Database Similarity Searching

A main application of pairwise alignment is retrieving biological sequences in databases based on similarity. This process involves submission of a query sequence and performing a pairwise comparison of the query sequence with all individual sequences in a database. Thus, database similarity searching is pairwise alignment on a large scale. This type of searching is one of the most effective ways to assign putative functions to newly determined sequences.

HEURISTIC DATABASE SEARCHING

Searching a large database using the dynamic programming methods, such as the Smith–Waterman algorithm, although accurate and reliable, is too slow and impracticalwhen computational resources are limited.Anestimate conducted nearly a decade ago had shownthat querying a database of 300,000 sequences using a query sequence of 100 residues took 2–3 hours to complete with a regular computer system at the time. Thus, speed of searching became an important issue. To speed up the comparison, heuristic methods have to be used. The heuristic algorithms perform faster searches because they examine only a fraction of the possible alignments examined in regular dynamic programming.

Currently, there are two major heuristic algorithms for performing database searches: BLAST and FASTA. These methods are not guaranteed to find the optimal alignment or true homologs, but are 50–100 times faster than dynamic programming. The increased computational speed comes at a moderate expense of sensitivity and specificity of the search, which is easily tolerated by working molecular biologists. Both programs can provide a reasonably good indication of sequence similarity by identifying similar sequence segments.

Both BLAST and FASTA use a heuristic *word method* for fast pairwise sequence alignment. This is the third method of pairwise sequence alignment. It works by finding short stretches of identical or nearly identical letters in two sequences. These short strings of characters are called *words*, which are similar to the windows used in the dot matrix method. The basic assumption is that two related sequences must have at least one word in common. By first identifying word matches, a longer alignment can be obtained by extending similarity regions from the words. Once regions of high sequence similarity are found, adjacent high-scoring regions can be joined into a full alignment.

BASIC LOCAL ALIGNMENT SEARCH TOOL (BLAST)

The BLAST program was developed by Stephen Altschul of NCBI in 1990 and has since become one of the most popular programs for sequence analysis. BLAST uses heuristics to align a query sequence with all sequences in a database. The objective is to find high-scoring ungapped segments among related sequences. The existence of such segments above a given threshold indicates pairwise similarity beyond random chance, which helps to discriminate related sequences from unrelated sequences in a database. BLAST performs sequence alignment through the following steps. The first step is to create a list of words from the query sequence. Each word is typically three residues for protein sequences and eleven residues for DNA sequences. The list includes every possible word extracted from the query sequence. This step is also called *seeding*. The second step is to search a sequence database for the occurrence of these words. This step is to identify database sequences containing the matching words. The matching of the words is scored by a given substitution matrix. A word is considered a match if it is above a threshold. The third step involves pairwise alignment by extending from the words in both directions while counting the alignment score using the same substitution matrix. The extension continues until the score of the alignment drops below a threshold due to mismatches. The resulting contiguous aligned segment pair without gaps is called *high-scoring segment pair*. In the original version of BLAST, the highest scored HSPs are presented as the final report. They are also called maximum scoring pairs.

Variants

BLAST is a family of programs that includes BLASTN, BLASTP, BLASTX TBLASTN, and TBLASTX. BLASTN queries nucleotide sequences with a nucleotide sequence database. BLASTP uses protein sequences as queries to search against a protein sequence database. BLASTX uses nucleotide sequences as queries and translates them in all six reading frames to produce translated protein sequences, which are used to query a protein sequence database. TBLASTN queries protein sequences to a nucleotide sequence database with the sequences translated in all six reading frames. TBLASTX uses nucleotide sequences, which are translated in all six frames, to search against a nucleotide sequence database that has all the sequences translated in six frames. In addition, there is also a bl2seq program that performs local alignment of two user-provided input sequences. The graphical output includes horizontal bars and a diagonal in a two-dimensional diagram showing the overall extent of matching between the two sequences.

