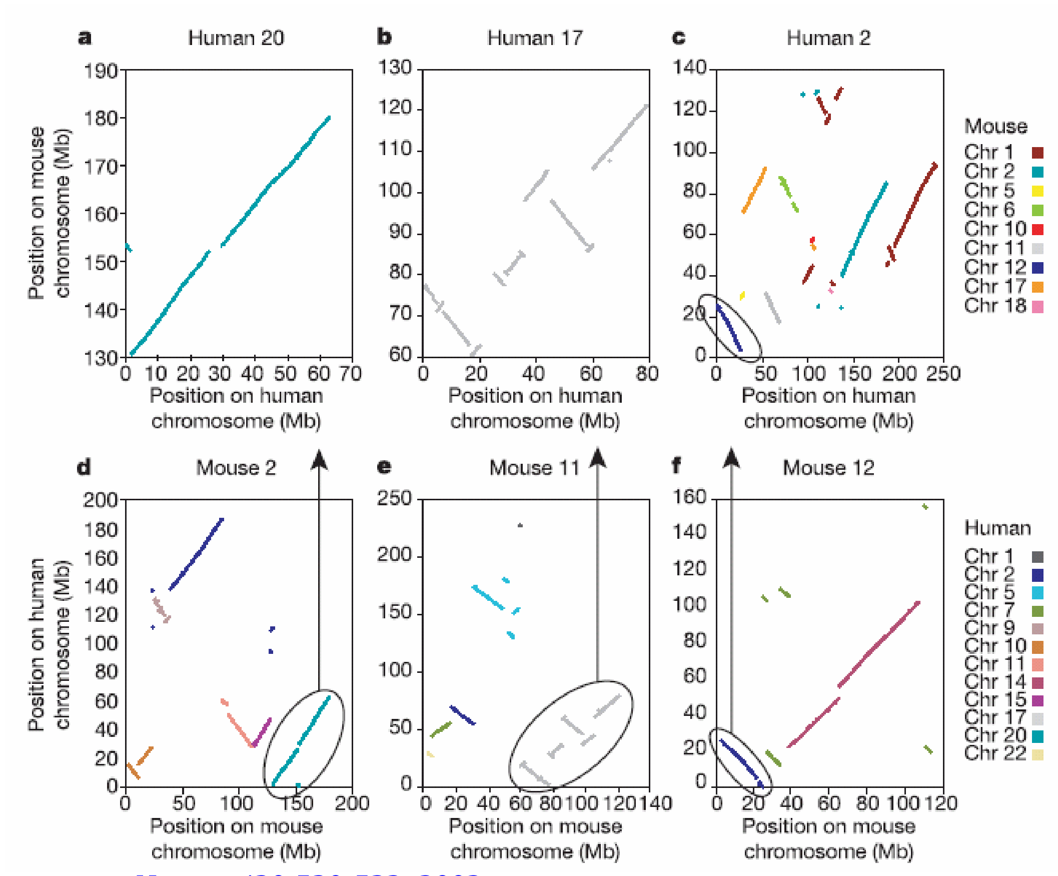
- **For *L = 1017***
  - BLAST seed expects *$(1017 – 11 + 1) * p^{11}$ = $1007 * p^{11}$* hits
  - But ~*1/4* of these overlap each other. So likely to have only ~*$750 * p^{11}$* distinct hits
  - Our example spaced seed expects *$(1017 – 18 + 1) * p^{11}$ = $1000 * p^{11}$* hits
  - But only *$1/4^6$* of these overlap each other. So likely to have ~*$1000 * p^{11}$* distinct hits

Spaced seeds likely to be more sensitive & more efficient

Sensitivity of PatternHunter I



Speed of PatternHunter I



Nature, 420:520-522, 2002

- **Mouse Genome Consortium used PatternHunter to compare mouse genome & human genome**

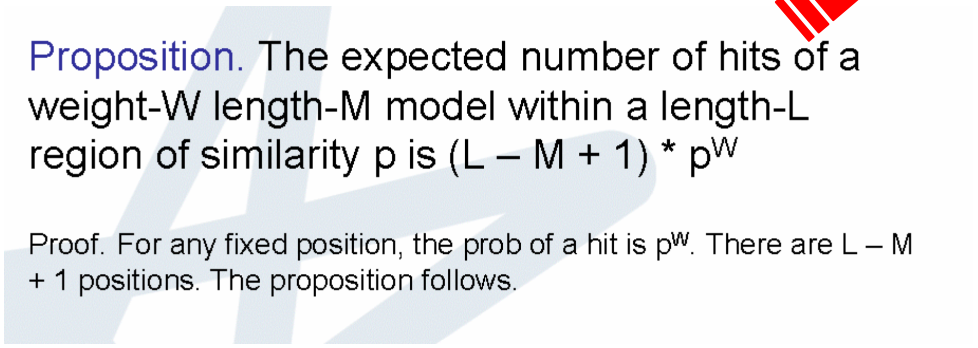
- **PatternHunter did the job in a 20 CPU-days --- it would have taken BLAST 20 CPU-years!**

How to Increase Sensitivity?

- **Ways to increase sensitivity:**
  - “Optimal” seed
  - Reduce weight by 1
  - Increase number of spaced seeds by 1
- **Intuitively, for DNA seq,**
  - Reducing weight by 1 will increase number of matches 4 folds
  - Doubling number of seeds will increase number of matches 2 folds
- **Is this really so?**

How to Increase Sensitivity?

- **Ways to increase sensitivity:**
  - "Optimal" seed
  - Reduce weight by 1
  - Increase number of spaced seeds by 1



Proposition. The expected number of hits of a weight-W length-M model within a length-L region of similarity p is (L – M + 1) * p^W

Proof. For any fixed position, the prob of a hit is p^W. There are L – M + 1 positions. The proposition follows.

- **For *L = 1017* & *p = 50%***
  - 1 weight-11 length-18 model expects *$1000/2^{11}$* hits
  - 2 weight-12 length-18 models expect *$2 * 1000/2^{12} = 1000/2^{11}$* hits

$\Rightarrow$ When comparing regions w/ >50% similarity, using 2 weight-12 spaced seeds together is more sensitive than using 1 weight-11 spaced seed!

