GLOBAL ALIGNMENT OF PAIR OF GENES

CS-327 Computational Methods in Biology

Northeastern Illinois University

Submitted By:
Dikshya Acharya

Submitted To:
Dr. Rachel Trana

ABSTRACT

A gene consists of DNA and RNA, which are used for the synthesis of Protein. DNA, RNA and protein are fundamental biological sequences in the cell. DNA is composed of four nucleotide bases and thus are sequences with a combination of A, C, T, G for Adenine, Cytosine, Thymine and Guanine. Similarly, RNA is a sequence of Adenine, Guanine, Cytosine and Uracil thus has a combination of A, G, C, U in the RNA string. Like-wise for Protein sequence, we have 20 amino acids to make different combination of the sequences. There are essentially many other structures in Bioinformatics arranged in sequences.

One deep concern in bioinformatics is determining the similarity between pairs of these genetic sequences. The alignment of sequences is a very well-known method adopted in bioinformatics for this purpose. The significance of the sequence alignment study ranges from discovering the functional and structural similarity of sequences to characterizing the common ancestor, forming phylogenetic trees, predicting the protein structure etc. Sequence alignment is a very vast field of research involving high computational complexity. The objective of this research paper is to study about global alignment, one of the methods adopted in sequence alignment of pairs of genes and the optimal algorithm used to solve the problem of alignment.

GLOBAL ALIGNMENT OF PAIRS OF GENES

Background

The change in gene's structure is a naturally occurring phenomenon during the biological evolution. More frequently, a DNA sequence changes during the replication. Replication is a process of DNA cells creating a copy of self so as to pass the genetic information to the daughter cells. So when there is a change in DNA's original structure due to any error during the replication, it is a mutation. Some of the mutations are:

- Substitutions: One base is replaced with another base
- Insertions: Where new bases are added to the existing sequence
- Deletions: Where existing bases are deleted from the sequence

[1]

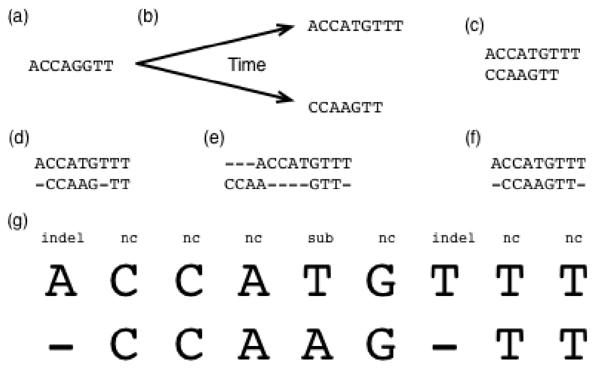


Figure 1.Sequence evolution and pairwise sequence alignment

Sequence alignment

Sequence alignment is aligning or positioning of elements in two sequences such that most of their similar elements are aligned together. The aligned elements are then compared for similarity. Sequence alignment constructs correspondence between bases or codons of DNA,

[1] http://readiab.org/

RNA or sequences of amino acids. Thus, Sequence alignment in bioinformatics is a research-based area dedicated to come up with tools to study and find the similarity in DNA or amino acid sequences through the means of computation. It is significant to gather information on genes, their structure, function or genetic evolution for various research purposes. It is very crucial in determining if two sequences are very much similar and have a similar structure and function, which can be a result of existence from the same ancestor. By computing distances between the aligned sequences, we can refer about the evolutionary processes they have gone. The study can be estimating time between emerging from common ancestor to present form, but also a chance of presenting a hypothesis or recreating one evolutionary event in the past or a sequence of them. Aligning the different sequences allows the researchers to observe various things like detecting their overlap or to take a note that one is a part of another or that they share a common subset of sequences. Aligning is not limited to pairwise sequences but there could also be a multiple alignment among sequences.

