12-2005

Correlogram Method for Comparing Bio-Sequences

Gandhali P. Samant

Correlogram Method for Comparing Bio-Sequences

Gandhali P. Samant, and Debasis Mitra
dmitra@cs.fit.edu

Technical Report FIT-CS-2006-01
Content of a Master's Thesis
Submitted to
Florida Institute of Technology
In partial fulfillment of the requirements
For the degree of

Master of Science
in
Computer Science

Melbourne, Florida
December 2005

ABSTRACT

Title: Correlogram method for Comparing Bio-Sequences

Author: Gandhali P. Samant, and Debasis Mitra

Sequence comparison is one of the most primitive operations used in bio-informatics. It is used as a basis for many other complex manipulations in the field of Computational Molecular Biology. Many methods and algorithms were developed to compare and align sequences effectively. Most of these methods use linear comparison and some standard scoring schemes to calculate the similarity between sequences. We described an alternative approach to compare sequences based on the correlogram method. This method has already been used in the past for comparing images. By using the correlogram method, a sequence is projected on a 3-D space and the difference between two sequences is calculated. This research describes construction of a correlogram and how correlograms can be used for sequence comparison. It also adds additional functionality to the basic correlogram method. Experiments with protein sequences corresponding to different strains of influenza virus and parvovirus will be described and the phylogeny/evolutionary trees constructed using the correlogram method will be compared, with the ones available in literature.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I greatly appreciate the generous help and guidance provided by my advisor, Dr. Debasis Mitra throughout the development of this thesis. Dr. Mitra first introduced me to the field of Bioinformatics and without his constant mentoring this thesis would not have been a distinct possibility.

I also want to thank Dr. William Shoaff and Dr. Alan Leonard who graciously accepted to serve on my thesis committee. I would also like to thank Dr. Mavis Mackenna and her team at University of Florida, Gainesville for their help on providing various data used in this research.

Special thanks to my friend Mridula Anand for her help in running the experiments and analyzing the results, Sanchi Chandan for explaining the biological aspects of this research and to my fiancé Chirag, for his help throughout the development of this thesis. Also I would like to thank my friends Amit Paspunattu, Tejas Rao, Anjali Ram and Raghunath Vemuri for their constant support during the time I spent at FIT.

Last but not the least I thank my family for believing in me and constantly supporting me throughout both good and not so good times.

DEDICATION

To My Parents, Anjali and Pramod, My Brother Kedar and to my Fiancé Chirag, for their continued support and encouragement throughout my career.

Chapter 1

Introduction

1.1 Purpose of Study

Sequence Comparison is one of the most important functions in the field of Bioinformatics. Many algorithms have been proposed for comparing two sequences and finding distance or similarity between them. The purpose of this study was to use new methods for the comparison of biological sequences. This study resulted in the development of correlogram algorithms, which can be used for sequence comparison, for creation of a phylogeny tree and also for finding occurrences of a given pattern within the target sequence.

1.1.1 What is Sequence Comparison?

The basic problem of sequence comparison can be defined as 'Finding resemblance or similarity between given sequences.' The most commonly addressed problem for sequence comparison is 'String Matching.' String Matching can be defined as 'The problem of finding occurrence(s) of a pattern string within another string or body of text.'[i] In the context of bioinformatics, strings are referred to as sequences.

Two important notions related to sequence comparison include:

1. Similarity – It is the extent to which sequences are related. The extent of similarity between the two sequences can be based on percent sequence identity and/or conservation.[ii]
2. Alignment – It is the process of lining up two or more sequences to achieve maximal levels of identity or conservation for the purpose of assessing the degree of similarity and the possibility of homology.[iii]

1.1.2 Importance of Sequence Comparison in Biology

In bio-molecules such as DNA, RNA & proteins, sequence similarity usually implies functional or structural similarity.[iv] A requirement of life is transfer of information from one generation to another through these bio-molecular sequences. The molecular structures and functionalities show up repeatedly in the genome of single species and other divergent species. It is said that 'Duplication and modification' is the central paradigm of protein evolution. Therefore, redundancy is very common in bio-molecular sequences. The genomes of two entirely different species can be similar. For example, some genes that exist in fruit flies also exist in humans. It is obvious that there must also be important differences between the two genomes. Hence, sequence comparison plays a very important role in modern biology. Similarity in sequences can be used to study the evolution of species, to cluster different species which are structurally or functionally similar in a group or to find the phylogeny between organisms.

1.2 Scope of Study

The correlogram concept is used for image comparison and is described by Huang et. al.[v] A color correlogram expresses how the spatial correlation of pairs of colors changes with distance (the term "correlogram" is adapted from spatial data analysis). In this research, the concept of the correlogram was used to develop an algorithm to compare biological sequences. These algorithms were then used to compare synthetic sequences. This research was further extended to real protein sequences and results of this study were compared with the previous research.

1.3 Thesis Outline

In Chapter 2 background information is presented to explain the different biological information researched for this study. In Chapter 3 literature in the area of sequence comparison is reviewed including past research done in biological algorithms for finding sequence similarity, aligning sequences, database search and finding phylogenies. In Chapter 4 research aims are outlined. Research Methodology is covered in Chapter 5. Chapter 6 contains further discussion of correlogram algorithms and their extensions. Chapter 7 describes different experiments and their results using these algorithms. Conclusions and further research opportunities are found in Chapter 8.

Chapter 2

Background

2.1 Introduction

This chapter covers the basic concepts in molecular biology, and use of BLAST algorithm and clustering of sequences. In this research, various virus sequences or protein sequences e.g. Influenza virus, Parvovirus were used to test new algorithms and these viruses are described along with the PHYLIP software which was used for creation of phylogeny trees.

2.2 Basic Concepts of Molecular Biology

Life started evolving around 3.5 billion years ago, and the first form of life was very simple.[vi] Interactions between various substances and energy led to systems capable of passing information over different generations. Over time, the structure of living organisms became more and more complex, but simple and complex organisms have similar biochemistry. Important substances required for life are Proteins and Nucleic Acids.[vii]

Proteins

Proteins play important structural and enzymatic roles in cells. A protein is a chain of simple molecules called Amino Acids. Every Amino acid has one central carbon atom, known as alpha carbon or Cα. To the Cα atom are attached a hydrogen atom

(H), an amino group (NH2), a carboxy group (COOH), and a side chain. The side chain distinguishes one amino acid from another. In nature, 20 amino acids can be found which are listed in the following table.

1-letter code	3-letter code	Amino Acid
A	Ala	Alanine
R	Arg	Arginine
N	Asn	Asparagine
D	Asp	Aspartate
C	Cys	Cysteine
Q	Gln	Glutamine
E	Glu	Glutamate
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
L	Leu	Leucine
K	Lys	Lysine
M	Met	Methionine
F	Phe	Phenylalanine
P	Pro	Proline
S	Ser	Serine
T	Thr	Threonine
W	Trp	Tryptophan
Y	Tyr	Tyrosine
V	Val	Valine

Table 2-1: Amino Acids

Nucleic Acids

Nucleic Acids encode information necessary to produce proteins and are responsible for passing this recipe to subsequent generations. There are two types of nucleic acids present in living organisms, RNA (ribonucleic acid) and DNA (deoxyribonucleic acid).

Like proteins, DNA is also a chain of simpler molecules. It's a double chain made up of two strands. Each strand has a backbone consisting of repetitions of the same basic unit. This unit is formed by a sugar molecule called 2'-deoxyribose attached to a phosphate residue. The sugar molecule contains five carbon atoms, and they are labeled 1' through 5.' The bond that creates the backbone is between 3' carbon of one unit, the phosphate residue, and the 5' carbon of the next unit. Attached to each 1' carbon in the backbone are other molecules called bases. There are four different kinds of bases. These are:

1. Adenine (A)
2. Guanine (G)
3. Cytosine (C)
4. Thymine (T).

The basic unit of a DNA molecule consisting of the sugar, the phosphate, and its base is called a nucleotide. RNA molecules are much like DNA molecules except in RNA the sugar is ribose instead of 2'-deoxyribose and instead of Thymine (T), Uracil (U) is present.

2.3 Blast (Basic Local Alignment Search Tool)

BLAST algorithms are heuristic search methods that seeks words of length W (default=3 in blastP) that score at least T when aligned with the query and scored with the substitution matrix (e.g. PAM). Words in the database that score T or

greater are extended in both directions in an attempt to find a locally optimal un-gapped alignment or HSP (High Scoring Pair).

1. PAM Matrix (Point Accepted Mutation or Percentage of Accepted Mutation)

   In bioinformatics, scoring matrices for computing alignment scores are often based on observed substitution rates, derived from the substitution frequencies seen in multiple alignments of sequences. The need for these matrices arise because of the fact that amino acid residues have biochemical properties which influence their replace-abilities and also the fact that Amino acids of similar sizes get substituted with each other, hence, simple scoring methods are not sufficient. PAM matrices are ***functions of evolutionary distances.*** Basic 1-PAM matrix reflects an amount of evolution producing on average one mutation per 100 amino acids.

2. Working of BLAST

   BLAST finds certain "seeds", which are very short segment pairs between the query and the database sequence. These seeds are then extended in both directions, without including gaps, until the maximum possible score for extensions of this particular seed is reached. Not all extensions are looked at. The program has a criterion to stop extensions when the score falls below a carefully computed limit. There is a very small chance of the right

extension not being found due to this time optimization, but in practice this tradeoff is highly acceptable.

2.4 Clustering

Clustering can be defined as "The process of organizing objects into groups whose members are similar in some way."[viii] Clustering can be used in the field of biology for grouping animals and plants. In bioinformatics, sequence clustering algorithms attempt to group sequences that are somehow related. For proteins, homologous sequences are typically grouped into families. Generally, the clustering algorithms are single linkage clustering, constructing a transitive closure of sequences with a similarity over a particular threshold. The similarity score is often based on sequence alignment. Sequence clustering is often used to make a non-redundant set of representative sequences. [ix]

2.5 Influenza Virus

Influenza, also known as the flu, is a contagious disease that is caused by the influenza virus. It attacks the respiratory tract in humans (nose, throat, and lungs).[x] There are three main types of influenza virus. Influenza A, B, and C. Influenza types A or B viruses cause epidemics of disease almost every winter. In the United States, these winter influenza epidemics can cause illness in 10% to 20% of population and are associated with an average of 36,000 deaths and 114,000 hospitalizations per year. Getting a flu shot can prevent illness from types A and B

influenza. Influenza type C infections cause a mild respiratory illness and are not believed to cause epidemics. The flu shot does not protect against type C influenza. Influenza A viruses are found in many different animals, including ducks, chickens, pigs, whales, horses, and seals. Influenza B viruses circulate widely only among human beings. Influenza type A viruses are divided into subtypes based on two proteins on the surface of the virus. These proteins are called hemagglutinin (H) and neuraminidase (N). The current subtypes of human influenza A viruses are A(H1N1) and A(H3N2). Influenza B virus is not divided into subtypes. Influenza A(H1N1), A(H3N2), and influenza B strains are included in each year's influenza vaccine.

Influenza virus can change in two ways:

1. Drift - Antigenic drift is due to small changes in the virus that happen continually over time. New virus strains are produced that may not be recognized by the body's immune system. A person infected with a particular flu virus strain develops antibody against that virus. As newer virus strains appear, the antibodies against the older strains no longer recognize the "newer" virus, and reinfection can occur. In most years, one or two of the three virus strains in the influenza vaccine are updated to keep up with the changes in the circulating flu viruses.
2. Shift - Antigenic shift is an abrupt, major change in the influenza A viruses, resulting in new hemagglutinin and/or new hemagglutinin and

neuraminidase proteins in influenza viruses that infect humans. Shift results in a new influenza A subtype. When a shift happens, most people have little or no protection against the new virus. While influenza viruses are changing by antigenic drift all the time, antigenic shift happens only occasionally. Type A viruses undergo both kinds of changes; influenza type B viruses change only by the more gradual process of antigenic drift.

2.6 Parvovirus

Parvovirus, commonly called parvo, is a genus of the Parvoviridae family of DNA viruses. Parvoviruses are some of the smallest viruses found in nature. Like all members of the parvoviridae family, they infect only mammals. [xi] Parvoviruses can cause disease in some animals. For example, Canine parvovirus is a potentially deadly disease among young puppies, causing gastrointestinal tract damage and dehydration. Mouse parvovirus, on the other hand, causes no symptoms but can contaminate immunology experiments in biological research laboratories. The most accurate diagnosis of parvovirus is by ELISA (Enzyme-Linked Immunosorbent Assay). [xii] Dogs and cats can be vaccinated against parvovirus. Many types of mammalian species have a type of parvovirus associated with them. A parvovirus tends to be specific about the taxon[xiii] of animal it will infect. That is, a canine parvovirus will affect dogs, wolves, and foxes, but will not infect cats or humans.

Parvovirus B19, which causes fifth disease in humans,[xiv] is a member of the Erythrovirus of parvoviridae rather than parvovirus.

2.7 Phylogenetic Trees/Phylip

All species of organisms undergo a slow transformation process called evolution. Developing the relationship between different species and their common ancestors is one of the important problems in biology. This is usually done by constructing trees of which leaves are the present day species and interior nodes are the hypothesized ancestors. These trees are called Phylogenetic trees. These trees can be built for species, population or other taxonomical units. The construction of the trees is based on the comparisons between the present day species and the input data for comparisons can be classified into two main categories:

1. Character State Matrix - The character state matrix is a table in which the character states for a set of characters are compared across all taxa[xv] included in the analysis.[xvi]
2. Distance Matrix – Distance matrix is used to present the results of the calculation of a pair wise comparison score. The matrix field (i ,j) is the comparison score calculated between any two input sequences.

