Covenant Journal of Physical and Life Sciences (CJPL) Vol. 1, No. 1 (Maiden Edition), August, 2013

Covenant University Journal

A Two-Phase Dynamic Programming Algorithm Tool for DNA Sequences

Oyelade Jelili &
Fatumo Segun

Abstract:

Sequence alignment has to do with the arrangement of DNA, RNA, and protein sequences to identify areas of similarity. Technically, it involves the arrangement of the primary sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Similarity may be a consequence of functional, structural, or evolutionary relationships between the sequences. If two sequences in an alignment share a common ancestor, mismatches can be interpreted as mutations, and gaps as insertions. Such information becomes of great use in vital areas such as the study of diseases, genomics and generally in the biological sciences. Thus, sequence alignment presents not just an exciting field of study, but a field of great importance to mankind. In this light, we extensively studied about seventy (70) existing sequence alignment tools available to us. Most of these tools are not user friendly and cannot be used by biologists. The few tools that attempted both Local and Global algorithms are not ready available freely. We therefore implemented a sequence alignment tool (CU-Aligner) in an understandable, user-friendly and portable way, with click-of-a-button simplicity. This is done utilizing the Needleman-Wunsh and Smith-Waterman algorithms for global and local alignments, respectively which focuses primarily on DNA sequences. Our aligner is implemented in the Java language in both application and applet mode and has been efficient on all windows operating systems.

Keywords — sequence, global alignment, local alignment, dynamic programming and java applet.

1. INTRODUCTION

DNA sequencing is a process of determining the exact order of over 3 billion chemical building blocks i.e. the A, C, T, and G bases that make up the DNA. This process was an important scientific discovery, as before sequencing was actually done, it was assumed that DNA consisted hundreds of thousands of genes. Upon performing sequencing, though, it was realized that there are approximately 20000-25000 genes in the human DNA. It is with the foundation of sequencing that sequence alignment has relevance.

According to Batzoglou [1], sequence alignment is the analysis of the sequence of the elements of genes or proteins. It involves the arrangement of DNA, RNA, or protein sequences to identify regions of similarity between them. Such similarity may be a consequence of functional, structural, or evolutionary relationships between the sequences. If two sequences in an alignment share a common ancestor, mismatches can be interpreted as mutations. Aligned sequences of nucleotide or amino acid residues are represented as rows within a matrix. Gaps are introduced between the residues so residues with identical or similar characters are aligned in the successive columns.

In situations where two sequences in an alignment share a common ancestor, mismatches can be interpreted as point mutations and gaps as indels (that is, insertion or deletion mutations) introduced in either or both of the mutations in the period within which they diverged from one another. In protein sequence alignment, the degree of similarity between amino acids occupying a particular position in the sequence can be interpreted as a rough measure of how conserved a particular region or sequence motif is among lineages. The absence of substitutions, or the presence of only very conservative substitutions (that is, the substitution of amino acids whose side chains have similar biochemical properties), in a particular region of the sequence, suggests that such a region has structural or functional importance. Though DNA and RNA nucleotide are more similar to each other than to amino acids, the conservation of base pairing still can indicate a similar functional or structural role. It is of course of some importance to note that sequence alignment can be used for non-biological sequences, such as those present in financial data or in natural language [9]. In the light of this, we observe that sequence alignment can be used to identify similarities in a series of letters and words present in human language.

2. PROBLEM STATEMENT

Very short and similar sequences can adequately be aligned by hand, but most interesting problems require lengthy, highly variable or extremely numerous sequences which cannot be aligned solely by human effort. We can see, for example that two 12 residue sequences, would require considering approximately one million alignments, and two 150 residue sequences would require considering approximately 10^{88} alignments (interesting considering that 10^{78} is the estimated number of atoms in the universe). Hence, human effort is thus applied in formulating algorithms to simplify and reduce such a lengthy process, and also occasionally in adjusting results to reflect patterns difficult to express algorithmically. Computational approaches to sequence alignment generally fall into two categories, these being Global and Local Sequence Alignment [2].