History

In retrospect, the sequence alignment history is not so old. As significant the sequence alignment method is, it equally took time to develop. In the year of 1970 was when history was made for sequence alignment by presenting a dynamic approach, which was a Needleman – Wunsch algorithm. Although towards the end of 1950's protein sequencing was beginning to emerge through multiple alignments. But they were used only for data presentations or the structure and function of the proteins but not for evolutionary analysis.
They were also slow and gathering the data was very tedious. Additionally, major of the sequence alignments were done by hand studying the biological factors that caused variation in sequences. In 1984, Hogeweg and Hesper developed computerized procedures based on pairwise sequence alignment and progressive alignment strategy for multiple alignments.

Various uses of alignment

Alignment of genes has the following significance:

1. It can be used to categorize and classify the DNA or protein sequences. The comparison between an unknown sequence and a known sequence can yield an insight into whether the two sequences are in the same category.
2. Pairwise sequence assembly that identifies the common parts of two sequences can obtain evolutionary relationships.
3. The function of the DNA sequence can be determined by aligning the protein sequence with the DNA sequence.
4. Similarly, comparing a known one with the unknown can also identify the unknown DNA sequence’s function.
5. Again, by comparing with a known structure of a sequence, the physical structure of the protein sequences can be predicted.
6. Pairwise sequence alignment is helpful in sequencing new genes.

Pairwise alignment and Multiple Alignment

The pairwise alignment is the alignment in which we compare the two biological sequences of DNA, RNA or proteins to locate regions of similarity that may indicate functional, structural and evolutionary relationships between pairs. These can be either local or global alignments. Pairwise alignment is basically done to find out the conserved regions between the two sequences and similarity searches in the database. Examples these are: BLAST, LALIGN, EMBOSS Needle, EMBOSS Water.

Whereas, in multiple alignment, we compare three or more biological sequences. This kind of alignment is generally a global alignment. For multiple alignments, progressive alignment is adopted. In this technique, the iterative call is made to pairwise alignment. First, the most closely related pairs are aligned and then the alignment is done for next closely related sequence to that pair and so on. Examples of multiple alignments are: MUSCLE, T-Coffee, MAFFT and CLUSTALW. Applications of multiple alignments are given as below:

- To detect variable and conserved regions in a family of genetic structure.
- For the phylogenetic analysis that relates to a tree, estimating a rate of substitution etc.
- To observe homology between an existing gene family and a newly developed gene of protein structure
- To exhibit homology in multi-gene families

Types of Alignment

It is equally challenging for biologist to accomplish alignment by hand calculation as there may be millions long sequences to compare for similarity. Thus, comes the role of computational approaches to solve the problem. There are two types of sequence alignment based on the techniques of alignment. First one is **Global Alignment** and other is **Local Alignment**.

Global vs. Local Alignment

Global alignment is aligning two sequences throughout their entire length. In this kind of alignment, the two sequences of roughly same sized are used for comparison. And the alignment is done end to end, so incase when the size is very different, you end up with lots of gaps in global alignment. The genes that are closely related are taken and are studied for mutations or polymorphism in them. We shall continue our further study on global alignment of genes through a dynamic programming approach using Needleman-Wunsch pairwise alignment algorithm.

----QVERYSEQ----

ABCDQVERYSEQTYWZ

Local alignment is matching the region of most similarity between two sequences. Thus it covers only a part of the sequences. It is very useful in aligning more dissimilar sequences that have certain region of similar sequences within their entire region. The smith-walterman algorithm finds the best local alignment between two sequences.