1. Query: **MRDPYNKLIS**

2. Scan every three residues to be used in searching BLAST word database.

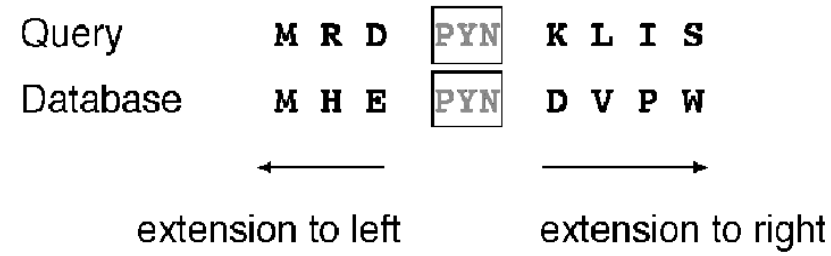
3. Assuming one of the words finds matches in the database.

Query	**PYN**	**PYN**	**PYN**	**PYN**	**...**
Database	**PYN**	**PFN**	**PFQ**	**PFE**	**...**

4. Calculate sums of match scores based on BLOSUM62 matrix.

Query	**PYN**	**PYN**	**PYN**	**PYN**	**...**
Database	**PYN**	**PFN**	**PFQ**	**PFE**	**...**
Sum of score	**20**	**16**	**10**	**10**	**...**

5. Find the database sequence corresponding to the best word match and extend alignment in both directions.



6. Determine high scored segment above threshold (22).

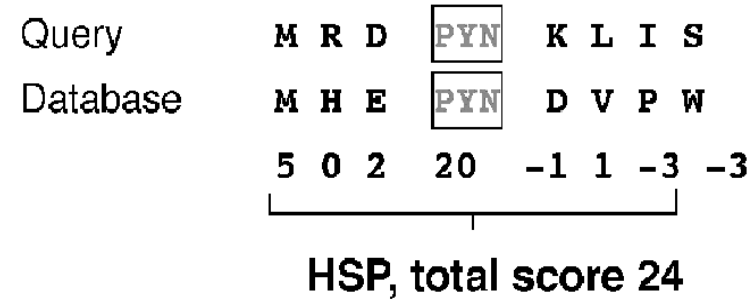


Illustration of the BLAST procedure using a hypothetical query sequence matching with a hypothetical database sequence. The alignment scoring is based on the BLOSUM62 matrix.

Statistical Significance

The BLAST output provides a list of pairwise sequence matches ranked by statistical significance. The significance scores help to distinguish evolutionarily related sequences from unrelated ones. Generally, only hits above a certain threshold are displayed.

Deriving the statistical measure is slightly different from that for single pairwise sequence alignment; the larger the database, the more unrelated sequence alignments there are. This necessitates a new parameter that takes into account the total number of sequence alignments conducted, which is proportional to the size of the database. In BLAST searches, this statistical indicator is known as the *E*-value (expectation value), and it indicates the probability that the resulting alignments from a database search are caused by random chance. The *E*-value is related to the *P*-value used to assess significance of single pairwise alignment. BLAST compares a query sequence against all database sequences, and so the *E*-value is determined by the following formula:

$E = m \times n \times P$.

Where m is the total number of residues in a database, n is the number of residues in the query sequence, and P is the probability that an HSP alignment is a result of random chance.

Low Complexity Regions

For both protein and DNA sequences, there may be regions that contain highly repetitive residues, such as short segments of repeats, or segments that are overrepresented by a small number of residues. These sequence regions are referred to as *low complexity regions* (LCRs). To avoid the problem of high similarity scores owing to matching of LCRs that obscure the real similarities, it is important to filter out the problematic regions in both the query and database sequences to improve the signal-to-noise ratio, a process known as *masking.* There are two types of masking: hard and soft. *Hard masking* involves replacing LCR sequences with an ambiguity character such as *N* for nucleotide residues or *X* for amino acid residues. The ambiguity characters are then ignored by the BLAST program. *Soft masking* involves converting the problematic sequences to lower case letters, which are ignored in constructing the word dictionary, but are used in word extension and optimization of alignments.