Phylip

PHYLIP, the Phylogeny Inference Package, is a package of programs for inferring phylogenies (evolutionary trees) from University of Washington.[xvii]

Phylip can infer phylogenies by parsimony, compatibility, distance matrix methods, and likelihood. It can also compute consensus trees, compute distances between trees, draw trees, resample data sets by bootstrapping or jackknifing, edit trees, and compute distance matrices. Phylip handles data that are nucleotide sequences, protein sequences, gene frequencies, restriction sites, restriction fragments, distances, discrete characters, and continuous characters.

Following are some important programs from Phylip[xviii] which were used in experiments based on phylogenies constructed from correlogram data.

FITCH - Estimates phylogenies from distance matrix data under the "additive tree model" according to which the distances are expected to equal the sums of branch lengths between the species. [xix]

KITSCH - Estimates phylogenies from distance matrix data under the "ultrametric" model which is the same as the additive tree model except that an evolutionary clock is assumed.

NEIGHBOR - An implementation of "Neighbor Joining Method," and of the UPGMA (Average Linkage clustering) method. Neighbor joining produces an unrooted tree without the assumption of a clock. UPGMA (Unweighted Pair Group Method with Arithmetic Mean) does assume a clock.

DRAWGRAM - Plots rooted phylogenies, cladograms, circular trees and phenograms in a wide variety of user-controllable formats. The program is interactive.[xx]

DRAWTREE – Similar to DRAWGRAM but plots unrooted phylogenies.

Chapter 3

Sequence Comparison

3.1 Some Existing Algorithms for Sequence Comparison

Many algorithms have been devised to compare sequences. These sequences can be commonly classified in the following three categories:

a. Dynamic programming algorithms for sequence alignment - Global, Local or Semi-Global Alignment
b. Heuristic and Database Search Algorithms - BLAST, FASTA
c. Multiple sequence alignment Algorithms.

The most widely used algorithms for sequence comparison are Dynamic programming algorithms and BLAST (Basic Local Alignment Search Tool).

3.1.1 Dynamic Programming Algorithms

Dynamic Programming algorithms are algorithms where an optimization problem is solved by saving the optimal solutions for every sub-problem instead of recalculating them. Different kinds of string alignments can be done with dynamic programming. Local alignment algorithms find out the conserved similarity between subsequences whereas Global alignment algorithms try to find the overall similarity between two sequences.

The following definitions are important to understand the concept of dynamic programming:

1. Similarity – This gives a numeric score of similarity between two sequences by means of some defined scoring method.

In DP algorithms, match receives +1 value whereas a mismatch receives -1 value.

A G T C T C

A T T G T C

---

1 -1 1 -1 1 1 = 2

2. Alignment – Way of placing one sequence above the other to make clear the correspondence between them.

A G T C G T C

A _ T C _ T C

The two sequences being compared may have different sizes. Hence, alignment of these sequences may contain spaces. These spaces are called gaps. In dynamic programming algorithms, these gaps are typically scored as -2.

1. Global Alignment

In this algorithm, the optimal alignment for every substring is computed and the scores are saved in a matrix. For two strings, s of length m and t of length n, D[i,j] is defined to be the best score of aligning the two substrings s[1..j] and t[1..i]. The best score is the last value in the table i.e. D[m,n]. It is computed by computing D[i,j] for all values of i and j where i ranges from 0 to m and j ranges from 0 to n. These scores, the D[i,j] are the optimal scores for aligning every substring, s[1..j] and t[1..i].

The formula is

```
D[i,j]  =  max{  D[i-1,j-1]  +  similarity
score([s[j],t[i]]),
                 D[i-1,j] + gap_score,
                 D[i,j-1] + gap_score}
```

To create the alignment, the path is reconstructed from position [m+1,n+1] to D[1,1] that led to the highest score.

2. Local Alignment

A local alignment between s and t is an alignment between a substring of s and a substring of t. The algorithm is same as the global alignment except the following difference in the formula.

```
D[i,j] = max{ D[i-1,j-1] + similarity score([s[j],t[i]]),
              D[i-1,j] + gap_score,
              D[i,j-1] + gap_score,
                    0 }
```

For any entry (i,j) there is always the alignment between the empty suffixes of [1..j] and t[1..i], which has score zero.

3. Semi global Alignment.

In a semiglobal alignment, the alignments are scored by ignoring some of the end spaces in the sequences.

3.1.2 BLAST (Basic Local Alignment Search Tool)

BLAST provides rapid searching for nucleotide and protein databases. It detects both local and global alignments hence regions of similarities embedded in unrelated proteins can also be found out by BLAST.[xxi] The BLAST algorithm was devised by Altschul et. al. in 1990.[xxii] The algorithm could be implemented in a number of ways in a variety of problems including DNA and protein sequence database searches, motif finding, gene identification etc. BLAST algorithm uses PAM (Point Accepted Mutation) matrix for scoring the match between sequences.

BLAST undertakes the following steps:

1. Compile list of high-scoring strings (or words, in BLAST jargon)
2. Search for hits - each hit gives a seed
3. Extend seeds.

For protein sequence, the list of high-scoring words consists of of all words with *w* characters (called *w*-mers) that score at least *T* with some *w*-mer of the query. The recommended value for *w*, the seed size, is 4 for protein searches.

K A L M R

V A K N S

-4 3 -4 -3 -1 –> Total:-9

Segment pair and its score under PAM120

Original BLAST algorithm was used to get the un-gapped alignment between sequences; however in 1997 a new generation of BLAST programs such as Gapped BLAST and PSI-BLAST (Position Specific Iterated BLAST) were introduced. Gapped BLAST uses PSSM (Position Specific Score Matrix) for scoring the alignment.[xxiii] The ability to generate gapped alignments was added in the Gapped BLAST version of the algorithm.

In PSI- BLAST, BLAST searches are iterated, with a position-specific score matrix generated from significant alignments found in round *i* used for round '*i* + 1'. Motif or profile search methods frequently are much more sensitive than pair wise comparison methods at detecting distant relationships.

3.1.3 FASTA

FASTA algorithm is another heuristic method for string comparison. It was developed by *Lipman* and *Pearson* in 1985 and further improved in 1988. [xxiv] The FASTA algorithm uses a fast search to initially identify sequences with high degree of similarity to the query sequence and then it does a secondary comparison on selected sequences. FASTA is slower than BLAST but sometimes it gives more sensitive results.

FASTA uses 3 steps[xxv] to calculate three scores to characterize sequence similarity:

1. Identify regions shared by the two sequences with the highest density of identities or pairs of identities. FASTA achieves much of its speed and selectivity in this step by using a lookup table to locate all identities or groups of identities between two DNA or amino acid sequences during the first step of the comparison.

2. Rescan the 10 regions with the highest density of identities using the PAM250 matrix. Trim the ends of the region to include only those residues contributing to the highest score. Each region is a partial alignment without gaps.

3. If there are several initial regions with scores greater than the cutoff value, check to see whether the trimmed initial regions can be joined to form an approximate alignment with gaps. Calculate a similarity score that is the sum of the joined initial regions minus a penalty (usually 20) for each gap. This initial similarity score is used to rank the library sequences. The score of the single best initial region found in step 2 is reported.

3.1.4 Multiple Sequence Alignment Algorithms

Multiple sequence alignment methods are mainly used when there is a need to extract information from a group of sequences. Examples of situations in which these techniques are used include the determination of secondary or tertiary structures, characterization of protein families, identification of similar regions etc.

[xxvi] The most commonly used methods are the hierarchical extension of pair wise sequence alignment methods. A multiple alignment with a group of sequences is obtained by inserting gaps in the sequences making them all of the same size. One important thing in multiple alignments is the quality of the alignment and for that a scoring function like SP Measure [xxvii] can be used. One more way of achieving multiple alignment is placing the sequences in a tree structure rather than placing them vertically in the pile. Examples of scoring functions used for this way of multiple alignments are star alignment and tree alignment.

Along with these traditional methods, there have been consistent efforts for finding different functions for scoring multiple alignments.

3.1.5 Miscellaneous Techniques

In addition to the techniques described above, new methods have been introduced. For example, Contact-based sequence alignment is a method which uses Contact accepted mutation matrices. [xxviii] CAO scores are said to reflect evolutionary relatedness better than PAM score. Also in 1990, there was an attempt to use a visualization technique called Correlation Images (CI). A CI is produced by comparing two sequences in all possible alignments with each image row corresponding to a distinct alignment. [xxix] CI was also used to compare three sequences in which each sequence was assigned to one axis so that the sequences define a three dimensional matrix which will contain every element of a sequence

compared to every element of the other two sequences. [xxx] Methods are also being proposed that do not include the fundamental tool of sequence alignment. For e.g. Haubold et. al. propose a new technique of genome comparison without sequence alignment. This procedure is based on the shortest unique substring. A shortest unique substring is a substring that occurs only once within the analyzed sequence or set of sequences and which cannot be further reduced in length without losing the property of uniqueness. Such substrings can be detected using generalized suffix trees. [xxxi]

3.2 Concept of Correlogram

3.2.1 What is a Correlogram?

The correlogram can be defined as a measure of spatial dependence (correlation) of a regionalized variable over some distance. [xxxii] The concept of correlograms have been widely used in many fields such as statistical analysis, computer graphics, environmental sciences, signal processing etc. The 'Correlogram concept for image comparison' described by Huang et. al. [xxxiii] provides the basis for using correlograms for sequence comparison.

3.2.2 Color Correlogram

A color correlogram expresses how the spatial correlation of pairs of colors changes with the distance (the term "correlogram" is adapted from spatial data analysis). The color correlogram is neither an image partitioning method nor a

histogram refinement method. Unlike purely local properties, such as pixel position, gradient direction, or purely global properties, such as color distribution, correlograms take into account the local color spatial correlation as well as the global distribution of this spatial correlation.

This method rectifies the major drawbacks of histogram methods. However, sequence correlogram is much simpler than color correlogram as at a time it takes into account only two sequences and finds the correlation between them.

Color correlogram has also been used recently for object tracking.[xxxiv] In this method a simplified version of color correlogram is used as object feature. Spatial information is incorporated into object representation, which allows variations of rotation to be detected throughout the tracking therefore rotational objects can be more accurately tracked.

3.3 Correlogram Usage in the Field of Bioinformatics

The correlogram can be defined as a graph showing correlation of one biological sequence with the other. The concept of the correlogram has been already used in the field of bioinformatics in numerous ways. In 1985, Macchiato et. al. used correlograms to analyze autocorrelation characteristics of active polypeptides.[xxxv] Further, correlograms have been widely used for analyzing spatial patterns in various experiments. E.g. Bertorelle and Barbujani used correlograms to study DNA Diversity.[xxxvi] Rosenberg Subramanium and Kumar used correlograms in

their studies regarding patterns of transitional mutation biases within and among mammalian genomes. [xxxvii]

In the research under study, correlogram were constructed as a 3-D matrix of which 2 dimensions are the set of entities (e.g. Amino Acids, Nucleic Acids) and the third dimension is distance. An individual correlogram for each sequence was constructed and then these correlograms were compared to measure distance between 2 sequences at a time. The correlogram computed distances were then used to create distance matrices and were also used to find repeating patterns in a long sequence.

Chapter 4

Research Questions

4.1 Research Questions

In this work, three research questions were examined related to sequence comparison algorithms[xxxviii] and the use of correlogram for it. These questions were used to determine the usefulness of the correlogram method and its advantages over other method and to find how this new method can be used to extract more information about sequences.

4.1.1 What are the Advantages of the Correlogram Technique Over the Previous Techniques?

Correlograms are a three dimensional structures which hold the sequence information projected in a 3-D structure. I examined whether the correlogram is a better method of comparing sequences than comparing them as linear structures.

4.1.2 Can the Correlogram Method be Modified to Take into Account the Biological Gaps Between Sequences?

Gaps in sequences are results of mutations (changes in DNA) that occur during evolution. In pair wise sequence comparison, gap penalty is assigned to gaps. I examined a modification to the correlogram method to accommodate the gaps that might have occurred in the sequences.

4.1.3 Can the Correlogram Method be Used to Find the Occurrence of a Given Pattern Over a Long Sequence?

Nucleotide and amino acid sequences contain patterns or motifs that have been preserved through evolution because they are important for the structure and function of the molecule. [xxxix] Discovery of these motifs is one of the important roles of bioinformatics. I examined whether the correlogram based sequence comparison algorithm can be extended to find these motifs or patterns in biological sequences.

Chapter 5

Research Methodology

5.1 Tools and Software used in this research

Tool	Application
Java 1.4.2	To code the algorithms
Java-api Excelreader	To generate the output in MS-Excel format
MS Excel	To create and analyze data and to create graphs.
Editplus	Editor for Java
Phylip	To Create phylogeny trees.

Table 5-1: Tools and Softwares

5.2 Chronological Order

This research follows a chronological order of the steps as follows:

Step 1: Development of the basic algorithm for sequence comparison using the correlogram method. - The first step in this research consisted of developing an algorithm to compare two input sequences using correlogram and finding the scalar distance between them. The algorithm was written in Java. This first algorithm is referred to as the *Basic Correlogram*.

Step 2: Compare the distances found using correlogram method with those found using BLAST and local alignment methods to determine the advantages and disadvantages of each method.

Step 3: Extension of the basic algorithm to include the gap penalty in the sequences - The algorithm was further modified to consider the potential gaps between sequences. This algorithm is referred to as the *Delta Correlogram*.

Step 4: Using the algorithms to find similarities between synthetically generated sequences - In this step, synthetic sequences were created. The results of these comparisons were recorded and studied to find patterns.

Step 5: Influenza virus data and parvovirus data from previous experiments were used to compute distances between real protein sequences. The distance matrices were computed using correlogram algorithms. Using these matrices, phylogeny trees were created and then those trees were compared to the phylogenies computed by other methods.