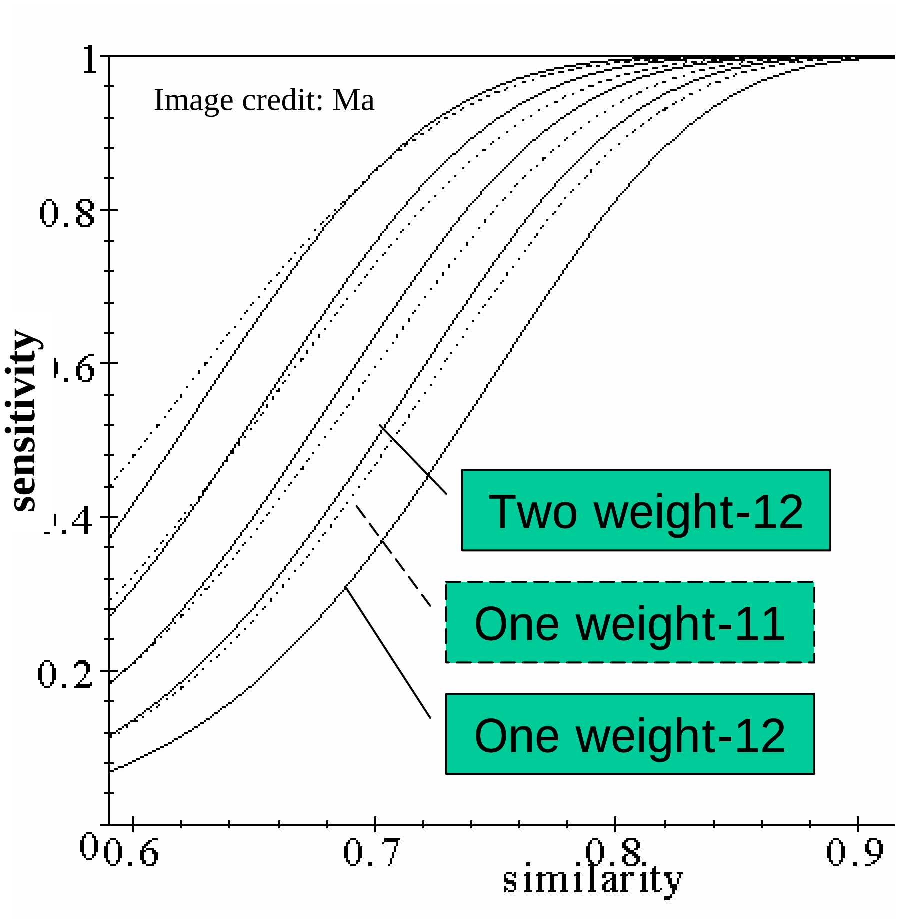
Exercise: Proof this claim

PatternHunter II

Li et al, *GIW*, 164-175, 2003

- **Idea**
  - Select a group of spaced seed models
  - For each hit of each model, conduct extension to find a homology
- **Selecting optimal multiple seeds is NP-hard**
- **Algorithm to select multiple spaced seeds**
  - Let A be an empty set
  - Let s be the seed such that A ∪ {s} has the highest hit probability
  - A = A ∪ {s}
  - Repeat until |A| = K
- **Computing hit probability of multiple seeds is NP-hard**

Sensitivity of PatternHunter II



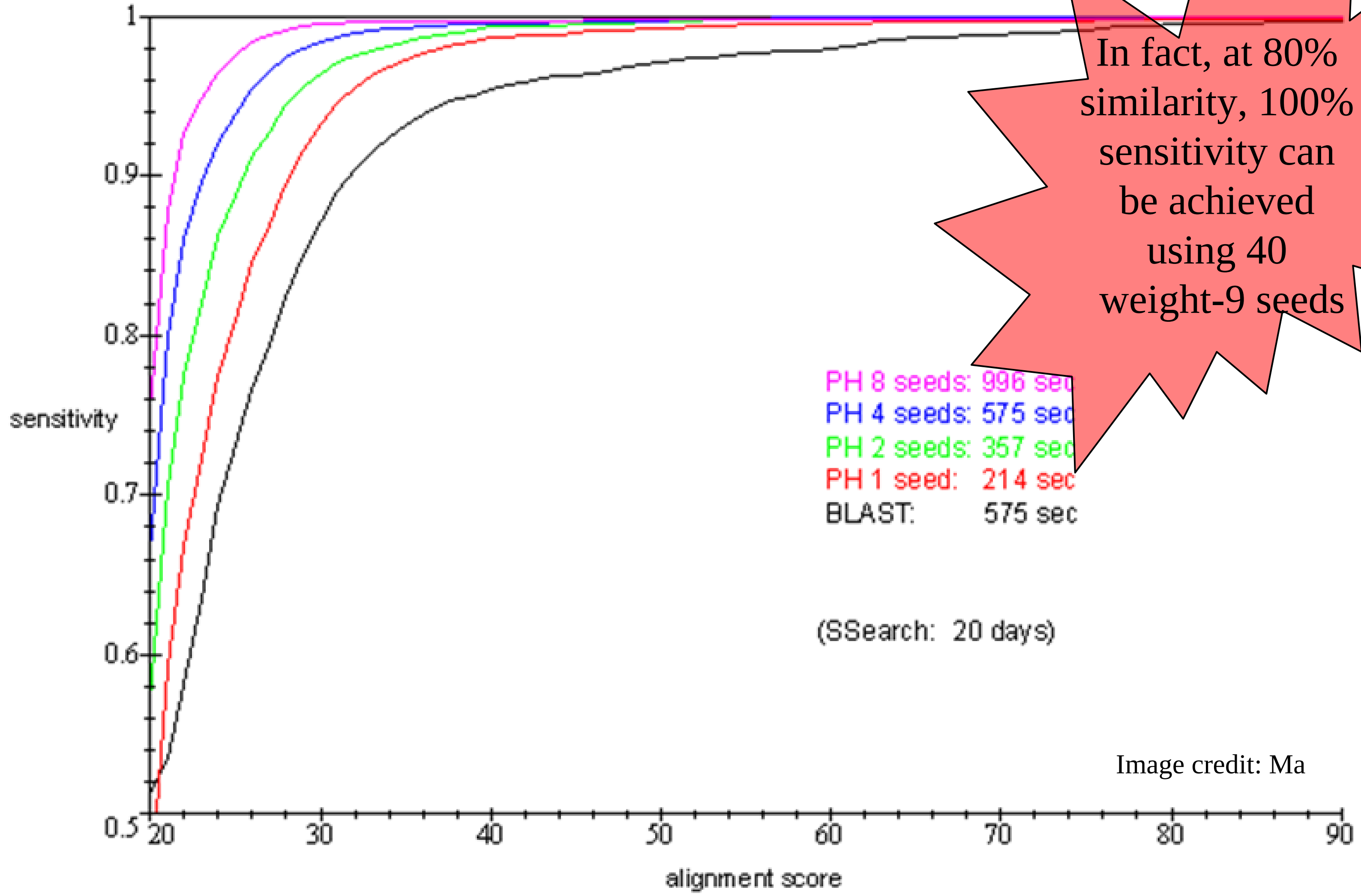
- **Solid curves: Multiple (1, 2, 4, 8,16) weight-12 spaced seeds**
- **Dashed curves: Optimal spaced seeds with weight = 11,10, 9, 8**