2.1 Pair wise Alignment

Pair wise sequence alignment methods, are used to find the best matching local or global alignments of two query sequences. It can only be utilized when considering two sequences at a time, and are often used for methods that do not require extreme precision. The three primary techniques for pair wise alignment are Dot-Matrix methods, Dynamic Programming, and Word Methods, though multiple sequence alignment techniques can be used to solve pair wise alignment cases. All three methods have their strong and weak points, but all have difficulty with highly repetitive sequences of low information content, especially where the number of repetitions differs in the two sequences to be aligned.

2.2 Smith-Waterman algorithm

The Smith-Waterman algorithm was first proposed by Temple Smith and Michael Waterman in 1981. It is a well-known algorithm for performing local sequence alignment [7]; that is, for determining similar regions between two nucleotide or protein sequences. Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure

2.3 Needleman–Wunsch algorithm

The Needleman–Wunsch algorithm performs a global alignment on two sequences. It is commonly used in bioinformatics to align protein or nucleotide sequences [8]. The Needleman–Wunsch algorithm is an example of dynamic programming, and was the first application of dynamic programming to biological sequence comparison.

2.4 Dynamic Programming

Global alignments, which attempt to align every residue in every sequence, are most useful when the sequences in the query set are similar and of roughly equal size. Local alignments are more useful for dissimilar sequences that are suspected to contain regions of similarity or similar sequence motifs within their larger sequence context. Very common and useful alignment techniques for the implementation of global and local sequence alignment are Needleman Wunsch and Smith Waterman algorithms [8]. These techniques are based on the dynamic programming principle.

Dynamic Programming Principle: The technique of dynamic programming is theoretically applicable to any number of sequences; however, because it is computationally expensive in both time and memory, it is rarely used for more than three or four sequences in its most basic form. This method requires constructing the *n*-dimensional equivalent of the sequence matrix formed from two sequences, where *n* is the number of sequences in the query. Standard dynamic programming is first used on all pairs of query sequences and then the "alignment space" is filled in by considering possible matches or gaps at intermediate positions, eventually constructing an alignment essentially between each two-sequence alignment. Although this technique is computationally expensive, its guarantee of a global optimum solution is useful in cases where only a few sequences need to be aligned accurately.

The dynamic programming principle involves three (3) major steps:

1. Initialization: This is the first step in dynamic programming. It involves the creation of a matrix with M+ 1 column and N+ 1 row, where M and N are the sizes of the sequences to be aligned.
2. Matrix fill (scoring): This is where each cell in the created matrix is filled with calculated values.
3. Trace back (alignment): The trace back step simply determines the actual alignment that results in a maximum score.

This work, utilizes the Needleman-Wunsch and Smith-Waterman algorithms, which are examples of dynamic programming, to implement the pairwise sequence alignment of DNA sequences. CU-Aligner is a standalone application but can also be deployed over the internet, thus being available to a much wider audience.

In Table 1, we show list of existing sequence alignment tools available with brief description of the feature they offer.

3. BASIC FEATURES OF CU-ALIGNER

Our tool presents the following features together. It makes the work of a biologist easier.

User-Friendly Nature- CU-Aligner presents what could otherwise be a cumbersome task by hand, in a plain and understandable format with click-of-a-button simplicity in carrying out alignments. Also its specialization in focusing on DNA sequences aids its simplicity. This is a solution to situations where multiple functionality of aligners, adversely affect understandability of such programs.

Portability- As this Aligner has been implemented in the Java language it presents the advantage of being able to be run on all platforms, this being possible as a result of the portable nature of the Java Language.