Step 6: Extension of the algorithm to scan for recurring patterns (motifs) over long sequences. - This algorithm was developed to research the third research question whether correlogram can be used to find the occurrence of a given pattern over a long sequence. The algorithm finds the locations of recurring patterns over a long sequence. This algorithm was referred to as the *Scan Correlogram*.

Step 7: The scanning algorithm was used to find the motifs in the long sequences of Parvovirus data.

All the three correlogram algorithms are explained in the next chapter.

Chapter 6

Correlogram Algorithms

6.1 Construction of Correlogram

Correlogram Definition and Construction

A correlogram is a graph showing correlation of one sequence with another. It is a 3-D matrix of which 2 dimensions are the set of entities (e.g. Amino Acids, Nucleic Acids) and the third dimension is distance.

Correlogram (x, y, d) is a function of frequency of x & y appearing at distance d in the sequence where x, y € ∑, 0<= d< =C, where ∑ is a set of characters and C is constant.

For example if the sequence for which correlogram is being constructed is an amino acid sequence then the two entity dependent dimensions will be 20 X 20 (for 20 amino acids in total). On the other hand if the correlogram is being constructed for DNA sequences these 2 dimensions will be 4 X 4 (The reason is that there are only four deoxyribo nucleotide).

Each correlogram plane has a fixed distance ‘d’ associated with it. To construct a correlogram, every character ‘x’ in the sequence will be compared with the character ‘y’ at distance ‘d’ from it and the cell corresponding to ‘x’ row and ‘y’ column will be incremented by count 1. The base plane of a correlogram is constructed for distance 0. At distance 0 every character in the sequence is compared with itself.

Similarly for distance 1 every character is compared with the character at distance 1 from it and accordingly the cell from correlogram plane corresponding to those 2 characters will be incremented by count 1. So as we can see that a correlogram is a set of different correlogram planes measured at different distances.

Consider a sequence **agcttagtct.**

If we calculate the appearance of every pair of characters at distance 1, the first pair is **a** and **g,** so the cell at the intersection of row A and column G will get incremented by 1. (Order of characters is important as pair ag and pair ga will increment different cells.) The Correlogram Plane for distance 1 for sequence **agcttagtct** will be as shown in Figure 6-1.

	A	T	G	C
A			**2**	
T	**1**	**1**		**1**
G		**1**		**1**
C		**2**		

Figure 6-1 : Correlogram plane for distance 1

A correlogram can be constructed as a set of frequencies for different distances such as 1, 2, 3, 4...etc. Any plane in a Correlogram is by definition a histogram. It can also be said that Correlogram is a set of histograms at different distances.

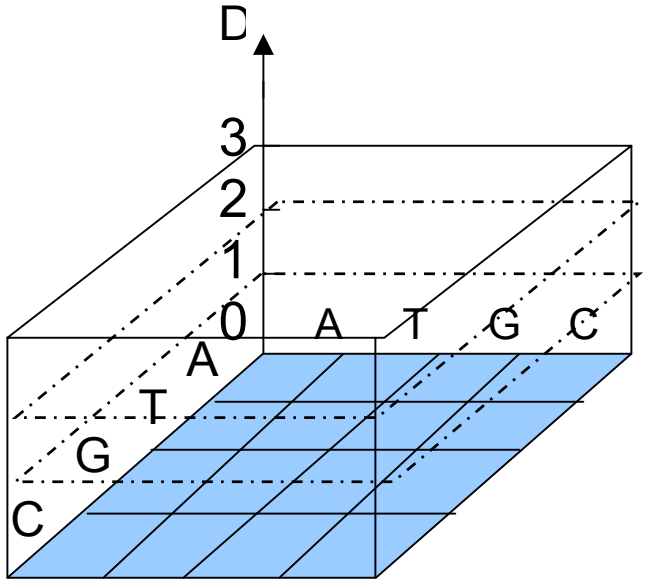


Figure 6-2: 3-D structure of correlogram for a nucleic acid sequence

The example shown above is the representation of nucleic acid sequence. Similarly if we want to construct correlogram for amino acid sequences, every plane will have 20 rows and 20 columns (There are a total of 20 amino acids).

6.2 Algorithm 1 - Basic Correlogram

Purpose – This algorithm has been designed to compare two sequences using the concept of correlogram. The first part of this algorithm is construction of the correlogram. The correlograms are then compared to find the corresponding distances between the corresponding sequences.

Introduction - This method calculates the difference between two sequences. A set of different distances is used to create a correlogram. A correlogram plane is constructed for each of these distances.

Steps -

1. Creation and Normalization of one Correlogram plane.
    a) Input – Sequence *s* and distance *d*
    b) Output – One Normalized Correlogram plane
    c) Process – Given sequence is scanned for all the pairs of amino acids at the given distance. A two dimensional array of size 20X20 (Total no. of amino acids) is created. For every pair of amino acids at the given distance the array cell is incremented by a count. Thus the sequence is scanned for its entire length.

After creating the Correlogram the total volume is calculated by adding all the elements of it. And then each element is divided by the volume to normalize it.

Normalization is needed for the sequence length parameter

E.g. Let us assume that while comparing 2 similar sequences of length 100, we got a score of S1. Similarly while comparing 2 sequences of length 20 we obtain a score of S2. The first correlogram (Score S1) will have a higher number of values as compared to the second correlogram (score S2). Even if the score S1 is greater than S2, it does not mean greater similarity. To make sure that we normalize the scores, we divide every value in the array by the total number of occupied cells. This ensures normalization with respect to the length of the sequence.

Algorithm *createCorrelogramPlane*

Input - sequence *s*, distance *d*

Output – array *a*

Variables

l = |s|;

volume y /*v is the volume of one correlogram plane*/

/* Initialization of Correlogram – *The array for creating one correlogram plane is initialized* */

for i = 0 to 20 (4 in case of nucleic acids sequence)

for j = 0 to 20 (4 in case of nucleic acids sequence)

a[i][j] = 0;

/* Creation of Correlogram *plane for a distance d* */

for i = 0 to l -1

if (i+d < l)

a1 = s[i];

a2 = s[i+d];

i1 = acidSeq[a1];

i2 = acidSeq[a2];

a[i1][i2] = a[i1][i2]+1 ;

v++;

/* Normalization of Correlogram - *Every value in the correlogram plane is normalized by dividing by the total volume of the plane* */

for i = 0 to 20 (4 in case of nucleic acids sequence)

for j = 0 to 20 (4 in case of nucleic acids sequence)

if (a[i][j] != 0.0)

a[i][j] = a[i][j]/v;

return a;

2. Creation of Correlogram Vector

   a) Input – Sequence *s* and array of distances *dist* denoted by dist[].

   b) Output – Normalized Vector *ve*

   c) Process – A new vector is created to hold the data sequences. This vector is used to store the data as it is a dynamic data structure and it grows whenever a new object is added. A method to create Correlogram plane is called to create Correlogram planes for different distances and they are added to a vector one by one.

   Algorithm *createCorrelogramVector*

   Input - sequence s, array dist

   Output - Vector ve

   Variables -

   Array t

   /**A vector is created to store all the correlogram planes for different distance values**/

for i = 0 to number of values in distances

t= createCorrelogramPlane (sequence s, dist[i]);

ve.add(t);

return ve;

3. Comparison of two vectors to score actual difference between them.

a) Input – two vectors.

b) Output – difference *diff* between them.

c) Process – given two vectors are scanned at each position the difference between their elements is calculated by using the following formula.

$x = ((f1-f2)^2 / 1+f1+f2)$

All the x values are added and the square root of the total is the final difference between 2 sequences.

$diff = \sqrt{\sum x}$

Algorithm *compareCorrelograms*

Input - Vector v1, Vector v2

Output - double diff

Variables -

Array c1, c2

l = size of v1 = size of v2

/* *Two correlogram for 2 different sequences are compared* */

```
for i = 0 to l
        c1 = v1.elementAt(i);
        c2 = v2.elementAt(i);
        for j = 0 to 20 (4 in case of nucleic acids sequence)
                for k = 0 to 20 (4 in case of nucleic acids sequence)
                t = (Square(c1[j][k] - c2[j][k]))/ (1 + c1[j][k] + c2[j][k]);
                        diff += t;
        diff  = sqrt(diff);
        return diff
```

Complexity of Basic Correlogram algorithm is linear i.e. O(n) where n is the length of the sequence. Hence basic correlogram algorithm is faster the Smith Waterman Dynamic programming algorithm as DP algorithm has complexity O(mn) where m and n are the lengths of sequences being compared.

6.3 Algorithm 2 - Delta Correlogram

Purpose – This algorithm has been designed to compare two sequences using the concept of a gapped correlogram. This technique was developed for two sequences which align against each other with gaps involved.

For example consider following two sequences:

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

A T G T C T A G A G A T T.

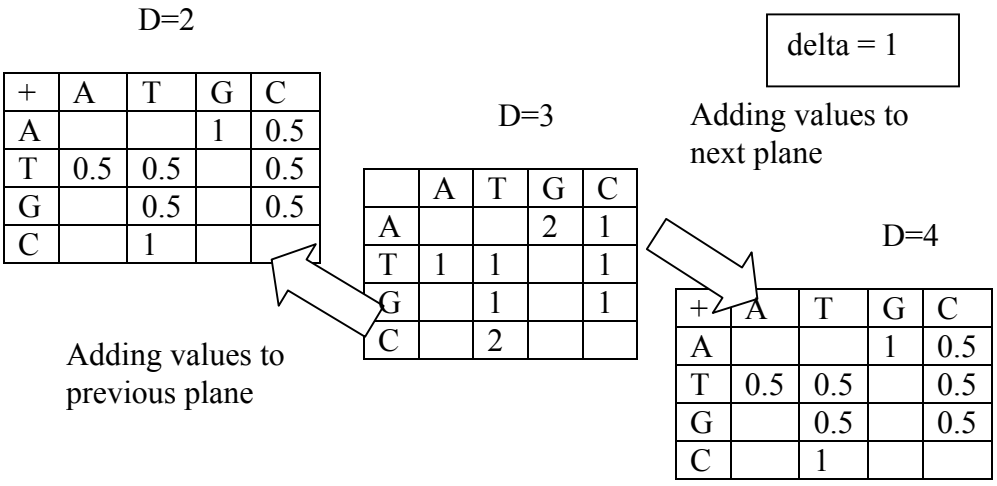
If we align the sequences as they are, they don't look very similar, but if we insert gap in the sequences as follows:

A __G T C T __G T A G A G T T

A T G T C T A G __A G A __T T.

Now their similarity is clearly seen. We used this concept of gap insertion for the development of the delta correlogram algorithm. The concept we used in the correlogram construction is that if two amino acids (or nucleic acids) are at a distance 'x' in one sequence there is a possibility that they can be at distance x-1 or x+1 in the other sequence with which it is being compared. Sometimes more than 1 gap needs to be inserted in order to align the sequences so there can be possibility that those two amino acids are at distance x-2 or x+2. The probability was used as half of the prior distance.

In the figure 6.5 the correlogram plane for D=3 has some values shown in it. So half of those values will be added to the correlogram plane at D= 4 and D= 2 and ¼ of those values will be added to the correlogram plane at D= 5 and D= 1.



+	A	T	G	C
A			1	0.5
T	0.5	0.5		0.5
G		0.5		0.5
C		1		

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

+	A	T	G	C
A			1	0.5
T	0.5	0.5		0.5
G		0.5		0.5
C		1		

Figure 6-3: Adding delta values to correlogram planes

The important part of this algorithm is the construction of the correlogram. The gap score is calculated while constructing the correlogram. After constructing one correlogram for each sequence, they are compared to find the difference between them. This algorithm is an extension to basic correlogram method to consider the gapped alignment of sequences.

Introduction - This algorithm takes care of gaps between the sequences, and this is done at the time of creation of correlogram vector. The reason is if a pair of character is at distance d, there is probability that in other sequence it might appear at distance d-1 or d+1.

For every pair at distance *n*, frequency *f* and with *delta = 1*, a fraction of frequency *f/2* is added at distance *n-1* and distance *n+1*.

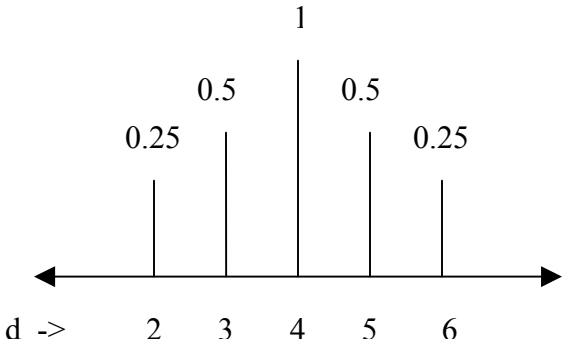


Figure 6-4: Delta value for different distances

Steps

1. Creation of Correlogram plane

    a) Input – Sequence *s* and distance *d*

    b) Output – One Correlogram plane

    Process – This method is same as creation of one correlogram plane for Basic correlogram. The only difference is the correlogram plane is not normalized.

