ISLAMIC UNIVERSITY OF TECHNOLOGY

Motif Finding Using Non-Deterministic Finite Automata

Authors:

Touhidul Alam Tapos (114410)

H.M. Amio Rahman (114448)

Supervisor:

Tareque Mohmud Chowdhury

Assistant Professor

Department of Computer Science and Engineering

A thesis submitted to the Department of CSE in partial fulfillment of the requirements for the degree of B.Sc. Engineering in CSE

Academic Year: 2014-15

A Subsidiary Organ of the Organization of Islamic Cooperation

Dhaka, Bangladesh

October 2015

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 Touhidul Alam Tapos and H.M. Amio Rahman under the supervision of Tareque Mohmud Chowdhury in 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 Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.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:

Touhidul Alam Tapos	H.M. Amio Rahman
Student ID – 114410	Student ID – 114448

Supervisor:

Tareque Mohmud Chowdhury
Assistant Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)

Abstract

Motif Finding is one of the topmost concerns on the basis of characteristics of the species and its primary concept is finding the common DNA sequence that specifies a certain characteristics of a species. In this thesis, here we propose a methodology of finding method using an algorithm called non-deterministic finite automata. Our primary concern is finding the best common sequence that follows the pattern and gives an accurate result with minimization of errors. And following that solving complexity issue that occurs in automaton system, we try to minimize the system time and space complexity so that it can find pattern with less storage needed. Using non-deterministic approach so far we provided a better solution approach with patterns both in matched and gapped situation.

Table of contents

Declaration of Authorship i

Abstract ii

Chapter 1: Introduction **3**

1.1 Overview 3

1.2 Problem Statement 4

1.3 Motivation 5

1.4 Outline 6

Chapter 2: Background **7**

2.1 Motif Models....7

2.2 Non-Deterministic Finite Automata 9

Chapter 3: Literature Review **11**

3.1 Greedy Profile Motif Search 11

3.2 Gibbs Sampling 12

3.3 Randomized Projection 13

3.4 Micron Automata Processor....14

3.5 Significant score of Motif in Biological Sequence 14

3.6 Common Motif with Gaps using Finite Automata 15

Chapter 4: Proposed Method **17**

4.1 Overall Concept 17

4.2 Proposed Algorithm 18

4.3 Methodology 19

4.4 Experiments and Data Analysis 21

4.4.1 Data set 21

4.4.2 Results 22

4.4.3 Time Complexity 24

4.4.4 Time Comparison 26

4.4.5 Result Verification 26

Chapter 5: Conclusion **28**

5.1 Summery of the thesis 28

5.2 Future works 28

Bibliography.......... 29

Introduction *Chapter 1*

1.1 Overview

Bioinformatics is one of the most promising sectors in modern era for research purpose and DNA (Deoxyribonucleic acid) is the most common term used in the sector of Biology and Bioinformatics. Without sequence information encoded as DNA, none of the known living cells could exist. Understanding sequences is therefore fundamental to Understanding biology. In this thesis, we study theoretical foundations and algorithms for the motif discovery problem, where, given a set of such sequences over a finite alphabet, we are asked to find exceptional patterns called as Motif. The reason for searching this pattern is that this Motif carries an important Biological meaning. Therefore, discovered patterns can serve as a basis for new biological hypotheses and direct further experimentation.

The goal of this thesis is to analyze the motif finding algorithms and comparison of our proposed algorithms with others:

- How motifs are searched in a given DNA sequence?
- How a motif with mismatches is computed?
- How the time and space complexity are computed?

These questions can be attributed to the areas of biology, statistics and algorithmic, respectively. Our goal of the thesis is to compute an approach finding motif with a user given mismatches and find the most occurred pattern.
'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.

1.2 Problem Statement

Motif finding problem of this thesis is different from other paper in different ways like , constructing data sets of the DNA sequence, matching given sequences and finding the exact and non-exact matches in the data sequences. 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 on a procedure using non-deterministic finite automata (NFA), making the NFA from the given data set and finding a given l-mer pattern from the dataset.

Main challenge in finding motif problem is 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

Our main goal is to redefine this procedure of NFA and optimize the result and comparison of this algorithm with other NFA approach.

