CSE300

Report on
BLAST: Basic Local Alignment Search Tool

`Your Roll` - Your Name
`Partner's Roll` - Partner's Name

August 29, 2022

Contents

Chapter 1

Introduction

Basic Local Alignment Search Tool(abbr. BLAST) is a program for finding regions of similarity between two string sequences. The tool employs a heuristic algorithm of the same name producing approximate results- efficient than known previous methods. The tool is used in various fields such as Bioinformatics, Molecular Biology, etc.

Easily put, a BLAST search compares two biological sequences such as nucleotide(DNA, RNA), proteins, and finds similarities by measuring certain parameters. By taking a heuristic approach, this algorithm outperforms previous well-known methods.

1.1 History

Initially, BLAST came from the stochastic model proposed by **Samuel Karlin** and **Stephen F. Altschul**. After a subsequent work behind the model, a group of five researchers- **Stephen F. Altschul**, **Warren Gish**, **Webb Miller**, **Eugene W. Myers**, and **David J. Lipman** first published their paper on "*Basic Local Alignment Search Tool*". The paper was later accepted by the *Journal of Molecular Biology* in 1990. By the time of writing this article, it has been cited 63801 times.

1.2 Types of BLAST

There are five types of BLAST, each for its own kind of query and database sequence type. The five types of BLAST and their relative query and database sequence type are as below:

Nature	Program	Query	Database
Nucleotide BLAST	blastn	Nucleotide (DNA, RNA)	Nucleotide (DNA, RNA)
Protein BLAST	blastp	Protein	Protein
Mixed BLAST	blastx	Translated Nucleotide	Protein
	tblastn	Protein	Translated Nucleotide
	tblastx	Translated Nucleotide	Translated Nucleotide

Table 1.1: Different types of BLAST

Chapter 2

BLAST Tool

BLAST tool is available in various formats such as-

- Standalone Executable File
- Public API by NCBI
- Docker Image for Cloud

There are also public databases available to download and use with the standalone BLAST program. NCBI website provides access to over 300+ unique databases.

2.1 Working Procedure

During the detailed study, we had used the BLAST tool. We used a Python package provided by NCBI to perform a BLAST search through the NCBI public API. In the following section, our follow-through approach to using BLAST tool has been described.

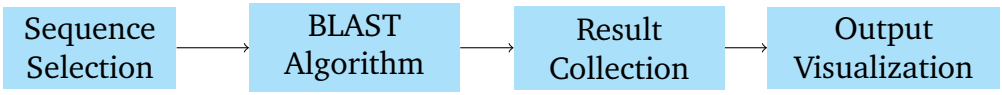

Figure 2.1: Basic working procedure of BLAST tool

2.1.1 Sequence Selection

A BLAST search requires two basic sequence- one which the search are going to be executed. One of them is the unknown sequence- it is referred as query sequence and, another is the database on which the tool is going to compare the sequences to.

During our BLAST search using the tool, we used Drosophila's *(Drosophila yakuba)* Mitochondrion DNA as our query sequence. And then the *nt* database was selected as it contains the records of different nucleotide sequence.

2.1.2 BLAST Algorithm

The algorithm takes the input and database sequence, and then processed them using different methods. Finally, a scoring scheme is used to denote the similarity between sequences. In next chapter, the BLAST algorithm is going to be explained in detail.

2.1.3 Result Collection

The output result given by the NCBI BLAST API defaults in XML format. The BLAST tool returns the results along with other meta-data such as Hit Score/BLAST Score [1], E-value [2], Query Range, Hit Range etc.

Formatted results are shown in the figure below-

```
Program: blastn (2.12.0+)
Query: unknown_yakuba_sequence (11001)
Target: nt
Hits: ----  -----  ------------------------------------------------------------
        #  # HSP  ID + description
     ----  -----  ------------------------------------------------------------
        0      7  gi|1976360327|ref|XM_039377129.1|  PREDICTED: Drosophil...
        1      6  gi|1976360325|ref|XM_039377128.1|  PREDICTED: Drosophil...
        2      7  gi|1976360324|ref|XM_002099563.3|  PREDICTED: Drosophil...
        3      7  gi|1976360323|ref|XM_015191338.2|  PREDICTED: Drosophil...
        4      7  gi|1988900005|ref|XM_039640670.1|  PREDICTED: Drosophil...
        5      6  gi|1988900003|ref|XM_039640669.1|  PREDICTED: Drosophil...
        6      7  gi|1988900001|ref|XM_039640668.1|  PREDICTED: Drosophil...
        7      7  gi|1988899999|ref|XM_039640667.1|  PREDICTED: Drosophil...
        8      6  gi|1509833196|ref|XM_026983418.1|  PREDICTED: Drosophil...
        9    507  gi|1243416815|gb|CP023340.1|  Drosophila melanogaster s...
```

Figure 2.2: Result from NCBI BLAST API

[1]The BLAST Score/Hit Score/Bit Score indicates the quality of the best alignment between the query sequence and the found sequence (hit). The higher the score, the better the alignment.