    Algorithm createDeltaCorrelogramPlane

    Input - sequence s, distance d

    Output – array a

    Variables -

```
l = |s|;
/* Initialization of Correlogram – The array for creating one correlogram plane is initialized*/
for i = 0 to 20 (4 in case of nucleic acids sequence)
        for j = 0 to 20 (4 in case of nucleic acids sequence)
                a[i][j] = 0;
/* Creation of Correlogram plane for a distance d */
for i = 0 to l -1
        if (i+d < l)
                a1      = s[i];
                a2      = s[i+d];
                i1      = acidSeq[a1];
                i2      = acidSeq[a2];
                a[i1][i2] = a[i1][i2]+1 ;
return a;
```

2. Creation and Normalization of Correlogram Vector
    a) Input – Sequence, array of distances and delta.
    b) Output – Normalized Vector
    c) Process – A new vector is created to hold the data sequences. Vector is used to store the data as it is a dynamic data structure and it grows

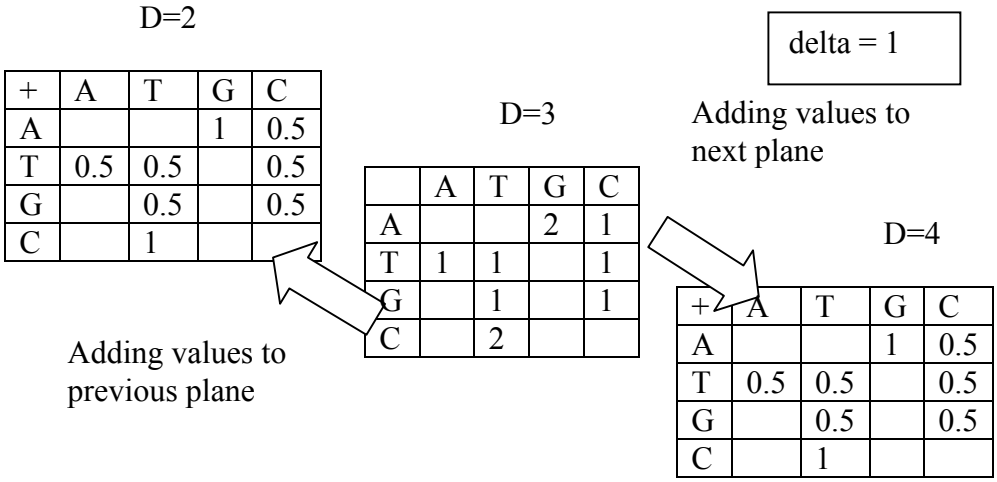
whenever a new object is added. A method *createCorrelogram* is called to create Correlogram of different distances and they are added to a vector one by one.

After creation, the vector is updated once again. For each Correlogram plane, previous and next Correlogram planes are updated. The number of correlograms being updated depends on the value of delta.

For e.g. - If delta is 2 for every Correlogram plane previous two and next two correlogram planes (if they exist) are updated.

This updating is done as follows.

In following figure delta = 1, so for every plane in correlogram, one plane in each direction (previous and next) will be updated for delta. If the plane for D = 4 has values shown in the figure, the plane for D=3 and D=5 will be updated as shown in the figure. The values shown in the figure will be added to the original values existing for that plane.



+	A	T	G	C
A			1	0.5
T	0.5	0.5		0.5
G		0.5		0.5
C		1		

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

+	A	T	G	C
A			1	0.5
T	0.5	0.5		0.5
G		0.5		0.5
C		1		

Figure 6-5: Adding delta values to correlogram planes

Normalization in basic Correlogram method is done after each correlogram plane is created. In this algorithm volume of the correlogram plane changes after addition of delta values to each plane and delta values can be added only after all the planes (i.e. the whole correlogram vector) are created.

Algorithm *createDeltaCorrelogramVector*

Input – Sequence s, Array dist, int delta)

Output – Vector *ve*

Variables -

Vector ve, cVe

```
Array temp, cTemp
Volume v          /* v is the volume of one correlogram plane */
for i=0 to dist.length
        temp =createDeltaCorrelogramPlane (s,dist[i]);
        ve.add(temp);
        cTemp =createDeltaCorrelogramPlane (s,dist[i]);
        cVe.add(temp);
        /* The delta values are added to all the correlogram planes
*/
        for i= to ve.size
                currentVe = cVe.elementAt(i);
                for j = 0 to 20 (4 in case of nucleic acids sequence)
                        for k = 0 to 20 (4 in case of nucleic acids
sequence)
                                w = currentVe[j][k];
                                if (i >= delta)
                                        for p=0  to delta
                                                dMin= ve[i-p-1]
                                                ve.remove(i-p-1);
                                                dMin[j][k] + =
(w/(2*(p+1)));
```

```
                                        ve.add(i-p-1,dMin);
                                    if (i+delta < ve.size())
                                        for p= 0 to delta
                                            dPlu = ve[i+p+1]
                                            ve.remove(i+p+1);
                                            dPlu[j][k] +=
(w/(2*(p+1)));
                                            ve.add(i+p+1,dPlu)

        /* Normalization of Correlogram- Every value in the
        correlogram plane is normalized by dividing by the total
        volume of the plane */
        for k = 0 to ve.size
            tempArr = ve[k]
            for i= 0 to 20 (4 in case of nucleic acids)
                for (int j0 to 20 (4 in case of nucleic acids)
                    if tempArr [i][j] <> 0
                        v+=tempArr[i][j];
            for i= 0 to 20 (4 in case of nucleic acids)
                for (int j0 to 20 (4 in case of nucleic acids)
                    if tempArr [i][j] <> 0
```

tempArr [i][j] =

tempArr[i][j] /v;

return ve;

3. Comparison of two vectors to score actual distance between them.
    a. Input – two vectors.
    b. Output – difference *diff* between them.
    c. Process – The same method c*ompareCorrelograms* as for Basic Correlogram algorithm is used here to calculate the total difference *diff* between the two sequences.

6.4 Algorithm 3 - Scan Correlogram

Purpose – This algorithm has been designed to find the occurrences of a given pattern over a long sequence.

Introduction – This method is an extension to basic correlogram method to find the given pattern in the target sequence using correlograms. First a correlogram is created for pattern sequences. If the length of pattern sequence is *l* then from target sequence a correlogram is created using only first *l* characters. The distance between these two correlograms is calculated and recorded. Next, all the pair scores for first character in target sequence are deleted from all the correlogram planes and

all the pair scores for the l +1 th character are added to all the correlogram planes. And again the distance between this updated correlogram and the correlogram for pattern sequence is calculated and recorded. Thus the whole sequence is scanned till the end of the target sequence is reached. After recording all the distances, a graph is plotted using MS excel visualizing all the distances. The best match is found where the score is minimum.

Steps:

1. Creation of Correlogram for pattern
    a) Input – pattern, distance matrix
    b) Output – Correlogram vector for pattern sequence
    c) Process – Correlogram vector for pattern sequence is created using the algorithms to create basic correlogram.
2. Creation Correlogram for target sequence
    a) Input – target sequence, distance matrix.
    b) Output – Correlogram vector for pattern sequence
    c) Process – correlogram for pattern sequence is created using the algorithms to create basic correlogram. Initially the correlogram is created only for first l characters where l is the length of pattern sequence.
3. Comparison of two correlogram vectors and scanning

a) Input – two correlogram vectors, target sequence and distance matrix.

b) Output – location *loc* of each substring and difference score *diff*.

   Process – The algorithm c*ompareCorrelograms* is used to get the difference between two given correlogram vectors. The difference and the location are recorded. (E.g. for first pair of vectors the location is 0.). Then the vector is de-normalized by multiplying each plane by its original volume. All the scores associated with the first character in the target sequence are deleted and the scores associated with the *l*+1 st character in the target sequence are added. The newly updated correlogram is then normalized and compared with the correlogram for pattern sequence. Again the distance and location are recorded. This process is repeated till the end of the sequence is reached.

   E.g.

   Pattern – atcgt

   Target – atcgattgctag

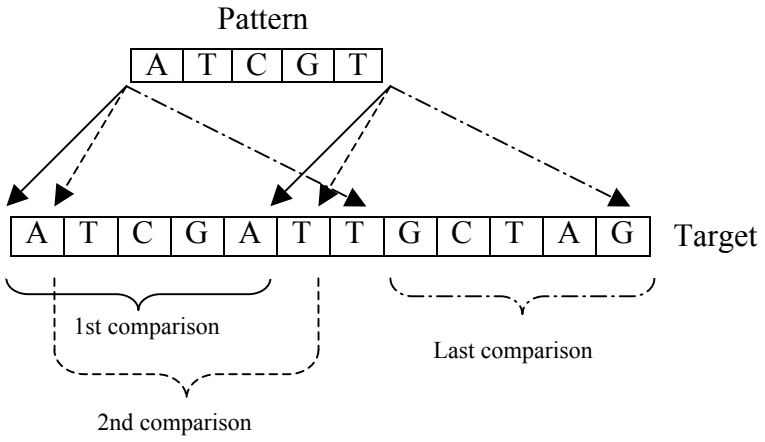


Figure 6-6: Scanning of sequence

Algorithm *scanSequence*

Input – Target squence s, Pattern sequence p, Array dist)

Variables -

m = |s|;

n = |p|;

char aIn,aOut,aNext,aPrev

int iOut, iIn, iNext, iPrev

String tempS = s[0,n]

Array diffValues [m-n+1]

```
array vol []
for i = 0 to vol.length
        vol[i] = n - dist[i];
/* pattern sequence patterns is searched for the possible matches in
the target sequence targets */
Vector patternS =  createCorrelogramVector (p, dist)
Vector targetS =  createCorrelogramVector (tempS, dist)
distValues[0] = compareCorrelograms (patternS, targetS);
for i=0 to (m-n)
        aOut  = s[i]
        aIn  = s[i+n]
        iOut = acidSeq.indexOf(acidOut);
        iIn  = acidSeq.indexOf(acidIn);
        for j = 0 to targetS.size()
                tempC[][] = targetS.elementAt(j);
                targetS.remove(j);
                int z  = dist[j];
                aNext = s[i+z]
                aPrev = s[i+n-z]
                iNext = acidSeq.indexOf(aNext);
                iPrev = acidSeq.indexOf(aPrev);
```

```
                    tempC [iOut][iNext] = tempC [iOut] [iNext] - (1/vol
[j]);
                    tempC [iPrev][iIn] = tempC [iPrev][iIn] + (1/vol[j]);
                    targetS.add(j,tempC);
                distValues[i+1] = compareCorrelograms(targetS,patternS);
            return distValues;
```

Complexity of Scan correlogram algorithm is linear. BLAST is one of the standard algorithms for scanning. The worst case complexity is same for both BLAST and Scan Correlogram algorithm. However, BLAST does the simple scanning whereas Scan correlogram algorithm goes into much more detail.

All the three algorithms are used in the various experiments performed for comparison of protein and virus sequences. These experiments are explained in the next chapter.

Chapter 7

Experiments & Results

7.1 Experiment 1 – Experiments with Synthetic Data

Purpose – The purpose of the synthetic experiments was to compare the capability of the correlogram method with one of the "traditional" sequence comparison techniques i.e. Smith-Waterman Dynamic Programming algorithms. The reason for using DP algorithms was that they are the standard method for sequence comparison. Additionally the sequences used in this experiment were changed (For example, wrapping around, deleting a specific character, adding a specific character etc.) and the changes in the scores were observed for both the methods. The sequences used in these experiments were nuclear acid sequences (i.e. they consist of only 4 characters A, T, G and C).

Steps

1. First of all measure the difference between reference sequence S1 and the target sequence S2. Let the difference be “Score1”. Now, reverse the target sequence S2 (Inv (S2)) and measure the difference between S1 and Inv (S2). Let the difference be “Score2”. Now choose S2 as the reference sequence and compare it with Inv (S2). Let the difference be “Score3”. Finally choose S1 as the reference sequence again and compare it with Inv

(S1). Let the difference be "Score4". The correlogram scores and DP scores for all the above combinations are obtained as shown below.

Sequence	Reversed Sequence	Correlogram Score	DP Score
S1:ATTCGGAGTT	**S2:**GTAGATGCTC	4.05	-0.2
S1:ATTCGGAGTT	**Inv(S2):**CTCGTAGATG	3.51	-1
S2:GTAGATGCTC	**Inv(S2):**CTCGTAGATG	1.92	-0.5
S1:ATTCGGAGTT	**Inv(S1):**TTGAGGCTTA	3.36	-0.5

Table 7-1: DP score and correlogram score values using reversed sequence

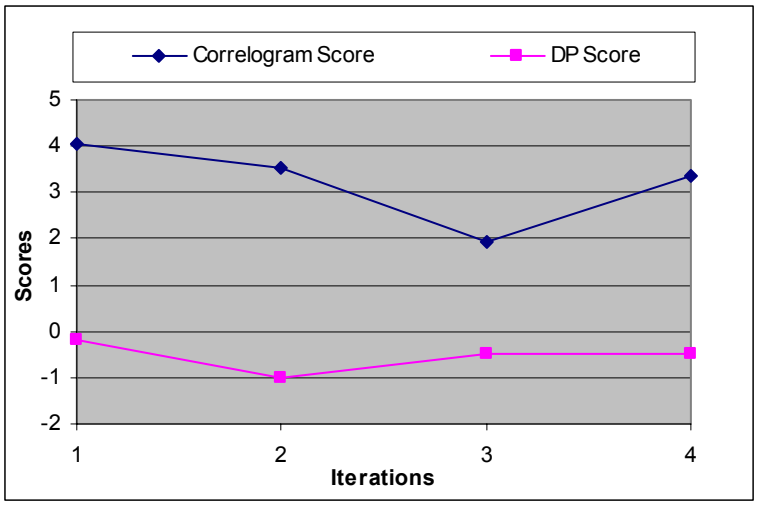


Figure 7-1: Comparison of DP score and correlogram score using reversed sequence

2. Wrap around the target sequence at different character length and measure difference with respect to the reference sequence each time. In the example shown below, sequence S1 GTAGATGCTC was chosen as the reference sequence. By wrapping, it means that the last character "C" of the sequence

is brought to the very start of the sequence to form a wrapped sequence "CGTAGATGCT" and the reference sequence is then compared with this wrapped sequence. For the second iteration, the character "T" (last character of the wrapped sequence) is brought to the first position and is compared with the reference sequence. This is done until all the characters have been wrapped and the wrapped sequence is the same as the original sequence as is shown below.