1.3 Motivation

In this section, we identify desired properties of the motif discovery algorithm to be developed in this thesis. Before doing that, we discuss the user input to be made to such an algorithm.

Every motif discovery algorithm should require that the user provides three pieces Of information; first, a set of input sequences to be searched, second, some hypothesis on the kind of pattern to be looked for, and third, knowledge on sequence composition, that is, a specification of sequence features that are to be expected and should hence not be reported as new motifs. While it is clear that input sequences must be provided, the second and third aspects are often not recognized as input to be made by the user. Both are, however, fundamental to motif discovery. To exemplify that, let us assume that no assumptions on pattern type are made. Then, in an extreme case, an algorithm might report the whole input text itself as a long motif with one occurrence and a high significance score. Testing a larger space of motifs is therefore not always an advantage, even when runtimes are not considered. A motif space should therefore consciously be chosen by the user based on the scientific question to be addressed. When searching for transcription factor binding sites, for instance, we can use knowledge on known binding motifs to specify a suitable range of motif lengths to be considered and possibly a maximal allowed degree of degeneracy, that is, a limit on the use of wildcard characters. The third requirement, providing information on expected sequence composition, can be seen as answering the question of what is already known about the studied sequences. For example, if nothing is known, it is a new result that a genome contains overrepresented patterns comprised of many Cs and Gs. If, in contrast, a biologist knows that the GC content (i.e. the fraction of nucleotides that are C or G) is high, these news are hardly surprising. As the goal of automated motif discovery is to find novel sequence features, it is inevitable that information about what is already known needs to be available. Therefore, the user should provide a background model or null model. This analysis of input to be made by the user directly translates into the first two requirements on a motif discovery algorithm listed below. Additionally, we formulate two further requirements.

Flexibility of background models: Background text models should be general enough to describe complex dependencies. I.i.d. Models and first order Markovian models are insufficient for many applications, especially those concerning DNA.
Appropriate scoring functions: The scoring function used to assess motifs should reflect the statistical significance with respect to the background model. The statistical properties of motifs can strongly be affected by self-overlaps (see Example 1.8). Hence, overlapping must be accounted for.
Exactness: To eliminate the risk of missing significant motifs, the algorithm should not work heuristically but extract a motif that is provably optimal with respect to the scoring function.
Practicality: The algorithm should be able to solve practical problem instances on current hardware in reasonable time.
The development of mathematical and algorithmically means to achieve this is the main goal of the present thesis. Before we can formalize the problem, we need to introduce some notation.

1.4 Outline

We first start with an introduction of motif models in Section 2.1, a general introduction of finite automata in Section 2.2, a background study of our work related papers discussed in Chapter 3.Chapter 4 consists of the our proposed method and its algorithm with full methodology, Chapter 5 analyze the data set of our given algorithm and comparison with other existing algorithms. Chapter 6 discussion of the algorithm and further work that also can be implemented with this approach.

Background *Chapter 2*

This chapter is divided into two categories. First part includes the fundamental of motif, motif discovery, models, different way to express motif in biological sequences. Second part of the chapter includes basic automata, non-deterministic finite automata and its basic structure, working procedure of NFA.

2.1 Motif Models

A DNA molecule consists of two anti-parallel chains of nucleotides forming a double helix. In DNA, each nucleotide contains one of the four bases adenine, cytosine, guanine, or thymine and, thus, a strand of DNA can be formalized as a string over the alphabet $\sum = \{A, C, G, T\}$. Transcription factors are proteins that bind to DNA in a sequence specific manner. That means they recognize special (sets of) sequences of nucleotides, called binding motifs. The motif-finding problem arises when DNA sequences contain instances of binding sites of a transcription factor but the binding motif of this factor is unknowns. Then finding motif algorithm might reveal a pattern that is shared by all sequences that can then by hypothesized to be the transcription factor's binding motif. 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 over many sequences.
- They are small, have constant size, and are repeated very often.
- They are statistically over-represented in regulatory regions.

