


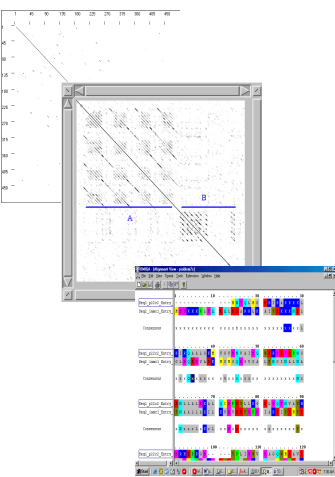
Introduction to Bioinformatics

Sami Khuri
Department of Computer Science
San José State University
San José, California, USA
khuri@cs.sjsu.edu
www.cs.sjsu.edu/faculty/khuri

@2010 Sami Khuri

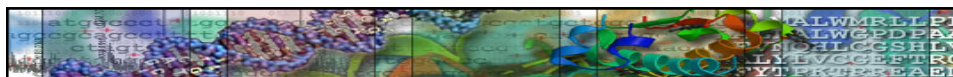


Pairwise and Multiple Sequence Alignment



- Homology
- Similarity
- Global string alignment
- Local string alignment
- Dynamic programming
- Scoring matrices:
 - PAM and BLOSUM
- BLAST family

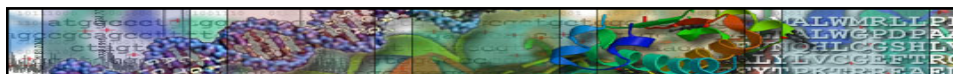
@2010 Sami Khuri



Sequence Alignment

- **Sequence alignment** is the procedure of comparing sequences by searching for a series of individual characters or character patterns that are in the same order in the sequences.
 - Comparing two sequences gives us a **pairwise alignment**.
 - Comparing more than two sequences gives us **multiple sequence alignment**.

©2010 Sami Khuri



Why Do We Align Sequences?

- The basic idea of aligning sequences is that **similar DNA sequences** generally produce **similar proteins**.
- To be able to predict the characteristics of a protein using only its sequence data, the **structure** or **function** information of known proteins with similar sequences can be used.
- To be able to check and see whether two (or more) genes or proteins are evolutionarily related to each other.

©2010 Sami Khuri



Query Sequence

If a query sequence is found to be significantly similar to an already annotated sequence (DNA or protein), we can use the information from the annotated sequence to possibly infer **gene structure** or **function** of the query sequence.

©2010 Sami Khuri



Homology and Similarity

Homology

- Evolutionary related sequence.
- A common ancestral molecular sequence.

Similarity

- Sequences that share certain sequence patterns.
- Directly observable from alignment.

©2010 Sami Khuri



Homology and Similarity

In other words:

- Sequence **similarity** is a measure of the matching of characters in an alignment.
- Sequence **homology** is a statement of common evolutionary origin.

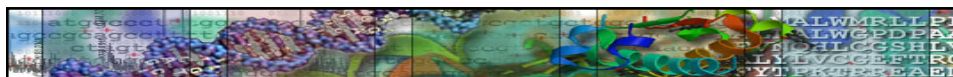
©2010 Sami Khuri



Evolution and Alignments

- Alignments reflect the **probable** evolutionary history of two sequences.
- Residues that align and that are not identical represent **substitutions**.
- Sequences without correspondence in aligned sequences are interpreted as **indels** and in an alignment are **gaps**.

©2010 Sami Khuri



Problem Definition

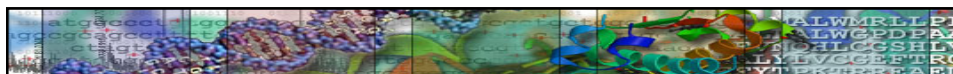
Given:

- Two sequences.
- A scoring system for evaluating match or mismatch of two characters.
- A penalty function for gaps in sequences.

