ISLAMIC UNIVERSITY OF TECHNOLOGY

A Modified Algorithm For DNA Motif Finding & Ranking Considering Variable Length Motif & Mutation

Authors:

Adnan Ferdous Ashrafi (104414)
A.K.M Iqtidar Newaz Adit (104427)

Supervisor:

Prof. Dr. M.A Mottalib
Head of Department,
Department of Computer Science and Engineering, IUT.

Co-Supervisor:

Moin Mahmud Tanvee
Lecturer,
Department of Computer Science and Engineering, IUT.

A thesis submitted to the Department of CSE

In partial fulfillment of the requirements for the degree

B. Sc. Engineering in CSE

Academic Year: 2013-2014

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and investigation carried out by Adnan Ferdous Ashrafi and A.K.M Iqtidar Newaz Adit under the supervision of Prof. Dr. M.A Mottalib and Moin Mahmud Tanvee of the Department of Computer Science and Engineering (CSE), IUT, Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

_____

Adnan Ferdous Ashrafi
Student ID – 104414

_____

A.K.M Iqtidar Newaz Adit
Student ID – 104427

_____

Supervisor:
Prof. Dr. M.A Mottalib
Head of Department
Department of Computer Science and Engineering
Islamic University of Technology (IUT)

Abstract:

With the evolution of time the gene composition of species have changed a lot. Consequently it has mutated to generate new diseases and traits. In order to identify genes or coding sections of a DNA sequence it is imperative to find out the promoter regions or the conserved regions of the DNA code first. But the main problem stands that the databases for these information are quite messy and needs to be researched. The main problems in finding motifs in a DNA sequence are finding a good and fast algorithm, considering mutations in those motifs, representing variable length motifs and being species general. In this thesis work we tried to formulate a new algorithm which is fast, accurate and effective. Instead of general string matching of DNA sequences we have done integer mapping and matching which are comparatively fast and accurate. Besides in order to formulate a complete DNA motif finding algorithm we also need a generalized and rational fitness function for evaluating the potential motifs. Thus we have formulated a desirable fitness function that enables us to compare the relativity among potential motifs and finally to predict a certain motif for the input DNA sequences.

Keywords: Motifs, De Novo Motif Finding, Hash Table, Mutation, Fitness Function, Integer matching.

Table of Contents

Table of Figures :

Chapter 1

Introduction

1.1 Overview

Bioinformatics is the application of computer technology to the management of biological information. Computers are used to gather, store, analyze and integrate biological and genetic information which can then be applied to gene-based drug discovery and development. The need for Bioinformatics capabilities has been precipitated by the explosion of publicly available genomic information resulting from the Human Genome Project. In experimental molecular biology, bioinformatics techniques such as image and signal processing allow extraction of useful results from large amounts of raw data.

In the field of genetics and genomics, it aids in sequencing and annotating genomes and their observed mutations. It plays a role in the textual mining of biological literature and the development of biological and gene ontology's to organize and query biological data. It plays a role in the analysis of gene and protein expression and regulation. Bioinformatics tools aid in the comparison of genetic and genomic data and more generally in the understanding of evolutionary aspects of molecular biology. At a more integrative level, it helps analyze and catalogue the biological pathways and networks that are an important part of systems biology. In structural biology, it aids in the simulation and modeling of DNA, RNA, and protein structures as well as molecular interactions.

Accelerated sequencing of genomes in the past decade has been accompanied by a rise in the use of bioinformatics tools by microbiologists. It would be difficult to find a microbiologist today who has never accessed online biological databases. Bioinformaticians and software developers have facilitated this process by increased emphasis on convenience and ease of use of software and databases, which in turn allowed microbiologists to use increasingly sophisticated techniques with minimal effort. The discovery of patterns in DNA, RNA, and protein sequences has led to the solution of many vital biological problems. For instance, the identification of patterns in nucleic acid sequences has resulted in the determination of open reading frames, identification of promoter elements of genes, identification of intron/exon splicing sites, identification of SH RNAs, location of RNA degradation signals, identification of alternative splicing sites etc.Pattern search in biological sequences has numerous applications and hence a large amount of research

has been done to identify patterns. In Bioinformatics, it is common to search biological sequences (DNA, RNA, proteins) for functional motifs such as cross-over hotspot instigators (chi), restriction sites, regulation motifs, binding sites, active sites in proteins, etc. Sequence motifs are important tools in molecular biology. Sequence motifs can describe and identify features in DNA, RNA and protein sequences such as transcription factor binding sites, splice junctions and protein-protein interaction sites.

'Motif discovery' (or 'motif finding') in biological sequences can be defined as the problem of finding short similar sequence elements (building the 'motif') shared by a set of nucleotide or protein sequences with a common biological function. The identification of regulatory elements in nucleotide sequences, like transcription factor binding sites (TFBSs), has been one of the most widely studied flavors of the problem, both for its biological significance and for its bioinformatics hardness. Motifs are fundamental functional elements in proteins vital for understanding gene function, human disease, and may serve as therapeutic drug targets.

Regulatory motifs are used to control the expression of genes, dictating under which condition a gene will be turned on or off. Motif discovery typically involves the discovery of binding sites, conserved domains. Regulation of gene expression is an essential feature of all living organisms and viruses. It is vital for understanding gene Function, human disease, drug design. They are often helpful in finding transcriptional regulatory elements, transcription factor binding sites. Sequence motifs are becoming increasingly important in the analysis of gene regulation. Within a sequence or database of sequences, researchers search and find motifs using computer-based techniques of sequence analysis, such as BLAST. Such techniques belong to the discipline of bioinformatics.

1.2 Problem Statement

The motif finding algorithms of this paper differ from one another in three ways of defining and representing motifs, making hash table approach, the fitness function for calculating motif significance and the search techniques used to find the optimal motifs. Numerous algorithms have been developed for discovering motifs, as well as algorithms for scanning databases for matches to a given motif or motifs. Some are specialized for discovery of DNA motifs. The main problem of large amount of data is the inclusion of noisy and irrelevant data in the information set. As the datasets become large the number of noisy, redundant and uninformative gene also increases resulting in space-time complexity. We mainly focused on finding the motifs accurately. So we work with on an efficient algorithm for motif discovery based on linear PSO, linear PSO(binary search), hash table approach and variable length motif ranking considering mutation evaluating them with fitness function.

1.3 Research Challenges

With the large amount of biological data generated due to DNA sequencing of various organisms, it is becoming necessary to identify techniques that can help in finding useful information amongst all the data. Finding motifs involves determining meaningful short sequences that may be repeated over many sequences in various species. Various approaches for the motif discovery problem have been proposed in the literature. Computational methods for identifying and modeling DNA sequence motifs and regulatory elements have been developed over the past 25 years. These methods either use DNA patterns or position weight matrices to model target DNA-binding sites. Some of these methods have been used successfully to discover the cis-regulatory elements in genes that are thought to be regulated by a common transcription mechanism. Orthogonal information from comparative genomics or co-regulation at the expression level have been incorporated into these methods to identify cis-regulatory sites. Because many of the regulatory elements are functional in vivo only in certain temporal or spatial contexts and require other nearby sites that together function as cis-regulatory modules, methods have also been developed to address composite regulatory elements or regulatory modules that consist of DNA sites bound by multiple regulatory factors. There are now many programs available for DNA motif discovery and putative regulatory element identification, and several have web interfaces for easy use. Bioinformatics is such a research area that is the interface between the biological and computational sciences. Identification of all of the functional elements in genomes, including genes and regulatory elements, is a fundamental challenge now that the complete genomic sequences of a number of prokaryotes and eukaryotes are available. The ultimate goal of bioinformatics is to uncover the wealth of biological information hidden in the mass of data and obtain a clearer insight into the fundamental biology of organisms. This new knowledge could have profound impacts on fields as varied as human health, agriculture, the environment, energy and biotechnology. The identification of DNA motifs remains an active challenge for the researchers in the bioinformatics domain. In recent years a considerable number of algorithms have been designed for identifying regulatory elements in DNA sequence.

1.4 Motivation

In recent decades, scientists have extracted genetic sequences—DNA, RNA, and protein sequences—from numerous organisms. These sequences hold the information for the construction and functioning of these organisms, but as yet we are mostly unable to read them. It has long been known that these sequences contain many kinds of “motifs”, i.e. re-occurring patterns, associated with specific biological functions. Thus, much research has been devoted to computer algorithms for automatically discovering subtle, recurring motifs in sequences. However, previous algorithms search for rigid motifs whose instances vary only by substitutions, and by insertions or deletions. Real motifs are flexible, and not only do vary by insertions and deletions but also on other things. This study describes a new computer algorithm for discovering motifs, which allows for arbitrary insertions and deletions. This algorithm can discover real, flexible motifs, and should be able to help us determine the functions of many biological molecules. Motif discovery is one of the classical sequence analysis problems but still has not been satisfactorily solved in an exact and efficient manner. Motifs are important patterns that are helpful in finding transcriptional regulatory elements, transcription factor binding site s, function al genomics, drug design, etc. As a result, numerous papers have been written to solve the motif search problem. Our motivation was finding an improved approach for motif discovery.

1.5 Scopes

Biology is encoded in molecular sequences: deciphering this encoding remains a grand scientific challenge. Functional regions of DNA, RNA, and protein sequences often exhibit characteristic but subtle motifs; thus, computational discovery of motifs in sequences is a fundamental and much-studied problem. This study aims at giving a new approach for better motif finding approach. However, most current algorithms do not allow for insertions or deletions (indels) within motifs, and the few that do have other limitations. But in our approach we try to solve that by using PSO. But still lot of work is needed for improving our algorithm. For that purpose the door for implementation is open for all interested ones. There can be work done like- for reducing the time complexity, giving scores to the alignments, for overcoming the generalization problem and others.

1.6 Thesis Outline

In Chapter 1 we have talked about the introduction of our study in a précised manner. Chapter 2 deals with the basic motif finding method and some highlighted evolutionary approaches with a brief discussion about “motif” discovery and PSO method, a modified algorithm for variable length motif discovery. Chapter 3 will be discussed about our proposed algorithm and some elaborate discussion. Here we present a method “Species Specific Motif Discovery Using HAH table and Variable Length Motif ranking considering mutation” for discovering motifs allowing simple comparison among sequences from databases in a fully general manner. Chapter 4 will consist of the experimental analysis and result comparisons.

Chapter 2

Literature Reviews

This chapter is divided into two categories. In the first part of this chapter, we will describe the fundamentals of Motif Detection. It gives the basics idea about the motif definition, detection and the decisions made latter part of this book. Information about motif detection and other relevant topics are mostly taken from various research articles . In the last section, a short description about the different properties of the Motif finding methods will be provided.

2.1Motif

To classify and identify the features of DNA, RNA, Protein sequences in molecular biology, motif plays an important role. It has the capabilities to describe and find out the special characteristics of DNA, RNA, Protein sequences. Sequence motifs are becoming important in the analysis of gene regulation day by day. Actually Motifs are short repeating pattern in DNA, RNA, Protein sequences in molecular biologythat is conserved during the process of evolution. Motifs are short, recurring patterns in DNA that are presumed to have a biological function.Often they indicate sequence-specific binding sites for proteins such as nucleases and transcription factors (TF). Others are involved in important processes at the RNA level, including ribosome binding, mRNA processing (splicing, editing) and transcription termination. Basically it has a great impact on biological science. It resides in a DNA, RNA or Protein sequences and plays an important role in molecular biology.