[2]E-value or 'expect value' is the number of matches by chance to the provided sequence one can expect in a database of a given size. Lower e values indicate more "significant" or better alignments.

```
Query: unknown_yakuba_sequence
Hit: gi|1976360323|ref|XM_015191338.2| (4736)
PREDICTED: Drosophila yakuba protein BCL9 homolog (LOC6523724), transc...
HSPs:  ----  --------  ---------  ------  ---------------  --------------------
          #   E-value  Bit score    Span      Query range              Hit range
       ----  --------  ---------  ------  ---------------  --------------------
          0         0    3627.86    2011       [3363:5374]             [992:3003]
          1         0    2774.87    1538       [1014:2552]            [3198:4736]
          2  3.2e-154     563.04     314       [9166:9480]             [181:495]
          3    8e-143     524.26     300       [8818:9118]             [483:783]
          4  1.8e-100     383.60     212       [8103:8315]             [781:993]
          5     2e-93     360.16     204       [2605:2809]            [2996:3200]
          6     8e-86     334.91     185       [9655:9840]                [0:185]
```

Figure 2.3: BLAST Hit Score Output

```
      Query: unknown_yakuba_sequence
        Hit: gi|1976360323|ref|XM_015191338.2| PREDICTED: Drosophila yakuba p...
Query range: [3363:5374] (1)
  Hit range: [992:3003] (-1)
Quick stats: evalue 0; bitscore 3627.86
  Fragments: 1 (2011 columns)
     Query - CAGCAGAGGACTGACCGAATGAGTCCAATTCCTTTGGTGATAGATGGGATAGAGGGGTA~~~TTGAC
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||~~~|||||
       Hit - CAGCAGAGGACTGACCGAATGAGTCCAATTCCTTTGGTGATAGATGGGATAGAGGGGTA~~~TTGAC

      Query: unknown_yakuba_sequence
        Hit: gi|1976360323|ref|XM_015191338.2| PREDICTED: Drosophila yakuba p...
Query range: [1014:2552] (1)
  Hit range: [3198:4736] (-1)
Quick stats: evalue 0; bitscore 2774.87
  Fragments: 1 (1538 columns)
     Query - TTATTTGTTGACAAAGAACGCTGGATTCGGGGATAAATTCGGCGGCATTGTTATCATGT~~~ACGCT
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||~~~|||||
       Hit - TTATTTGTTGACAAAGAACGCTGGATTCGGGGATAAATTCGGCGGCATTGTTATCATGT~~~ACGCT

      Query: unknown_yakuba_sequence
        Hit: gi|1976360323|ref|XM_015191338.2| PREDICTED: Drosophila yakuba p...
Query range: [9166:9480] (1)
  Hit range: [181:495] (-1)
Quick stats: evalue 3.2e-154; bitscore 563.04
  Fragments: 1 (314 columns)
     Query - CCGGCGACATAGTAGAAAAGAGTTCGTTTTTAAGGTTCTTTGGAGAACTTATGTTGGTG~~~TCTAG
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||~~~| |||
       Hit - CCGGCGACATAGTAGAAAAGAGTTCGTTTTTAAGGTTCTTTGGAGAACTTATGTTGGTG~~~TTTAG
```

Figure 2.4: HSP Score

2.1.4 Output Visualization

The output result produced by the BLAST tool are most of them incomprehensible. Since the search is executed upon hundreds and thousands of different database sequences, it becomes hard to get an understanding of the alignment matching. Hence, the tool visualizes the output in a graphical manner which helps a user to identify most matching sequences.

The BLAST tool identifies different matching scores with individual colors so that it becomes easy to identify the regions. The following figure shows the output produced by the BLAST tool.

