DNA sequence alignment by translation of codons into protein.

Nadav, Shir, Amit, Reut

1. The Aim of the Experiment

1.1. What is Sequence Alignment?

Sequence alignment is the process of comparing and detecting similarities between DNA/RNA sequences. The "similarities" that are being detected depend on the goals of the particular alignment process.

1.2. Why is Sequence Alignment Important?

Sequence alignment appears to be extremely useful in a number of bioinformatics applications. Finding similarities between sequences helps us achieve:

a. Function inference - function of newly sequenced genes by alignment with known genes.
b. New members of a gene family prediction.
c. Discover evolutionary relationships.
d. Finding structurally similar regions in a protein.

And more.

1.3. The Aim of Protein (Codon) Alignment

The aim of our project is to use a protein alignment instead of a DNA alignment because proteins are built from 20 amino acids while DNA only contains four different bases, meaning that the 'signal-to-noise ratio' in protein sequence alignments is much better than in alignments of DNA.

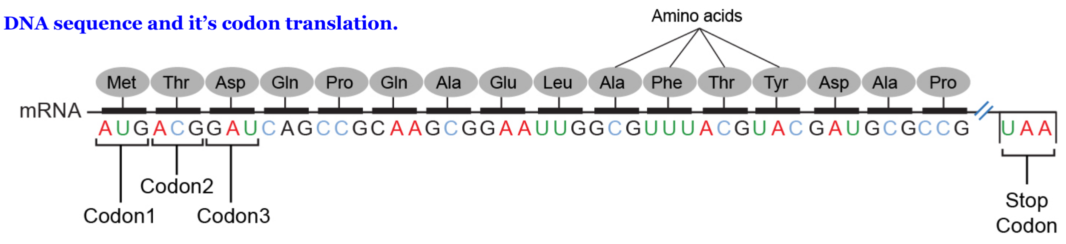
2. The codon alignment problem

2.1. What is a codon?

A codon is a trinucleotide sequence of DNA or RNA that corresponds to a specific amino acid. The genetic code describes the relationship between the sequence of DNA bases (A, C, G, and T) in a gene and the corresponding protein sequence that it encodes. The cell reads the sequence of the gene in groups of three bases. There are 64 different

codons: 61 specify amino acids while the remaining three are used as stop signals.

Figure 1. DNA sequence and it's codon translation.



DNA sequence alignment by translation of codons into protein.

Nadav, Shir, Amit, Reut

2.2. Introduction To the Problem

Owing to the degeneracy of the genetic code, where a single amino acid can be encoded by multiple codons, it is often preferable to align protein sequences rather than the underlying coding DNA as it increases sensitivity at longer evolutionary distances and prevents the introduction of frame shifts into an alignment. Thus the construction of a protein alignment first and then reverse translating this into a codon-based DNA alignment is invariably the optimal solution and provides reliable alignments to perform correct evolutionary analyses.

3. DATA

3.1. DNA fasta file

Files taken from NCBI containing thousands of nucleotides for each sequence.
For testing our implementation we used files from home assignments. We process the fasta files and run the algorithm below.

4. The Algorithm

4.1. Needleman Wunsch Background

This algorithm is being used in order to align two sequences, particularly protein or nucleotide sequences. It uses dynamic programming -an algorithmic technique for solving a problem by recursively breaking it down into simpler subproblems and using the fact that the optimal solution to the overall problem depends upon the optimal solution to it's individual subproblems.

The Needleman–Wunsch algorithm assigns a score to every possible alignment in each iteration, and the purpose of the algorithm is to find all paths that give an optimal alignment by using a target function. The main steps in Needleman-Wunsch algorithm are:

- Initialization of the matrix with the possible scores.
- Matrix filling with maximum score (or minimum penalty).
- Trace back the residues for the desired alignment.

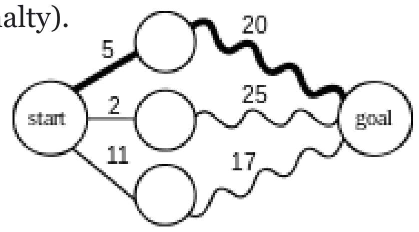


Figure 2. Dynamic Programing Visualization

DNA sequence alignment by translation of codons into protein.