Some common features of motif has been invented to identify it in the long sequences of DNA, RNA or Protein. Some important features that are helpful to identify motif or know about motif are given below:

- Motifs are patterns are of length 5 to 20 bases and are repeated overmany sequences.
- They are small, have constant size, and are repeated very often.
- They are statistically over-represented in regulatory regions.

Here we have given an example of motif in DNA sequences to understand the characteristic of motif. From the example we can have a brief idea about the short recurring sequence Motif. In

the following example, we can see a conserved short sequence (Motif) in few DNA Sequences. From the example we can see that the pattern **ATGCAACT** is unchanged and conserved. This short DNA sequence ATGCAACT is called motif.

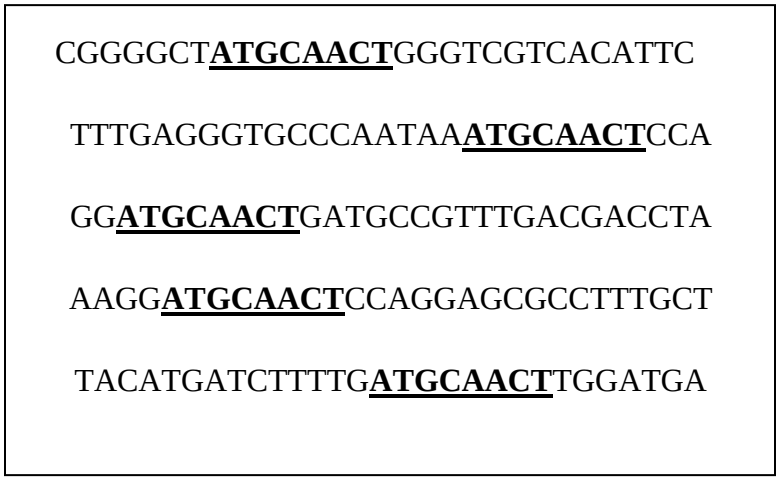


Figure 1 shows the typical representation for a motif

2.2 Overview of Motif Discovery

Motif discovery is a process of finding a meaningful pattern for DNA, RNA and protein sequence that is commonly shared by two or more human and animal molecules. Motif sequences are patterns found in biological sequences (DNA sequences, Protein sequences) that are important for understanding gene function, human disease, drug design, etc. It is helpful to find out and know the transcriptional regulatory elements, transcription factor binding sites, and so on. Basically it plays a great role to find out the biological function in human evolution. However, to find motif sequences is a very challenging task for the people of this sector.

Identification of motifs is becoming very important because they represent conserved sequences which can be biologically meaningful. Some of the areas where motif discovery can be useful include finding binding sites in amino acids, finding regulatory information within either DNA or RNA sequences, searching for splicing information and protein domains. The motifs can represent patterns which activate or inhibit the transcription process and are responsible for regulating gene expression. Motif identification can be thought of as finding the best local multiple alignments for the sequences under consideration. So, motif discovery is a challenging field for the people of bioinformatics. But motif discovery is very interesting and important in biological science though it is challenging task. Motif discovery can be used to categorize unknown DNA sequences into their corresponding families which is very important in biological science. If we can find out the similar family of DNA sequences it is easy to know

the biological function of them. Motif discovery also helps to classify the features of DNA, RNA, Protein sequences. From the motif we can know the characteristics of DNA, RNA, and Protein. Motifs represent regions that have been conserved through evolution. So, those regions are likely to be important for the function of the protein (e.g. an active site).Motifs can be used to classify proteins into families based on their functions, or predict the function of a new protein. It is very important to know the functions of protein in biological functions. For this reason motif finding is very important feature today for its demand. The motifs can represent patterns which activate or inhibit the transcription process and are responsible for regulating gene expression. Not only that but also motif is helpful in finding transcriptional regulatory elements, transcription factor binding sites. For these reasons researchers are very much interested to find out the best algorithm to detect the short conserved sequence Motif.

They are taking the challenges to find out the Motif sequences which have great impact on biological function. Motif identification can be thought of as finding the best local multiple alignments for the sequences under consideration. But there are some causes for what it is becoming challenging and difficult task for the bioinformatics people.

The challenges present in motif identification include:

- We do not know the actual length of Motif sequence. So, it's difficult to find out motif of unknown length.
- The motifs are never exactly the same as the actualconserved sequence. There is always a lot of sequence variability present with respect to a single motif.
- Motifs are very short signals as compared to the size of the DNA sequence under consideration.
- The regulatory sequences containing the motifs may sometimes be located very far away from coding regions that they regulate. This makes it difficult to determine the portion of the DNA sequence that should be analyzed.
- The regulatory sequences may, at times, be present on the opposite strand from the coding sequence they regulate.

For these few topics today motif finding is still one of the hardest task and becoming the challenging field for the people of this field. They are trying hard to find the path to overcome the given problem and find out the optimal path for detecting the motif from the long sequences of DNA, RNA and Protein.

2.2.1 Types of Motif Finding Approach

Three versions of the motif search problem have been identified by researchers:

- Simple Motif Search (SMS),
- Planted Motif Search (PMS) – also known as (l, d)- motif search and
- Edit-distance-based Motif Search (EMS).

PMS problem takes as input n sequences of length m each and two integers l and d. The problem is to identify a string M of length l such that M occurs in each of the n sequences with a Hamming distance of at most d.

For example, if the input sequences are GCGCGAT, CACGTGA, and CGGTGCC; l =3 and d=1, then GGT is a motif of interest. EMS is the same as PMS, except that edit distance is used instead of the Hamming distance. SMS takes as input n sequences and an integer l .The problem is to identify all the patterns of length l (with up to l /2 wild card characters), together with a count of how many times each pattern occurs.

2.3 Modeling and representation of DNA motifs

2.3.1 Representation of DNA motifs with sequence patterns

Motifs in DNA sequences can be modeled by sequence patterns, which are simply 'words' in the four-letter alphabet (A, C, G, T) of DNA. In order to capture the information about degeneracy, where more than one base may occur in a specific position of a binding site, the International Union of Pure and Applied Chemistry (IUPAC) nucleic acid codes are used (see Fig. 2). Once a sequence pattern for a binding site is determined, there are very fast methods for finding all occurrences of that pattern in a DNA sequence. The limitation of sequence patterns is that many binding sites do not match the pattern exactly, although they are usually very similar to it. Fig. 2shows an alignment of binding sites for the Pho4 TF from *Saccharomyces cerevisiae*, taken from the Promoter Database of *Saccharomyces cerevisiae*(SCPD) , and shows that at most positions more than one base is tolerated, even though there is always a predominant, or preferred, base. A consensus sequence, which indicates the most common base at each position, is easily determined for this site: ATTCGCACGTGG. However, as the occurrence matrix (see Fig. 3) shows, only positions 6–9 always match the consensus sequence, and every other position tolerates variants; in a larger sample size of functional sites, even the apparently conserved positions may show variations. A search for the consensus sequence would miss all

of the known sites.If we were to encode every position with the IUPAC degeneracy letter from the observed sites; we would get the pattern HBHHNCACGYKN (invariant positions in bold). A search of the genome with this pattern would find all of the known sites, but would also produce a large excess of nonfunctional sites that match the pattern. For example, the sequence CGATACACGCTA matches the degenerate IUPAC pattern but has almost no similarity to the consensus sequence, matching only the four conserved positions, and is unlikely to be a binding site. Alternatively, we could search using the consensus sequence allowing one mismatch, but we would miss 13 of 14 known sites that way. The known binding sites have between one and five mismatches to the consensus sequence, so to find all of them in the genome one would have to allow up to five mismatches. However, there are more than 200 000 different sequences that match the consensus allowing for five mismatches, the vast majority of which are unlikely to be true binding sites.

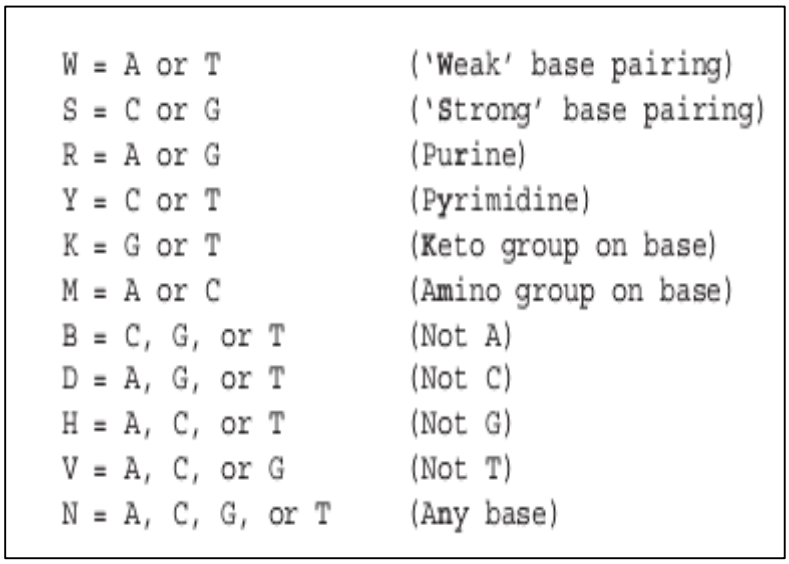


Figure 2 IUPAC nucleotide codes for representation of degenerate DNA sequence patterns

.

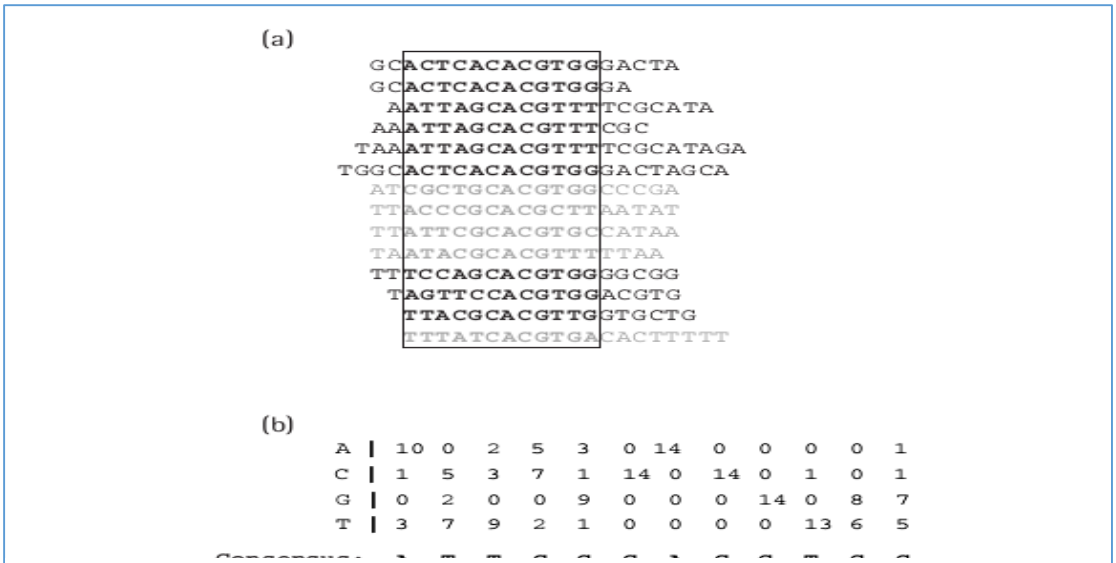


Figure 3 Alignment and analysis of yeast pho4-binding sites

a) Alignment of the 14 S. cerevisiaepho4-binding sites generated by the consensusprogram reveals a 12 nt. representative DNA motif. The binding site is boxed; sequences shown in gray are on the opposite strand of the DNA and have been reverse-complemented in this figure.
b) The table shows the frequency of each base at each of the 12 positions in the motif. Beneath it are shown the consensus sequence and the IUPAC representation, which shows all degeneracy in the sequence.