FASTA

FASTA was in fact the first database similarity search tool developed, preceding the development of BLAST. FASTA uses a "hashing" strategy to find matches for a short stretch of identical residues with a length of *k*.

The string of residues is known as *ktuples* or *ktups*, which are equivalent to words in BLAST, but are normally shorter than the words. Typically, a ktup is composed of two residues for protein sequences and six residues for DNA sequences.

The first step in FASTA alignment is to identify ktups between two sequences by using the hashing strategy. This strategy works by constructing a lookup table that shows the position of each ktup for the two sequences under consideration. The positional difference for each word between the two sequences is obtained by subtracting the position of the first sequence from that of the second sequence and is expressed as the offset. The ktups that have the same offset values are then linked to reveal a contiguous identical sequence region that corresponds to a stretch of diagonal in a two-dimensional matrix. The second step is to narrow down the high similarity regions between the two sequences. Normally, many diagonals between the two sequences can be identified in the hashing step. The top ten regions with the highest density of diagonals are identified as high similarity regions. The diagonals in these regions are scored using a substitution matrix. Neighboring high-scoring segments along the same diagonal are selected and joined to form a single alignment. This step allows introducing gaps between the diagonals while applying gap penalties. The score of the gapped alignment is calculated again. In step 3, the gapped alignment is refined further using the Smith–Waterman algorithm to produce a final alignment. The last step is toper form a statistical evaluation of the final alignment as in BLAST, which produces the *E*-value.

1. Given two amino acid sequences for comparision:

sequence 1 **AMPSDGL**
sequence 2 **GPSDNAT**

2. Construct a hashing table:

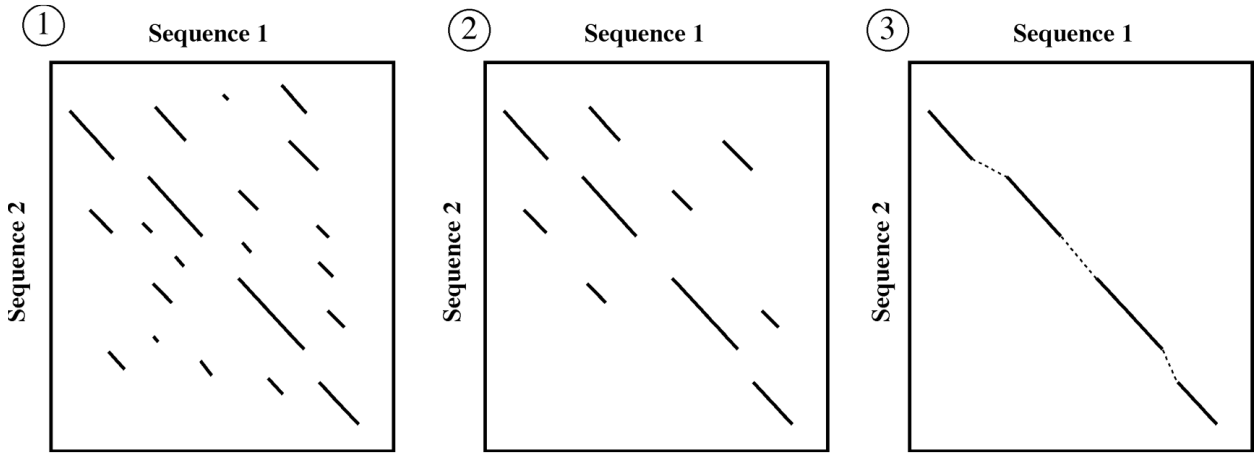
amino acid	sequence position		offset
	seq 1	seq 2	
A	1	6	-5
D	5	4	1
G	6	1	5
L	7	–	–
M	2	–	–
N	–	5	–
P	3	2	1
S	4	3	1
T	–	7	–

3. Identify residues with the same offset values (highlighted in grey).

4. Find the matching word of three residues in the order of 3, 4 and 5 in one sequence and 2, 3,and 4 in the other.

5. This allows establishment of alignment between the two sequences.

```
sequence 1   AMPSDGL-
                |||
sequence 2   -GPSDNAT
```