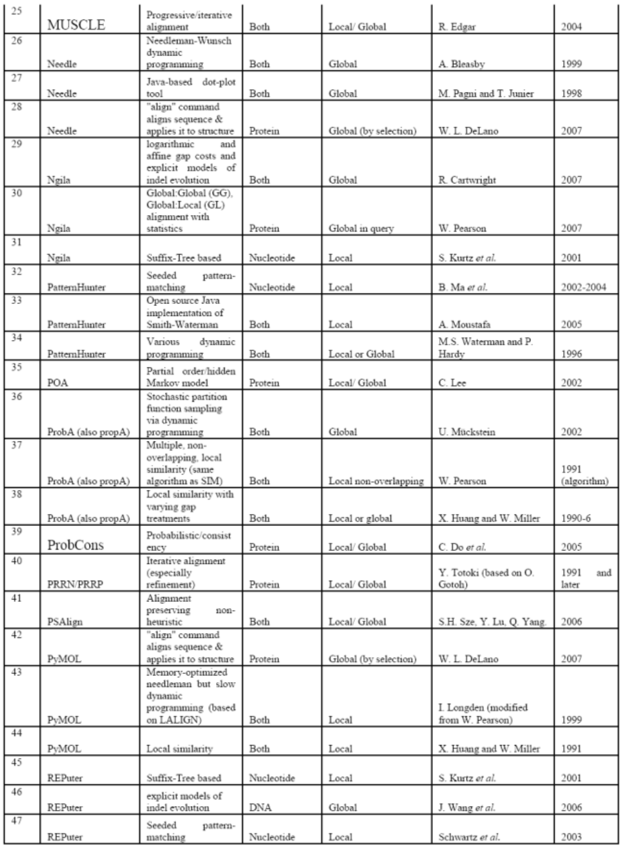
Understandable Output- CU-Aligner presents the result of an alignment in a manner not requiring special knowledge to understand. It also presents results as soon as they are computed, as opposed to a situation where output is sent to the user at a later time.

Open Source: CU-Aligner is offered free of charge upon request.

Flexibility- CU-Aligner presents a user with the choice of parameters to use in performing an alignment, these being the gap penalty, the match and mismatch scores.

Table 1: *List of different sequence alignment tools available and their features*

S/N	Name	Description	Sequence Type	Alignment Type	Author	Year
1	BioPerl dpAlign	dynamic programming	Both	Both + Ends-free	Y. M. Chan	2003
2	BLASTZ	Seeded pattern-matching	Nucleotide	Local	Schwartz *et al.*	2003
3	DNA Baser	Multi alignment	Both	Local/Global+ Post processing	M. Gabriel	2005
4	DNADot	Web-based dot-plot tool	Nucleotide	Global	R. Bowen	1998
5	DOTLET	Java-based dot-plot tool	Both	Global	M. Pagni and T. Junier	1998
6	Ed'Nimbus	Seeded filtration	Nucleotides	Local	P. Peterlongo *et al.*	2006
8	Geneious	Progressive/Iterative alignment; ClustalW plugin	Both	Local/ Global	A.J. Drummond *et al.*	2006
9	GGSEARCH, GLSEARCH	Global:Global(GG), Global:Local(GL) alignment with statistics	Protein	Global in query	W. Pearson	2007
10	JAligner	Open source Java implementation of Smith-Waterman	Both	Local	A. Moustafa	2005
11	Kalign	Progressive alignment	Both	Global	T. Lassmann	2005
12	LALIGN	Multiple, non-overlapping, local similarity (same algorithm as SIM)	Both	Local non-overlapping	W. Pearson	1991 (algorithm)
13	MAFFT	Progressive/iterative alignment	Both	Local/ Global	K. Katoh *et al.*	2005
14	Matcher	Memory-optimized needleman but slow dynamic programming (based on LALIGN)	Both	Local	I. Longden (modified from W. Pearson)	1999
15	MAVID	Progressive alignment	Both	Global	N. Bray and L. Pachter	2004
16	MCALIGN2	explicit models of indel evolution	DNA	Global	J. Wang *et al.*	2006
17	MCALIGN2	Seeded pattern-matching	Nucleotide	Local	Schwartz *et al.*	2003
18	MCALIGN2	Seeded pattern-matching	Nucleotide	Local	B. Ma *et al.*	2004
19	MSA	Dynamic programming	Both	Local/ Global	D.J. Lipman *et al.*	1989 (modified 1995)
20	MULTALIN	Dynamic programming/clustering	Both	Local/ Global	F. Corpet	1988
21	Multi-LAGAN	Progressive dynamic programming alignment	Both	Global	M. Brudno *et al.*	2003
22	MUMmer	Suffix-Tree based	Nucleotide	Global	S. Kurtz *et al.*	2004
23	MUMmer	Web-based dot-plot tool	Nucleotide	Global	R. Bowen	1998
24	MUMmer	Stochastic partition function sampling via dynamic programming	Both	Global	U. Mückstein	2002