2.3.2 Representation with weight matrices

A more general representation of TF-binding sites is a position weight matrix (PWM) (alternative names include weight matrix, position-specific scoring matrix or PSSM, specificity matrix, and others. In this representation (see Fig. 4), an l-long sequence motif is represented by a 4 × lmatrix where each possible base, at each position in the binding sites, is assigned a score. The score of any specific sequence is just the sum of the position scores from the weight matrix corresponding to that sequence. For example, the score of the consensus sequence, is the sum of the bold numbers in the matrix (see Fig. 4d). Any other sequence can be scored similarly, so that an entire genome can be scanned by a matrix and the score at every position obtained.

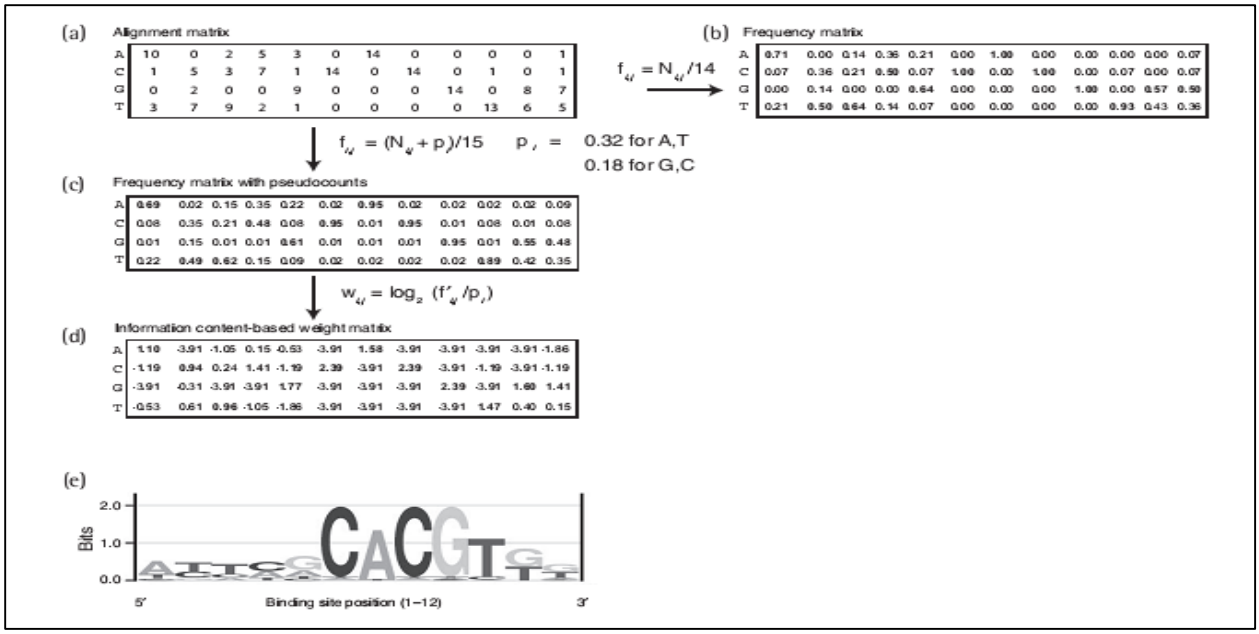


Figure 4 Matrix representations of the 14 pho4-binding sites in Fig. 3

2.4 Related Works for Motif Finding

Day by day Motif Detection is becoming a challenging and interesting topic to the concerned people of this area. Many researchers are working to develop the methods to find the short sequence from the DNA or Protein sequences. Motif discovery is a very important problem in biology. It finds applications in DNA or protein sequence analysis, comprehending disease susceptibility and disease cure. A considerable effort in this area was concentrated on understanding the evolution of the genome by identifying the DNA binding sites for transcription factors. So, now-a-days we can see a number of existing methods to get the motif in the effective way. In this study, we present some of the computational methods that exist and try to evaluate their performance in case of long sequences. Some computational algorithms along with some Graph-based algorithms have been found to identify motif. Here, we will discuss some of the methods that are needed to develop our proposed method.

2.4.1 Motif discovery using linear-PSO with linear search

Lailee et al [1] used a modified motif search algorithm based on a population based stochastic optimization technique called Linear Particle Swarm Optimization (PSO).

For this study PSO was modified for discovering motif. The modified Linear-PSO is chosen even though it is slower because linear search is not a choice but a necessary criteria for identifying motif for specific species. Linear-PSO is used linear selection and linear search in order to compare all possible motifs existing in DNA sequences for representing specific species. Particle Swarm Optimization algorithm is an evolutionary optimization technique by Kennedy &Eberhart. In Linear PSO they use linear number for population initializing and next position updating.

Main Objective:
Search a species specific motif existed in each DNA sequence from the same species.

Process:
First a DNA set is selected as Target set. Then target motif(s) is extracted from the Target set .Then target motif is compared with other DNA sequences.

Target motif is the possible combination of bases from the 'Target Set' of pre specified length.

Number of target motif , $t = (n – t_m) + 1$

where n = length of target DNA sequence

t_m = length of target motif

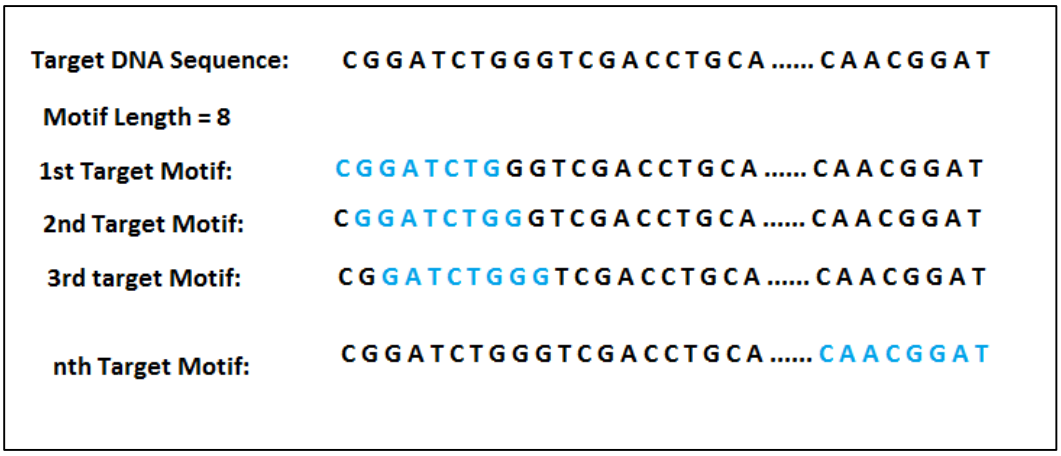


Figure 5 Target motif finding

Fitness calculation:

Fitness function is used to maximize the similarity of the sequence of the motif found while avoiding low complexity solution .In case of exact match for each base count 1 point and for exact match count an additional point (total point multiply by 2).

2.4.2 Motif discovery using linear-PSO with binary search

Lailee et al [4] used linear-PSO for binary search. Searching is one of the most basic operations performed in a computer. Searching is widely used in fields such as database management, compiler construction and natural language processing. Searching in bioinformatics is important because there is a need to identify pattern in DNA sequences as quickly as possible. Therefore, speed is one of the most important criteria when determining the quality of a search process. The most popular method of searching is Binary Search.The Linear-PSO algorithm [2] which uses linear search requires more time compared to the original PSO. Even though the speed of Linear-PSO is slow, the validity and accuracy of the result is high because all possible combinations of motifs are tested.

However, linear search slows down the algorithm, when the speed is one of the most important factors under consideration. In this paper we present an improved version of Linear -PSO that is integrated with a Binary Search.The first DNA sequence was selected as a Reference Set and other DNA sequences will be used to compare for similarity of motif among different animals from the same species as shown in Fig 4. Target motif is found as like as in the linear search algorithm. From the 1st sequence target motif is selected and compare with the other sequences. In this study, length of motif is set to 6 bases and without any mutation (6, 0).Linear-PSO with Binary Search is an improved version of Linear-PSO. There were two improvements made to the Linear-PSO algorithm. First, the preprocessed data needed to be sorted before applying this algorithm. Second, for similarity searching, Binary Search is used instead of Linear Search.

Figure 6 Target Motif representation

2.4.3 Motif discovery using De Bruijn Graph

Hong Zhou et al [5] used De Bruijn graph for motif discovery.Graph theory is playing an important role in computational biology. Graph-based algorithms provide a simpler and quicker solution to computationally intensive problems such as DNA fragment assembly and motif discovery.

This paper focuses on developing a time-efficient algorithm for motif discovery using a de Bruijn graph. We have found that the algorithm was successful in mining signals for larger number of sequences and at a faster rate when compared to some popular motif searching tools such as MEME.

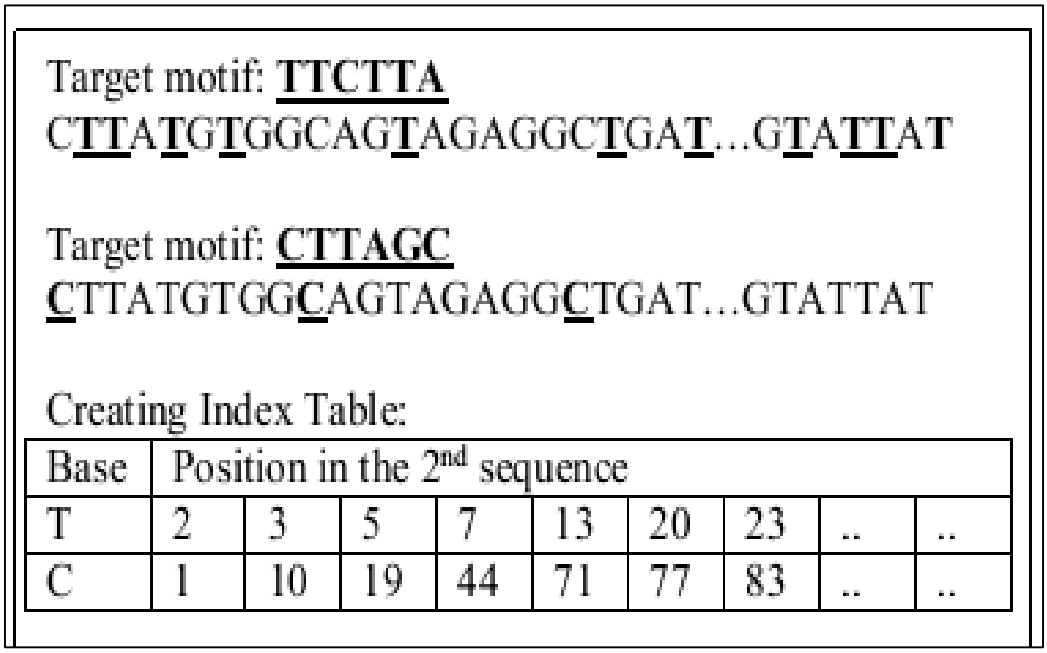
First a deBruijn graph is constructed. Then cycles are removed and transformed it into a directed acyclic graph.Then a path on the directed graph is computed and later on the conserved regions are extracted.