Find:

- An **optimal pairing** of sequences that retains the order of characters in each sequence, perhaps introducing gaps, such that the total score is optimal.

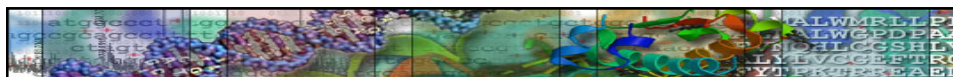
©2010 Sami Khuri



Quantifying Alignments

- How should alignments be scored?
 - Do we use +1 for a match and -1 for a mismatch?
- Should we allow **gaps** to open the sequence so as to produce better matches elsewhere in the sequence?
 - If gaps are allowed, how should they be scored?

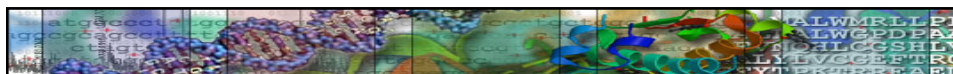
©2010 Sami Khuri



Local and Global Alignments

- Global alignment
 - find alignment in which the **total score** is highest, perhaps at the expense of areas of great local similarity.
- Local alignment
 - find alignment in which the **highest scoring subsequences** are identified, at the expense of the overall score.
 - Local alignment can be obtained by performing minor modifications to the global alignment algorithm.

©2010 Sami Khuri



Dynamic Programming

- Dynamic programming provides a reliable and optimal computational method for aligning DNA and protein sequences.
- The **optimal alignments** provide useful information to researchers, who make **functional, structural, and evolutionary predictions** of the sequences.

©2010 Sami Khuri



Dynamic Programming

A **dynamic programming** algorithm solves every subproblem just once and then saves its answer in a table, avoiding the work of recomputing the answer every time the subproblem is encountered.

Dynamic Programming reduces the amount of enumeration by avoiding the enumeration of some decision sequences that cannot possibly be optimal.

©2010 Sami Khuri



The String Alignment Problem

- A string is a sequence of characters from some alphabet.
- Given two Strings S and T ;
how **similar** are they?
- To answer this question we need to define a good "**alignment**" function between S and T .

©2010 Sami Khuri



String Alignment: An Example

Example: $S = acdbcdbc$ and $T = bcdcbcb$.

A possible alignment:

```

a c d b c d b c
b c d b c - b b
    
```

where the special character "-" represents an insertion of a space.

As for the **alignment function**, each column receives a certain value and the total score for the alignment is the sum of the values assigned to its columns.

©2010 Sami Khuri



String Alignment Function

A column $\begin{pmatrix} x \\ y \end{pmatrix}$ receives the value

- **+1** if $x = y$, i.e. we have a match
- **-1** if $x \neq y$, i.e. we have a mismatch
- **-2** if $x = -$ or $y = -$, i.e. we have a gap

Apply the above alignment function to the example:

	a	c	d	b	c	d	b	c	
	b	c	d	b	c	-	b	b	
Score	-1	1	1	1	1	-2	1	-1	= +1

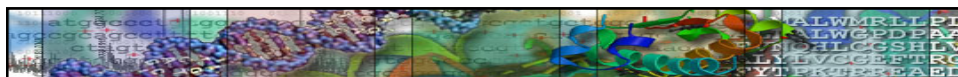
©2010 Sami Khuri



String Alignment: Remarks

- The **string alignment function**:
 - rewards matches,
 - penalizes mismatches and spaces.
- For any pairs of strings S and T and an alignment function, there are many possible alignments.
- The **string alignment problem** (SAP) consists in finding the best alignment between two strings while allowing certain mismatches.
- SAP can be solved by using Dynamic Programming.

©2010 Sami Khuri

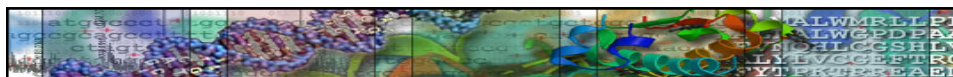


String Alignment Problem and DP

DP solves an instance of the String Alignment Problem by taking advantage of already computed solutions for smaller instances of the same problem.