⇒ **"Doubling the seed number" gains better sensitivity than "decreasing the weight by 1"**

Expts on Real Data

- **30k mouse ESTs (25Mb) vs 4k human ESTs (3Mb)**
  - downloaded from NCBI genbank
  - “low complexity” regions filtered out
- **SSearch (Smith-Waterman method) finds “all” pairs of ESTs with significant local alignments**
- **Check how many percent of these pairs can be “found” by BLAST and different configurations of PatternHunter II**

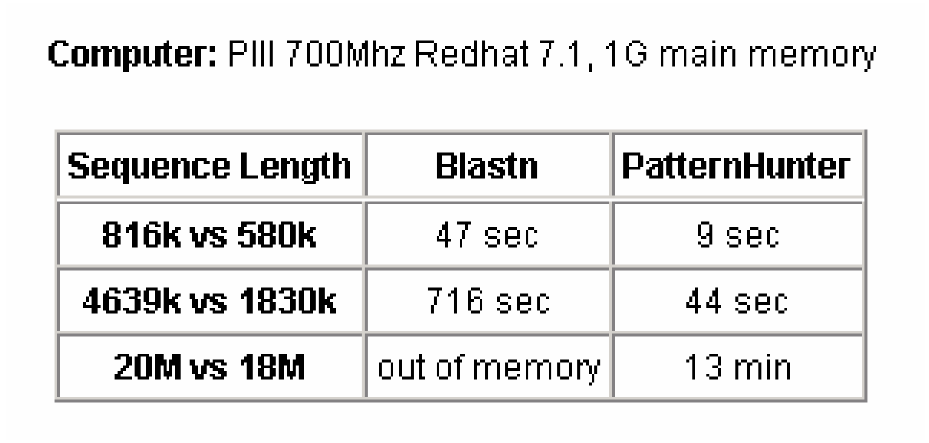
Results



In fact, at 80% similarity, 100% sensitivity can be achieved using 40 weight-9 seeds

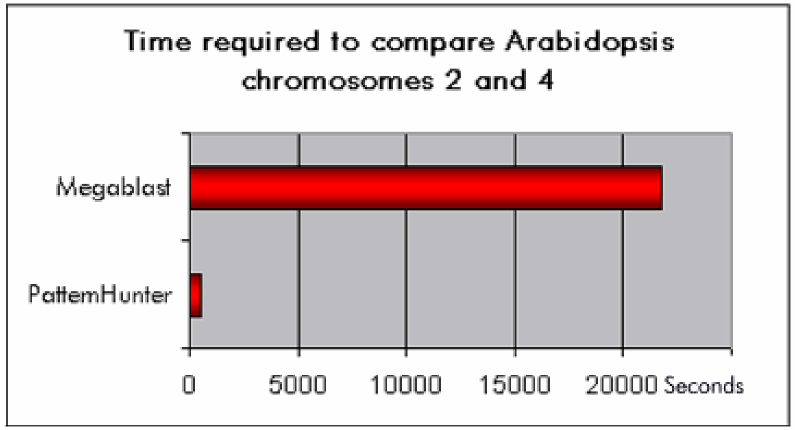
Image credit: Ma

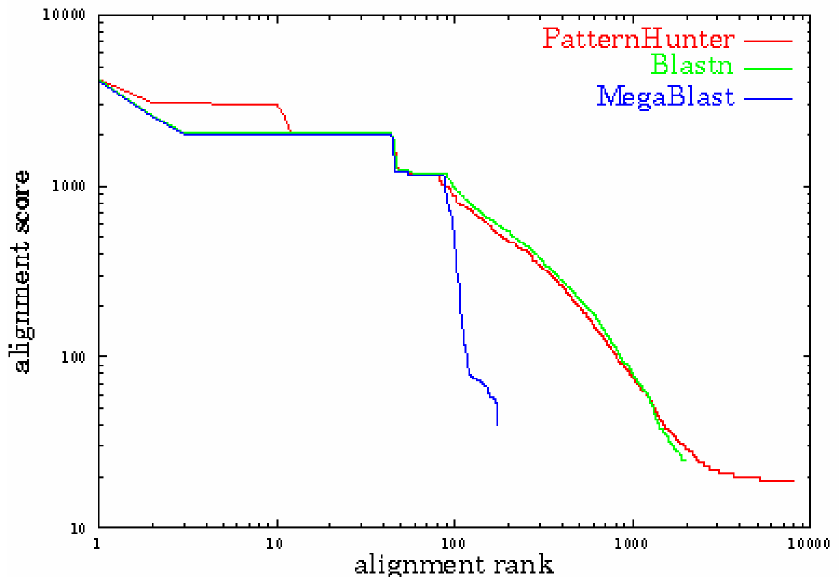
Farewell to the Supercomputer Age of Sequence Comparison!