2.4.4 A Modified Algorithm for Variable Length DNA Motif Discovery

Samiul Islam et al [11] used variable length motif discovery, introducing index table concept in motif searching, In their paper they model the solution by following steps: Extract target motif, compare with other DNA sequence, calculating the target motif using fitness function, find the best motif using fitness value.A DNA sequence is selected as a target set and motifs(particles) are generated from the target set of a specific length which are known as ‘target motif’.

Figure 7 Target Motif

For searching purpose they have made a index table where the positions of the first character of the target motif existed in other DNA sequences will be indexed.



Base	Position in the 2nd sequence								
T	2	3	5	7	13	20	23	..	..
C	1	10	19	44	71	77	83	..	..

Figure 8 Hash Table

Directly search those positions of the DNA sequence for further comparison and compare the characters sequentially.

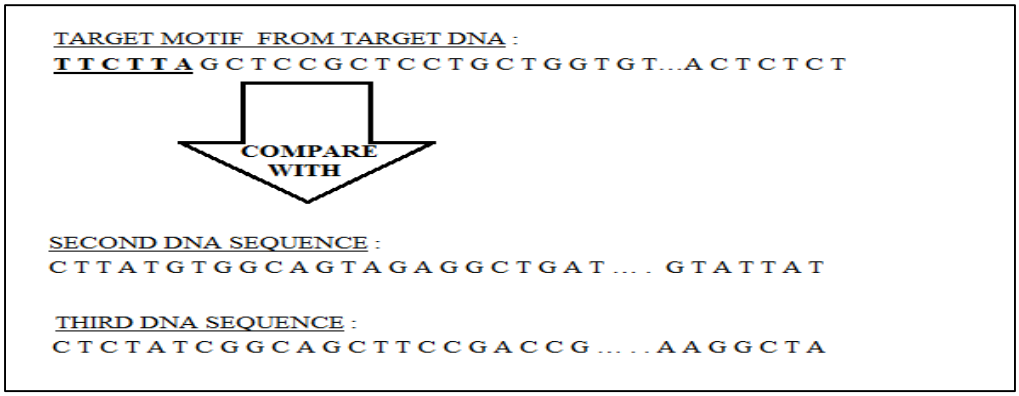


Figure 9 Comparing DNA sequence

For Fitness calculation and Ranking, maximize the similarity of the sequence motif avoid low complexity solutions. For example 'TTTTT' has lower complexity than 'ATTTA'. For exact matching, Fitness value = motif length*2. For one mutation, Fitness value = motif length-1. For ranking highest fitness value is considered.

Chapter 3

Proposed Method

3.1 Overall Concept

PSO algorithm starts with the random initialization of a population of individual particle in the search space. Each particle goes through the fitness calculation. However, previous researchers who used methods based on PSO only focus on motif detection in individual DNA sequence that permit randomization of selected population.In our method we propose to select variable length target motif where as previously some researchers selected constant length target motifs. For that reason the output motif sequence may not be the best motif sequence because there can be other motif sequence of different length whose fitness value might be higher than the previous. In our proposed method we focus on not only finding the exact motif sequence and also the mutation.

In our thesis work, we are going to propose a new searching algorithm based on the PSO where for each cycle only one particle called 'target motif' are selected and compared with the DNA sequences for fitness calculation.

The selection of target motif in our proposed method plays an important role to get the exact and accurate motif sequence. Here, we consider the variable length to choose the target motif from the selected DNA sequence which will to correctly identify the motif. Because motif length is an important fact in finding motif sequences. So, we consider the variable length to calculate the fitness value and from the fitness values we can identify the best suited motif. In our method we have shown that when the target motif length is increasing it's tough to get the perfect match. But here we can see that for which length of target motif we are getting the motif frequently and for which length we are getting the highest fitness value.

We can see the experimental result from our next result section from where we can get the idea of our proposed variable length target motif.

Secondly we develop a new searching algorithm to compare the target motif with other DNA sequences in effective way. Basically we introduce the 'hash table' in searching process which brings the flexibility and simplicity. Each sequence is converted to a corresponding index hash table. So basically instead of string matching we are trying for integer matching which is faster

in all sense. It's also helpful for time consuming. It reduces the redundancy which we face in the linear search. Actually linear search is a time consuming and lengthy process for long DNA sequences. So, we are planning to improve the method by effectively the hash table concept to compare the long DNA sequences in an effective way.

3.2 Proposed Method

The information in DNA is stored as a code made up of four chemical bases A, T, C, and G. In our proposed method the main focus is on finding the best suited motif sequence from a set of DNA sequences. To do that we assume the chemical bases as characters and the DNA sequences as strings. Initially we will select some random DNA sequences of same species. But finally we will choose a specific dataset (GenBank:SusScrofa) for DNA sequences.

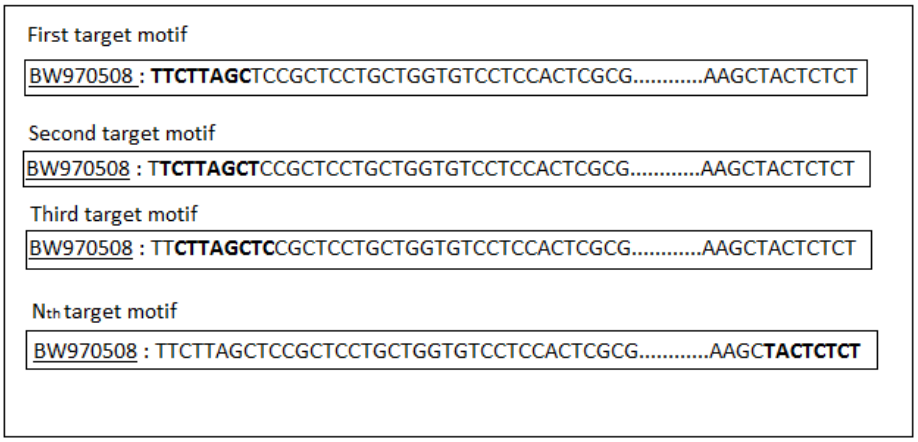


Figure 10 Target motif finding

From those DNA sequences first we have selected a random DNA sequence as target DNA sequence and from it we will extract the target motifs. For the every DNA sequences we have made the hash table. Below we have shown our reference genome and 2nd DNA sequence and its hash table:

Reference Genome:

A	A	T	C	G	T	C	A	T	G	C	T	A	T	G	A	A	T	G	C	C	A	T	G	C	G	C	A	T	T
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

2nd DNA sequence:

A	A	T	C	G	T	A	A	T	C	C	T	A	A	G	A	A	T	C	G	G	A	T	G	G	G	C	G	T	T
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

2nd DNA sequence hash table:

A:	1	2	8	13	14	16	17	22
T:	3	6	9	12	18	23	29	30
G:	5	15	20	21	24	25	26	28
C:	4	7	10	11	19	27		

Figure 11 Hash Table for 2nd DNA sequence

Our target motif is AATCGT and we are going to the 2nd DNA sequence hash table and looking for A's position and we have found it in 1, 2, 8, 13, 14, 16, 17, 22 index position. Now our second letter is also A it must be in next of 1st A position. And that's why 8 and 22 indices are omitted.

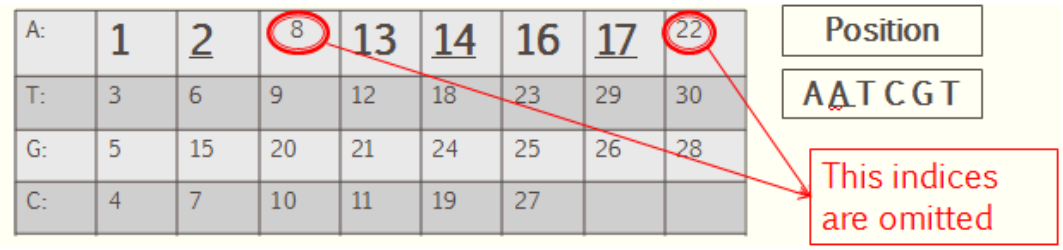


Figure 12 Matching index

Now, our next letter is T and its position is 3, 6, 9, 12, 18, 23, 29, 30. And we need to check whether it's in the next position of the second A's index. And we have found it in the 3rd and 18th positon.

Figure 13 Matching index

Similarly we need to check for C's and G's position.

A:	1	2	8	13	14	16	17	22
T:	3	6	9	12	18	23	29	30
G:	5	15	20	21	24	25	26	28
C:	4	7	10	11	19	27		

AATCGT

15

Figure 14 Matching index

A:	1	2	8	13	14	16	17	22
T:	3	6	9	12	18	23	29	30
G:	5	15	20	21	24	25	26	28
C:	4	7	10	11	19	27		

AATCGT

Figure 15 Matching index

But now we have to check T's index which position must be 21 but here in 21 position its G's index so this doesn't match and we can say that it's 1 letter mutation.

A:	1	2	8	13	14	16	17	22
T:	3	6	9	12	18	23	29	30
G:	5	15	20	21	24	25	26	28
C:	4	7	10	11	19	27		

AATCGT

This one should be in T's index.

This indicates possible mutation

Figure 16 Matching index

For target motif: A=1, A =2, T=3, C=4, G=5, T=6

For 2nd DNA sequence if we take A's position as i then 2nd A's position must be i+1. This sequence will be continued up to six characters and if all matches then it is (6,0) potential motif and if one mismatch then it is (5,1) potential mutated motif.

Index	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
Target Motif (index)	A = 1	A = 2	T = 3	C = 4	G = 5	T = 6
2nd DNA Sequence hash table (index)	A = 1	A = 2	T = 3	C = 4	G = 5	T = 6
2nd DNA Sequence hash table (index)	A = 16	A = 17	T = 18	C = 19	G = 20	G = 21

Now, we can say that it is a potential motif.