25	MUSCLE	Progressive/iterative alignment	Both	Local/ Global	R. Edgar	2004
26	Needle	Needleman-Wunsch dynamic programming	Both	Global	A. Bleasby	1999
27	Needle	Java-based dot-plot tool	Both	Global	M. Pagni and T. Junier	1998
28	Needle	"align" command aligns sequence & applies it to structure	Protein	Global (by selection)	W. L. DeLano	2007
29	Ngila	logarithmic and affine gap costs and explicit models of indel evolution	Both	Global	R. Cartwright	2007
30	Ngila	Global:Global (GG), Global:Local (GL) alignment with statistics	Protein	Global in query	W. Pearson	2007
31	Ngila	Suffix-Tree based	Nucleotide	Local	S. Kurtz *et al.*	2001
32	PatternHunter	Seeded pattern-matching	Nucleotide	Local	B. Ma *et al.*	2002-2004
33	PatternHunter	Open source Java implementation of Smith-Waterman	Both	Local	A. Moustafa	2005
34	PatternHunter	Various dynamic programming	Both	Local or Global	M.S. Waterman and P. Hardy	1996
35	POA	Partial order/hidden Markov model	Protein	Local/ Global	C. Lee	2002
36	ProbA (also propA)	Stochastic partition function sampling via dynamic programming	Both	Global	U. Mückstein	2002
37	ProbA (also propA)	Multiple, non-overlapping, local similarity (same algorithm as SIM)	Both	Local non-overlapping	W. Pearson	1991 (algorithm)
38	ProbA (also propA)	Local similarity with varying gap treatments	Both	Local or global	X. Huang and W. Miller	1990-6
39	ProbCons	Probabilistic/consistency	Protein	Local/ Global	C. Do *et al.*	2005
40	PRRN/PRRP	Iterative alignment (especially refinement)	Protein	Local/ Global	Y. Totoki (based on O. Gotoh)	1991 and later
41	PSAlign	Alignment preserving non-heuristic	Both	Local/ Global	S.H. Sze, Y. Lu, Q. Yang.	2006
42	PyMOL	"align" command aligns sequence & applies it to structure	Protein	Global (by selection)	W. L. DeLano	2007
43	PyMOL	Memory-optimized needleman but slow dynamic programming (based on LALIGN)	Both	Local	I. Longden (modified from W. Pearson)	1999
44	PyMOL	Local similarity	Both	Local	X. Huang and W. Miller	1991
45	REPuter	Suffix-Tree based	Nucleotide	Local	S. Kurtz *et al.*	2001
46	REPuter	explicit models of indel evolution	DNA	Global	J. Wang *et al.*	2006
47	REPuter	Seeded pattern-matching	Nucleotide	Local	Schwartz *et al.*	2003

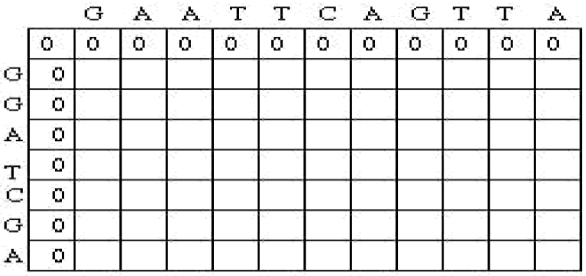
48	RevTrans	Combines DNA and Protein alignment, by back translating the protein alignment to DNA.	DNA/Protein (special)	Local/ Global	Wernersson and Pedersen	2003 (newest version 2005)
49	SAGA	Sequence alignment by genetic algorithm	Protein	Local/ Global	C. Notredame *et al.*	1996 (new version 1998)
50	SAM	Hidden Markov model	Protein	Local/ Global	A. Krogh *et al.*	1994 (most recent version 2002)
51	SEQALN	Various dynamic programming	Both	Local or Global	M.S. Waterman and P. Hardy	1996
52	SEQALN	Suffix-Tree based	Nucleotide	Global	S. Kurtz *et al.*	2004
53	SEQALN	Web-based dot-plot tool	Nucleotide	Global	R. Bowen	1998
54	SIM	Local similarity	Both	Local	X. Huang and W. Miller	1991
55	SIM	logarithmic and affine gap costs and explicit models of indel evolution	Both	Global	R. Cartwright	2007
56	SIM	Global:Global (GG), Global:Local (GL) alignment with statistics	Protein	Global in query	W. Pearson	2007
57	SIM, GAP, NAP, LAP	Local similarity with varying gap treatments	Both	Local or global	X. Huang and W. Miller	1996
58	SIM, GAP, NAP, LAP	Needleman-Wunsch dynamic programming	Both	Global	A. Bleasby	1999
59	SIM, GAP, NAP, LAP	Java-based dot-plot tool	Both	Global	M. Pagni and T. Junier	1998
60	SLIM Search	Ultra-fast blocked alignment	Both	Both	L. Bloksberg	2004
61	SSEARCH	Local (Smith-Waterman) alignment with statistics	Protein	Local	W. Pearson	1981
62	Stretcher	Memory-optimized but slow dynamic programming	Both	Global	I. Longden (modified from G. Myers and W. Miller)	1999
63	T-Coffee	More sensitive progressive alignment	Both	Local/ Global	C. Notredame *et al.*	2000
64	Tranalign	Aligns nucleic acid sequences given a protein alignment	Nucleotide	NA	G. Williams (modified from B. Pearson)	2002
65	Water	Smith-Waterman dynamic programming	Both	Local	A. Bleasby	1999
66	wordmatch	*k*-tuple pairwise match	Both	NA	I. Longden	1998
67	YASS	Seeded pattern-matching	Nucleotide	Local	L. Noe and G. Kucherov	2006