Sequence	Wrapped Sequence	Correlogram Score	DP Score
GTAGATGCTC	CGTAGATGCT	0.94	0.2
GTAGATGCTC	TCGTAGATGC	1.4	0
GTAGATGCTC	CTCGTAGATG	1.92	-0.5
GTAGATGCTC	GCTCGTAGAT	2.2	-0.333
GTAGATGCTC	TGCTCGTAGA	2.4	-0.125
GTAGATGCTC	ATGCTCGTAG	2.4	-0.2
GTAGATGCTC	GATGCTCGTA	1.94	-0.5
GTAGATGCTC	AGATGCTCGT	1.42	0
GTAGATGCTC	TAGATGCTCG	0.9	0.2
GTAGATGCTC	GTAGATGCTC	0	0.1

Table 7-2: DP score and correlogram score values using wrapped sequence

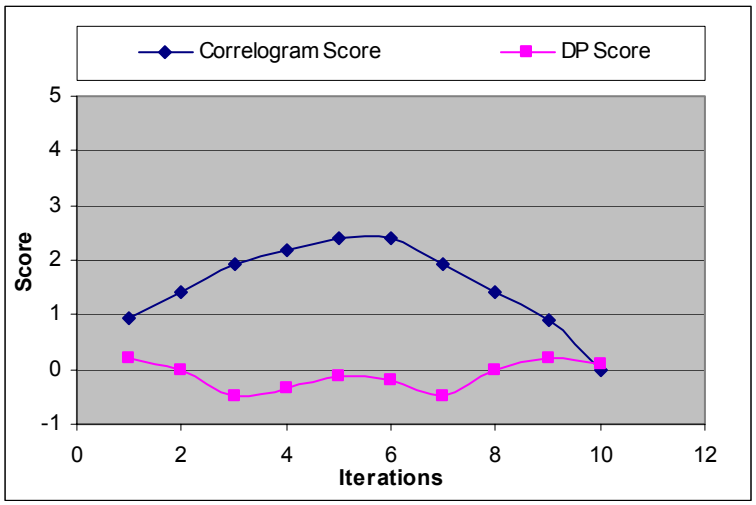


Figure 7-2: Comparison of DP score and correlogram score using wrapped sequence

3. Delete a character from target sequence and measure the difference.- In this experiment first sequence (S1) ATTCGGAGTT was compared with second sequence (S2) GTAGATGCTC, but for every comparison one character from S2 was deleted from different positions. For example in the first iteration, the first character "G" is deleted from the target sequence S2 (GTAGATGCTC) and is compared with the reference sequence. Similarly in the second iteration, the character "T" is deleted from S2 and compared with the reference sequence till all the characters in the target sequence have been exhausted.

Sequence S1	Delete a character from S2	S2 after deletion	Correlogram Score	DP Score
ATTCGGAGTT	~~G~~TAGATGCTC	TAGATGCTC	4.49	-0.1667
ATTCGGAGTT	G~~T~~AGATGCTC	GAGATGCTC	4.18	-0.1429
ATTCGGAGTT	GTA~~A~~GATGCTC	GTGATGCTC	4.37	-0.2
ATTCGGAGTT	GTA~~G~~ATGCTC	GTAATGCTC	4.59	-0.2
ATTCGGAGTT	GTAG~~A~~TGCTC	GTAGTGCTC	3.9	-0.2
ATTCGGAGTT	GTAGA~~T~~GCTC	GTAGAGCTC	4.14	-0.25
ATTCGGAGTT	GTAGAT~~G~~CTC	GTAGATCTC	4.41	-0.2
ATTCGGAGTT	GTAGATG~~C~~TC	GTAGATGTC	4.03	-0.3333
ATTCGGAGTT	GTAGATGC~~T~~C	GTAGATGCC	4.3	-0.2
ATTCGGAGTT	GTAGATGCT~~C~~	GTAGATGCT	4.13	-0.3333

Table 7-3: DP score and correlogram score values using deletion of a character from sequence

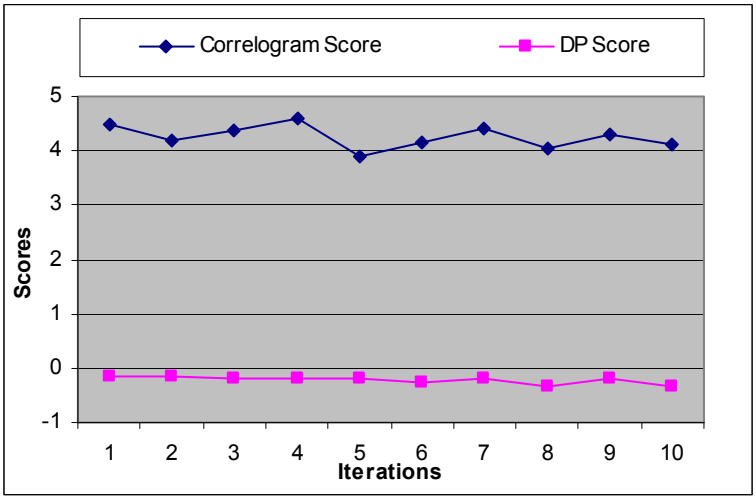


Figure 7-3: Comparison of DP score and correlogram score using deletion of a character from sequence

4. Then replace a character and measure the difference – In this experiment, first sequence (S1) ATTCGGAGTT was first compared with second sequence (S2) GTAGATGCTC and then first character for S2 was replaced by all

three other characters one at a time and S1 was compared with S2. Same steps were repeated for characters at 5th position and 10th position. For example in the sequence shown below the first character "G" is replaced in the second iteration by character "A" in S2 and is compared with the reference sequence S1 and so on.

	Sequence S1	Replace a character from S2	Correlogram Score	DP Score
Replace 1st character	ATTCGGAGTT	**G**TAGATGCTC	4.05	-0.2
	ATTCGGAGTT	**A**TAGATGCTC	4.59	-0.3333
	ATTCGGAGTT	**T**TAGATGCTC	3.78	-0.3333
	ATTCGGAGTT	**C**TAGATGCTC	4.6	-0.2
Replace 5 th character	ATTCGGAGTT	GTAG**A**TGCTC	4.05	-0.2
	ATTCGGAGTT	GTAG**T**TGCTC	3.92	-0.2
	ATTCGGAGTT	GTAG**G**TGCTC	3.85	-0.3333
	ATTCGGAGTT	GTAG**C**TGCTC	4.54	-0.2
Replace 10 th character	ATTCGGAGTT	GTAGATGCT**C**	4.05	-0.2
	ATTCGGAGTT	GTAGATGCT**A**	3.94	-0.2
	ATTCGGAGTT	GTAGATGCT**T**	3.64	-0.3333
	ATTCGGAGTT	GTAGATGCT**G**	3.67	-0.2

Table 7-4: DP score and correlogram score values using replacement of a character from a sequence

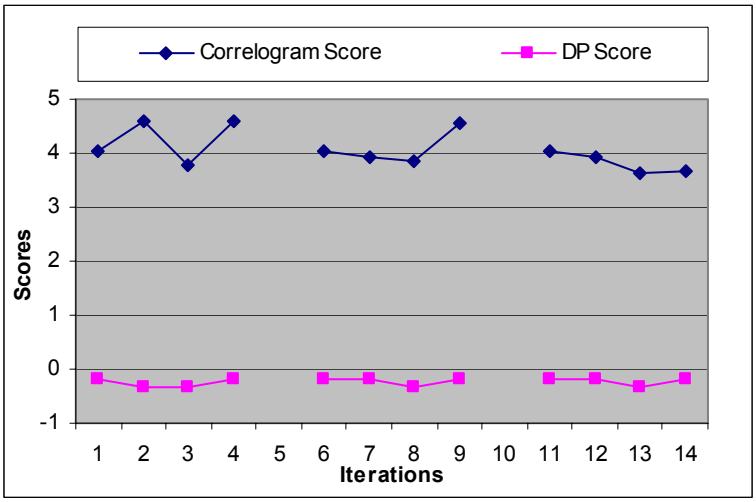


Figure 7-4: Comparison of DP score and correlogram score using replacement of a character from a sequence

5. Finally add a character and measure the difference. – In this experiment all the 4 characters were added to S2 at first position one by one and S1 was compared with S2. Same steps were repeated for adding characters at 5th position and 10th position. For example, in the sequence shown below a character "G" was added before the target sequence S2 in the first iteration, followed by "A," "T" and "C" in subsequent iterations. This was repeated for the 5th position and the 10th position in the target sequence.

	Sequence S1	Sequence S2	Add a character to S2	Corre-logram Score	DP Score
Add at the 1st position	ATTCGGAGTT	GTAGATGCTC	**G**GTAGATGCTC	4.19	-0.25
	ATTCGGAGTT	GTAGATGCTC	**A**GTAGATGCTC	4.13	-0.5
	ATTCGGAGTT	GTAGATGCTC	**T**GTAGATGCTC	4.05	-0.25
	ATTCGGAGTT	GTAGATGCTC	**C**GTAGATGCTC	3.81	-0.25
Add at the 5th position	ATTCGGAGTT	GTAGATGCTC	GTAG**G**ATGCTC	3.90	-0.3333
	ATTCGGAGTT	GTAGATGCTC	GTAG**A**ATGCTC	4.24	-0.2
	ATTCGGAGTT	GTAGATGCTC	GTAG**T**ATGCTC	3.78	-0.2
	ATTCGGAGTT	GTAGATGCTC	GTAG**C**ATGCTC	4.64	-0.2
Add at the 10th position	ATTCGGAGTT	GTAGATGCTC	GTAGATGCTC**G**	4.18	-0.1429
	ATTCGGAGTT	GTAGATGCTC	GTAGATGCTC**A**	4.18	-0.1429
	ATTCGGAGTT	GTAGATGCTC	GTAGATGCTC**T**	4.07	-0.2
	ATTCGGAGTT	GTAGATGCTC	GTAGATGCTC**C**	4.39	-0.1429

Table 7-5: DP score and correlogram score values using addition of a character for a sequence

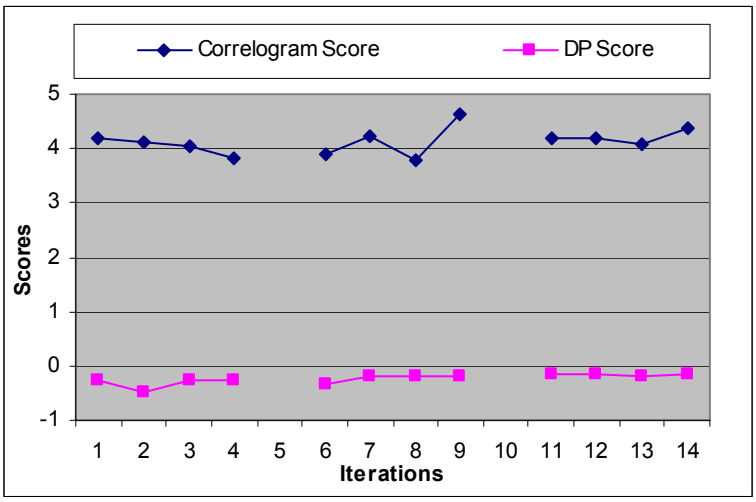


Figure 7-5: Comparison of DP score and correlogram score using addition of a character for a sequence

Observations and Results-

1. Correlogram method gives the difference score between 2 sequences whereas DP method gives the similarity score.
2. In the reverse sequence comparison DP method gives same score when any sequence is compared with its reverse sequence, whereas the correlogram method score changes with each sequence.
3. The wrapped around sequence correlogram method gives a maximum difference score when the sequence is halfway wrapped. The DP method gives the minimum similarity score approximately when 1/3rd and 2/3rd wrapped.
4. When a character is deleted, the DP score does not change whereas the correlogram score varies depending on which character is getting deleted.
5. Overall, DP method is more related to alignment of characters. The score does not vary much with which character is getting added or deleted. But correlogram score is more sensitive to which character is getting added or deleted.

7.2 Experiment 2 – Equine-2 (Horse) Influenza Virus

Purpose – This experiment was designed to compare the correlogram method with other methods. Lai et. al.[xl] describe the phylogenetic and antigenic analysis of recent circulating equine-2 influenza viruses in the United States. The

hemagglutinin (HA) gene of different strains of equine-2 influenza viruses was studied. This gene is the dominant surface antigen of the influenza viri and the principal mediator of immunity. Together with the NA (neuraminidase) gene it defines the antigenic phenotype of the virus, which in turn, classifies the influenza A viruses into subtypes. [xli]

Steps –

Influenza sequence data was compiled and analyzed using GeneTool version 1.1. Phylogenetic analysis was performed by using the deduced HA1 amino acid sequence and the PHYLIP software package (University of Washington). Distance matrix was calculated by using the PROTDIST program, and an un-rooted tree generated by using the FITCH program.

Virus	Abbreviation	Accession number	Country of origin
A/Eq/Saskatoon/90	SA90	AF197243	Canada
A/Eq/Suffolk/89	SU90	X68437	United Kingdom
A/Eq/Lambourne/92	LM92	X85087	United Kingdom
A/Eq/Hong Kong/92	HK92	L27597	China
A/Eq/Kentucky/91	KY91	L39918	United States
A/Eq/Kentucky/92	KY92	L39917	United States
A/Eq/Kentucky/94	KY94	L39914	United States
A/Eq/Kentucky/95	KY95	AF197247	United States
A/Eq/Kentucky/96	KY96	AF197248	United States
A/Eq/Kentucky/97	KY97	AF197249	United States
A/Eq/Kentucky/98	KY98	AF197241	United States
A/Eq/Florida/93	FL93	L39916	United States
A/Eq/Florida/94	FL94	AF197242	United States
A/Eq/Argentina/93	AR93	L39913	Argentina
A/Eq/Argentina/94	AR94	AF197245	Argentina
A/Eq/Argentina/95	AR95	AF197244	Argentina
A/Eq/Argentina/96	AR96	AF197246	Argentina
A/Eq/New York/99[a]	NY99	AY273167	United States
A/Eq/Oklahoma/00[a]	OK00	AY273168	United States

[a] The HA_1 sequences determined in this study.