17

Figure 17 Finding potential motif

Now we need to do this process repeatedly to get the potential motif and below we have shown our flow chart:

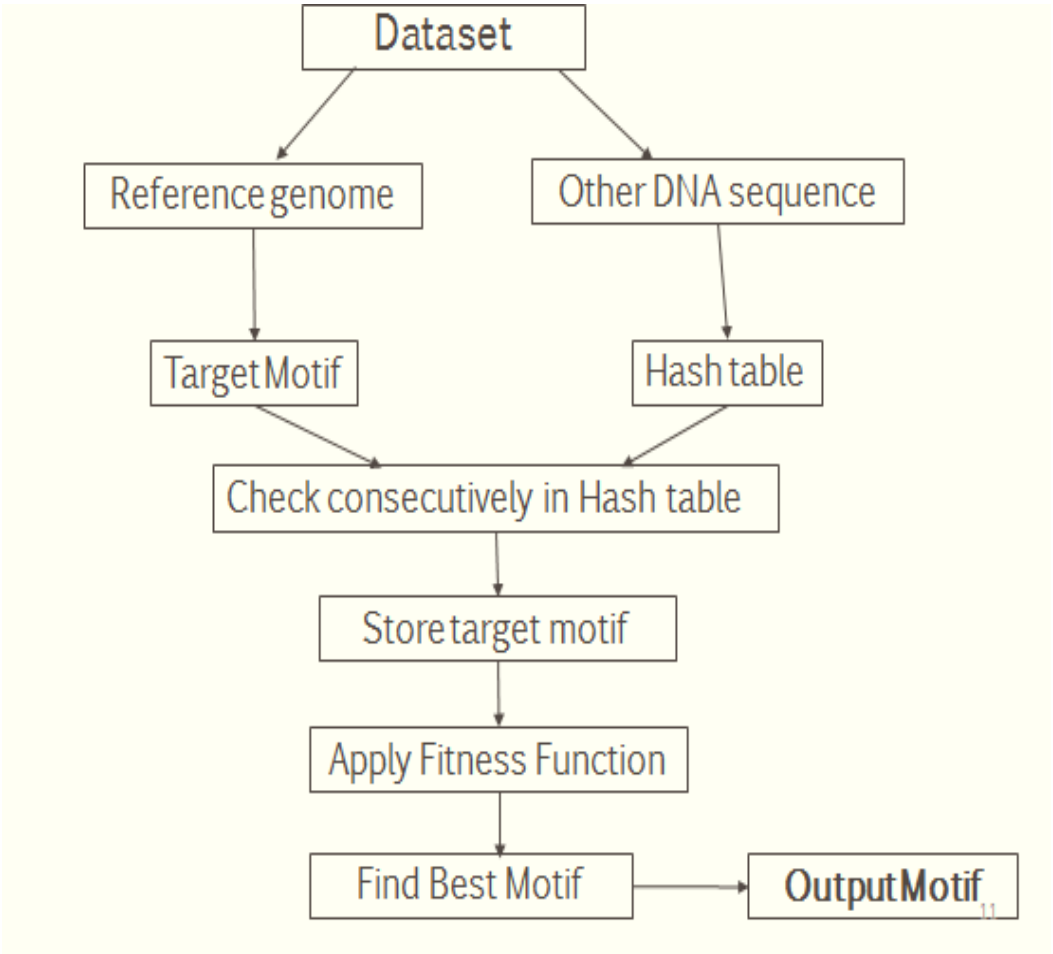


Figure 18 our proposed motif discovery flow chart

3.3 Proposed Algorithm

Algorithm: MOTIF _FINDER

1. Select the database for finding motifs, constraint is the database files must be DNA sequences of the same species.
2. Select randomly any one of the DNA sequences to be the reference DNA from where we will extract the motifs.
3. Make hash table consisting of 4 rows for the 4 letters A,T,G & C for the remaining DNA sequences.
4. Extract motifs of length 5 to 20 from the reference DNA sequence.
5. Take 1 motif at a time and begin checking.
    I. For each target motif check indices of corresponding characters in hash table of next sequences.
    II. Consider the mutation flag set to 1. [1 mutation allowed in each motif]
        A. If corresponding indices match output it as a motif.
        B. If corresponding indices match with 1 mismatch maximum, output it as a mutated motif.
        C. Else go to step 5.
    III. Check for 1st letter mutation.
        A. Check for strings of size 1 less than the target motif length from the hash table of other sequences.
        B. If all corresponding indices match check its left position by comparing the character at the leftmost of the motif and corresponding index from the hash table.
        C. If the character does not match output it as a possible 1 character mutated motif.
        D. Else go to step IV.

IV. Calculate the fitness value according to the fitness function.

V. Increase motif length.

VI. Go back to step 4 and repeat the process until motif length is greater than 20.

6. Make a ranking table from the fitness values of target motifs.

Choose the high ranked motif as output.

3.4 Fitness Calculation:

Similarity and complexity are the two main factor required to evaluate a motif [9]. Along with the necessity of being the two sequences to be similar, their complexity also needs to be addressed. For example 'TTTTT' has lower complexity than 'ATTTA'. The objective of the fitness function is to maximize the similarity of the sequence motif as well as avoiding low complexity solutions.

3.4.1 Fitness Function [12] :

Fitness function used in this experiment is adapted from Sarwar Topon, Dianhui Wang [12] that we intend to use in our algorithm:

$$r(k,m) = \frac{d(K,M)}{d(K,Mref) + c(K)}$$

r(K,M) is used to find the potential motif.

$$d(k,m) = 1 - 1/k\left[\sum_{i=1}^{k} \sum_{\forall bi \epsilon x} \mathrm{f(bi,i)}k(bi,i)\right] \ldots\ldots\ldots\ldots 0 \leq \mathrm{d(k,m)} \leq 1$$

d(K,M) is generalized hamming distance, where $f(b_i,i)$ and $k(b_i,i)$ are the observed frequencies of base b_i at position i in K and M.

$$c(k) = 4/3[1 - \frac{1}{k^2\left[\sum_{\forall bi \epsilon x}\left(\sum_{i=1}^{k} k(bi,i)\right)^2\right]}] \ldots\ldots\ldots\ldots 0 \leq c(k) \leq 1$$

c(K) is used to find the composition complexity of K where the complexity is scored according to the distribution of bases (A, C, G, T) in the K.

Example:

Let's say K = AATCTC M = AATGTC Mref = ACTATC

d(K,M) = $\frac{1}{6}$ d(K,Mref) = $\frac{2}{6}$ c(K) = 0.72

- r(K,M) = 0.15822

As d(K,M) > d(K,Mref) so it is a potential motif of r(K,M) = 0.60096

Let's say K = TTTTTT M = AATGTC Mref = ACTATC

d(K,M) = $\frac{4}{6}$ d(K,Mref) = $\frac{4}{6}$ c(K) = 0.0

- r(K,M) = 1.0

So the lowest r(K,M) is their desired motif.

3.4.2 Fitness Function [13]

Total_Fitness (P) = w_1*L + w_2*$\sum$ FS(Sm, Pn)……[13]

Here,

w1 = weight given to the length and

w2 = weight given to similarity.

If S = AATCTC , P = AATCTC

w1 = w2 = 1 [we assume]

L =length of the motif = 6

Then $\sum$ FS(Sm, Pn) means how many times P has been found in S (DNA sequences).

Example :

If it matches exactly then 1 otherwise 0. Here for this example:

- Total_Fitness(P) = w_1*L + w_2* ∑FS(Sm, Pn)
- Let's say P has been found in S for 19 times that means:
- Total_fitness(P) = 1*6+1*19 = 25 [w_1=w_2=1 and ∑FS(Sm,Pn) =19]
- Here for low complexity like "TTTTTT" it will show the highest value but actually it is not a motif.

3.4.3 Proposed Fitness Function:

$$\textbf{Fitness\_Score} = w_1 * L + w_2 * r(K,M)$$

r(k,m) will now calculate the similarity

Let's say K = AATCTC M = AATGTC Mref = ACTATC

d(K,M) = $\frac{5}{6}$ d(K,Mref) = $\frac{4}{6}$ c(K) = 0.72

- r(K,M) = 0.60096

As d(K,M) > d(K,Mref) so it is a potential motif of r(K,M) = 0.60096

Fitness Score = 1*6+1*(0.60096+0.72) = 7.32096

Let's say K = TTTTTT M = AATGTC Mref = ACTATC

d(K,M) = $\frac{2}{6}$ d(K,Mref) = $\frac{2}{6}$ c(K) = 0.0

- r(K,M) = 1.0

Fitness_Score = 1 * 6 + 1 * 1.0 = 7

Now, it will give lower value for lower complexity.

So in this example* AATCTC *is the potential motif.

3.5 Experimental Data and Results:

In this paper, we have used *Sus Scrofa* , *Mus musculus*, *Homo sapiens*, *Aedes aegypti* , *Felis catus* dataset. The data was taken from GenBank database. The lengths of selected sequences are from 829 based to 850 bases. Table shows the dataset we have used.

3.5.1 Datasets:

The datasets that we used are as follows (accession numbers specified)

SusScrofa	***Musmusculus***	***Feliscatus***	***Homo sapiens***	***Aedesaegypti***
BW970508	JK707008.1	KJ933256	NM_001166002.2	KJ736826
BW971304	JK707007.1	KJ933223	NM_001166004.2	KJ736827
BW972295	JK707011.1	KJ933191	NM_001166003.2	KJ736825
BW973679	JK707010.1	KJ933160	NM_181773.4	KJ736824

Table 1: Dataset

3.4.2 Results:

We have considered variable length motif discovery. As mentioned earlier, any of the DNA sequences can be chosen for generating the target motifs. For our experiment, we have selected the first DNA sequence to generate target motif. When we have considered the motif length as six without mutation (6, 0), the total number of potential motif presents in the DNA sequence is 824 ((829 – 6) +1). The set of target motifs then compared with all possible combinations of motifs from 2^{nd}, 3^{rd} and 4^{th} DNA sequences with the help of index table generated for respective DNA sequence. The same procedure has been repeated while the motif length are (7, 0) and (8, 0). Only one iteration is required in this process to discover any unknown motifs. While searching for a motif, the fitness function assigns points for each of the potential motifs for each

of the other DNA sequences. The motifs are ranked based on the fitness values acquired while comparing with other DNA sequences.We have also considered the possibility of mutation in motifs. We have allowed our calculation to consider one or more mutation (i.e. (6, 1,(6,2)) for motifs of length six and one or two or no mutation) while comparing motifs with other sequences. Below we have given our target motif results for 2^{nd}, 3^{rd} and 4^{th} sequence.

