

## Database Searching and BLAST

Tuesday, October 27th

Dannie Durand

### Data base searching overview

The goal of a database search is to find all “high-scoring” local alignments (*i.e.*, local alignments with a score above a given threshold) and to determine the significance of alignments found. A database search can be used to compare a protein or a cDNA sequence with genomic DNA, e.g. to find gene location or identify intron/exon boundaries. Another application is to find homologous protein sequences for structural and functional predictions and evolutionary analyses.

Database searching is essentially a local alignment problem. In theory, dynamic programming could be used to search a database for sequences that are similar to a query sequence. However, the running time for dynamic programming is  $O(mn)$ , where  $m$  is the length of the query sequence and  $n$  is the length of the database.

For large data bases, the complexity of this approach is prohibitive. The “typical” amino acid query sequence is 250 - 300 residues long, although query sequences can be much longer. For example, the genomic sequence of the BRCA2 gene, including introns and exons, is 84,193 base pairs long. The length of the transcribed BRCA2 mRNA is 11,386 base pairs, including untranslated regions. The BRCA2 amino acid sequence contains 3418 residues. Currently, the GenBank nr database, which includes non-redundant coding sequence translations and sequences from PDB, SwissProt, PIR and PRF (excluding environmental samples from whole genome sequencing projects), contains almost 70 billion base pairs and 18 billion amino acids. Instead of dynamic programming, heuristics are used to search data bases of this size.

|            | Nonredundant (nr) sequence database |                      |
|------------|-------------------------------------|----------------------|
|            | Nucleic Acid Sequences              | Amino Acid Sequences |
| Date:      | Oct 21, 2015 1:57 PM                | Oct 21, 2015 1:54 PM |
| Letters:   | 106,026,924,574                     | 26,861,771,399       |
| Sequences: | 33,081,743                          | 73,792,625           |

### BLAST

BLAST (Basic Alignment Local Search Tool) is a heuristic that searches a database for significant local alignments in less than  $O(mn)$  time. BLAST takes as input a query sequence,  $Q$ , of length  $m$  and a database,  $D$ , of length  $n$ . The data base is a series of concatenated nucleic or amino acid sequences  $M_1, M_2, \dots, M_j, \dots$

The BLAST literature uses the following terminology:

- A *Segment Pair* is an ungapped local alignment.
- A *Maximal Segment Pair (MSP)* is a segment pair whose score cannot be improved by extending or shortening the alignment.
- A *High scoring Segment Pair (HSP)* is a maximal segment pair with score  $\mathcal{S} \geq \mathcal{S}_T$ , where  $\mathcal{S}_T$  is a similarity score threshold (typically defined by the user).

Figure 1 shows an MSP of length  $l$  between query sequence,  $Q$  and matching sequence,  $M_j$ . We use  $\sigma$  and  $\tau$ , respectively, to designate the subsequences of  $Q$  and  $M_j$  that participate in the alignment.

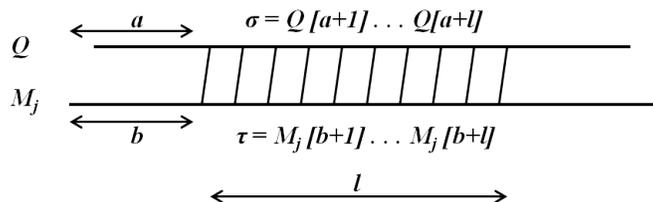


Figure 1: An ungapped, local alignment (MSP) between query sequence,  $Q$  and matching sequence,  $M_j$ . The MSP is  $l$  residues long and starts at residue  $a+1$  in  $Q$  and at residue  $b+1$  in  $M_j$ . We use  $\sigma$  and  $\tau$ , respectively, to designate the subsequences of  $Q$  and  $M_j$  that participate in the alignment.

## BLAST 90

Given a query sequence,  $Q$ , of length  $m$  and a database,  $D$ , of length  $n$ , BLAST attempts to find all database sequences that contain a maximal segment pair with a score above the reporting threshold,  $\mathcal{S}_T$ . Instead of comparing the query to every sequence in the data base, the BLAST heuristic restricts the search for high scoring ungapped alignments to regions of the data base that are “promising”; i.e., that are likely to contain an HSP. This strategy requires a method for predicting accurately which regions contain an HSP and which do not. BLAST does this by using a fast scan to find tiny, high scoring matches, which he calls “hits,” and then extending the hit to obtain HSP’s with scores at least  $\mathcal{S}_T$ . A *hit* is an ungapped alignment of a word in the query sequence and a word in the database, that has a score of at least  $T$ , where  $T$  is a BLAST performance parameter. A *word* or  $w$ -mer is a string of  $w$  consecutive letters. Typically,  $w$  is small.