- Given two sequences, S and T , instead of determining the similarity between S and T as whole sequences only, DP builds up the final solution by determining all similarities between arbitrary prefixes of S and T .
- DP starts with shorter prefixes and uses previously computed results to solve the problem for large prefixes until it finally finds the solution for S and T .

©2010 Sami Khuri



SAP: Optimal Alignments

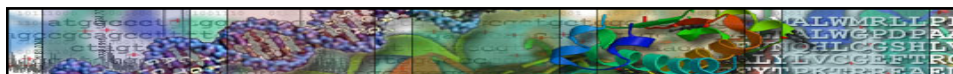
- Use the alignment function previously seen.
- Given two strings S and T over some alphabet, with $|S| = n$ and $|T| = m$.

Define $a(i, j)$ to be the value of an **optimal alignment** of strings:

$$S[1], S[2], \dots, S[i] \text{ and} \\ T[1], T[2], \dots, T[j]$$

$a(n, m)$ is the value of an **optimal alignment** of S and T .

©2010 Sami Khuri



SAP: Basis Relation

- The dynamic programming algorithm will compute each $a(i, j)$, $0 \leq i \leq n$ and $0 \leq j \leq m$, only **once**, by considering the values already computed for smaller indexes i and j .
- Define

$$a(i, 0) = \sum_{k=1}^i p(S[k], -)$$

and

$$a(0, j) = \sum_{k=1}^j p(-, T[k])$$

where p is the alignment function.

$a(i, 0)$ means that the first i characters of S are aligned with no characters of T . In other words, the i characters of S are matched with i spaces (i.e. "-"). Similarly for $a(0, j)$.

©2010 Sami Khuri



SAP: Recurrence Relation

In general:

$$a(i, j) = \max \begin{cases} a(i-1, j-1) + p(S[i], T[j]) \\ a(i-1, j) + p(S[i], -) \\ a(i, j-1) + p(-, T[j]) \end{cases}$$

$$\text{Recall: } p(S[i], T[j]) = \begin{cases} +1 & \text{if } S[i]=T[j] \\ -1 & \text{if } S[i]\neq T[j] \end{cases}$$

$$\text{and } p(S[i], -) = -2$$

$$p(-, T[j]) = -2.$$

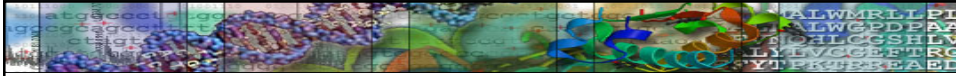
©2010 Sami Khuri



SAP: Computing $a(n,m)$

- DP uses a table of size $(n+1) \times (m+1)$.
- $a(i, j)$ corresponds to the optimal alignment of the i^{th} prefix of S with the j^{th} prefix of T .
- The dynamic programming algorithm fills in the entries of the table (matrix) by computing the values of $a(i, j)$ from top to bottom, left to right.
- The value of the optimal alignment is given by $a(n, m)$.

©2010 Sami Khuri



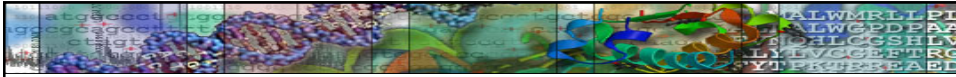
Filling Entry $a(i,j)$ in the Table

$a(i-1, j-1) + p(S[i], T[j])$	$a(i-1, j) + (-2)$
$a(i, j-1) + (-2)$	$a(i, j)$

align S[i] with “-”

align T[j] with “-”

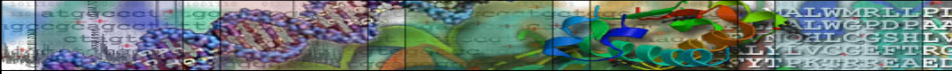
©2010 Sami Khuri



DP: Bookkeeping and Retracing

- Draw lines crossing the entries in the matrix to show from which entry in the matrix we derived the maximum score for each entry $a(i, j)$.
- To determine the solution of the optimal alignment, simply retrace the steps from entry $a(n, m)$ to entry $a(0, 0)$.