Nadav, Shir, Amit, Reut

4.2. Steps of Our Algorithm

Input:

a. Two NOC sequences
b. Score matrix (for amino acids, e.g 'Blosum62')

Steps:

a. Initializing a matrix $(n + 1 \cdot m + 1)$
b. Filling the first row and the first col by the summing of the Gap values.
c. Continuing filling the matrix from left to right and up down as a function of the dependency of one letter to a triplet.
d. Each cell is being filled by the maximization of the target function.
e. We consider one or two shifts, a gap (three shifts) and a stop codon in each cell.
f. Tracing back to an optimal alignment.

Output:

a. An alignment for the input sequences.
b. The score that was found in the matrix.

4.3. The Difference Between NW to Our Implementation

The most significant difference between the algorithms is that we consider a sequence as an instance of ordered triples representing proteins while NW aligns sequences according to the underlying coding DNA. This puts us in a problem where we have to jump three consecutive letters whenever we want to add a Gap to one sequence.
Shifts (!) come in order to solve that problem.
In addition, we have added a reference to Stop codons.

4.4. What is a Shift and Why Do We Use It?

In our implementation, a Shift is equivalent to a Gap in NW, meaning that it is a single letter gap of one alignment in contrast to the other.
In our case, where we align proteins (codons triplets), instead of stepping over one codon (Gap in our case), we **omit** one/two letters from the shifted sequence in order to suffice the greater alignment - reach maximum in the target function. Omitting means that the sampled letter that was shifted, is not being used in the protein interpretation.
Today there are a lot of algorithms that process DNA quickly and efficiently but still there is a certain confidence interval for errors.
The Shifting approach helps us deal with this problem.

However, Shifting means losing valuable information from the DNA sequence, so we penalize it accordingly.

4.5. Advantages and Disadvantages

4.5.1. Advantages:

There are a lot of ways to align sequences; however, when dealing with protein-coding sequences, they do not take in mind the corresponding amino-acid translations. Because NT sequences are less conserved, clear amino-acid level similarities can be blurred, hence complicating alignment;
In addition, the current optimization criteria during the alignment process doesn't penalize insertion/ deletion events while our algorithm does (Frameshifts).

4.5.2. Disadvantages:

As we mentioned above, the shifting process inherently omits information. Moreover, the algorithm depends heavily on RAM usage (we will discuss it in the complexity analysis section).

4.6. Run Time and Space Complexity

Our algorithm has the same space and time complexity as the classical NW algorithm and at the same time take under consideration biological deviations and sequencing errors. As we recall from class, NW algorithm with the simple implementation (also the implementation in our project) has a run time and space complexity of $O(n \cdot m)$**

4.7. Improvements

a. Adding a decay parameter to the scoring function - only on the penalties (gaps, frameshifts and stop codons).
b. Reducing space complexity by saving the last three columns in each iteration.
c. Allowing a first/second shift at the beginning of the sequence.

5. Results

5.1. Way of Comparing Results

The comparison will be made as follows:

- For a particular sequence we will compare the alignments obtained from the running of the algorithm with a different score table each time.

- We will also take a particular nucleotide sequence (PCNA) as it appears in 2 different specieses(mice, elephant) and compare it to another species that will serve as a reference(human).
- We would like to test whether there is a relationship between an estimated evolutionary distance and the score obtained in the alignment of the protein in the 2 appropriate species.

5.2. Heuristics We Used

BLOSUM (BLOcks SUbstitution Matrix) matrix is a substitution matrix used for sequence alignment of proteins. BLOSUM matrices are used to score alignments between evolutionarily divergent protein sequences.
The **EDSSMat** series of matrices were developed and described in Trivedi and Nagarajaram (2019).

5.3. Results Visualization and Comparison

5.3.1. Human-X Score table:

Amino Acid Seq	**BLOSUM62**	**BLOSUM75**	**EDSS50**
Human/Elephant	6882	7000	11427
Human/Mouse	8170	8360	12835
Human/Human - control	24838	26667	30192

We will do the following calculating for each matrix score:

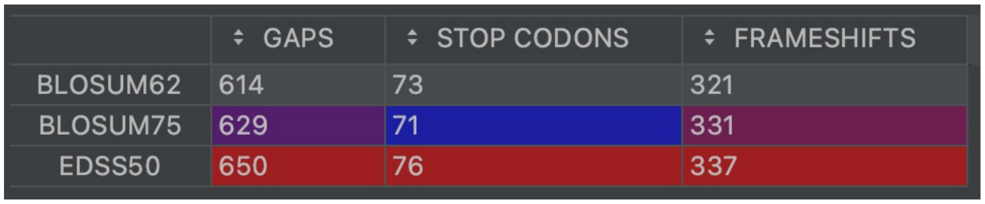
$$\frac{Human\_Mouse(matrix\ score) - Human\_Elephant(matrix\ score)}{Human\_Human(matrix\ score) - Human\_Elephant(matrix\ score)}$$

Results:

For **BLOSUM62:** 7.173%
For **BLOSUM75:** 6.915%
For **EDSS50:** 7.503%
Average: 7.197
Std: 0.295

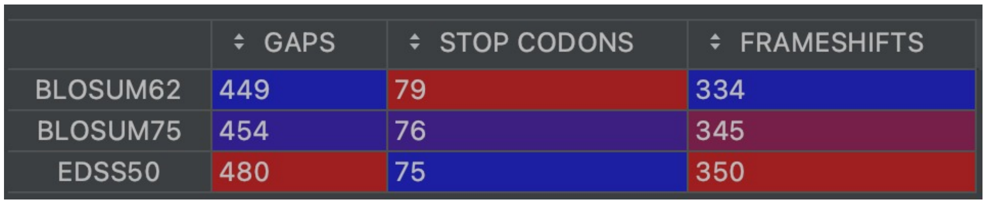
5.3.2. Human-X Alignment attributes comparison-

Human-Elephant:



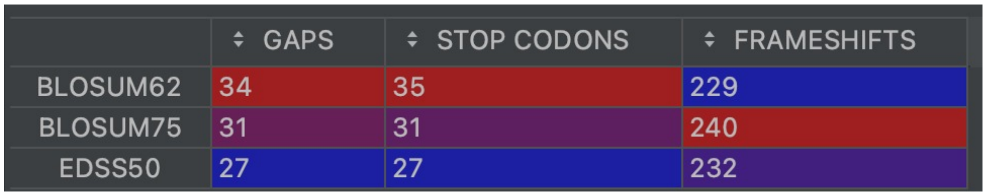
	GAPS	STOP CODONS	FRAMESHIFTS
BLOSUM62	614	73	321
BLOSUM75	629	71	331
EDSS50	650	76	337

Human/Mouse:



	GAPS	STOP CODONS	FRAMESHIFTS
BLOSUM62	449	79	334
BLOSUM75	454	76	345
EDSS50	480	75	350

Human-control group:



	GAPS	STOP CODONS	FRAMESHIFTS
BLOSUM62	34	35	229
BLOSUM75	31	31	240
EDSS50	27	27	232

A interesting comparison is as following:

	Estimated Divergence Time(M.Y.A- 10^6 years)
Human/Elephant	105
Human/Mouse	90
Human/Human - control	0

As we can see Human/Mouse gets the higher score for alignment for each matrix we used than Human/Elephant. Unsurprisingly the closest estimated divergence time in M.Y.A is for Human/Mouse (calculated in Timetree website- screenshots attached in the Appendix).

6. Appendix

6.1. Signal-to-noise ratio (SNR or S/N)

SNR is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. SNR is defined as the ratio of signal power to the noise power, often expressed in decibels. A ratio higher than 1:1 (greater than 0 dB) indicates more signal than noise. In our case, when considering a DNA sequence as a sequence of protein triplets, noise in the sequence (Letter that was documented miss correctly), will have a smaller impact on the end result in contrast to the alignment process by the underlying coding DNA.

6.2. Run Time and Space Complexity :

There is a way to improve the running time of the algorithm to$O(nm\ /log(n))$ by using the method of Four Russians. There is also a way to improve the space complexity of the algorithm to O($max(n, m)$) by using the Divide-and-conquer algorithm.