The purpose of motif models is to generalize a limited number of observed motif instances to a larger set of similar strings that we predict to be motif instances as well. That means, when we encounter one of these strings in an unknown piece of DNA, we would annotate this position as a putative motif instance. Motif models differ in the way this generalization is done. The most widely used models are consensus strings with a

Hamming neighborhood (i.e. the set of strings with a limited Hamming distance to the consensus), position weight matrices (PWMs) along with a score threshold, and generalized strings (also known as IUPAC strings in the context of DNA). Following, we explain the three models using the example of a DNA binding site, namely record MA0107.1 in the Jaspar database (Sandelin et al., 2004) which represents the binding site of the Rel protein domain as described by Kunsch et al. (1992). The database contains 18 experimentally verified binding sites that are shown in Figure 1.1a. Some of these sites occur multiple times such that the set of all sites contains 14 distinct sequences.

Consensus Strings with Hamming Neighborhood: When confronted with a set of motif instances, a consensus string can be obtained by majority vote at each position as depicted in Figure 1.1b. In the shown case, the consensus GGGAATTTCC has a Hamming distance of up to three to the original sequences. Therefore, the motif might be modeled as the set of all strings with a Hamming distance of at most three to the consensus. Thus, the given 18 strings are generalized to all 3676 strings with at most three differences to GGGAATTTCC. The underlying assumption is that only the number of mismatches is important and not their position. Whether or not this assumption is justified depends on the application. In case of transcription factor binding sites, some positions are usually more conserved than others, challenging this assumption. Nonetheless, the Weeder algorithm by Pavesi et al. (2004), which uses this motif model, outperformed many other methods in a benchmark study by Tompa et al. (2005).

Position Weight Matrices: A position weight matrix (PWM) of length l is a | ∑ | * l matrix, where each entry w_i gives the score to be counted when character _ is found at position i (Staden, 1984). Given a putative motif instance, the scores for the characters at all positions are summed up to obtain a score for the instance. When the score is above a pre-chosen threshold, the string is said to be a motif instance. A common method to obtain a PWM is the translation of a position frequency matrix (PFM) into log-odds scores, where the PFM contains, for each position, the number of times each character occurred at this position in the sample sequences. An example is shown in Figure 1.1d.

(a)

```
GGGAATTTCC
GGGAATTTCC
GGGAATTTCC
TGGAATTTCC
TGGAATTTCC
CGGAATTTCC
GGGGATTTCC
GGGACTTTCC
CGGAGTTTCC
GGGAATTCCC
CGGACTTTCC
CGGACTTTCC
GGGGAATTCC
TGGGGTTTCC
GGGGATTCCC
GGGGTTTTCC
GGGGGATTCC
GTGGGTTTCC
```

(b) Consensus: GGGAATTTCC

(c) IUPAC string: BKGRNWTYCC

(d)

A:	0	0	0	11	10	2	0	0	0	0
C:	4	0	0	0	3	0	0	2	18	18
G:	11	17	18	7	4	0	0	0	0	0
T:	3	1	0	0	1	16	18	16	0	0

(e)

A:	-2.30	-2.30	-2.30	0.83	0.74	-0.69	-2.30	-2.30	-2.30	-2.30
C:	-0.11	-2.30	-2.30	-2.30	-0.36	-2.30	-2.30	-0.69	1.31	1.31
G:	0.83	1.25	1.31	0.41	-0.11	-2.30	-2.30	-2.30	-2.30	-2.30
T:	-0.36	-1.20	-2.30	-2.30	-1.20	1.19	1.31	1.19	-2.30	-2.30

(f)

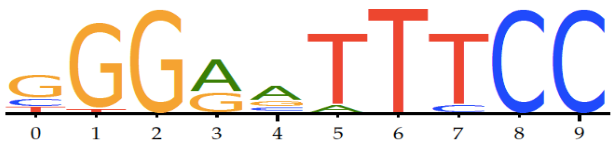


Figure 2.1: Different motif models of the binding site MA0107.1 from the Jaspar database are shown: (a) original sequences (b) consensus string obtained by position-wise majority vote (c) minimal IUPAC string matching all sequences (d) position frequency matrix (e) position weight matrix containing log-odds scores computed assuming uniform background distribution and 0.5 pseudo counts (f) sequence logo.

Generalized Strings: A generalized string is a sequence of character sets. Each set gives the allowed characters at that position. The generalized string g = {A, G}, {A}, {A,T} for example, matches AAA, AAT, GAA, and GAT. Sets containing more than one letter (like {A, G} or {A, T} are called wildcards. In case of nucleotide sequences, all possible sets are commonly abbreviated by IUPAC one-letter codes (IUPAC stands for International Union of Pure and Applied Chemistry, the one-letter codes for sets of DNA characters have been proposed by Cornish-Bowden, 1985).