Sequence Number	Motif Length	Target Motifs	Total Fitness	r(k) value	d(k, m) value	c(k) value
2nd Sequence	6,0	TGTGGC	27.7776	03.7776	01.3332	02.4444
		GTGGCT	27.7776	03.7776	01.3332	02.4444
		AAAAGG	27.1112	03.1112	01.3332	01.7776
	7,0	ATTTCAG	30.2448	02.2448	00.8572	01.3878
		GTGGCTG	29.7142	01.7142	00.5714	01.1428
		ATGTGTA	15.5102	01.5102	00.8571	00.6531
	8,0	ATTTCAGC	25.5938	01.5938	00.8750	00.7188
		ATGTGTAC	25.3438	01.3438	00.6250	00.7188
		TGTGGCAC	25.0938	01.0938	00.3750	00.7188
	6,1 and 6,2	GTGGCT	42.6666	06.6666	03.0000	03.6666
		AAAAGG	33.8890	03.8890	01.6665	02.2220
		CAAAAA	31.3890	01.3890	00.0000	01.3890
	7,1 and 7,2	GTGGCTG	76.4285	06.4285	03.5715	02.8570
		GTGTACA	30.6122	02.6122	01.1428	01.4694
		GTGGAGT	30.5714	02.5714	01.4286	01.1428
	8,1 and 8,2	GTGGAGTT	50.1876	02.1876	01.0000	01.1876
		CTGCCATG	25.4688	01.4688	00.7500	00.7188
		CGGTCCTG	25.2812	01.2812	00.6250	00.6562

Sequence Number	Motif Length	Target Motifs	Total Fitness	r(k) values	d(k, m) values	c(k) values
3rd Sequence	6,0	AAGAAG	108.4448	12.4448	05.3328	07.1104
		GTGGAG	43.0002	07.0002	04.0002	03.0000
		TGTGGA	42.6666	06.6666	03.0000	03.6666
	7,0	CAAGAAG	59.4284	03.4284	01.1428	02.2856
		AAGAAGG	44.7552	02.7552	01.2858	01.4694
		GAAGAAG	44.7552	02.7552	01.2858	01.4694
	8,0	GAAGAAGG	50.2500	02.2500	01.2500	01.0000
		AAGAAGGC	50.1876	02.1876	01.0000	01.1876
		CGCGGAGC	25.2188	01.2188	00.6250	00.5938
	6,1 and 6,2	AAGAAG	145.8324	19.8324	10.5000	09.3324
		TTGTGG	56.0000	08.0000	04.0000	04.0000
		TGTGGA	49.7777	07.7777	03.5000	04.2777
	7,1 and 7,2	CAAGGTG	61.0612	05.0612	02.2856	02.7756
		TGACACG	60.0816	04.0816	01.1428	02.9388
		CAAGAAG	60.0000	04.0000	01.7144	02.2856
	8,1 and 8,2	CAAGGTGG	50.5624	02.5624	01.2500	01.3124
		TGGACCGG	50.3124	02.3124	01.0000	01.3124

Sequence Number	Motif Length	Target Motifs	Total Fitness	r(k) values	d(k, m) values	c(k) values
4th sequence	6,0	CCTGCA	43.0002	07.0002	03.0000	04.0002
		TCATCA	28.0000	04.0000	01.3332	02.6668
		ACCAGG	28.0000	04.0000	01.3332	02.6668
	7,0	CATCATC	44.3877	02.3877	00.4287	01.9593
		GTGGCCA	30.8164	02.3877	01.4286	01.3878
		TTCAGCA	30.0408	02.0408	00.5714	01.4694
	8,0	GGCACCAT	25.3438	01.3438	00.6250	00.7188
		GTGGCCAT	25.3438	01.3438	00.6250	00.7188
		GCGGCGAA	25.2500	01.2500	00.6250	00.6250
	6,1 and 6,2	ATGCTG	70.5560	10.5560	03.3330	07.2220
		AAGAAG	47.4446	05.4446	02.3331	03.1108
		ACCAGG	43.9998	07.9998	04.0002	04.0002
	7,1 and 7,2	TGCCGCC	75.7145	05.7145	02.8570	02.8570
		CATCATC	44.8164	02.8164	00.8571	01.9593
		ATGCTGC	30.8980	02.8980	01.4286	01.4694
	8,1 and 8,2	CAAGAGTC	49.9376	01.9376	00.5000	01.4376
		ATGCTGCA	25.3750	01.3750	00.6250	00.7500
		CTGCAGTG	25.3438	01.3438	00.6250	00.7188

Table 2: Potential target motif(after fitness calculation)

3.4.3 Time Comparisons:

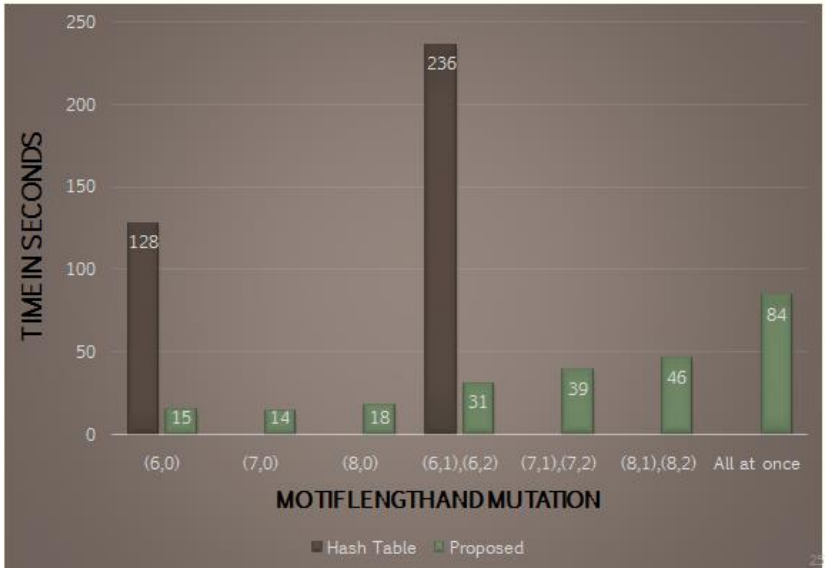


Figure 19 Time comparison between hash table and our proposed method

Here for hash table approach for (6,0) motif it took 128 seconds where for our proposed method it takes 15 seconds. For (7,0) motif it takes 14 seconds and for (8,0) motif it takes 18 seconds. For hash table approach (6,1) motif took 236 seconds where our method take only 31 seconds for both (6,1) and (6,2) motifs. And all at once it takes 84 seconds.

3.4.4 Result Verification:

For our result evaluation we used Fasta, Blast, CompareMotif(Online de novo motif finding tool) http://bioware.ucd.ie/~compass/biowareweb/Server_pages/comparimotif.php . For input it takes motif in one text file and DNA sequence in another text file then there is a field of variation in motif if there is 0.5 then it will consider 50 percent mutation from its parents and if it is 1 then it will show the exact match of the motif.For the following output inputs were

I. Compare motif	II. Proposed system
• Motifs file (motifs_length_6.txt) • 2nd DNA sequence from Sus scrofa dataset (2.txt, accession number BW971304) • IC value 1.0 (means only exact matching)	• Motifs file (motifs_length_6.txt) • 2nd DNA sequence from Sus scrofa dataset (2.txt, accession number BW971304) • Allowed Mutation is 2

Table 3 : Inputs for result verification

CompareMotif (Online de novo motif finding tool)	Proposed System

Sim1	Sim2	Match	Pos	MatchIC	NormIC	Score	Info1	Info2
ES	EP	TTCTTA	6	6.000	1.000	6.000	6.00	60.00
ES	EP	TCTTAG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	TCTTAG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CTTAGC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	TAGCTC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CTGCTG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CTGCTG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	TGGTGT	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CCTCCA	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GCTTTC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CTGCCG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	TGTAGC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GCTCTT	6	6.000	1.000	6.000	6.00	60.00
ES	EP	TTCAGC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GCAGCT	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CAGCTC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CCCAGC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CGTGGC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GGCTCG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GCTCGG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GCCTGG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GAGCTG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GGCAAG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	AGGTGG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	AAGTTC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	TTCAGC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	AGACCA	6	6.000	1.000	6.000	6.00	60.00
ES	EP	GAAAGC	6	6.000	1.000	6.000	6.00	60.00
ES	EP	CGTGAG	6	6.000	1.000	6.000	6.00	60.00
ES	EP	AGGGGG	6	6.000	1.000	6.000	6.00	60.00

Motif Number	Motif	Position
1	TTCTTA	1 2 3 4 5 6
2	TCTTAG	2 3 4 5 6 7
2	TCTTAG	250 251 252 253 254 255
3	CTTAGC	381 382 383 384 385 386
5	TAGCTC	575 576 577 578 579 580
16	CTGCTG	293 294 295 296 297 298
16	CTGCTG	555 556 557 558 559 560
19	CTGGTG	20 21 22 0 24 25
20	TGGTGT	512 513 514 515 516 517
26	CCTCCA	604 605 606 607 608 609
30	CACTCG	11 12 0 14 0 16
37	GCTTTC	346 347 348 349 350 351
45	GCTCTC	346 347 348 0 350 351
46	CTCTCG	465 466 467 468 0 470
70	CTGCCG	499 500 501 502 503 504
72	GCCGCC	711 712 0 714 0 716
96	TTGTAG	60 61 62 63 64 65
97	TGTAGC	573 574 575 576 577 578
110	GCAACC	600 0 602 603 604 605
116	ATGCCC	743 744 0 746 0 748
118	GCCCAA	136 0 138 139 140 141
118	GCCCAA	431 0 433 434 435 436
130	GCTCTT	248 249 250 251 252 253
130	GCTCTT	831 0 833 0 835 836
134	TTCAGC	342 343 344 345 346 347
141	GCAGCT	568 569 570 571 572 573
142	CAGCTC	246 247 248 249 250 251
145	CTCGCA	533 534 535 536 0 538
152	CAGGAC	332 333 334 335 0 337
154	GGACCT	469 470 0 472 0 474
162	CCCAGC	244 245 246 247 248 249
164	CAGCGC	623 624 625 626 0 628
169	CGTGGC	528 529 530 531 532 533
180	GGCTCG	531 532 533 534 535 536
181	GCTCGG	532 533 534 535 536 537

Figure 20 Result comparison

Chapter 4

Conclusion

4.1 Summary of the research

In our thesis, mainly we proposed a method to identify motif in an effective way. As motif detection is an important part of biological researches due its demand on biological science, we have tried our best to find out the most efficient way to detect motif in DNA sequences. Actually we have considered the length of the target motif very seriously and we build up our proposed algorithm according to the variable length of target motif. We mainly developed a new searching algorithm concept where we introduce hash table concept from the index of the target motif using linear search and we have used new fitness function to calculate the potential motif and also we consider the first letter mutation and mutation in any possible position.

Experimental results show that the proposed method identifies the motifs from Sus Scrofa,Homo Sapiens, Cat, Mus muculus dataset with similar accuracy as Linear-PSO and with greater efficiency. By incorporating Linear-PSO we have achieved the benefit of allowing all possible motifs in a sequence of variable length and along with it, the addition of the concept of index table made the pervious algorithm an improved method. As a result, the algorithm becomes faster and the validity of the result is also high.

4.2 Future works

We have completed our simulation work and have got better result than using the Linear-PSO method and modified algorithm of motif finding..

During the process of creating index table, we have created hash table for every DNA sequence and for every character we are going back to the hash table again and it has been taken less time then previous method. And the fitness function that we have used is more efficient than the previous one which is too trivial. And here we also considered the first letter mutation. So next time we will try to make the search faster though we have used exhaustive search here.