6.3. Pseudo-code

```
Data: Two Sequences $S_1$ and $S_2$, a $cost$ method that returns the cost of frameshift, deletion and
      STOP codon, a $\sigma$ method returning the AA substitution cost.
Result: An array $C$ such that $C[i][j] = score(\mathcal{A}(S_1[1:i], S_2[1:j]))$.
for $i = 0$ to $len(S_1)$ do
    for $j = 0$ to $len(S_2)$ do
        if $i == 0$ AND $j = 0$ then
            C[i][j]=0;
        else
            $AA_1$= "?"; $AA_2$= "?";;
            if $i - 3 > 0$ then $AA_1 = (\pi(S_1[i-3:i])$;
            if $j - 3 > 0$ then $AA_2 = (\pi(S_2[j-3:j])$;
            stop$S_1$=0; stop$S_2$=0;
            if $AA_1 ==$ "$*$" then stop$S_1$ = cost("*");
            if $AA_2 ==$ "$*$" then stop$S_2$ = cost("*");
            if $AA_1 ==$ "$*$" OR $AA_2 ==$ "$*$" then
                $subst\_AA$=stop$S_1$ +stop$S_2$
            else
                if $i - 3 > 0$ AND $j - 3 > 0$ then
                    $subst\_AA = \sigma(AA_1, AA_2)$
                else
                    $subst\_AA = +\infty$
```

$$
C[i][j] = min \begin{cases}
get\_C(i-3, j-3) + subst\_AA \\
get\_C(i-3, j) + stopS_1 + cost("\_") \\
get\_C(i, j-3) + cost("\_") + stopS_2 \\
\\
get\_C(i-3, j-2) + stopS_1 + cost("!") \\
get\_C(i-3, j-1) + stopS_1 + cost("!") \\
get\_C(i-2, j-3) + cost("!") + stopS_2 \\
get\_C(i-1, j-3) + cost("!") + stopS_2 \\
\\
get\_C(i, j-1) + cost("\_") + cost("!") \\
get\_C(i, j-2) + cost("\_") + cost("!") \\
get\_C(i-1, j) + cost("!") + cost("\_") \\
get\_C(i-2, j) + cost("!") + cost("\_") \\
\\
get\_C(i-1, j-1) + 2 * cost("!") \\
get\_C(i-1, j-2) + 2 * cost("!") \\
get\_C(i-2, j-1) + 2 * cost("!") \\
get\_C(i-2, j-2) + 2 * cost("!")
\end{cases}
$$

```
return $C$;
```

6.4. EDSSMat Explanation:

These are substitution scoring matrices used to align proteins or regions which experience intrinsic disorder. Alignment blocks, used to compute the matrix values, were composed of predicted intrinsically disordered regions. When compared to other, more frequently used substitution matrices (like BLOSUM), EDSSMat had significantly smaller E-values when aligning regions of disorder.

6.5. Link to git:

https://github.cs.huji.ac.il/shirmo/algbio_hackathon

7. Bibliography:

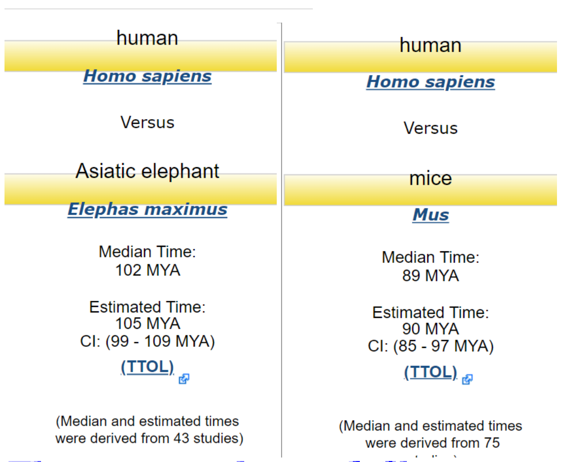
a. **https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0022594#s3**

b. **https://teach.genetics.utah.edu/content/evolution/biochemistry/pdfs/comparing-amino-acids.pdf**

c. **https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm#Complexity**

d. **http://www.timetree.org/**

e. **https://www.ncbi.nlm.nih.gov/**



human
Homo sapiens
Versus
Asiatic elephant
Elephas maximus
Median Time:
102 MYA
Estimated Time:
105 MYA
CI: (99 - 109 MYA)
(TTOL)
(Median and estimated times were derived from 43 studies)

human
Homo sapiens
Versus
mice
Mus
Median Time:
89 MYA
Estimated Time:
90 MYA
CI: (85 - 97 MYA)
(TTOL)
(Median and estimated times were derived from 75

Figure 3. estimated divergence time in years calculated in Timetree website