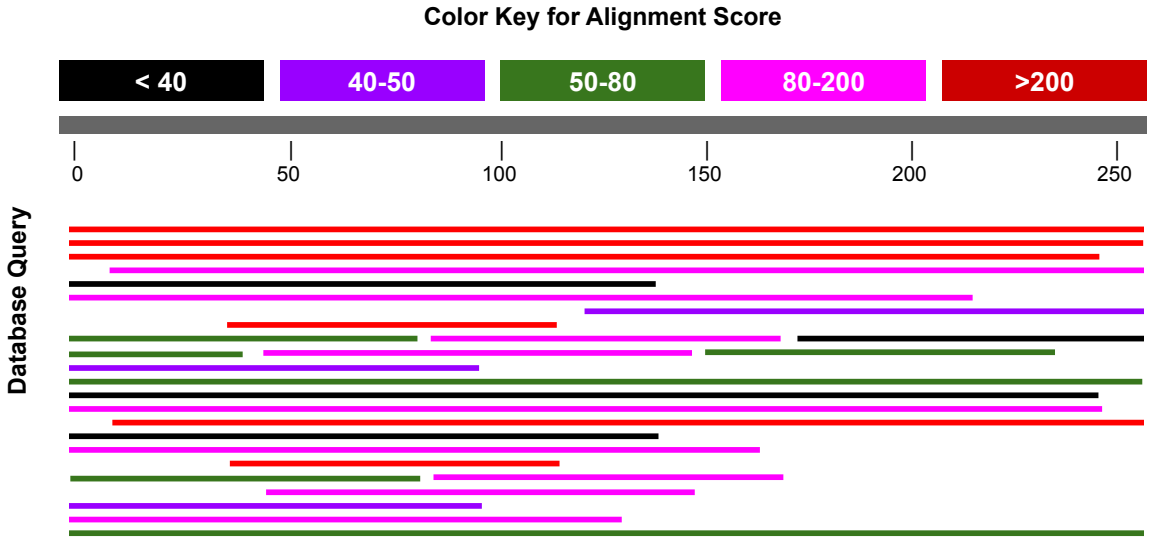

Figure 2.5: Output Visualization of a BLAST search

Chapter 3

BLAST Algorithm

In this chapter, we are going to describe the BLAST algorithm with the help of an example.

3.1 Basic Terminologies

3.1.1 Numeric Representation of Sequences

Bioinformatics algorithms, such as BLAST uses numerical representation to identify each sequences uniquely. The BLAST algorithm uses a **Base 4 Numeric Representation** to identify each unique nucleotide sequence by assigning each character in the sequence with a unique value. In the following table, numerical representation for the four nucleotides of DNA are shown.

Nucleotide	Letter	Value
Adenine	A	0
Cytosine	C	1
Guanine	G	2
Thymine	T	3

Table 3.1: Numeric Representation of Nucleotides

3.1.2 K-mer Construction

In **Bioinformatics**, k-mers are substrings of length k contained within a biological sequence. Primarily used within the context of computational genomics and sequence analysis.

A sequence of length L will have $L - k + 1$ k-mers and n^k total possible k-mers, where n is number of possible monomers. For example, all the possible k-mers of a DNA sequence (**GTAGAGCTGT**) are shown below:

k	k-mers
1	G, T, A, G, A, G, C, T, G, T
2	GT, TA, AG, GA, AG, GC, CT, TG, GT
3	GTA, TAG, AGA, GAG, AGC, GCT, CTG, TGT
4	GTAG, TAGA, AGAG, GAGC, AGCT, GCTG, CTGT
5	GTAGA, TAGAG, AGAGC, GAGCT, AGCTG, GCTGT
6	GTAGAG, TAGAGC, AGAGCT, GAGCTG, AGCTGT
7	GTAGAGC, TAGAGCT, AGAGCTG, GAGCTGT
8	GTAGAGCT, TAGAGCTG, AGAGCTGT
9	GTAGAGCTG, TAGAGCTGT
10	GTAGAGCTGT

Table 3.2: Different k-mers of the sequence *GTAGAGCTGT*

3.2 Pre-processing of Database

3.2.1 Constructing 3-mers

We select GGACGGATTCC as a database sequence. Now we need to construct the 3-mers and calculate the position and key values of them. Duplicate 3-mers should be avoided and base 4 representation of the 3-mers should be selected as their key values. Then the obtained table is sorted with respect to the key value.