Computer: PIII 700Mhz Redhat 7.1, 1G main memory

Sequence Length	Blastn	PatternHunter
816k vs 580k	47 sec	9 sec
4639k vs 1830k	716 sec	44 sec
20M vs 18M	out of memory	13 min





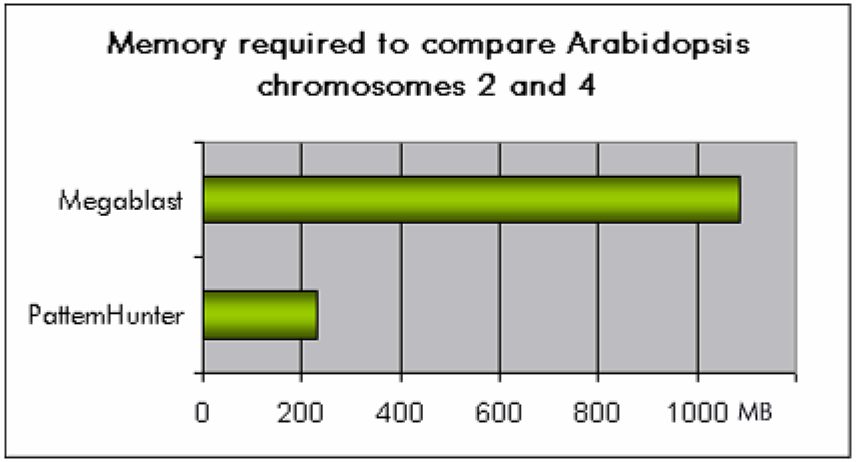


Image credit: Bioinformatics Solutions Inc

More Advanced Sequence Comparison Methods

PHI-BLAST

Iterated BLAST

PSI-BLAST

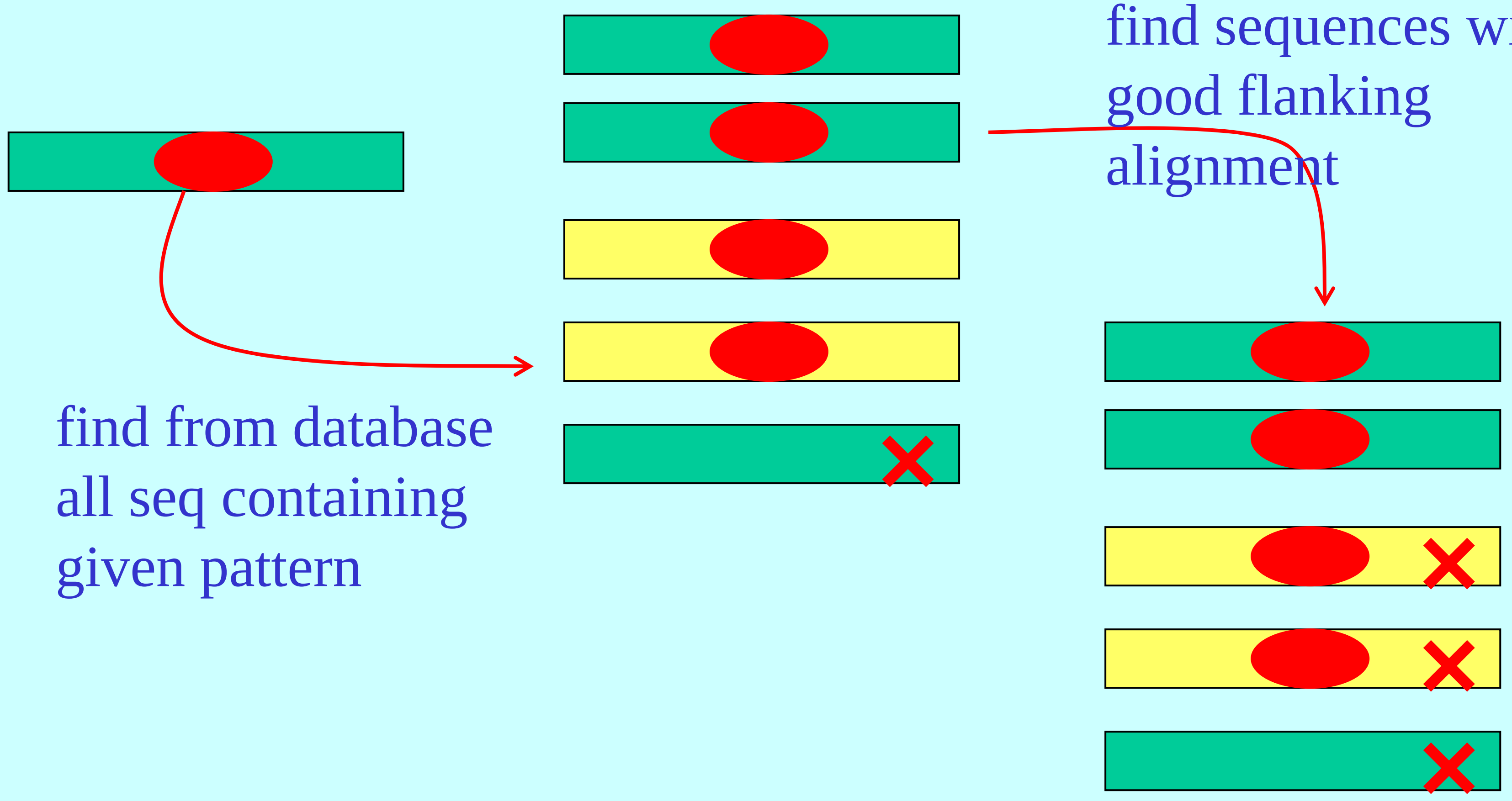
SAM

NUS
National University of Singapore

PHI-BLAST (Pattern-Hit Initiated BLAST)