Chapter 5

5.1 References

1. Davila, J., Balla, S., Rajasekaran, S.: Fast and practical Algorithm for Panted (l, d) Motif Search. In: IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 4, pp. 544-552, IEEE Press (2007)
2. Pradhan, M.: Motif Discovery in Biological Sequences. Master's Projects (2008)
3. Kennedy, J., and Ebehart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, Perth, Australia (1995).
4. Chang, B., C., H., Ratnaweera, A., and Halagmuge, S., K.,: Particle Swarm Optimization for Protein Motif Discovery. In Genetic Programming and Evolvable Machine, vol. 5, pp. 203-214. (2004)
5. Akbari, R., and Ziarati, K.,: An Efficient PSO Algorithm for Motif Discovery in DNA. In IEEE International Conference of Emerging Trends in Computing, Tamil Nadu, India 2009.
6. Hardin, C., T., and Rouchka, E., C.: DNA Motif Detection Using Particle Swarm Optimization and Expectation-Maximization. In IEEE Symposium on Swarm Intelligence, 2005.
7. Zhou, W., Zhu, H., Liu, G., Huang, Y., Wang, Y., Han, D., and Zhou C.: A Novel Computational Based Method for Discovery of Sequence Motifs from Coexpressed Genes. In International Journal of Information Technology, vol. 11 (2005).
8. Lei, C., and Ruan, J.: A Particle Swarm Optimization Algorithm for Finding DNA Sequence. In IEEE International Conference on Bioinformatics and Biomedicine, Philadelphia, 2008.
9. Sharifa, L., S., A., Harun, H., and Taib, M., N.: A Modified Algorithm for Species Specific Motif Discovery. In International Conference on Science and Social Research (CSSR 2010), Kuala Lumpur, Malaysia, Dec 5-7, 2010.
10. Sharifa, L., S., A. and Harun, H.: Motif Discovery using Linear-PSO with binary Search. In AWERProcedia Information Technology & Computer Science. Pp 458 – 462. (2012)

11. CompariMotif: quick and easy comparisons of sequence motifs Richard J. Edwards1,2,*, Norman E. Davey1 and Denis C. Shields. Received on February 11, 2008; revised on March 18,2008; accepted on March 19, 2008.Advance Access publication March 28, 2008
12. Dianhui Wang, SarwarTapan. : MISCORE: a new scoring function for characterizing DNA regulatory motifs in promoter sequences.From 23rd International Conference on Genome Informatics (GIW 2012) Tainan, Taiwan. 12-14 December 2012.
13. ShripalVijayvargiya, Pratyoosh Shukla.: A Structured Evolutionary Algorithm for Identification of Transcription Factor Binding Sites in Unaligned DNA Sequences. International Journal of Advancements in Technology http://ijict.org/ ISSN 0976-4860
14. Matt Stine, DipankurDashgupta,SurajMukatira. : Motif Discovery in Upstream Sequences of Coordinately Expressed genes. sequences.From 20rd International Conference on Genome Informatics (GIW 2011) Tainan, Taiwan. 11-13 December 2011.

Chapter 6

6.0 Appendix

Appendix 6.1 - Textprocessor.java

```
Package text_processor;

Import java.io.BufferedReader;
Import java.io.Buffe redWriter;
Import java.io.DataInputStream;
Import java.io.File;
Import java.io.FileInputStream;
Import java.io.FileNotFoundException;
Import java.io.FileWriter;
Import java.io.IOException;
Import java.io.InputStreamReader;
Import java.io.PrintWriter;
Import java.text.DecimalFormat;

publicclassfinal_output{
publicfinal_output(String
input,intmotif_length,intsequence_no)throwsFileNotFoundException,IOException{
create_directory();
PrintWriter result =newPrintWriter(newBufferedWriter(new
FileWriter("final_output\\"+"final_output_sequence_no"+sequence_no+"_motif_le
ngth_"+motif_length+".txt",false)));
String[] motifs =new String[1500];
double[]total_fitness=newdouble[1500],dkm=newdouble[1500],rk=newdouble[1500],
ck=newdouble[1500];
FileInputStreamfstream=newFileInputStream(input);
DataInputStream in =newDataInputStream(fstream);
BufferedReaderbr=newBufferedReader(newInputStreamReader(in));
intline_count=0;
        String line="";
int count=0;
while((line=br.readLine())!=null){
if(line_count>1){
String[] output=line.split("\\s+");
motifs[count]=String.valueOf(output[0]);
total_fitness[count]=Double.parseDouble(output[1]);
rk[count]=Double.parseDouble(output[2]);
dkm[count]=Double.parseDouble(output[3]);
ck[count]=Double.parseDouble(output[4]);
count++;
}
line_count++;

}
for(int j=0;j<count;j++){
for(int l=j+1;l<count;l++){
```

```
if((motifs[j]==null? motifs[l]==null:
motifs[j].equals(motifs[l]))&&!"".equals(motifs[j])&&!"".equals(motifs[l])){
total_fitness[j]=total_fitness[j]+total_fitness[l];
rk[j]=rk[j]+rk[l];
dkm[j]=dkm[j]+dkm[l];
ck[j]=ck[j]+ck[l];
motifs[l]="";
total_fitness[l]=0;
rk[l]=0;
dkm[l]=0;
ck[l]=0;
}
}
}
for(int j=1;j<count;j++){
for(int l=count-1;l>=j;--l){
if(total_fitness[l-1]>total_fitness[l]){
                    String temp_motif=motifs[l-1];
doubletemp_total_fitness=total_fitness[l-1];
doubletemp_rk=rk[l-1];
doubletemp_dkm=dkm[l-1];
doubletemp_ck=ck[l-1];

motifs[l-1]= motifs[l];
total_fitness[l-1]=total_fitness[l];
rk[l-1]=rk[l];
dkm[l-1]=dkm[l];
ck[l-1]=ck[l];

motifs[l]=temp_motif;
total_fitness[l]=temp_total_fitness;
rk[l]=temp_rk;
dkm[l]=temp_dkm;
ck[l]=temp_ck;
}
}
}

result.println("Motifs"+"              "+"Total Fitness"+"       "+"r(k)"+"
"+"d(k,m)"+"                    "+"c(k)");
for(int h=count-1;h>=0;h--){
if(motifs[h]!=""){
result.print(motifs[h]);
result.print("              ");
result.print(newDecimalFormat("00.0000").format(total_fitness[h]));
result.print("              ");
result.print(newDecimalFormat("00.0000").format(rk[h]));
result.print("              ");
result.print(newDecimalFormat("00.0000").format(dkm[h]));
result.print("              ");
result.print(newDecimalFormat("00.0000").format(ck[h]));
result.println();
}
}
br.close();
result.close();
}
```

```
privatevoidcreate_directory(){
        File theDir5 =newFile("final_output");
if(!theDir5.exists()){
boolean result =false;
try{
theDir5.mkdir();
result=true;
}catch(SecurityException se){
}
if(result){
System.out.println("DIR created");
}
}
}
}
```

Appendix 6.2 - motifGenerator.java

```
packagetext_processor;

importjava.io.BufferedWriter;
importjava.io.File;
importjava.io.FileWriter;
importjava.io.IOException;
importjava.io.PrintWriter;

publicclassmotifGenerator{

publicmotifGenerator(String DNA,intmotif_length)throwsIOException{
        String reference_motif;
int counter =0;
int total =0;
        File theDir=newFile("motifs");

// if the directory does not exist, create it
if(!theDir.exists()){

boolean result =false;
            try{
theDir.mkdir();
result=true;
}catch(SecurityException se){
//handle it
}
if(result){
System.out.println("DIR created");
}
}
PrintWriter update
=newPrintWriter(newBufferedWriter(newFileWriter("motifs\\motifs_length_"+moti
f_length+".txt",false)));
for(inti=0;i<DNA.length()-(motif_length-1);i++){
reference_motif=DNA.subSequence(counter,counter+motif_length).toString();
total++;
counter++;
update.write(reference_motif);
```

```
update.println("\n");
}
System.out.println("Total Number of motifs possible is   : "+ total +" bp.");
System.out.println("Total Length of DNA sequence 1 is    : "+DNA.length()+"
bp.");
update.close();
}

}
```

Appendix 6.3 - HashTable.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

packagetext_processor;

importjava.io.BufferedWriter;
importjava.io.FileWriter;
importjava.io.IOException;
importjava.io.PrintWriter;

public class HashTable{

int number;
        String DNA;
HashTable(String DNA,int number)throwsIOException{
this.DNA= DNA;
this.number=number;
table_sequence();
}
int[][]table_sequence()throwsIOException{
int hash[][];
int A,T,G,C;
int count;
intcount_A=0;

intcount_T=0;
intcount_G=0;
intcount_C=0;
hash=newint[4][DNA.length()];
        A=0;
        T=1;
        G=2;
        C=3;
PrintWriter viewer
=newPrintWriter(newBufferedWriter(newFileWriter("hash_table\\hash_table_seque
nce_"+ number +".txt",false)));
for(count=0;count<DNA.length();count++){
if(DNA.charAt(count)=='A'){
hash[A][count_A]=count+1;
count_A++;
}
```

```
if(DNA.charAt(count)=='T'){
hash[T][count_T]=count+1;
count_T++;
}
if(DNA.charAt(count)=='G'){
hash[G][count_G]=count+1;
count_G++;
}
if(DNA.charAt(count)=='C'){
hash[C][count_C]=count+1;
count_C++;
}

}
System.out.print("A : ");
viewer.print("A : ");
for(int m=0;m<count_A;m++){
viewer.print(hash[A][m]);
viewer.print(" ");
System.out.print(hash[A][m]);
System.out.print(" ");
}
System.out.print("\n"+"T : ");
viewer.println("\n");
viewer.print("\n"+"T : ");
for(int m=0;m<count_T;m++){
viewer.print(hash[T][m]);
viewer.print(" ");
System.out.print(hash[T][m]);
System.out.print(" ");
}
System.out.print("\n"+"G : ");
viewer.println("\n");
viewer.print("\n"+"G : ");
for(int m=0;m<count_G;m++){
viewer.print(hash[G][m]);
viewer.print(" ");
System.out.print(hash[G][m]);
System.out.print(" ");
}
System.out.print("\n"+"C : ");
viewer.println("\n");
viewer.print("\n"+"C : ");
for(int m=0;m<count_C;m++){
viewer.print(hash[C][m]);
viewer.print(" ");
System.out.print(hash[C][m]);
System.out.print(" ");
}
System.out.print("\n");
viewer.close();
System.out.println("Total Length of DNA sequence "+ number +" is    :
"+DNA.length()+" bp.");
return hash;
}
}
```

Appendix 6.4 - motifChecker.java

```
packagetext_processor;

importjava.io.BufferedReader;
importjava.io.BufferedWriter;
importjava.io.DataInputStream;
importjava.io.FileInputStream;
importjava.io.FileNotFoundException;
importjava.io.FileWriter;
importjava.io.IOException;
importjava.io.InputStreamReader;
importjava.io.PrintWriter;

public class motifChecker{
int count_motif=0;
motifChecker(int[][]hashtable,Stringpath,Stringoutput,intDNA_length,intmotif_
length,int mutation)throwsFileNotFoundException,IOException{
FileInputStreamfstream=newFileInputStream(path);
DataInputStream in =newDataInputStream(fstream);
BufferedReaderbr=newBufferedReader(newInputStreamReader(in));
        String motif;
char letter =0;
PrintWriter result
=newPrintWriter(newBufferedWriter(newFileWriter("frequent_motifs\\"+
output,false)));
PrintWriter result2
=newPrintWriter(newBufferedWriter(newFileWriter("candidates\\"+
output,false)));
PrintWriter result3
=newPrintWriter(newBufferedWriter(newFileWriter("mutation\\mutation"+mutation
+"_"+ output,false)));
int index =0;
int[][] results =newint[motif_length][DNA_length];
int length =0;
result.println("Motif Number"+"           "+"Motif"+"
"+"Position");
result.println("============"+"           "+"=====+"
"+"========");
//intoutput_motif=0;
intmotif_no=1;
intm_init=mutation;
while((motif =br.readLine())!=null){
length=0;
int[]check_count=newint[DNA_length];


intm_check=1;
for(int m =0; m <=motif.length()-1; m++){
letter=motif.charAt(m);
index=returnindex(letter);
int[] counter =hashtable[index];
inti=0;
if(length ==0){
for(i=0;i<DNA_length;i++){
```