The original BLAST90<sup>1</sup> heuristic has three main steps:

1. Construct a list,  $L$ , containing high scoring words derived from the query sequence.

*DNA:*  $L$  contains the  $m - w + 1$  words that are subsequences of  $Q$ .

*Proteins:* For each word,  $z$ , in  $Q$ , add to  $L$  all  $w$ -mers that have a similarity score  $\geq T$ , when aligned with  $z$  using the PAM120 substitution matrix. (I used BLOSUM62 in class.) The values of  $T$  and  $w$  are pre-specified parameters.

Note (1) Some words in  $Q$  will not appear in  $L$ . Specifically, any word with a score less than  $T$  when aligned with itself will not appear on  $Q$ . (2) Some words in  $L$  and not words in  $Q$ . The entries in  $L$  are stored in a hash table for fast lookup.

2. Find hits by scanning the database sequence for  $w$ -mers that correspond to a word in  $L$ . (Note that one could also make a list of the high scoring words in the database and compare each  $w$ -mer in the query sequence with all words in that list. Intuitively this might seem like a better option because the same hash table could be used for every query and would only have to be rebuilt when the database was updated. However, this would result in a much bigger hash table. In addition, this approach incurs a disk access performance penalty because it requires that the database be accessed randomly rather than scanned sequentially.)
3. Extend hits to obtain HSP's with scores at least  $\mathcal{S}_T$ . The time spent on this step is reduced by using a score cutoff. If the score of the extended alignment is lower than the best score seen so far by the amount of the cutoff, then BLAST stops extending the alignment in that direction.

The underlying assumption of the BLAST heuristic is that most HSPs will contain a hit and that hits that are not contained in an HSP are rare. If a region contains an ungapped alignment with score at least  $\mathcal{S}_T$ , but there is no word in that alignment with score at least  $T$ , then BLAST will not report this HSP, resulting in a false negative. On the other hand, if extending a hit does not lead to an ungapped alignment with score at least  $\mathcal{S}_T$ , then the time spent extending an alignment in the region of the hit is wasted. The trick is to select  $w$  and  $T$  to obtain a good balance between false negatives and unnecessary extensions. These scenarios are shown graphically in Fig. 2.

Three parameters influence the precision, recall, and speed of the heuristic. Increasing the value of  $\mathcal{S}_T$  (i.e., making the  $E$  value threshold more stringent) will result in fewer false positives and more false negatives. For a given  $\mathcal{S}_T$ , the values of  $w$  and  $T$  are selected to minimize the number of false negatives and the search time. Steps 1 and 2 in the heuristic are relatively fast. Step 3 is slow. Therefore, the goal is to select values for the parameters  $w$  and  $T$  that limit the number of hits that must be extended in Step 3 without incurring too many false negatives. If the hit threshold,  $T$ , is increased, the number of hits - and therefore the number of extensions - will decrease. However, the number of regions that contain a local, ungapped alignment with score greater than  $\mathcal{S}_T$ , but

---

<sup>1</sup>Karlin and Altschul, Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes, 1990, PNAS, 87:2264-2268.