Table 7-6: Horse influenza virus data ♦

The same test data was analyzed using the correlogram method. All the protein sequences were obtained from the EMBL-EBI (European Bioinformatics Institute)[xlii] Database. A distance matrix was created using correlogram distances for every pair among these sequences. The basic Correlogram algorithm was used to calculate the correlogram distances. From this distance matrix, a tree was created using PHYLIP software package. [xliii] The program 'FITCH' from Phylip software package was used for creating tree whereas the program 'DRAWTREE' was used for visualizing the tree.

♦ This table is obtained from the paper by Lai et. al. referred above.

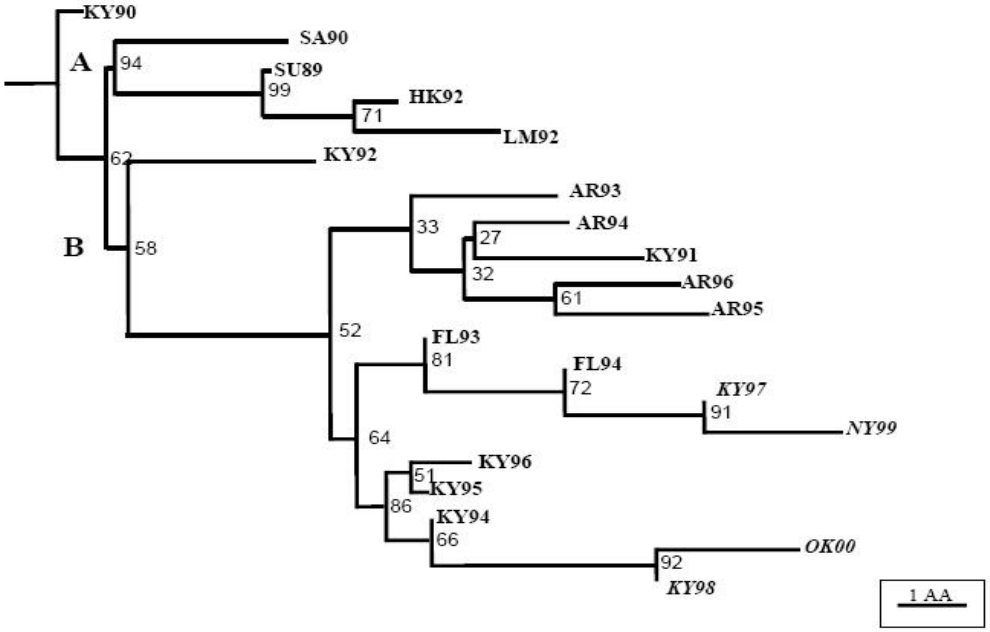


Figure 7-6: Phylogenetic Tree Generated with Lai et. al.'s Experiment

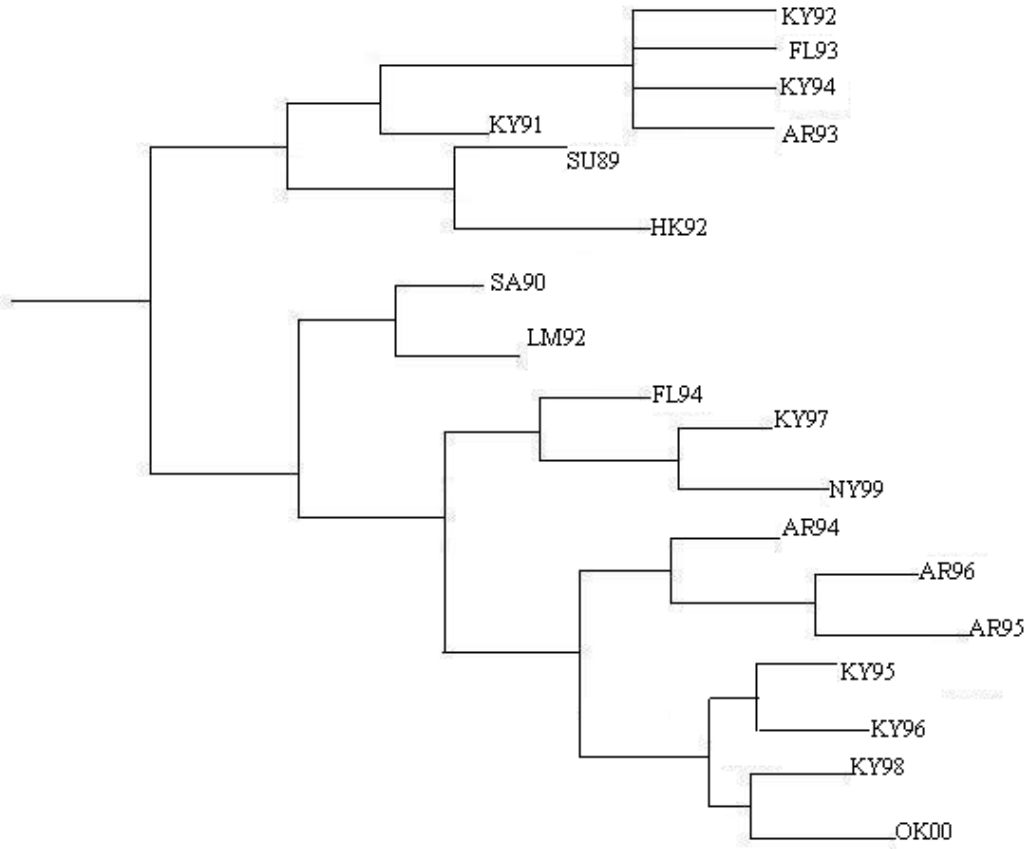


Figure 7-7: Phylogenetic Tree Using Correlogram Method *

Comparison between trees –

The tree obtained in this experiment (Figure 7-6) was compared with the tree generated in Chapman and Rossmann's experiment (Figure 7-7). Ideally distance matrices for both the trees could have been compared but since the distance matrix was not available I compared the tree topologies. Following table shows the difference matrix for the tree nodes.

* KY90 is not present in the tree as the accession number could not be found in the available data.

	SA90	SU89	LM92	HK92	KY91	KY92	KY94	KY95	KY96	KY97	KY98	FL93	FL94	AR93	AR94	AR95	AR
SA90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
SU89	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
LM92	-2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
HK92	2	-1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	
KY91	-2	-5	-4	-6	0	0	0	0	0	0	0	0	0	0	0	0	
KY92	2	0	1	-1	-3	0	0	0	0	0	0	0	0	0	0	0	
KY94	-1	-4	-3	-5	-5	-4	0	0	0	0	0	0	0	0	0	0	
KY95	-2	0	-3	-1	1	4	6	0	0	0	0	0	0	0	0	0	
KY96	-1	0	-3	-1	1	4	6	0	0	0	0	0	0	0	0	0	
KY97	-2	-2	-5	-3	-1	2	2	0	0	0	0	0	0	0	0	0	
KY98	-2	-1	-4	-2	0	3	7	-1	-1	-1	0	0	0	0	0	0	
FL93	0	-3	-2	-4	-4	-3	-4	5	5	5	4	0	0	0	0	0	
FL94	-3	-2	-5	-3	-1	2	3	0	0	0	-1	5	0	0	0	0	
AR93	1	-2	-1	-3	-1	-2	-4	4	4	2	3	-3	2	0	0	0	
AR94	-2	-1	-4	-2	6	3	1	-3	-3	-3	-4	2	-2	5	0	0	
AR95	-1	0	-3	-1	5	4	2	-2	-2	-2	-3	3	-1	6	-1	0	
AR96	-1	0	-3	-1	5	4	2	-2	-2	-2	-3	3	-1	6	-1	0	
NY99	-3	-2	-5	-3	-1	2	2	0	0	0	-2	5	0	2	-3	-2	
OK00	-1	-1	-4	-2	0	3	7	-1	-1	-1	0	4	-1	3	-4	-3	

Table 7-7: Difference Matrix for Tree Nodes

Observations and results

1. Some similarities were retained in both the experiments.

   E.g. HK92 and SU89 are close in both the trees. Also KY98 and OK00 are close in both the trees.

2. The correlogram scores grouped AR94, AR95 and AR96 together whereas in the other tree AR94 is far from AR95 and AR96.

7.3 Experiment 3 – Parvovirus

Purpose – The purpose of this experiment was to find the Phylogenetic relationships among different types of Parvoviri using Correlogram based distances. The different parvoviri were found in an article by Chapman and Rossmann. [xliv]

Steps -

1. The following viruse sequences were used for this experiment.
    a. B19 virus
    b. Bovine parvovirus
    c. Canine parvovirus strain B
    d. Feline panleukopenia virus (strain 193)
    e. Murine minute virus (STRAIN MVMI)
    f. Porcine parvovirus (STRAIN NADL-2)
    g. Raccoon parvovirus
    h. Adeno-associated virus 2
    i. Galleria mellonella densovirus
2. All VP1 protein sequences for the organisms used in the above experiment were downloaded from NCBI. [xlv]
3. Two different distance matrices were created using basic and delta correlogram method.
4. Using these distance matrices and NEIGHBOUR[xlvi] program from PHYLIP un-rooted trees were created.

5. These trees were viewed using DRAWTREE [xlvii] and DRAWGRAPM [xlviii] programs from PHYLIP.

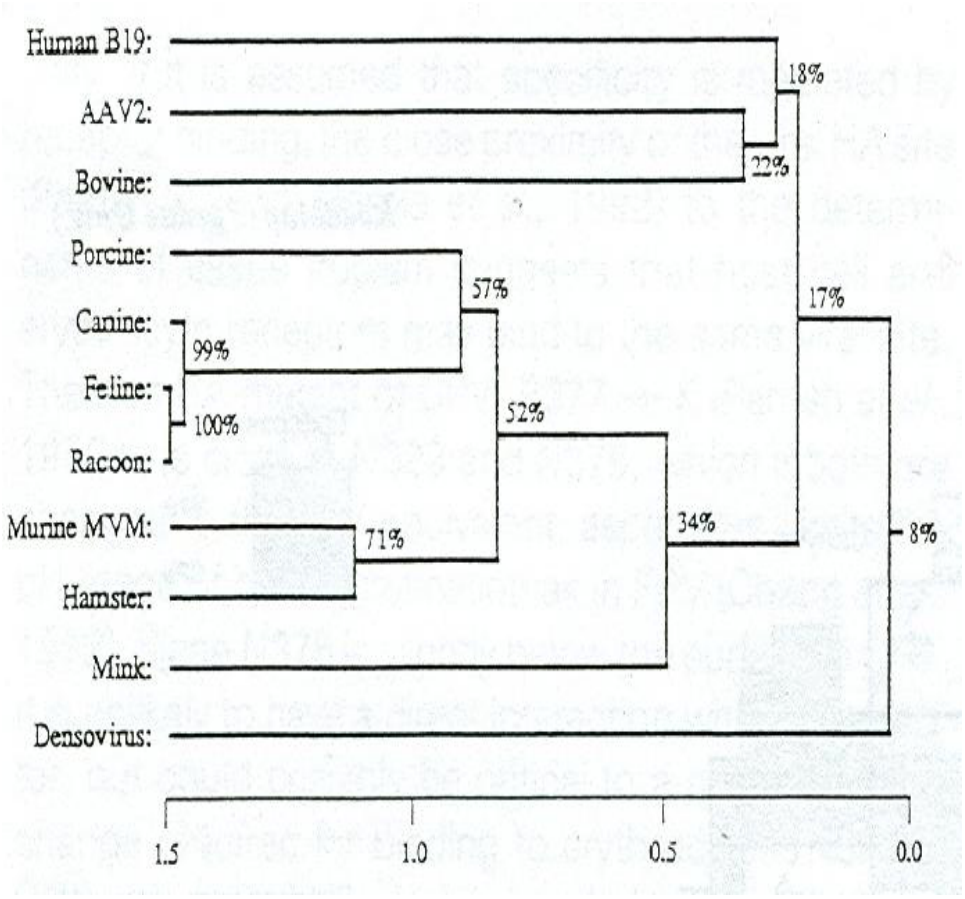
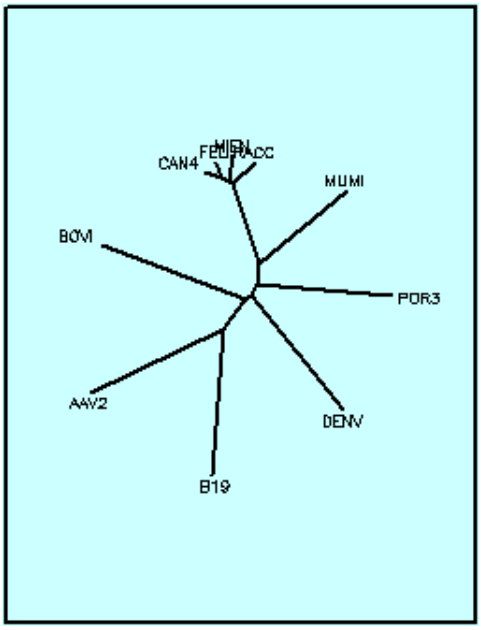
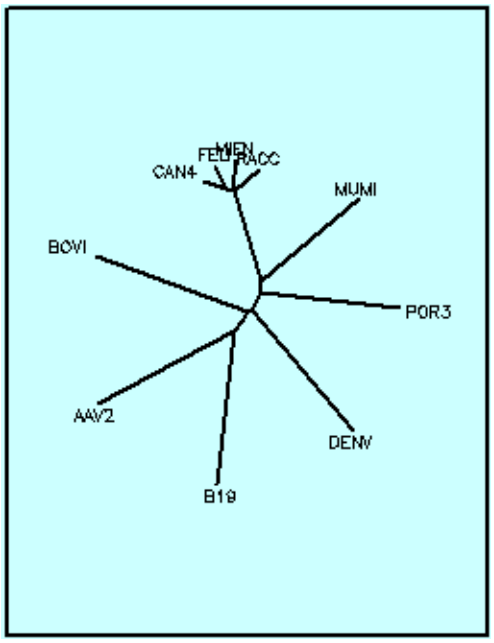