3-mers	Position	Key
GGA	1,5	40
GAC	2	33
ACG	3	6
CGG	4	26
GAT	6	35
ATT	7	15
TTC	8	61
TCC	9	53

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Table 3.3: (a) Generated 3-mers (b) Sorted 3-mers with respect to key values

3.2.2 Building Binary Search Tree

Now we are going to construct a binary search tree. We select the positions and keys of the 3-mers as nodes.

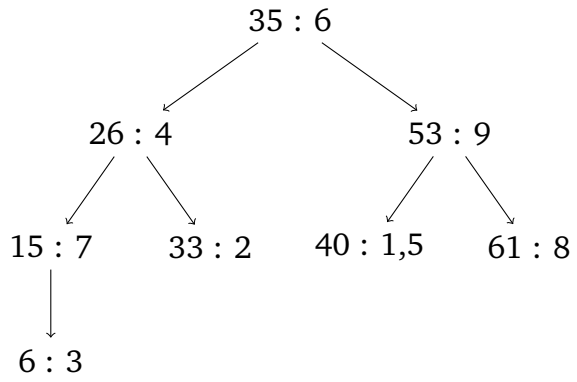

Figure 3.1: Constructed binary search tree

With this, our pre-processing of the database sequence is done. Now we proceed to the next step.

3.3 Construction of k-mers From Query Sequence

First of all, we should consider the following table representing the scoring scheme.

Scoring Scheme	
Match	1
Mismatch	-1
Gap Insertion	-1
HSP Threshold	1

Table 3.4: Scoring Scheme

Let us consider ATCG as a query sequence. The 3-mers of this sequence are ATC and TCG.

3-mers	Key
ATC	13
CTC	45
GTC	49
TTC	61
AAC	1
ACC	5
AGC	9
ATA	12
ATG	14
ATT	15

3-mers	Key
TCG	54
ACG	6
CCG	24
GCG	38
TAG	50
TGG	58
TTG	62
TCA	52
TCC	53
TCT	59

Table 3.5: Selected K-mers with minimum HSP threshold

Here we compare all possible 3-mers with the query 3-mers and select those whose matching score is at least the defined value of HSP threshold. We also calculate the key values of these 3-mers.

3.4 Comparing k-mer with Pre-processed Data

In this section, we are going to search the key value of the selected 3-mers with minimum HSP threshold in tree 3.3.

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Table 3.6: 3-mers of Database Sequence

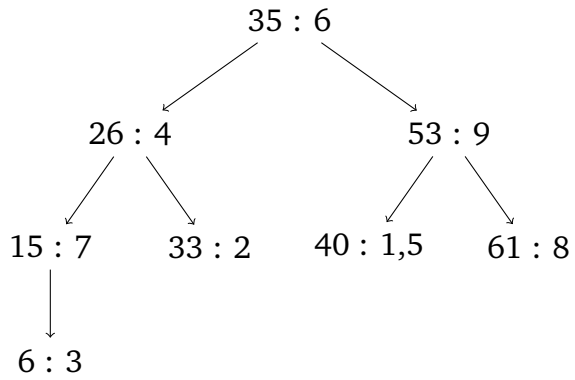

Figure 3.2: Constructed binary search tree

The 1st key found in the bst is 61. Its position is 8 and the corresponding 3-mer is TTC.

3-mers	Key
ATC	13
CTC	45
GTC	49
TTC	61
AAC	1
ACC	5
AGC	9
ATA	12
ATG	14
ATT	15

3-mers	Key
TCG	54
ACG	6
CCG	24
GCG	38
TAG	50
TGG	58
TTG	62
TCA	52
TCC	53
TCT	59

Table 3.7: 3-mers with Minimum HSP Value

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Table 3.8: 3-mers of Database Sequence

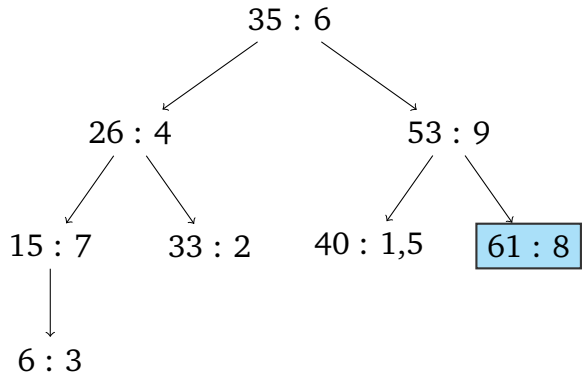

Figure 3.3: Constructed binary search tree

Following the similar process we have found 15, 6 and 53. The corresponding (3-mers, position) pairs are **(ATT, 7)**, **(ACG, 3)** and **(TCC, 9)** respectively.