WSEQVDNCEA

KSEQVENCEN

Local alignment is more flexible as oppose to global as scoring can be high as we are considering only a part of sequences. The benefit over global alignment is that similar regions appearing in various orders in two biological sequences can be figured out as related which is known as domain shuffling. It is used for cases like:

- Sequences of varying length are compared
- Long sequences with coding and non-coding regions are compared
- Proteins from different families are compared to find conserved region
- Global alignment doesn't give the expected score, but gives hint about the similarity

Global Alignment using Needleman-Wunsch Algorithm

As stated above, a general approach to global alignment is Needleman-Wunsch algorithm using dynamic programming. The Needleman-Wunsch algorithm is used for aligning protein or nucleotide sequences to find their structural and functional similarity. Developed by Saul B. Needleman and Christian D. Wunsch, the algorithm is the first applications of dynamic programming to compare biological sequences. The algorithm redesigns the solution from small solutions by dividing a large problem into small sets of problems.

For the alignment of sequences using this algorithm, let us consider the sequences *s1* and *s2* as CGCA and CACGTAT. To check how similar the genes are, they are assigned a score. The objective is to seek to determine the best alignment out of all the possible candidate alignments, the one that gives the highest overall similarity. And the alignment that gives the best score is taken into consideration. Then we have our scoring system as below:

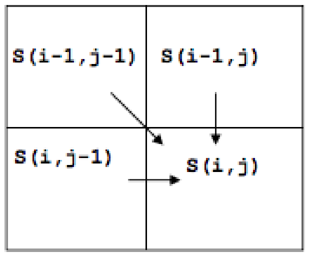
- For each letter in both sequences that match, they are given a MATCH score of +1
- For each substitution or mismatch, they are given a MISMATCH penalty score of 0
- For each insertion or deletion, they are given a GAP penalty score of -1

The best alignment given using the algorithm then will look like one of these:

```
CACGTAT        CACGTAT        CACGTAT
_ _ CGCA_      C_ _GCA_       CGC_ _ A _
```

To begin with the algorithm, we then have to create a matrix A of size *n by m* where n = Length of sequence *s1* + 1 and m = length of sequence *s2* + 1. The alignment starts from the left upper corner of the matrix and follows usually a diagonal path down towards the bottom right. When two letters each from s1 and s2 are aligned, the path is diagonal. Each cell corresponds to a pairing of one letter from each sequence and is given the maximum score that comes from neighboring cells plus either match, mismatch or gap score that we will discuss in detail below.

[2]Initialization and building a matrix



During initialization, the first rows and first columns are assigned a value same way regardless of the sequences compared. So the matrix A[0][0] will have a value of 0. Then for

[2] [2] http://www.cs.otago.ac.nz/cosc348/alignments/Lecture05_GlobalAlignment.pdf

each cell, scores in first row and first columns is set to be the gap score multiplied by the distance from the origin.

For each column in row 0 starting from column 1

A[0][column] = Gap_Score * column

For each row in all first column starting from row 1

A[row][0] = GapScore * row

Now we start filling the rest of the matrix with the best scores. We get the best score for the existing cell A[1][1] based on what its neighboring cells, i.e A[0][0], A[0][1] and A[1][0] in addition to the gaps /match /mismatch each has.

For each cell that is on top most of the existing cell, i.e. A[0][1] , we have to take this cell's score plus the vertical gap score which is -1, so it will be: **scoreTop = A[0][1] – 1**

Similarly, we calculate the horizontal gap score from the right cell i.e. A[1][0], so we have: **scoreLeft = A[1][0] - 1**

At last, we have also had to look for the left diagonal of the existing cell, i.e. A[0][0]. However, we have to also check if the letters at this position that is i-1th index for s1 and j-1th index for s2 matches or not. If they match then we will have: **scoreDiagonal = A[0][0] + 1**

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

Else if they don't match, then we assign the mismatch score of 0. So it will be: **scoreDiagonal = A[0][0] + 0**

Therefore the best score for cell A[1][1] would be the maximum score of scoreTop, scoreLeft and scoreDiagonal.

And then we fill rest of the cells iteratively and continue until we reach A[m-1][n-1], that is the last cell of our ***m by n*** matrix. The score that we get at the last cell gives the scoring of the alignment and that is the optimal score of the alignment.