- **Input**
  - protein sequence and
  - pattern of interest that it contains
- **Output**
  - protein sequences containing the pattern and have good alignment surrounding the pattern
- **Impact**
  - able to detect statistically significant similarity between homologous proteins that are not recognizably related using traditional one-pass methods

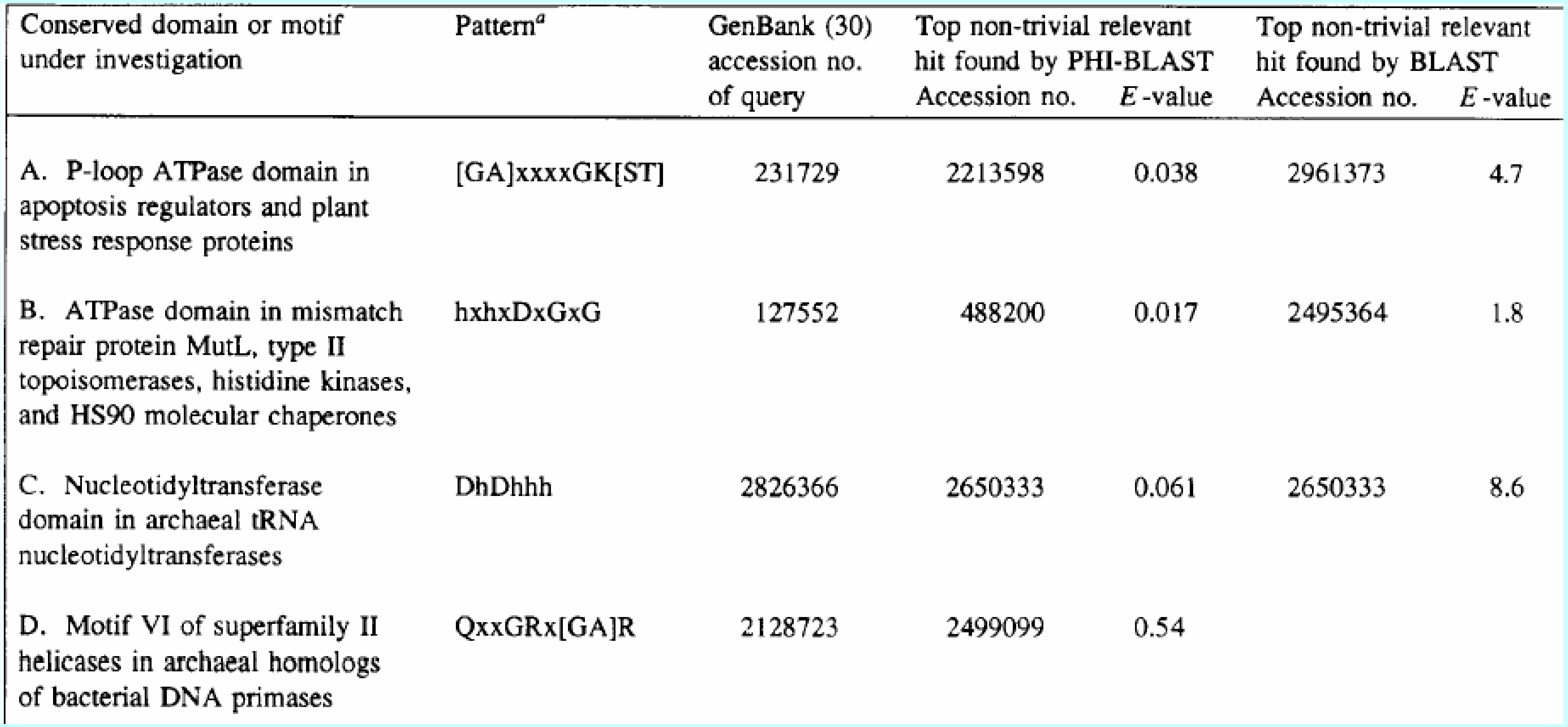
PHI-BLAST: How it works



find from database all seq containing given pattern

find sequences with good flanking alignment

PHI-BLAST: IMPACT

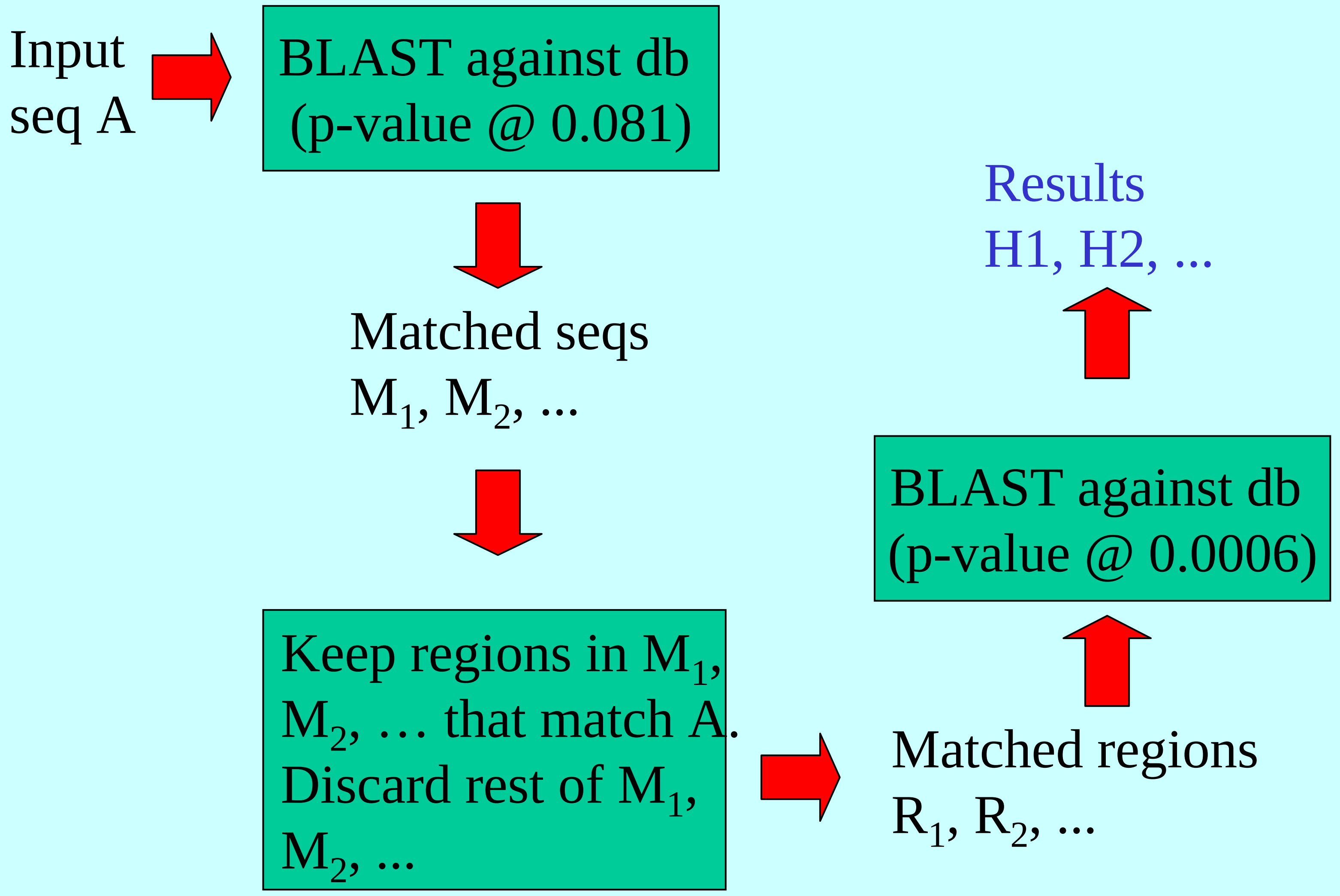


Conserved domain or motif under investigation	Pattern[a]	GenBank (30) accession no. of query	Top non-trivial relevant hit found by PHI-BLAST		Top non-trivial relevant hit found by BLAST	
			Accession no.	*E*-value	Accession no.	*E*-value
A. P-loop ATPase domain in apoptosis regulators and plant stress response proteins	[GA]xxxxGK[ST]	231729	2213598	0.038	2961373	4.7