©2010 Sami Khuri




Global Alignment

Example
Align the following sequences:

S = TCCA
T = TCGCA

Solution
Use Needleman Wunsch Algorithm

©2010 Sami Khuri



Three Possible Paths

	T	C	G	C	A
T					
C					
C					
A					

Any given point in the matrix can be reached from three possible positions.

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

©2010 Sami Khuri

Computing the Score (I)

	T	C	G	C	A
T					
C					
C					
A					

$$a(i,j) = \max \begin{cases} a(i-1,j-1) + p(i,j) \end{cases}$$

	T	C	G	C	A
T					
C					
C					
A					

$$a(i,j) = \max \begin{cases} a(i-1,j-1) + p(i,j) \\ a(i-1,j) - (\text{gap_penalty}) \end{cases}$$

©2010 Sami Khuri

Computing the Score (II)

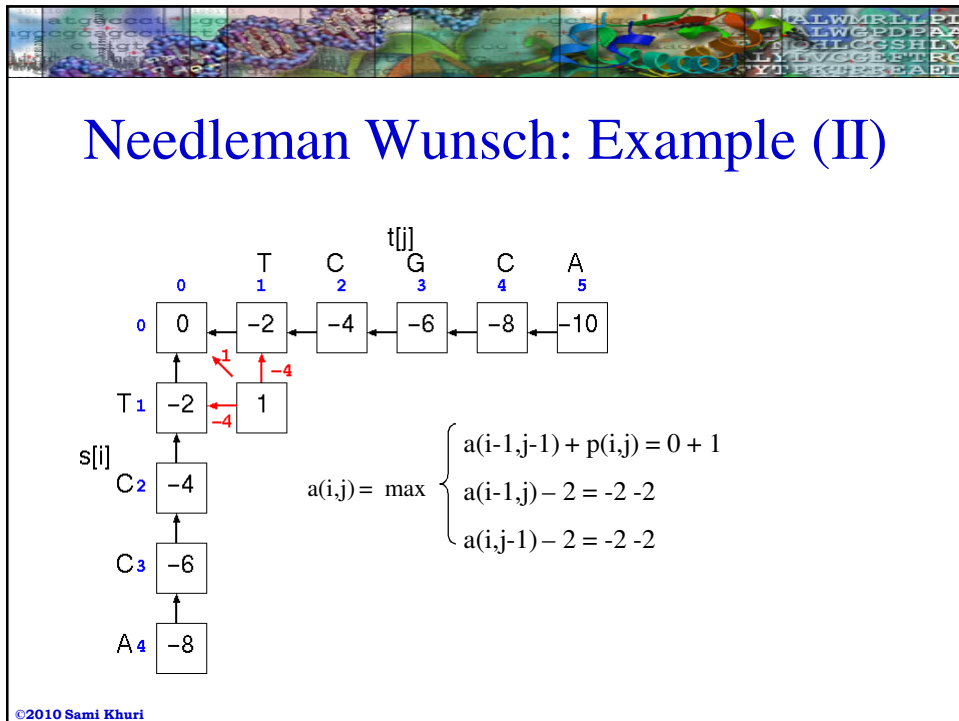
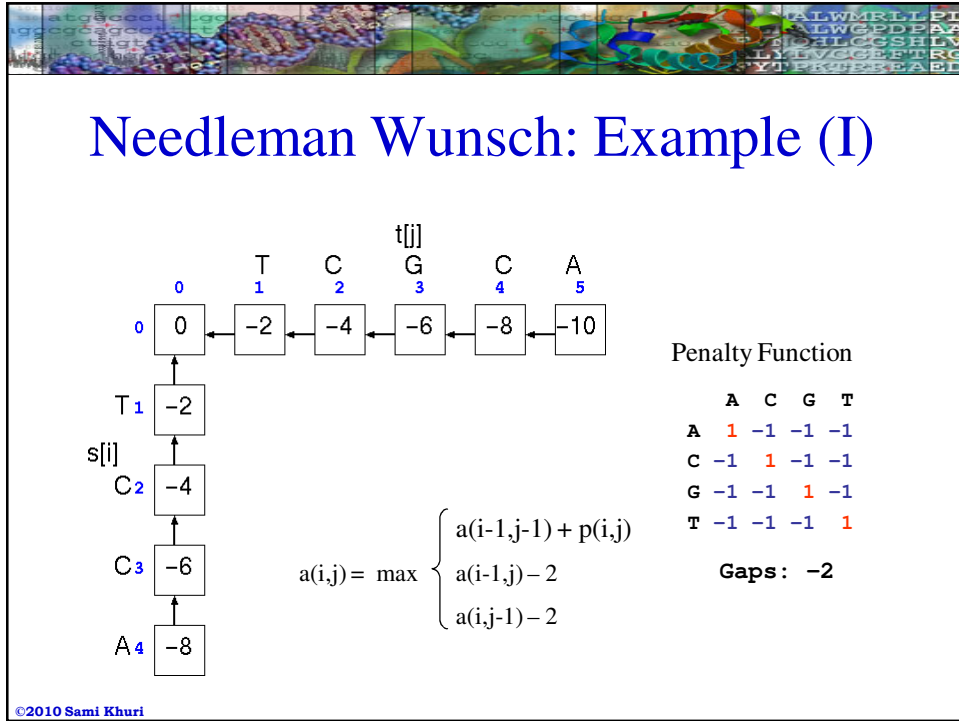
	T	C	G	C	A
T					
C					
C					
A					