So once our initialization of the scores is done and we reach to the bottom right

corner, our matrix will look like the table below that we did for our sequences CACGTAT and CGCA. And the matrix[n-1][m-1] gives us the optimal score of the alignment.

		C	G	C	A
	0	-1	-2	-3	-4
C	-1	1D	0	-1	-2
A	-2	0	1D	0	0
C	-3	-1	0	2D	1
G	-4	-2	0	1V	2
T	-5	-3	-1	0V	1
A	-6	-4	-2	-1	1D
T	-7	-5	-3	-2	0 V

Creating a diagonal string

Now that we have a matrix filled with scores, next step is to find the optimal path using this matrix. As we are working with computers so as oppose to using arrows, we are using strings to give us the direction. Thus, to represent a horizontal path, we are using ‘H’, to represent a vertical path we are using ‘V’ and for diagonal path we will use ‘D’. As we start making the path from the last cell which is at index *n-1 by m-1*. Thus, we do our initialization as below:

1. Set our current row to n-1; currRow = n-1
2. Set our current column to m-1; currColumn = m-1
3. Initialize our string to an empty string as dString = “”
4. While currRow or currColumn is not equal to zero, do the following iteratively.
    a. If currRow is equal to 0, add ‘H’ to dString, and decrement currColumn by 1
    b. Else if currColumn is equal to 0, add ‘V’ to dString and decrement currRow by 1

c. Else if score in current cell i.e A[currRow][currColumn] is equal to A[currRow][currColumn - 1] + gapScore , then add ‘H ’ to dString and decrement currColumn by 1

d. Else if score in current cell is equal to A[currRow -1][currColumn] + gap, the add ‘V’ to dString and decrement currRow by 1

e. Else it will be a diagonal path, so add ‘D’ to dString and decrement both currRow and currColumn by 1

5. After loop ends the reverse of the dString will be our path. Process ends.

   As we can see from the table above, we have marked our paths based on the comparisons of the scores we got in our scoring matrix and the end path we get from building our directional string would be: **VDVVDDD**

Finding Alignments from the Path

Now that we have a path, we can now derive an alignment for our strings s1 and s2 from this path. We will have two strings representing the alignments as sequence1Aligned and sequence2Alignned as an empty string first. Again, we set our starting index for sequence s1 and s2 as seq1pos initialized to N-1 and seq2pos initialized to M-1 respectively. Similarly, we have a counter to keep track of the index of path so that we stop as we reach the end of the dstring. Hence, call this counter as dirPos and set it to 0 at first. Thus, the algorithm would be:

1. While the value of dirPos is less than length of the path/dString, continue iterating and do:

   a. If dString[dirPos] is equal to ‘D’, then add letter at seq1Pos of s1 to sequence1Aligned and letter at seq2Pos of s2 to sequence2Aligned, i.e.

      sequence1Aligned += s1[seq1Pos] & sequence2Aligned += s2[seq2Pos]

Also, decrement seq1Pos and seq2Pos by 1.

b. Else if dString[dirPos] is equal to ‘V’, then add s1[seq1Pos] to sequence1Aligned and add ‘-’ to sequence2Aligned, i.e.

   Sequence1Aligned += s1[seq1Pos] & sequence2Aligned += ‘-’

   Also, decrement seq1Pos by 1

c. Else we know it is going to be ‘H’. So, add ‘-’ to sequence1Aligned and add s2[seq2Pos] to sequence2Aligned, i.e.

   Sequence1Aligned += ‘-’ & sequence2Aligned += s2[seq2Pos]

   Also, decrement seq2Pos by 1

d. At the end of the conditions, increment dirPos by 1

2. The two resulting strings are the alignments for sequences s1 and s2.

Thus, the resulting global alignments from this step will be:

s1(CACGTAT): CACGTAT

s2(CGCA)   : CGC--A-

And the score would be: 1+0+1-1-1+1-1 = **0** (same as the last cell on table above)

Time Efficiency of the algorithm

The algorithm for filling the matrix is O(n*m) time and creating a directional string is O(n*m) time, and yielding global alignment using directional strings is O also (n + m) steps.