```
results[length][i]= counter[i];
check_count[i]++;
}
}
int[] checker=hashtable[index];
if(length >0){
for(int j =0; j <DNA_length; j++){
intcheck_mut=0;
for(int k =0;k < checker[k]; k++){
if(counter[k]== results[length-1][j]+1&& results[length-1][j]!=0){
results[length][j]= counter[k];
check_count[j]++;
}
elseif(m_init==1){
if((mutation==1&& length>1&& counter[k]== results[length-2][j]+2&&
results[length-2][j]!=0&& results[length-1][j]!=results[length-2][j]+1)){
results[length][j]= counter[k];
results[length-1][j]=0;
check_count[j]++;
mutation--;
check_mut++;
m_check=mutation;

}
}
else{

}
if(m_init==1&&check_mut==0){
mutation=1;
}
}
}
}
length++;
}
if(length>=(motif_length-1)){
for(int k=0;k<DNA_length;k++){
if((check_count[k]==motif_length/*&&m_init==m_check ) ||
(m_init>m_check&&check_count[k]==motif_length*/)){
if(motif_no>=1&&motif_no<10){
result.print(motif_no+"            "+"          ");
}
if(motif_no>=10&&motif_no<100){
result.print(motif_no+"           "+"          ");
}
if(motif_no>=100){
result.print(motif_no+"          "+"          ");
}
result.print( motif +"          ");
count_motif++;
result2.println(motif);
for(int g=0;g<motif_length;g++){
result.print(" "+ results[g][k]+" ");
}
result.println("\n");
}
```

```
if(check_count[k]== motif_length-1){
if(motif_no>=1&&motif_no<10){
result3.print(motif_no+"             "+"           ");
}
if(motif_no>=10&&motif_no<100){
result3.print(motif_no+"            "+"           ");
}
if(motif_no>=100){
result3.print(motif_no+"           "+"          ");
}
result3.print( motif +"          ");
for(int g=0;g<motif_length;g++){
result3.print(" "+ results[g][k]+" ");
}
result3.println("\n");
}
}
}
mutation=1;
if(length>0){
motif_no++;
}
}
result3.close();
result2.close();
result.close();

}
intreturnindex(char in){
intindexs=0;
if(in =='A'){
indexs=0;
}
if(in =='T'){
indexs=1;
}
if(in =='G'){
indexs=2;
}
if(in =='C'){
indexs=3;
}
returnindexs;
}
intreturncount(){
returncount_motif;
}
}
```

Appendix 6.5 - fitness_function.java

```
packagetext_processor;

importjava.io.BufferedReader;
importjava.io.BufferedWriter;
importjava.io.DataInputStream;
importjava.io.FileInputStream;
importjava.io.FileNotFoundException;
importjava.io.FileWriter;
importjava.io.IOException;
importjava.io.InputStreamReader;
importjava.io.PrintWriter;
importjava.text.DecimalFormat;

public class fitness_function{

publicfitness_function(String path,intcount,intmotif_length,int
sequence)throwsFileNotFoundException,IOException{
PrintWriter result
=newPrintWriter(newBufferedWriter(newFileWriter("fitness\\fitness_"+"possible
_motifs_sequence"+sequence+"_length_"+motif_length+".txt",false)));
FileInputStreamfstream=newFileInputStream(path);
DataInputStream in =newDataInputStream(fstream);
BufferedReaderbr=newBufferedReader(newInputStreamReader(in));
        String motif;

String[] motifs=new String[count];
inti=0;
while((motif=br.readLine())!=null){
motifs[i]=motif;
i++;
}
char[]reference_motif=consensus_motif(motifs,count,motif_length);
result.print("Consensus Motif");
result.print("              ");
for(int u=0;u<motif_length;u++){
result.print(reference_motif[u]);
}
result.println();
result.println("Motif "+"              "+"Total Fitness"+"
"+"R(k)"+"                "+"D(k,m)"+"               "+"C(k)");
for(int y=0;y<count;y++){
if(!motifs[y].isEmpty()){
result.print(motifs[y]);
result.print("              ");
doubletotalfitness=
total_fitness(rk(reference_motif,motifs[y].toCharArray(),motif_length),motif_
length);
result.print(newDecimalFormat("00.0000").format(totalfitness));
result.print("              ");
double fitness =rk(reference_motif,motifs[y].toCharArray(),motif_length);
result.print(newDecimalFormat("00.0000").format(fitness));
result.print("              ");
doubledkm=dkm(reference_motif,motifs[y].toCharArray(),motif_length);
result.print(newDecimalFormat("00.0000").format(dkm));
```

```
result.print("               ");
doubleck=ck(motifs[y].toCharArray(),motif_length);
result.print(newDecimalFormat("00.0000").format(ck));
result.println();
}
}
result.close();

}
char[]consensus_motif(String[]motifs,intcount,intmotif_length){
char[]reference_motif=newchar[motif_length];
for(int n=0;n<motif_length;n++){
int a=0,t=0,g=0,c=0;
for(int m=0;m<count;m++){

if(motifs[m].charAt(n)=='A'){
a++;
}
if(motifs[m].charAt(n)=='T'){
t++;
}
if(motifs[m].charAt(n)=='G'){
g++;
}
if(motifs[m].charAt(n)=='C'){
c++;
}

}
if(a>=t && a>=g && a>=c){
reference_motif[n]='A';
}
elseif(t>=a && t>=g && t>=c){
reference_motif[n]='T';
}
elseif(g>=t && g>=a && g>=c){
reference_motif[n]='G';
}
elseif(c>=a && c>=g && c>=t){
reference_motif[n]='C';
}

}
returnreference_motif;
}
doublerk(char[]reference_motif,char[]motif,intmotif_length){
doublerk=0;

rk=dkm(reference_motif,motif,motif_length)+ck(motif,motif_length);
returnrk;
}
doubletotal_fitness(doublerk,double length){
doubletotal_fitness=0;
double w1=0,w2=0;
        w1=length-5;
        w2=1;
total_fitness= w1*length + w2*rk;
```

```
returntotal_fitness;
}
doubleck(char[]motif,intmotif_length){
doubleck=0;
double a=0,t=0,c=0,g=0;
for(inti=0;i<motif_length;i++){
if(motif[i]=='A'){
a++;
}
if(motif[i]=='T'){
t++;
}
if(motif[i]=='G'){
g++;
}
if(motif[i]=='C'){
c++;
}
}
ck=(4/3)*(1-(((a*a)+(t*t)+(c*c)+(g*g))/(motif_length*motif_length)));
returnck;
}
doubledkm(char[]reference_motif,char[]motif,intmotif_length){
doubledkm=0;

for(inti=0;i<motif_length;i++){
if(reference_motif[i]==motif[i]){
dkm++;
}
}
dkm=dkm/motif_length;
returndkm;
}
}
```

Appendix 6.6 - final_output.java

```
packagetext_processor;

importjava.io.BufferedReader;
importjava.io.BufferedWriter;
importjava.io.DataInputStream;
importjava.io.File;
importjava.io.FileInputStream;
importjava.io.FileNotFoundException;
importjava.io.FileWriter;
importjava.io.IOException;
importjava.io.InputStreamReader;
importjava.io.PrintWriter;
importjava.text.DecimalFormat;

public class final_output{
public final_output(String
input,intmotif_length,intsequence_no)throwsFileNotFoundException,IOException{
create_directory();
```

```
PrintWriter result =newPrintWriter(newBufferedWriter(new
FileWriter("final_output\\"+"final_output_sequence_no"+sequence_no+"_motif_le
ngth_"+motif_length+".txt",false)));
String[] motifs =new String[1500];
double[]total_fitness=newdouble[1500],dkm=newdouble[1500],rk=newdouble[1500],
ck=newdouble[1500];
FileInputStreamfstream=newFileInputStream(input);
DataInputStream in =newDataInputStream(fstream);
BufferedReaderbr=newBufferedReader(newInputStreamReader(in));
intline_count=0;
        String line="";
int count=0;
while((line=br.readLine())!=null){
if(line_count>1){
String[] output=line.split("\\s+");
motifs[count]=String.valueOf(output[0]);
total_fitness[count]=Double.parseDouble(output[1]);
rk[count]=Double.parseDouble(output[2]);
dkm[count]=Double.parseDouble(output[3]);
ck[count]=Double.parseDouble(output[4]);
count++;
}
line_count++;

}
for(int j=0;j<count;j++){
for(int l=j+1;l<count;l++){
if((motifs[j]==null? motifs[l]==null:
motifs[j].equals(motifs[l]))&&!"".equals(motifs[j])&&!"".equals(motifs[l])){
total_fitness[j]=total_fitness[j]+total_fitness[l];
rk[j]=rk[j]+rk[l];
dkm[j]=dkm[j]+dkm[l];
ck[j]=ck[j]+ck[l];
motifs[l]="";
total_fitness[l]=0;
rk[l]=0;
dkm[l]=0;
ck[l]=0;
}
}
}
for(int j=1;j<count;j++){
for(int l=count-1;l>=j;--l){
if(total_fitness[l-1]>total_fitness[l]){
                    String temp_motif=motifs[l-1];
doubletemp_total_fitness=total_fitness[l-1];
doubletemp_rk=rk[l-1];
doubletemp_dkm=dkm[l-1];
doubletemp_ck=ck[l-1];

motifs[l-1]= motifs[l];
total_fitness[l-1]=total_fitness[l];
rk[l-1]=rk[l];
dkm[l-1]=dkm[l];
ck[l-1]=ck[l];

motifs[l]=temp_motif;
```

```
total_fitness[l]=temp_total_fitness;
rk[l]=temp_rk;
dkm[l]=temp_dkm;
ck[l]=temp_ck;
}
}
}

result.println("Motifs"+"              "+"Total Fitness"+"       "+"r(k)"+"
"+"d(k,m)"+"                    "+"c(k)");
for(int h=count-1;h>=0;h--){
if(motifs[h]!=""){
result.print(motifs[h]);
result.print("               ");
result.print(newDecimalFormat("00.0000").format(total_fitness[h]));
result.print("               ");
result.print(newDecimalFormat("00.0000").format(rk[h]));
result.print("               ");
result.print(newDecimalFormat("00.0000").format(dkm[h]));
result.print("               ");
result.print(newDecimalFormat("00.0000").format(ck[h]));
result.println();
}
}
br.close();
result.close();
}

privatevoidcreate_directory(){
        File theDir5 =newFile("final_output");
if(!theDir5.exists()){
boolean result =false;
try{
theDir5.mkdir();
result=true;
}catch(SecurityException se){
}
if(result){
System.out.println("DIR created");
}
}
}
}
```