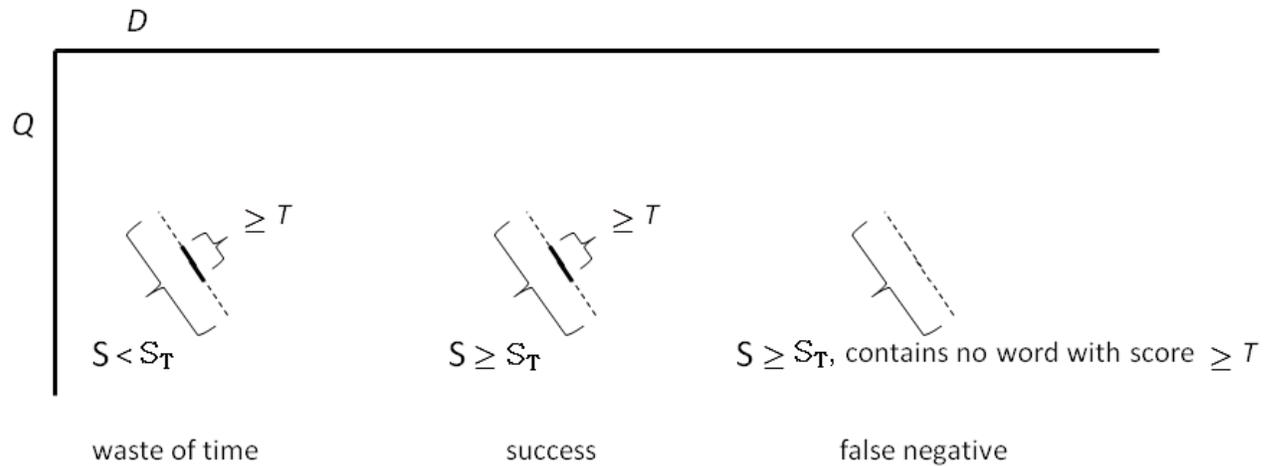


Figure 2: An unnecessary extension (left), a successful identification of a matching sequence (middle), and a false negative (right).

do not contain a hit with score at least  $T$ , will also increase, resulting in more false negatives. Decreasing  $w$  will similarly decrease the running time and increase the number of false negatives. Note that only  $S_T$  influences the false positive rate, but all three parameters contribute to the false negative rate. A false negative can occur because the heuristic fails to find a related sequence. This is determined by the values of  $w$  and  $T$ . A false negative also occurs when the search returns a related sequence, but it is not reported because it has a score below the reporting threshold,  $S_T$ .

Altschul and his colleagues used simulation studies to estimate the probability, for a given set of parameter values, that hits found in the data base would in fact be contained in local, ungapped alignments with score at least  $S_T$ . This is discussed in detail in Altschul *et al.*, 1990, which is on electronic reserve. Briefly, they used a statistical approach to minimize the probability of unnecessarily attempting to extend a hit. They determined empirically that a choice of  $w = 4$  and  $T = 17$  offered a good compromise between maximizing this probability and an excessive running time. Note that the values of  $w$  and  $T$  have been changed since 1990. Different values are used today.

## Gapped, two-hit BLAST

In 1997, three innovations<sup>2</sup> to the BLAST algorithm were introduced to address issues in sensitivity and running time:

1. Gapped extensions
2. Two hit BLAST
3. Position-Specific Iterative (PSI) BLAST

We discuss the first two innovations in detail below. PSI-BLAST, the third innovation, yields improved sensitivity by constructing a Position Specific Scoring Matrix (PSSM) of sequences with significant similarity to the query sequence. In the first iteration, PSI-BLAST compares candidate matches to the query sequence. In subsequent iterations, PSI-BLAST compares candidate matching sequences to the PSSM from the previous iteration. Improved sensitivity is obtained because the resulting PSSM captures the distribution of amino acid sequences observed at each conserved position in the protein family represented by the query sequence. PSI-BLAST will not be discussed further here. We will introduce Position Specific Scoring Matrices later in the semester.

Despite the early success of the BLAST heuristic as a sequence database search tool, the exponential growth of sequence data bases created a need for a faster heuristic. By 1997, parameters had been reduced to  $w = 3$  and  $T = 13$ , resulting in many more attempted ungapped extensions in Step 3 of the heuristic. Since ninety percent of the running time was expended in the third step of the procedure, improving efficiency required reducing the number of extensions without loss of sensitivity.

A second difficulty with the BLAST90 heuristic was that it only finds ungapped alignments. A simple solution might be to find two (or more) MSPs and merge them later. For this to work, the heuristic must be able to identify the individual MSPs so that they can be merged. This, in turn, requires that each MSP contain a hit (a word match with score at least  $T$ ). This can be achieved by decreasing the word score threshold from  $T = 13$  to  $T = 11$ . But, this will increase the number of hits found in Step 2 and the number of unnecessary extensions in Step 3, resulting in slower running times.