3.5 Alignment Extension

3.5.1 Seeding

The next step is seeding. Seeding means finding exact matches of the part of the database with the part of the query. For 3-mers ACG, ATT, TTC and TCC, the starting position to put 3 consecutive crosses are 3, 7, 8 and 9 respectively.

	G	G	A	C	G	G	A	T	T	C	C
A							X	X			
T			X					X	X		
C				X					X	X	
G					X						X

Table 3.9: Seeding

Now we are going to apply the **Smith-Waterman Algorithm** to determine the extension of the local alignments..

3.5.2 Extension of the Seeding

After applying the algorithm we obtain the following table. Now we are going to extend the alignments. Recall that, the score extension threshold was 1.

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table 3.10: Extension of alignment

This first local alignment found from the table is in table 3.11 . The upper segment is from the database sequence and the lower segment is from the query sequence.

C	G G
C	G -

Table 3.11: Alignments

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table 3.12: 3-mers of Database Sequence

Following the same approach, we get 6 local alignments.

C	G	G		A	T	T	C	C
C	G	-		A	T	-	C	G
A	T	T		A	T	T	C	C
A	T	C		A	T	-	C	-
A	T	-		A	T	T	C	-
A	T	C		A	T	-	C	G

3.6 Complexity of the Algorithm

In this section, we are going to determine the complexity of the BLAST algorithm. The number of k-mers in a X length string can be expressed as $X - K + 1$ where K is the length of a k-mer. So the number of k-mers in the M length database sequence and the number of k-mers in the L length query sequence is calculated below.

M length Database,

$$n_{k-mer} = M - K + 1 \tag{3.1}$$

L length Query,

$$n_{k-mer} = L - K + 1 \tag{3.2}$$

As we have used the binary search tree to find the k-mers, the complexity of the algorithm is,

$$\mathcal{O}(\{L - K + 1\} \times \log_2\{M - K + 1\})$$

Here, K = k-mer length, L = Length of query sequence and M = Total length of database sequences.
Since K is constant, we can express the complexity as,

$$\mathcal{O}(L \times \log_2 M)$$

Chapter 4

Conclusion

BLAST is a powerful tool for finding local alignments. Since BLAST uses **Smith Waterman Algorithm** to extend the local alignments, other methods can also be used in place in order to increase the efficiency of the overall process. FASTA, the predecessor to BLAST is however slower than BLAST but produces exact matches through a much wider range of scoring matrices, making it easier to tailor a search to a specific single sequence. On the other hand, an extremely fast but less sensitive alternative to BLAST is BLAT (Blast Like Alignment Tool).

The BLAST approach allows extremely fast approach to search database and to match and extend the alignments. BLAST is more time-efficient than FASTA by searching only for the more significant patterns in the sequences, yet with comparative sensitivity. This is what makes BLAST more convenient than other existing algorithms of the same kind.

References

1. Altschul, S. F., et al. “Basic Local Alignment Search Tool.” Journal of Molecular Biology, vol. 215, no. 3, Oct. 1990, pp. 403–10. PubMed, doi:10.1016/S0022-2836(05)80360-2.

2. Smith, T. F., and M. S. Waterman. “Identification of Common Molecular Subsequences.” Journal of Molecular Biology, vol. 147, no. 1, Mar. 1981, pp. 195–97. DOI.org (Crossref), doi:10.1016/0022-2836(81)90087-5.

3. BLAST: Basic Local Alignment Search Tool. https://blast.ncbi.nlm.nih.gov/Blast.cgi. Accessed 12 July 2021.

4. GitHub - NCBI-Hackathons/NCBIComputationalCookbook: Jupyter Notebooks to More Effectively Leverage Computational Resources at NCBI.” GitHub, https://github.com/NCBI-Hackathons/NCBIComputationalCookbook. Accessed 28 July 2021.

5. PR-INBRE BiRC [Bioinformatics Resources Core] - YouTube. https://www.youtube.com/channel/UC8kHK9I5NxHmW0j-RcWQ8cg. Accessed 12 July 2021.

6. Index of /Blast/Db. https://ftp.ncbi.nlm.nih.gov/blast/db/. Accessed 28 July 2021.