$$a(i,j) = \max \begin{cases} a(i-1,j-1) + p(i,j) \\ a(i-1,j) - (\text{gap_penalty}) \\ a(i,j-1) - (\text{gap_penalty}) \end{cases}$$

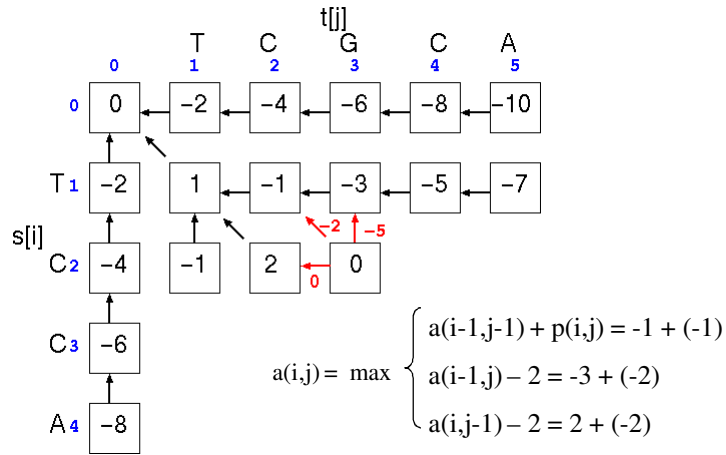
Each new score is found by choosing the maximum of three possibilities. For each square in the matrix: keep track of where the best score came from.

Fill in scores one row at a time, starting in upper left corner of matrix, ending in lower right corner.