4. MATERIALS AND METHODS

CU-Aligner was implemented in the Java language in both application and applet mode. Deployment on to a web page can be done with simple coding statements once, the web page is built using Macromedia Dreamweaver.

5. DESIGN OF CU-ALIGNER

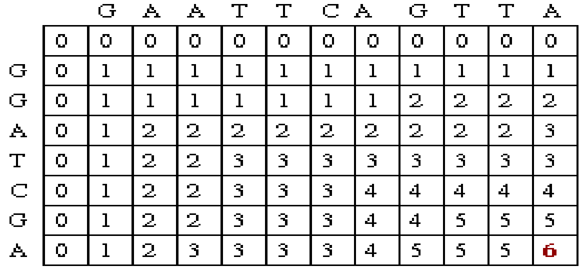
CU-Aligner is designed based upon the Needleman-Wunsch and Smith-Waterman algorithms (Wunsch and Waterman 1970), for Global and Local alignments, respectively. The aligner program implements each stage of each algorithms implementation: These being, the Initialization, Matrix-Fill and Trace back Stages as depicted in the figure 1, 2 and 3. The flow of the CU-Aligner is given is figure 4.



		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Figure 1: Initialization

The first step in the global alignment dynamic programming approach is to create a matrix with M + 1 columns and N + 1 rows where M and N correspond to the size of the sequences to be aligned.



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

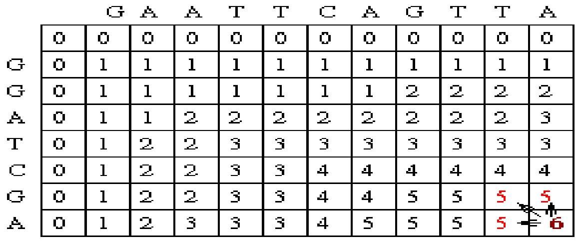
Figure 2: Matrix-Fill

This is where each cell in the created matrix is filled with calculated values.

In order to find $M_{i,j}$ for any i,j it is necessary to know the score for the matrix positions to the left, above and diagonal to i, j. In terms of matrix positions, it is necessary to know $M_{i-1,j}$, $M_{i,j-1}$ and $M_{i-1, j-1}$. i.e.

$$G[i,j] = \max \begin{cases} G[i-1,j-1] + s(x_i, y_j) \\ G[i-1,j] + g \\ G[i,j-1] + g \end{cases}$$

$$G[0,j] = j \cdot g, \quad j \in [0,\ldots,n]$$
$$G[i,0] = i \cdot g, \quad i \in [0,\ldots,m]$$



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

Figure 3: Traceback

The traceback step determines the actual alignment(s) that result in the maximum score. Note that with a simple scoring algorithm such as one that is used here, there are likely to be multiple maximal alignments. The traceback step begins in the M,J position in the matrix, i.e. the position that leads to the maximal score. In this case, there is a 6 in that location.