B. ATPase domain in mismatch repair protein MutL, type II topoisomerases, histidine kinases, and HS90 molecular chaperones	hxhxDxGxG	127552	488200	0.017	2495364	1.8
C. Nucleotidyltransferase domain in archaeal tRNA nucleotidyltransferases	DhDhhh	2826366	2650333	0.061	2650333	8.6
D. Motif VI of superfamily II helicases in archaeal homologs of bacterial DNA primases	QxxGRx[GA]R	2128723	2499099	0.54		

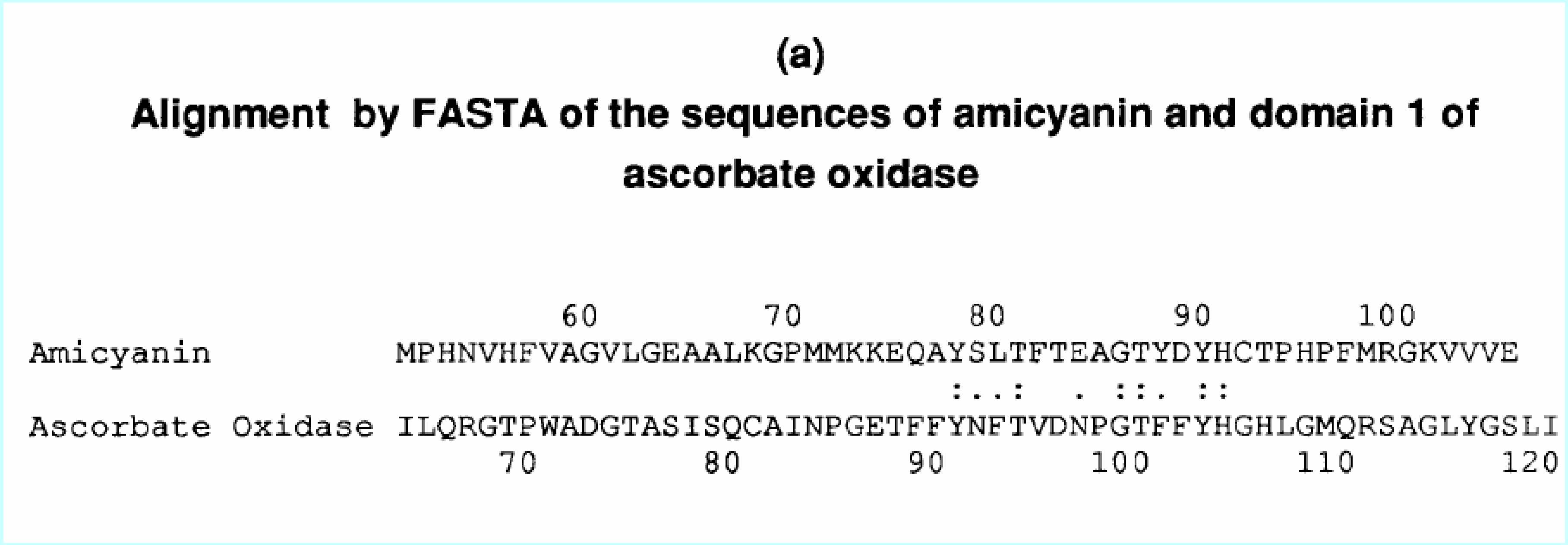
ISS (Intermediate Sequence Search)

- **Two homologous seqs, which have diverged beyond the point where their homology can be recognized by a simple direct comparison, can be related through a third sequence that is suitably intermediate between the two**
- **High score betw A & C, and betw B & C, imply A & B are related even though their own match score is low**

ISS: Search Procedure


Input seq A
BLAST against db (p-value @ 0.081)
Matched seqs M_1, M_2, ...
Keep regions in M_1, M_2, … that match A. Discard rest of M_1, M_2, ...
Matched regions R_1, R_2, ...
BLAST against db (p-value @ 0.0006)
Results H1, H2, ...