Figure 7-8: Phylogenetic Tree Generated with Chapman and Rossmann's Experiment

Results -

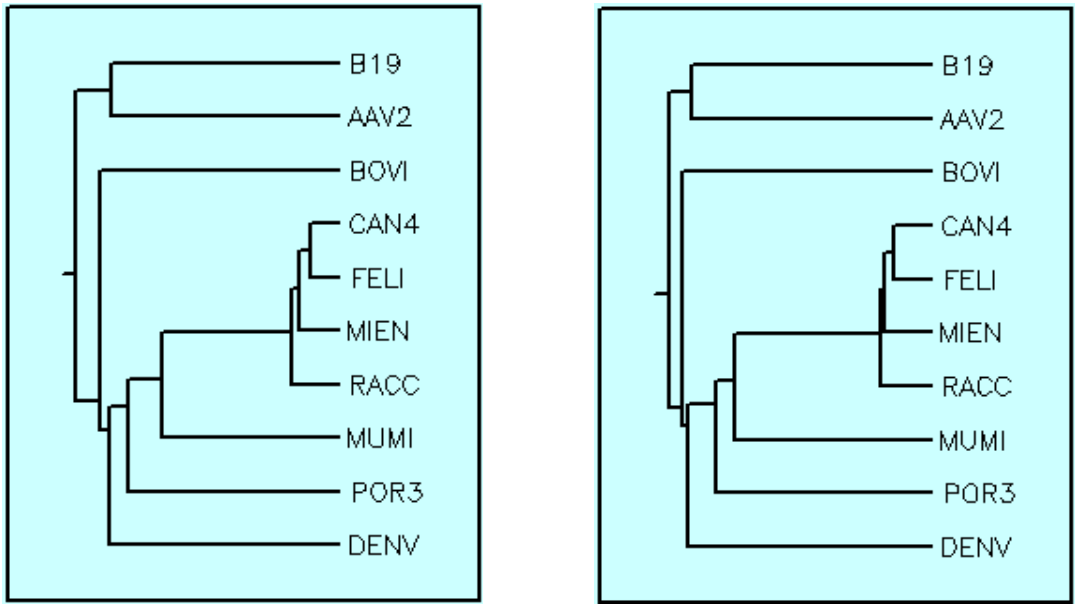




Delta Correlogram

Basic Correlogram

Figure 7-9: Phylogeny trees created using DRAWTREE



Delta Correlogram Basic Correlogram

Figure 7-10: Phylogeny trees created using DRAWGRAM

Distance Matrices

	DENV	BOVI	AAV2	B19	MIEN	POR3	CAN4	FELI	RACC	MUMI
DENV	0	0.078	0.083	0.085	0.076	0.077	0.077	0.077	0.078	0.076
BOVI	0.078	0	0.085	0.093	0.080	0.091	0.078	0.078	0.080	0.071
AAV2	0.083	0.085	0	0.076	0.081	0.087	0.081	0.081	0.084	0.076
B19	0.085	0.093	0.076	0	0.095	0.096	0.097	0.096	0.099	0.087
MIEN	0.076	0.080	0.081	0.095	0	0.068	0.014	0.012	0.014	0.059
POR3	0.077	0.091	0.087	0.096	0.068	0	0.067	0.070	0.072	0.076
CAN4	0.077	0.078	0.081	0.097	0.014	0.067	0	0.009	0.018	0.059
FELI	0.077	0.078	0.081	0.096	0.012	0.070	0.009	0	0.015	0.058
RACC	0.078	0.080	0.084	0.099	0.014	0.072	0.018	0.015	0	0.061
MUMI	0.076	0.071	0.076	0.087	0.059	0.076	0.059	0.058	0.061	0

Table 7-8: Distance Matrix for basic correlogram distances

	DENV	BOVI	AAV2	B19	MIEN	POR3	CAN4	FELI	RACC	MUMI
DENV	0	0.113	0.116	0.116	0.109	0.110	0.109	0.110	0.112	0.109
BOVI	0.113	0	0.118	0.125	0.113	0.124	0.111	0.111	0.113	0.106
AAV2	0.116	0.118	0	0.108	0.113	0.120	0.113	0.113	0.116	0.107
B19	0.116	0.125	0.108	0	0.124	0.126	0.125	0.124	0.128	0.117
MIEN	0.109	0.113	0.113	0.124	0	0.094	0.023	0.019	0.021	0.088
POR3	0.110	0.124	0.120	0.126	0.094	0	0.093	0.095	0.098	0.104
CAN4	0.109	0.111	0.113	0.125	0.023	0.093	0	0.017	0.027	0.087
FELI	0.110	0.111	0.113	0.124	0.019	0.095	0.017	0	0.021	0.087
RACC	0.112	0.113	0.116	0.128	0.021	0.098	0.027	0.021	0	0.091
MUMI	0.109	0.106	0.107	0.117	0.088	0.104	0.087	0.087	0.091	0

Table 7-9: Distance Matrix for delta correlogram distances

Comparison between trees –

The tree obtained in this experiment (Figure 7-9 Basic Correlogram) was compared with the tree generated in Chapman and Rossmann's experiment (Figure 7-7). Following table shows the difference matrix for the tree nodes.

	B19	AAV2	BOVI	CAN4	FELI	MIEN	RACC	MUMI	POR3	DENV
B19	0	-1	1	3	2	5	0	1	0	1
AAV2	-1	0	2	2	1	4	-1	0	-1	0
BOVI	1	2	0	0	-1	2	-3	-2	-3	-2
CAN4	3	2	0	0	-1	-2	1	0	3	0
FELI	2	1	-1	-1	0	-3	2	-1	2	-1
MIEN	5	4	2	-2	-3	0	-3	0	1	2
RACC	0	-1	-3	1	2	-3	0	-3	0	-3
MUMI	1	0	-2	0	-1	0	-3	0	-1	-2
POR3	0	-1	-3	3	2	1	0	-1	0	-3
DENV	1	0	-2	0	-1	2	-3	-2	-3	0

Table 7-10: Difference Matrix for Tree Nodes

Observations -

1. The overall similarity is maintained between both the trees.
2. The tree shown in above paper has Densovirus in a different branch, whereas our tree has it inside one branch of Parvoviri.
3. B19 and AAV2 are very near in both the trees.
4. The nearness is retained among Canine, Feline and Raccoon sequences.

7.4 Experiment 4 - Scanning of sequences

Purpose – The purpose of this experiment was to find the given pattern over the long sequence with the aid of correlogram.

Steps – The Scan correlogram algorithm explained in chapter 6 was used in this experiment. Every virus sequence was scanned for each of the given patterns and the score and location for each substring in the target sequence were recorded.

The following virus sequences were studied in this experiment: *

1. Porcine-parvovirus
2. Bovine Parvovirus
3. CPV Packaged Strand
4. H1 Complementary
5. MVM Packaged Strand
6. PhiX-Genome
7. AAV NC001401
8. AAV Complementary
9. ADV Complementary
10. Astell and Tattersall MVMi Packaged Sequence.

The three patterns which were searched in this sequence were:

1. ACACCAAAA
2. ATACCTCTTGC

* The Parvovirus data used in this experiment was given by Dr. Mavis McKenna and her team (Macromolecular Structure Group, University of Florida). They also supplied with the patterns which they wanted to search over long parvovirus sequences.

3. ATCCTCTATCAC.

The Scan Correlogram algorithm gives score for every substring of the target sequence. But we needed to find the most matching patterns; hence a cut off score was defined. The algorithm returned all the scores for each substrings. The purpose of this experiment was to find the most matching substrings and as for the correlogram method lowest score gives the best match. All the values less than the cut off were recorded to locate the substring and only values less than the cutoff score were accepted. The algorithm searches substrings of exact length as pattern sequence, but it can be extended to find the substrings or more of less length than the pattern. This extension can be further studied.

Following are the results shown for **Bovine Parvovirus.**

The length of sequence was 5517 and the cut off score used was 2.48 for all three patterns.

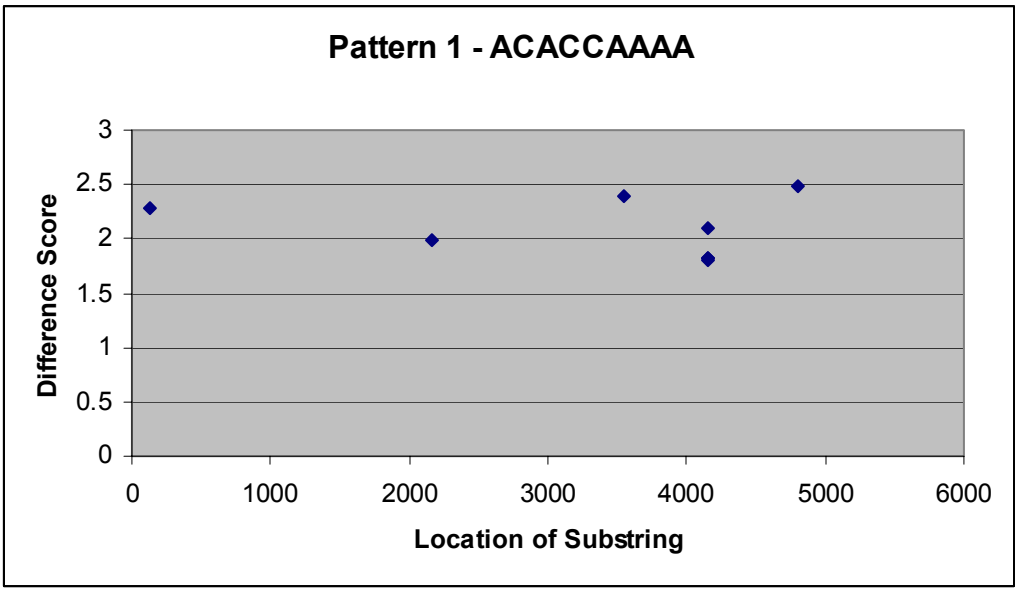


Figure 7-11: Graph showing locations and scores for pattern 1

The patterns found were as follows:

Location	Score Distance	Substring
129	2.28	Acaactaaa
2167	1.99	Acccaaata
3543	2.39	Aactccaaa
4149	1.83	Taccaccaa
4150	1.83	Accaccaaa
4151	2.09	Ccaccaaat
4152	1.81	Caccaaatc
4798	2.48	Acccccaat

Table 7-11: Locations and scores for Pattern 1

While comparing the above substrings with the pattern sequence "ACACCAAAA," we observe that the pattern at location **4152** having a score of **1.81** was observed to

be the longest common substring and obtained the lowest score. The next closest substrings had a score of 1.83 at locations 4149 and 4150.

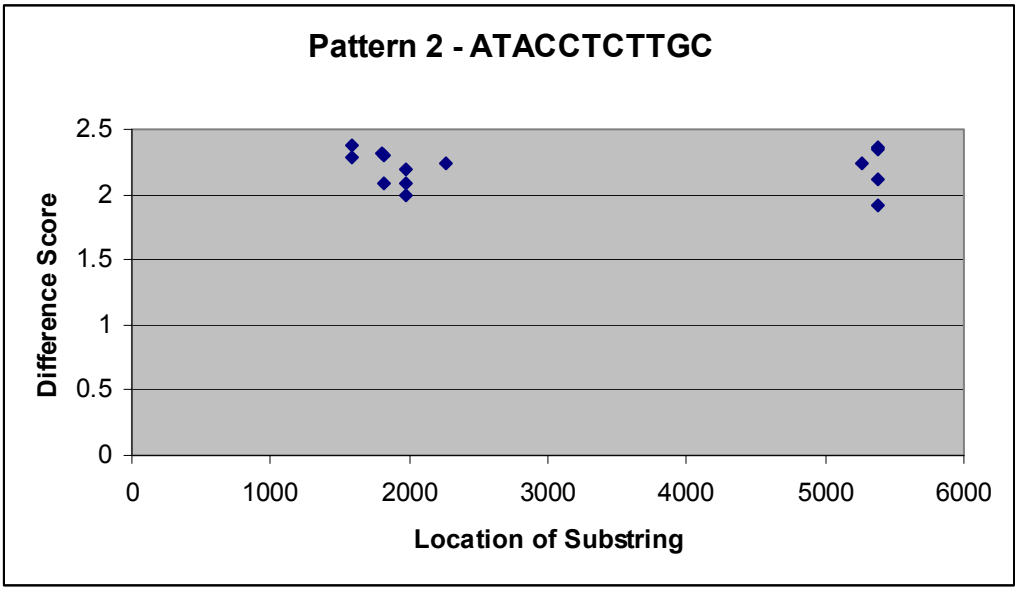


Figure 7-12: Graph showing locations and scores for pattern 2

The patterns were found as follows:

Location	Score Distance	Substring
1582	2.37	Cacctcgcttt
1583	2.28	Acctcgctttt
1808	2.31	Tttgatacctc
1812	2.08	Atacctccatg
1813	2.30	Tacctccatgt
1974	2.20	Acactcctctt
1975	2.00	Cactcctcttg
1976	2.08	Actcctcttgc
2270	2.24	Tatctcttctg
5267	2.24	Taccctctagt
5379	2.11	Actcatttcct
5380	1.92	Ctcatttcctg
5381	2.36	Tcatttcctgc
5382	2.34	Catttcctgca

Table 7-12: Locations and scores for pattern 2

While comparing the above substrings with the pattern sequence "ATACCTCTTGC," we observe that the pattern at location **5380** having a score of **1.92** obtained the lowest score. This was due to the fact that in all the substrings above it was the most matching substring.