Traceback takes the current cell and looks to the neighbor cells that could be direct predecessors. This means it looks to the neighbor to the left (gap in sequence #2), the diagonal neighbor (match/mismatch), and the neighbor above it (gap in sequence #1). The algorithm for traceback chooses as the next cell in the sequence one of the possible predecessors. They are all also equal to 5.

Since the current cell has a value of 6 and the scores are 1 for a match and 0 for anything else, the only possible predecessor is the diagonal match/mismatch neighbor.

If more than one possible predecessor exists, any can be chosen. This gives us a current alignment of :

(Seq #1) A
|
(Seq #2) A

Continuing on with the traceback step, we eventually get to a position in column 0 row 0 which tells us that traceback is completed. One possible maximum alignment is :

G A A T T C A G T T A
| | | | | |
G G A _ T C _ G _ _ A

And an alternate derived from the same procedure:

G _ A A T T C A G T T A
| | | | | |
G G _ A _ T C _ G _ _ A

There are more alternative solutions each resulting in a maximal global alignment score of 6. Since this is an exponential problem, most dynamic programming algorithms will only print out a single solution

The same principle is followed in the implementation of the Smith-Waterman algorithm, except that the matrix is filled on the following basis:

$$L[i,j] = \max \begin{cases} L[i-1,j-1] + s(x_i,y_j) \\ L[i-1,j] + g \\ L[i,j-1] + g \\ 0 \end{cases}$$

$$L[0,j] = 0, \quad j \in [0,\ldots,n]$$

$$L[i,0] = 0, \quad i \in [0,\ldots,m]$$

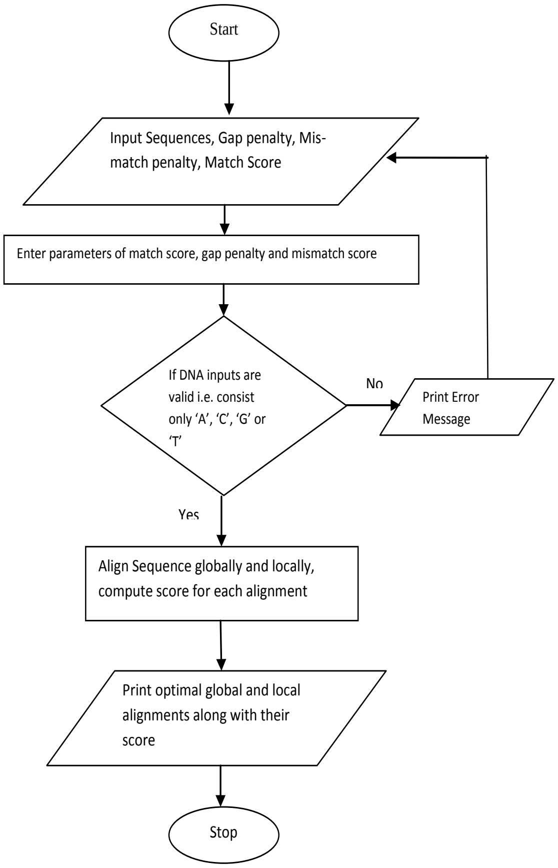
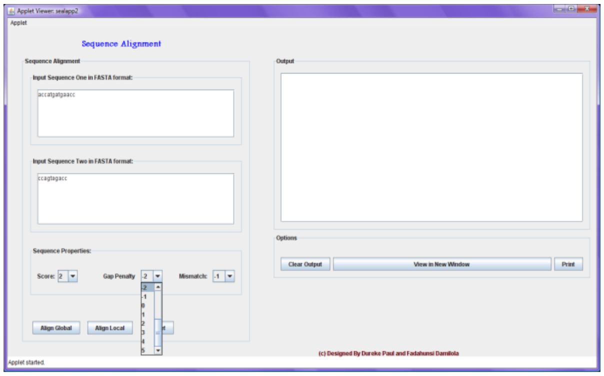


Figure 4: *Flowchart of Pairwise Aligner*

6. RESULTS

The pairwise aligner accepts two DNA sequences (see figure 5) and produces their global and local alignments, as well as their scoring matrix (see figure 6 and 7 respectively). A deployable version if CU-Aligner on the internet is shown in figure 8.