©2010 Sami Khuri

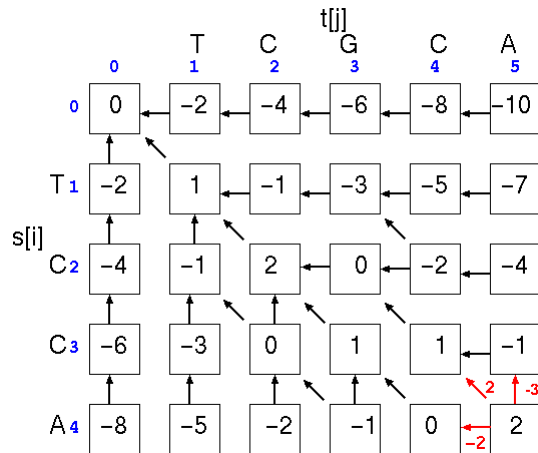


Needleman Wunsch: Example (III)



©2010 Sami Khuri

Needleman Wunsch: Example (IV)



©2010 Sami Khuri

Needleman Wunsch: Example (V)

		T	C	G	C	A
	0	-2	-4	-6	-8	-10
T	-2	1	-1	-3	-5	-7
C	-4	-1	2	0	-2	-4
C	-6	-3	0	1	1	-1
A	-8	-5	-2	-1	0	2

Solution:

```

T C - C A
: : : :
T C G C A
-----
1+1-2+1+1 = 2
                
```

©2010 Sami Khuri

Drawback of the DP for SAP

- The major drawback of dynamic programming is the fact that the table of size $(n+1) \times (m+1)$ uses $O(nm)$ space.
- It is easy to compute $a(n, m)$ in linear space since all we have to do at any given time during the computation is save two rows of the matrix, not more.
- The only values needed when computing $a(i, j)$ are found in rows i and $i-1$.
- But it is not easy to find the optimal alignment in linear space.

©2010 Sami Khuri



Sub-Optimal Alignment

- The best alignment from a biological point of view, may not be the best alignment from a computational point of view.
- The ultimate goal is to align **functional** regions.
- The software can only align regions of sequence **similarity**.
- Sub-optimal alignments may not have the best sequence alignment, but may have helical regions or active sites aligned better than the “optimal” alignment.

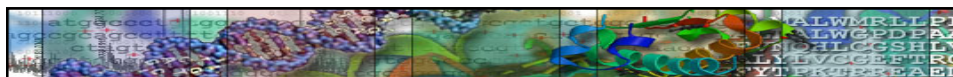
©2010 Sami Khuri



Global Alignment

- The **dynamic programming** method we studied so far was designed by Needleman and Wunsch (1970).
- Their dynamic algorithm gives a **global alignment** of sequences.
- We now turn our attention to **local alignments**.

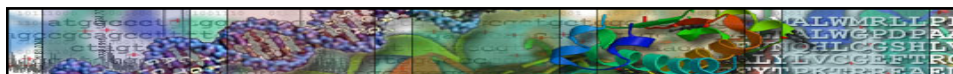
©2010 Sami Khuri



Local Alignment

- A modification of the dynamic programming algorithm for sequence alignment provides a **local sequence alignment** giving the highest-scoring local match between two sequences (Smith and Waterman 1981).
- **Local alignments** are usually more meaningful than global matches because they include patterns that are conserved in the sequences.


©2010 Sami Khuri



Local Alignment II

- The rules for calculating scoring values are slightly different with local alignment.
- The most important difference being:
 - Recall that the scoring system must include negative scores for mismatches
- With **local alignment**, when a dynamic programming scoring matrix value becomes negative, that value is set to zero, which has the effect of terminating any alignment up to that point.


©2010 Sami Khuri



Global and Local Alignments

- **Global Alignment:**
 - Are these two sequences generally the same?
- **Local Alignment:**
 - Do these two sequences contain high scoring subsequences?
- **Local similarities** may occur in sequences with different structure or function that share common substructure or subfunction.