Instead, to obtain gapped alignments without further increasing the running time, Gapped BLAST uses a two phase protocol for selecting regions for a full, gapped alignment: first, ungapped extensions are attempted in those regions that contain a word with score at least  $T = 13$ . To limit the computational cost, the ungapped extension is terminated if the alignment score drops below  $X_u$ , the ungapped extension cutoff. If the score of the resulting MSP exceeds a preset minimum score, then a gapped extension (using dynamic programming) is attempted. Again, to limit the computational cost of this step, the extension is terminated if the alignment score drops below  $X_g$ ,

---

<sup>2</sup>Altschul, *et al.* Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 1997, vol.25, n17, 3389-3402

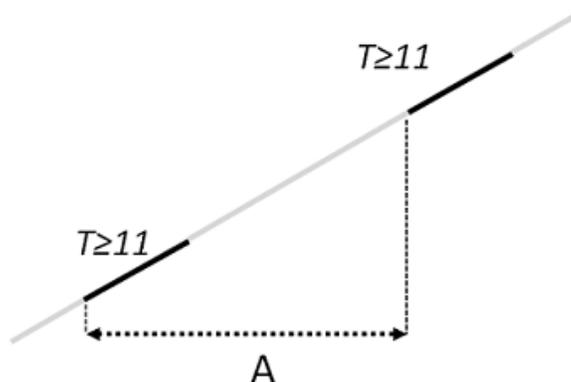


Figure 3: In Two-Hit BLAST, an extension is triggered if a pair of hits is found on the same diagonal within a distance of  $A$ . The current parameter values are  $w=3$ ,  $T=11$ , and  $A=40$ .

the gapped extension cutoff. If the score of this gapped alignment exceeds  $\mathcal{S}_T$ , the resulting match is reported.

This innovation delivered both gapped alignments and higher sensitivity, yet still achieved an improvement in running time. By increasing  $T$  from 11 to 13, the number of ungapped extensions was reduced by two thirds. Using the ungapped extensions as a filter for identifying candidates for gapped extension resulted in one gapped extension per 4000 ungapped extensions. Although the computational cost of gapped extensions is 500 times the cost of ungapped extensions, the total running time was reduced by more than a factor of 2.

The second innovation, Two-Hit BLAST, delivered further performance improvement without unduly compromising sensitivity. The underlying rationale is that an HSP will typically contain more than one hit. Better specificity, resulting in fewer extensions, can be obtained by reducing the threshold,  $T$ , to obtain more hits, but requiring two hits on the same diagonal in close proximity in order to trigger an ungapped extension (Fig. 3).

In Two-Hit Blast, the hit score threshold was reduced to  $T = 11$ . Ungapped alignments are attempted only when two hits are found on the same diagonal that are separated by a distance no greater than  $A = 40$ . This modification resulted in 3.2 times as many hits, but decreased the number of extensions by 0.14, yielding an additional two-fold speedup. An example showing the reduction in the number of extensions with Two-Hit Blast is shown in Fig. 4.

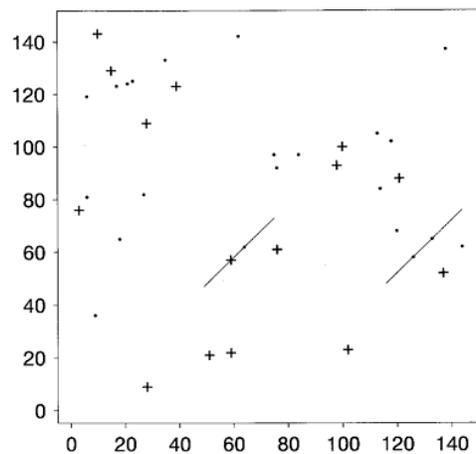


Figure 4: Hits with  $T=11$  (.) and  $T=13$  (+) in an alignment of Broad bean leghemoglobin and Horse beta globin, reproduced from Altschul *et al.* (1997). This alignment contains 37 hits when  $T=11$ , but only two pairs satisfy the requirements for an extension. In contrast, 15 hits are obtained when  $T=13$ , which would result in 15 extensions with the original 1990 BLAST algorithm.

### Putting it all together

1. Find hits of length  $w$  with similarity threshold  $T$ .
2. If there are two hits on same diagonal separated by a distance of at most  $A$ , perform an ungapped extension to obtain an HSP using cutoff,  $X_u$ .
3. If the HSP score in (3) exceeds  $\mathcal{S}_g$ , perform a gapped extension using dynamic programming with cutoff  $X_g$ .
4. If gapped local alignment score exceeds  $\mathcal{S}$ , report the match, the score, and the significance expressed as an E-value.