Analysis of the algorithm

When the quality of the global algorithm is of crucial essence, the Needleman-Wunsch Algorithm is always used for the global alignment. It is a powerful algorithm for finding the best alignment of two similar length sequences and similar across their entire length. However, considering the running time, this algorithm is very expensive. Hence, it is not so suitable and can be slow for aligning multiple and long sequences. It is always a challenge to achieve high quality sequence alignment while optimizing the running time. Several efforts have been made. The development of FOGSAA is one such effort on overcoming the challenges and solving the problem faced by algorithm. Stands for Fast Optimal Sequence Alignment Algorithm, FOGSAA, is remarkably faster than Needleman – Wunsch algorithm. It claims to be faster by 70% - 90% for sequences of higher similarity at the same time giving the same optimal score.

Different Sequence Alignment Software

Computer technologies and software has been an essential tool in computing complex algorithms and calculations. Similarly, for sequence alignment, there have been numerous software tools developed to aid the research process. The common ones and their uses in different areas of sequence alignment are as follows:

1. **BLAST**: Stands for Basic Local Alignment Search Tool, originated at National Center for Biotechnology Information (NCBI). It is a database search-based tool and is very important software package used in bioinformatics for rapid searching of nucleotide and protein databases. The program compares both protein and nucleotide bases to find the regions of local similarity between them. For software to run, BLAST requires a query sequence to search for and target sequence containing one or multiple sequences database, which we are searching against. Ion the database, it will then find sequences that are similar to the query sequence. Generally, query is much smaller than the ones in database. Different types of BLAST are available based on sequences to compare. For e.g.: CS-BLAST, DIAMOND etc. The popularity of BLAST is due to various reasons:

➢ Sequence similarity is a strong tool for identifying relationship between unknown sequences

- It is fast given how fast and big the field is growing.
- It is reliable
- It is flexible and can be adapted to many sequence analysis scenarios

2. **FASTA**: It is a DNA and protein sequence alignment software tool termed as FASTP by David J. Lipman and William R. Pearson in 1985. Pronounced as "Fast-A", it stands for "FAST-ALL", as it works with any alphabets, an extension of "Fast-P" for protein and "FAST-N" for nucleotide alignments. It searches a corresponding sequence of the given query nucleotide or protein using local sequence alignment to find the match in database. The main areas of FASTA package are:

- Precisely computing the similarity statistics to assist in decision making as to whether the alignment is coincidence or check if there is the homology.

3. Other software tools include softwares like LALIGN, FFAS, LAST, GeneWise, ACANA, YASS, Genome Compiler, needle, Bioperl, DNASTAR etc based on pairwise alignment approach and ALE, AMAP, Bali-phy, AlignMe, PRALINE, based on multiple sequence alignment.

References

- http://phylonetworks.blogspot.com/2016/05/the-early-history-of-sequence-alignment.html
- http://www.cs.utoronto.ca/~brudno/bcb410/lec2notes.pdf
- Bioinformatics Technologies; Yi Ping Phoebe Chen
- Bioinformatics; Andrzej Polanski, Marek Kimmel
- https://en.wikipedia.org/wiki/Sequence_alignment
- https://en.wikipedia.org/wiki/List_of_sequence_alignment_software
- https://en.wikipedia.org/wiki/BLAST
- http://bioinfo.mbb.yale.edu/mbb452a/2003/projects-2003/dinu.pdf
- Sequence Alignment, Michael S. Rosenberg
- https://www.nature.com/articles/srep01746
- https://pdfs.semanticscholar.org/9f61/e3f06f288b2fccc1ebcd3346fd0b8719bfa6.pdf
- http://www.cs.otago.ac.nz/cosc348/alignments/Lecture05_GlobalAlignment.pdf