POSITION-SPECIFIC SCORING MATRICES

One of the applications of multiple sequence alignments in identifying related sequences in databases is by construction of position-specific scoring matrices (PSSMs), profiles, and hidden Markov models (HMMs). These are statistical models that reflect the frequency information of amino acid or nucleotide residues in a multiple alignment. The purpose of establishing the mathematical model is to allow partial matches with a query sequence so they can be used to detect more distant members of the same sequence family, resulting in an increased sensitivity of database searches.

A PSSM is defined as a table that contains probability information of amino acids or nucleotides at each position of an ungapped multiple sequence alignment. The matrix resembles the substitution matrices, but is more complex in that it contains positional information of the alignment. In such a table, the rows represent residue positions of a particular multiple alignment and the columns represent the names of residues or vice versa. The values in the table represent log odds scores of the residues calculated from the multiple alignments. To construct a matrix, raw frequencies of each residue at each column position from a multiple alignment are first counted. The frequencies are normalized by dividing positional frequencies of each residue by overall frequencies so that the scores are length and composition independent. The values are converted to the probability values by taking to the logarithm (normally to the base of 2). In this way, the matrix values become log odds scores of residues occurring at each alignment position. In this matrix, a positive score represents identical residue or similar residue match; a negative score represents a no conserved sequence match. This constructed matrix can be considered a distilled representation for the entire group of related sequences, providing a quantitative description of the degree of sequence conservation at each position of a multiple alignment. The probabilistic model can then be used like a single sequence for database searching and alignment or can be used to test how well a particular target sequence fits into the sequence group. The probability values in a PSSM depend on the number of sequences used to compile the matrix. Because the matrix is often constructed from the alignment of an insufficient number of closely related sequences, to increase the predictive power of the model, a weighting scheme similar to the one used in the Clustal algorithm is used that down weights overrepresented, closely related sequences and up weights underrepresented and divergent ones, so that more divergent sequences can be included. Application of such a weighting scheme makes the matrix less biased and able to detect more distantly related sequences.

Position	1 2 3 4 5 6
Sequence 1	**A T G T C G**
Sequence 2	**A A G A C T**
Sequence 3	**T A C T C A**
Sequence 4	**C G G A G G**
Sequence 5	**A A C C T G**

Convert multiple alignment to a raw frequency table

Pos.	1	2	3	4	5	6	Overall freq.
A	0.6	0.6	—	0.4	—	0.2	0.30
T	0.2	0.2	—	0.4	0.2	0.2	0.20
G	—	0.2	0.6	—	0.2	0.6	0.27
C	0.2	—	0.4	0.2	0.6	—	0.23

Normalize the values by dividing them by overall freq.

Pos.	1	2	3	4	5	6	Overall freq.
A	2.0	2.0	—	1.33	—	0.67	0.30
T	1.0	1.0	—	2.0	1.0	1.0	0.20
G	—	0.74	2.22	—	0.74	2.22	0.27
C	0.87	—	1.74	0.87	2.61	—	0.23

Convert the values to log to base of 2

Pos.	1	2	3	4	5	6
A	1.0	1.0	—	0.41	—	-0.58
T	0.0	0.0	—	1.0	0.0	0.0
G	—	-0.43	1.15	—	-0.43	1.15
C	-0.2	—	0.8	-0.2	1.38	—

Match **AACTCG** in the matrix

Find nucleotides at respective pos. of the matrix

Pos.	1	2	3	4	5	6
A	(1.0)	(1.0)	—	0.41	—	-0.58
T	0.0	0.0	—	(1.0)	0.0	0.0
G	—	-0.43	1.15	—	-0.43	(1.15)
C	-0.2	—	(0.8)	-0.2	(1.38)	—

Calculate the sum of log odds scores

1.0 + 1.0 + 0.8 + 1.0 + 1.38 + 1.15 = 6.33