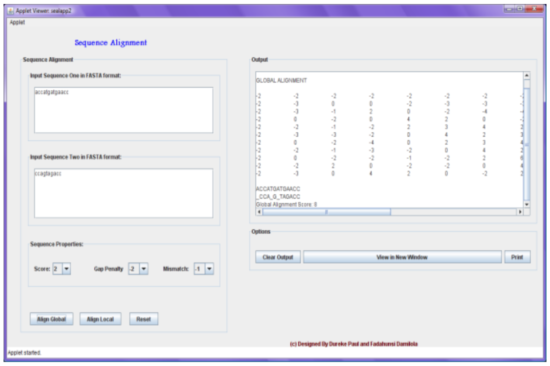


Input Sequence One in FASTA format: accatgatgaacc

Input Sequence Two in FASTA format: ccagtagacc

Sequence Properties: Score: 2 Gap Penalty -2 Mismatch: -1

Figure 5: *Pairwise Aligner Interface*



Input Sequence One in FASTA format: accatgatgaacc

Input Sequence Two in FASTA format: ccagtagacc

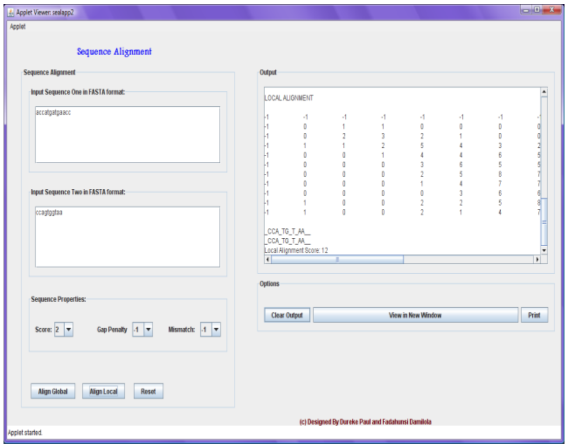
Sequence Properties: Score: 2 Gap Penalty -2 Mismatch: -1

```
GLOBAL ALIGNMENT

-2      -2      -2      -2      -2      -2      -2
-2      -3      0       0       -2      -3      -3
-2      -3      -1      2       0       -2      -4
-2      0       -2      0       4       2       0
-2      -2      -1      -2      2       3       4
-2      -3      -3      -2      0       4       2
-2      0       -2      -4      0       2       3
-2      -2      -1      -3      -2      0       4
-2      0       -2      -2      -1      -2      2
-2      -2      2       0       -2      -2      0
-2      -3      0       4       2       0       -2

ACCATGATGAACC
_CCA_G_TAGACC
Global Alignment Score: 8
```

Figure 6: *Local Alignment*



Input Sequence One in FASTA format: accatgatgaacc

Input Sequence Two in FASTA format: ccagtggtaa

Sequence Properties: Score: 2 Gap Penalty -1 Mismatch: -1

```
LOCAL ALIGNMENT

-1      -1      -1      -1      -1      -1      -1
-1      0       1       1       0       0       0
-1      0       2       3       2       1       0
-1      1       1       2       5       4       3
-1      0       0       1       4       4       6
-1      0       0       0       3       6       5
-1      0       0       0       2       5       8
-1      0       0       0       1       4       7
-1      0       0       0       0       3       6
-1      1       0       0       2       2       5
-1      1       0       0       2       1       4

_CCA_TG_T_AA__
_CCA_TG_T_AA__
Local Alignment Score: 12
```

Figure 7: *Local Alignment*

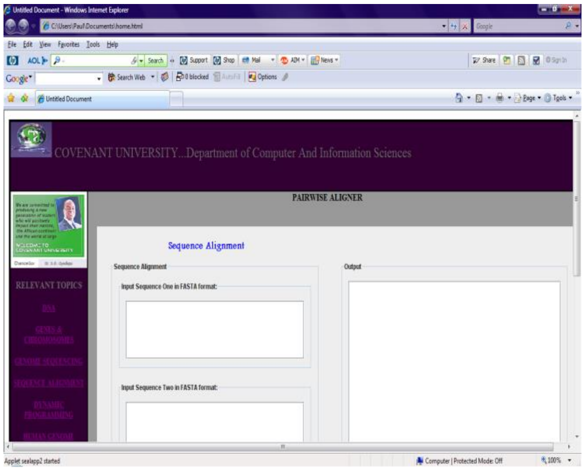


Figure 8: *Applet Deployed On Web Page*

A user, on accessing the pairwise aligner, enters in both sequences in the appropriate fields. Parameters are then selected so as to carry out the alignment as the user desires in terms of the match score, gap penalty and mismatch score.