ISS: IMPACT



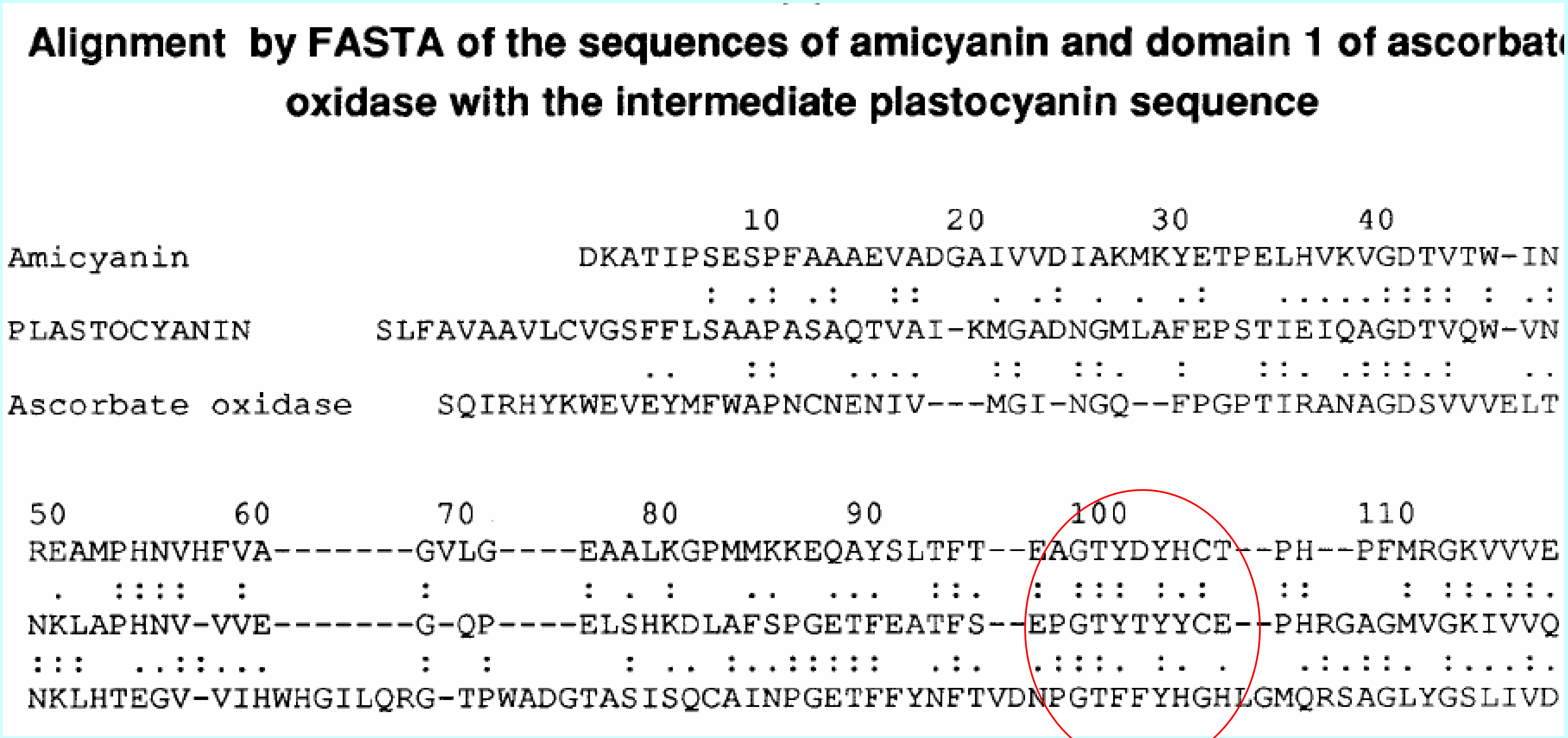
(a)

Alignment by FASTA of the sequences of amicyanin and domain 1 of ascorbate oxidase

```
                          60        70        80        90       100
Amicyanin         MPHNVHFVAGVLGEAALKGPMMKKEQAYSLTFTEAGTYDYHCTPHPFMRGKVVVE
                                             :..:  . ::. ::
Ascorbate Oxidase ILQRGTPWADGTASISQCAINPGETFFYNFTVDNPGTFFYHGHLGMQRSAGLYGSLI
                       70        80        90       100       110       120
```

No obvious match between
Amicyanin and Ascorbate Oxidase

ISS: IMPACT



Alignment by FASTA of the sequences of amicyanin and domain 1 of ascorbat oxidase with the intermediate plastocyanin sequence

```
                                    10        20        30        40
Amicyanin                   DKATIPSESPFAAAEVADGAIVVDIAKMKYETPELHVKVGDTVTW-IN
                                  : .: .:  ::   . .: . . .:   .....:::: : .:
PLASTOCYANIN      SLFAVAAVLCVGSFFLSAAPASAQTVAI-KMGADNGMLAFEPSTIEIQAGDTVQW-VN
                               ..   ::   ....   ::  ::.  :   ::. .:::.:   ..
Ascorbate oxidase    SQIRHYKWEVEYMFWAPNCNENIV---MGI-NGQ--FPGPTIRANAGDSVVVELT


50        60        70        80        90         100           110
REAMPHNVHFVA-------GVLG----EAALKGPMMKKEQAYSLTFT--EAGTYDYHCT--PH--PFMRGKVVVE
 .  ::::  :        :       : . :   ..  ...  ::.  : ::: :.:   ::    : ::.::.
NKLAPHNV-VVE-------G-QP----ELSHKDLAFSPGETFEATFS--EPGTYTYYCE--PHRGAGMVGKIVVQ
:::  ..::...       :  :      : .. :..:::::  .:.  .:::. :. .   .:.::. :...:.
NKLHTEGV-VIHWHGILQRG-TPWADGTASISQCAINPGETFFYNFTVDNPGTFFYHGHLGMQRSAGLYGSLIVD
```