©2010 Sami Khuri



Local Alignments

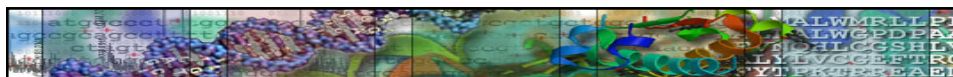
	G	A	A	C	G	T	A	G	G	C	G	T	A	T
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	1	1	0	0	1	0	0	0	0	0	1	0
T	0	0	0	0	0	1	0	0	0	0	0	1	0	2
A	0	0	1	1	0	0	2	0	0	0	0	0	2	0
C	0	0	0	0	2	0	0	1	0	1	0	0	0	1
T	0	0	0	0	0	1	1	0	0	0	0	1	0	1
A	0	0	1	1	0	0	2	0	0	0	0	0	2	0
C	0	0	0	0	2	0	0	0	0	1	0	0	0	0
G	0	1	0	0	0	3	1	0	1	1	0	2	0	0
G	0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

Thus, the best local alignment achieved from the above Dynamic Programming is:

```

A C G G A G G
A C G T A G G
    
```

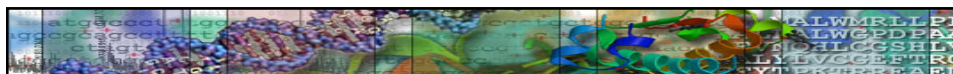
©2010 Sami Khuri



Scoring Systems

- Use of the **dynamic programming** method requires a scoring system for
 - the comparison of symbol pairs (**nucleotides** for DNA sequences & **amino acids** for protein sequences),
 - a scheme for insertion/deletion (gap) penalties.
- The most commonly used scoring systems for protein sequence alignments are the log odds form
 - of the **PAM250** matrix and
 - the **BLOSUM62** matrix.
- A number of other choices are available.

©2010 Sami Khuri



Scoring Matrices (I)

- Upon evaluating a sequence alignment, we are really interested in knowing whether the alignment is random or meaningful.
- A **scoring matrix** (table) or a **substitute matrix** (table) is a table of values that describe the probability of a residue (amino acid or base) pair occurring in an alignment.

©2010 Sami Khuri



Scoring Matrices (II)

- The alignment algorithm needs to know if it is more likely that a given amino acid pair has occurred **randomly** or that it has occurred as a result of an **evolutionary** event.
- Similar amino acids are defined by high-scoring matches between the amino acid pairs in the substitution matrix.

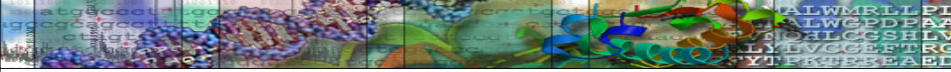
©2010 Sami Khuri



The Roles of the Scoring Matrices

The quality of the alignment between two sequences is calculated using a **scoring system** that favors the matching of related or identical amino acids and penalizes poorly matched amino acids and gaps.

©2010 Sami Khuri

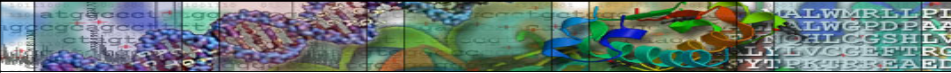


Comparison: PAM and BLOSUM Matrices

The **PAM** model is designed to track the evolutionary origins of proteins, whereas the **BLOSUM** model is designed to find their conserved domains.

BLOSUM 80	BLOSUM 62	BLOSUM 45
PAM 1	PAM 120	PAM 250
<i>Less divergent</i>	←————→	<i>More divergent</i>

©2010 Sami Khuri



BLAST

- Basic Local Alignment Search Tool
 - Altschul et al. 1990,1994,1997
- Heuristic method for local alignment
- Designed specifically for database searches
- Idea: Good alignments contain short lengths of exact matches.

©2010 Sami Khuri



The BLAST Family

- **blastp**: compares an amino acid query sequence against a protein sequence database.
- **blastn**: compares a nucleotide query sequence against a nucleotide sequence database.
- **blastx**: compares the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database.

©2010 Sami Khuri