On clicking the global alignment button, a check on inputted sequences is made to ensure compliance of the inputted sequences to DNA sequence structure (i.e. consisting alphabets A, C, G, and T). Once this check is cleared the two sequences are globally aligned.

In the case of local alignment, the same process takes place, except that the output is the local alignment of the two sequences.

7. CONCLUSION

Sequence alignment can be a useful technique for studying molecular evolution and analyzing sequence-structure relationships. We present a sequence alignment tool in an understandable, user-friendly and portable way, with click-of-a-button simplicity. This is done utilizing the Needleman-Wunsh and Smith-Waterman algorithms for global and local alignments, respectively which focuses primarily on DNA sequences. Our aligner is implemented in the Java language in both application and applet mode and has been tested working efficiently on all windows operating systems.

REFERENCES

[1] Flaninick J, Batzoglou S. Using multiple alignments to improve seeded local alignment algorithms. *Nicleic Acids Research*, 33(14), pp. 4563-4577, 2005.

[2]. K. Somasundaram and S. Radhakrishnan: “Nimble Protein Sequence Alignment in Grid (NPSAG)”, *Journal of Computer Science* 4 (1), pp. 36-41, 2008.

[3] Schwartz, S., Zhang, Z., Frazer, K.A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R.C., and Miller, W., “A Web server for aligning two genomic DNA sequences”. *Genome Res.* 10**,** pp. 577-586, 2000.

[4] Junier, T. and Bucher, P., “A Java applet for browsing molecular sequence data”. *In Silico Biol*. 1, pp. 13-20, 1998.

[5] T. Junier, *M. Pagni* and P. Bucher, mmsearch, “A motif arrangement language and the Eukaryotic Promoter Database (EPD)”. *Nucleic Acids Res*, 26, 353-357, 1998.

[6] P. Peterlongo, N. Pisanti, F. Boyer, A. P. do Lago, and M.-F. Sagot, “Lossless filter for multiple repetitions with Hamming distance”. *Journal of Discrete Algorithms*, 19, pp. 71-87, 2008.

[7] Smith T.F, Waterman M.S, “Identification of Common Molecular Subsequences”. *Journal of Molecular Biology,* 147, pp. 195–197, 1981.

[8] Needleman SB, Wunsch CD, “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. *Journal of Molecular Biology* 48 (3), 443-53, 1970.

[9] Brudno M, Malde S, Poliakov A, Do CB, Couronne O, Dubchak I, Batzoglou S*.* "Glocal alignment: finding rearrangements during alignment"*. Bioinformatics 19 Suppl 1: i54–6,(2003.*

[10] Schwartz S, Kent WJ, Smit A, Zhang Z, Baertsch R, Hardison RC, Haussler D, Miller W. Human-mouse alignments with BLASTZ. Genome Res. 2003;13:103–107

[11] Lassmann T, Sonnhammer EL. Quality assessment of multiple alignment programs. FEBS Lett. 2002;529:126–130.doi:10.1016/S0014-5793(02)03189

[12] Katoh K, Misawa K, Kuma K, Miyata T. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res. 2002;30:3059–3066

[13] Rice P, Longden I, Bleasby A. EMBOSS: the European Molecular Biology Open Software Suite. Trends Genet. 2000;16:276–27

[14] Bray N, Dubchak I, Pachter L. AVID: A Global Alignment Program. Genome Res. 2003;13:97–102

[15] Brudno M, Do CB, Cooper GM, Kim MF, Davydov E, Program NC, Green ED, Sidow A, Batzoglou S. LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNA. Genome Res

About the Authors:

Oyelade Jelili and Fatumo Segun: *Department of Computer and Information Sciences College of Science and Technology, Covenant University, Ota.*
Email: *Ola.oyelade@covenantuniversity.edu.ng, segunfatumo@yahoo.co.uk*