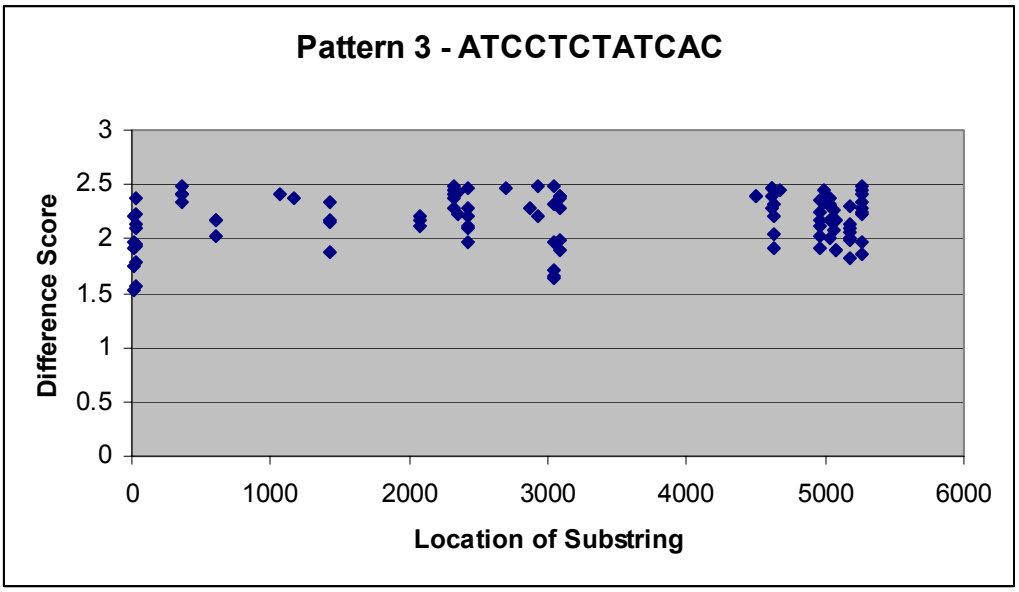


Figure 7-13: Graph showing locations and scores for pattern 3

For pattern 3, several substrings were found with scores less than the cut-off score. While comparing all these substrings with the pattern sequence "ATCCTCTATCAC," we observed that the pattern at location **21** having a score of **1.53** obtained the lowest score.

Observations and Results-

1. The substrings found were very similar to the given pattern.
2. The total count of matches varies with the cut off values. For the same cutoff value, the three patterns gave largely different total counts.

Chapter 8

Conclusions and Future Research

8.1 Conclusions and Future Research Directions

There are numerous ways by which sequences can be compared in the field of bioinformatics. This research developed the correlogram comparison method for comparing sequences. Experiments were performed on real sequences and on synthetic sequences. These experiments were designed to answer the research questions of whether the correlogram method can be utilized to compare biological sequences.

The synthetic data experiment compared the correlogram method and the dynamic programming methods used for comparing 2 sequences. The scores were compared by drawing graphs for both the correlogram score and the DP scores. The results showed some differences between the correlogram score and the DP score. Based on this, it was observed that the Dynamic Programming method was more sensitive to the positioning of characters in the sequence, whereas the Correlogram method was found to be more sensitive to the character itself. For e.g. if a character is added to a sequence and the sequence is compared with the original sequence, correlogram score changed if the character being added is changed, whereas DP score stayed constant. This result confirmed the answer for the first research

question “What are the advantages of the correlogram technique over the previous techniques?” Thus the correlogram method can be used and further researched for the biological sequence. The further study can be done to see how the array of distances used for correlogram computations can impact the results. Following is the comparison chart of DP algorithm and Correlogram algorithm with respect to synthetic experiments.

	Correlogram Algorithm	**DP algorithm**
Reverse Sequence	No significant difference	No significant difference
Wrap around	Minimum Similarity score when half wrapped	Minimum similarity score when 1/3rd and 2/3rd wrapped.
Delete character	Score changes depending on which chracter is deleted	Score does not change.
Replace character	Score changes depending on which chracter is replaced	score does not change.
Add character	Score changes depending on which chracter is added	Score changes depending on where chracter is added

Table 8-1: Comparison of Correlogram and DP algorithm with respect to Synthetic Experiments

The experiment was conducted with real data on different strains of the horse influenza virus and the parvovirus. For the horse influenza virus, we compared the phylogeny tree obtained from the paper by Lai et. al. [xlix] with the phylogeny tree obtained using correlogram distances. It was observed that the phylogeny was retained in most cases; however there were certain differences between the two.

This might suggest that there can be some biological reasoning behind these differences which can be a starting point for further research.

Both basic and gapped correlogram methods were used to find the difference score in the parvovirus experiment. Using these results, phylogeny trees were drawn for both sets of results. The results showed that there was some difference between the trees depicted in Figure 7-8 and Figure 7-9 found by Basic method and Gapped Correlogram method which was answer to our second research question "Can the correlogram method be modified to take into account the biological gaps between sequences?" The answer to this question is yes it can be modified, however the differences between the basic and gapped correlogram methods were not very significant. It will be interesting to study various delta values for Gapped correlograms and how they affect the scores. This gapped correlogram method can be further researched to see if the delta values are useful in determining global versus local alignments. The smaller delta values can be used to find the local alignments in the sequences or where there are small gaps within sequences. The bigger delta values can be used to find the bigger gaps in the sequences.

Using parvovirus data, the correlogram method was used to construct the phylogeny tree. The phylogenetic relationship was found to be in accordance with the characteristics of parvovirus. Further research can be conducted to compare these results (phylogeny trees) with trees drawn using other sequence comparison methods.

Finally, the research looked at finding an answer to the third research question "Can the correlogram method be used to find the occurrence of a given pattern over a long sequence." The scan correlogram algorithm was developed and used in this research to find motifs or patterns. The experiment was performed on certain parvovirus sequences to find 3 distinct patterns. The results of this experiment showed that the sub-sequences obtained were very similar to the given pattern. Further enhancements can be made to the scan correlogram method to use the gapped correlogram method for finding patterns and also to find the sub-sequences of more or less length than that of the pattern sequence.

[i] http://www.nist.gov/dads/HTML/stringMatching.html , February 5, 2005

[ii] http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/glossary2.html , February 5, 2005

[iii] http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/glossary2.html , February 5, 2005

[iv] http://www.cs.ucdavis.edu/~gusfield/extramaster.old/node2.html , February 3, 2005

[v] Huang, J., Kumar, S.R., Mitra, M., Zhu, W.J. and Zabih, R. (1999), "Image Indexing using color correlograms," International Journal of Computer Vision 35(3), pp 245-268

[vi] http://cmex-www.arc.nasa.gov/VikingCD/Puzzle/Evolife.htm, June 2005

[vii] Setubal and Meidanis, "Introduction to computational molecular biology," McGraw Hill Publications, 5th Edition, January 20, 2005

[viii] http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/index.html, August 2005

[ix] http://en.wikipedia.org/wiki/Sequence_clustering, September 2005

[x] http://www.cdc.gov/flu/about/disease.htm, September 2005

[xi] http://encyclopedia.laborlawtalk.com/Parvoviridae and http://cnx.rice.edu/content/m11062/latest/, October 2005

[xii] http://www.m.ehime-u.ac.jp/~yasuhito/Elisa.html, October 2005

[xiii] A **taxon** (plural **taxa**), or **taxonomic unit**, is an element of a taxonomy, most commonly used in the scientific classification in biology, where a taxon is a group of organisms that has been named. Adapted from http://en.wikipedia.org/wiki/Taxon, October 2005

[xiv] http://www.cdc.gov/ncidod/dvrd/revb/respiratory/parvo_b19.htm, April 2005

[xv] A **taxon** (plural **taxa**), or **taxonomic unit**, is an element of a taxonomy, most commonly used in the scientific classification in biology, where a taxon is a group of organisms that has been named. Adapted from http://en.wikipedia.org/wiki/Taxon, October 2005

[xvi] http://www.unm.edu/~jerusha/lab10.htm, December 2005

[xvii] http://evolution.genetics.washington.edu/phylip.html, May 2005

[xviii] http://evolution.genetics.washington.edu/phylip/progs.data.dist.html, June 2005

[xix] The web documentation for FITCH, Kitsch and NEIGHBOUR can be found at http://www.umanitoba.ca/afs/plant_science/psgendb/doc/Phylip/distance.html, June 2005

[xx] The web documentation for DRAWGRAM and DRAWTREE can be found at http://www.umanitoba.ca/afs/plant_science/psgendb/doc/Phylip/draw.html, May 2005

[xxi] http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/information3.html , February 3, 2005

[xxii] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990), "Basic Local Alignment Search Tool," Journal of Molecular Biology No. 215, pp 403–410

[xxiii] Stephen F. Altschul*, Thomas L. Madden, Alejandro A. Schäffer1, Jinghui Zhang, Zheng Zhang, Webb Miller and David J. Lipman (1997), "BLAST and PSI-BLAST: a new generation of protein database search programs," Nucleic Acids Research Vol. 25, No. 17, pp 3389–3402

[xxiv] W. R. Pearson and D. J. Lipman (1988), "Improved Tools for Biological Sequence Comparison," PNAS 85, pp 2444- 2448

[xxv] http://bimas.dcrt.nih.gov/fastainfo/fasta_algo, June 2005

[xxvi] http://www.infobiogen.fr/doc/MAcours/multalign.html , May 11, 2005

[xxvii] Carillo, H. and Lipman, D. (1988), "The multiple sequence alignment problem in biology," SIAM Journal of Applied Math. 48(5), pp 1073-1082

[xxviii] Jens Kleinjung, John Romein, Kuang Lin1 and Jaap Heringa (2004), "Contact-based sequence alignment," Nucleic Acids Research, Vol. 32, No. 8, pp 2464-2473

[xxix] Matthew O. Ward and David S. Admas (1990), "Nucleotide sequence analysis using correlation images Visualization in Biomedical Computing," Proceedings of the First Conference on 22-25 May 1990, pp 49 - 56

[xxx] Kenneth P. Hinckley, Matthew O. Ward (1991), "The Visual Comparison of Three Sequences," Proceedings of the 2nd conference on Visualization '91 (IEEE), pp 179-186

[xxxi] Haubold B., Pierstorff N., Moller F. and Wiehe T.(2005), "Genome comparison without alignment using shortest unique substrings," BMC Bioinformatics 6:123, pp 1-11

[xxxii] http://www.qgsi.com/Terms.html, June 2005

[xxxiii] Huang, J., Kumar, S.R., Mitra, M., Zhu, W.J. and Zabih, R. (1999), "Image Indexing using color correlograms," International Journal of Computer Vision 35(3), pp 245-268

[xxxiv] Qi Zhao and Hai Tao (2005), "Object Tracking using Color Correlogram**"** to appear in IEEE Workshop on VS-PETS, October 2005

[xxxv] MF Macchiato, V Cuomo and A Tramontano (1985), "Determination of the autocorrelation orders of proteins," European Journal of Biochemistry Vol 149, pp 375-379

[xxxvi] Giorgio Bertorelle and Guido Barbujanit (1995), "Analysis of DNA Diversity by Spatial Auto Correlation," Genetics Society of America, pp 811 - 819

[xxxvii] Michael S. Rosenberg, Sankar Subramanian, and Sudhir Kumar (2003), "Patterns of Transitional Mutation Biases Within and Among Mammalian Genomes," Molecular Biology & Evolution Vol 20, pp 988-993

[xxxviii] http://lectures.molgen.mpg.de/Pairwise/SeqAli/, October 2005

[xxxix] Brona Brejova, Chrysanne DiMarco, Tomas Vinar et. al. (2000), "Finding patterns in biological sequences," Technical report CS-2000-22, University of Waterloo, pp 1-49

[xl] Alexander C.K. Lai, Kristin M. Rogers, Amy Glaser, Lynn Tudor, Thomas Chambers "Alternate circulation of recent equine-2 influenza viruses (H3N8) from two distinct lineages in the United States," Virus Research Vol 100 (2), pp 159-64.

[xli] http://www.flu-archive.org/glossary.html, November 2005

[xlii] http://www.ebi.ac.uk/cgi-bin/expasyfetch, October 2005

[xliii] http://evolution.genetics.washington.edu/phylip.html, August 2005

[xliv] Micheal S. Chapman and Micheal G. Rossmann (1993), "Structure, Sequence and Function Correlations among Parvoviruses," Virology 194, pp 491-508

[xlv] http://www.ncbi.nlm.nih.gov/, October 2005

[xlvi] http://www.umanitoba.ca/afs/plant_science/psgendb/doc/Phylip/neighbor.html, August 2005

[xlvii] http://www.umanitoba.ca/afs/plant_science/psgendb/doc/Phylip/drawtree.html, September 2005

[xlviii]

http://www.umanitoba.ca/afs/plant_science/psgendb/doc/Phylip/drawgram.html, September 2005

[xlix] Alexander C.K. Lai, Kristin M. Rogers, Amy Glaser, Lynn Tudor, Thomas Chambers "Alternate circulation of recent equine-2 influenza viruses (H3N8) from two distinct lineages in the United States," Virus Research Vol 100 (2), pp 159-64.