Convincing homology via Plastocyanin

Previously only this part was matched

PSI-BLAST (Position-Specific Iterated BLAST)

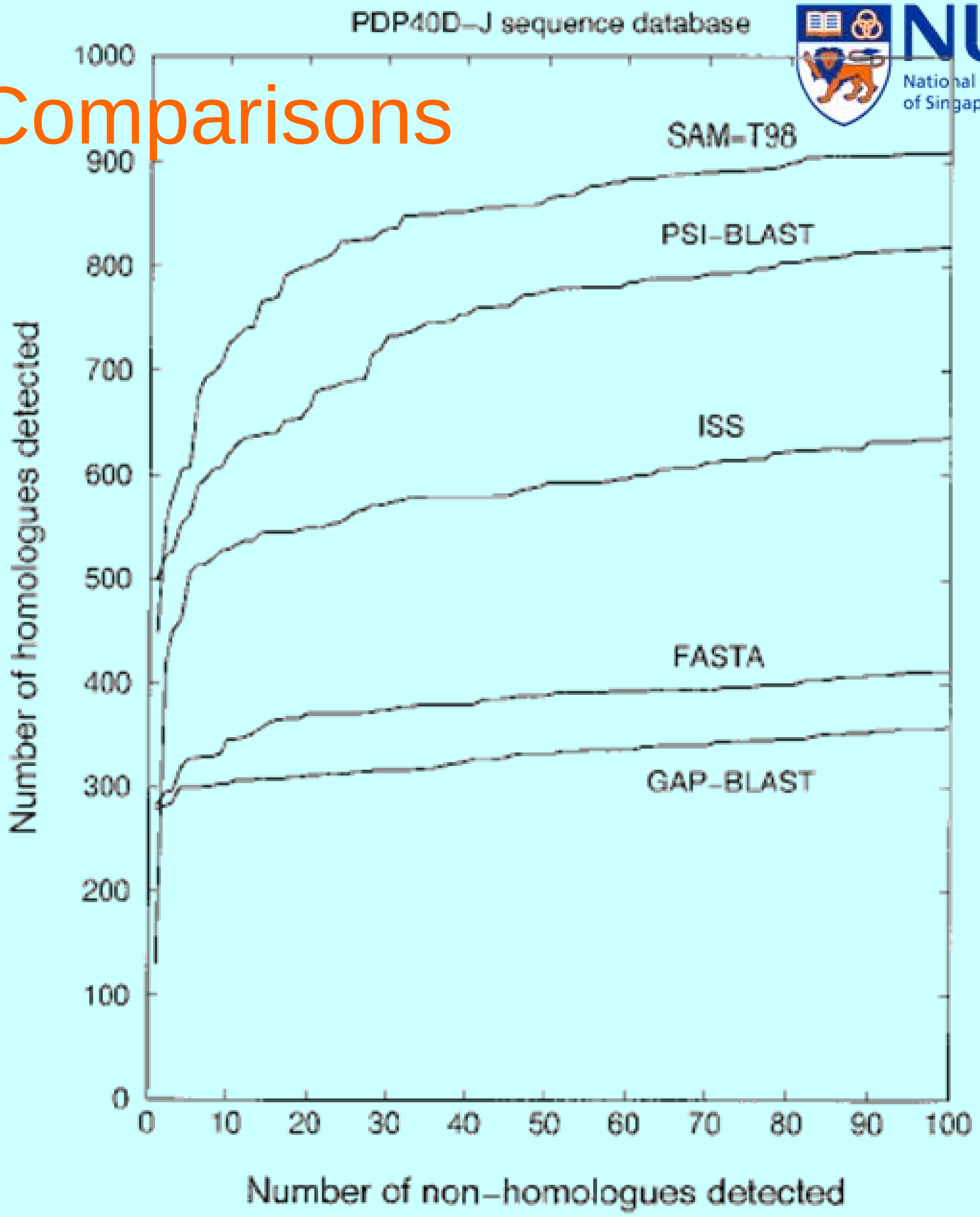
- **Given a query seq, initial set of homologs is collected from db using GAP-BLAST**
- **Weighted multiple alignment is made from query seq and homologs scoring better than threshold**
- **Position-specific score matrix is constructed from this alignment**
- **Matrix is used to search db for new homologs**
- **New homologs with good score are used to construct new position-specific score matrix**
- **Iterate the search until no new homologs found, or until specified limit is reached**

SAM-T98 HMM Method

- **Similar to PSI-BLAST**
- **But use HMM instead of position-specific score matrix**

Comparisons

Iterated seq. comparisons vs pairwise seq. comparison



Any Question?

NUS
National University of Singapore

Acknowledgements

- Some slides on popular sequence alignment tools are based on those given to me by Bin Ma and Dong Xu
- Some slides on Needleman-Wunsch and Smith-Waterman are based on those given to me by Ken Sung

References

- J. Park et al. “Sequence comparisons using multiple sequences detect three times as many remote homologs as pairwise methods”, *JMB*, 284(4):1201--1210, 1998
- J. Park et al. “Intermediate sequences increase the detection of homology between sequences”, *JMB*, 273:349--354, 1997
- Z. Zhang et al. “Protein sequence similarity searches using patterns as seeds”, *NAR*, 26(17):3986—3990, 1996
- B. Ma et al. “PatternHunter: Faster and more sensitive homology search”, *Bioinformatics*, 18:440—445, 2002
- M. Li et al. “PatternHunter II: Highly sensitive and fast homology search”, *GIW*, 164—175, 2003
- D. Brown et al. “Homology Search Methods”, *The Practical Bioinformatician*, Chapter 10, pp 217—244, WSPC, 2004

References

- S.F.Altshcul et al. “Basic local alignment search tool”, *JMB*, 215:403--410, 1990
- S.F.Altschul et al. “Gapped BLAST and PSI-BLAST: A new generation of protein database search programs”, *NAR*, 25(17):3389--3402, 1997
- S.B.Needleman, C.D.Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”, *JMB*, 48:444—453, 1970
- T.F.Smith, M.S.Waterman. “Identification of common molecular subsequences”, *JMB*, 147:195—197, 1981