2.2 Non-Deterministic Finite Automata

Nondeterministic finite automata (NFAs) are probably best known for being equivalent to right-linear context-free grammars and, thus, for capturing the lowest level of the Chomsky-hierarchy, the family of regular languages. It is well known that NFAs can offer exponential saving in space compared with deterministic finite automata (DFAs), that is, given some n -state NFA one can always construct a language equivalent DFA with at most 2n states [67]. This so-called power set construction turned out to be optimal, in general. That is, the bound on the number of states is tight in the sense that for an arbitrary n there is always some n -state NFA which cannot be simulated by any DFA with less than 2n states [63,64]. These two milestones from the early days of automata theory form part of an extensive list of equally striking problems of NFA related problems, and are the basis of description complexity. Moreover, it initiated the study of the power of resources and features given to finite automata see, e.g., [21] for a survey on limited resources for finite automata.

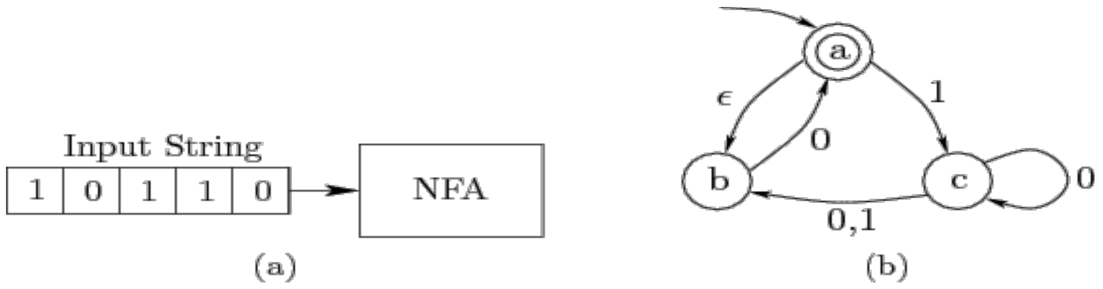


Figure 2.2: (a) Input string that is on a queue is entered into the NFA system. (b) NFA from where the given inputs are checked whether they are accepted or not.

Our tour on the subjects listed in the abstract of NFAs related problems cover some (recent) results in the field of description and computational complexity. It obviously lacks completeness, as NFAs fall short of exhausting the large selection of finite automata related problems considered in the literature. We give our view of what constitute the most recent interesting links to the considered problem areas. Our nomenclature of finite automata is as follows: A nondeterministic finite automaton (NFA) is a quintuple A = (Q,Σ, δ, q_0, F), where Q is the finite set of states , Σ is the finite set of input symbols , q0 ∈ Q is the initial state , F ⊆ Q is the set of accepting states , and δ : Q x Σ → 2^Q is the transition function . A finite automaton is deterministic (DFA) if and only if |δ (q, a)| = 1, for all states q ∈ Q and letters a ∈ Σ . The language accepted by the finite automaton A is defined as L (A) = {w ∈ Σ* | δ (q_0, w) ∩ F != ∅}, where the transition function is recursively extended to δ : Q x Σ* → 2^Q .

There are so many algorithms exist so far on the area of motif finding. Different algorithms have been approached by different data structure like – Suffix tree, Hash table, Automata etc. But no existing algorithms have followed the trade off between run time and efficiency of the algorithm. We have discussed here few existing algorithms that works on exact matched or gapped match, though our focus has always been on finding motif within gapped match. In Section 3.1 we discussed Greedy profile Motif Search which follows a greedy algorithm , Section 3.2 Gibbs Sampling , one of the widely used Method for Finding motif , Section 3.3 A Randomized projection , which follows random distribution pattern, Section 3.4 Micron Automata Processor , A built-in processor chips in which solve pattern finding using automaton process, Section 3.5 includes a NFA approach of scoring motif and finally in 3.6 a motif search with gap that uses finite automata process.

3.1 Greedy Profile Motif Search

This algorithm Use P-Most probable *l*-mers to adjust starts positions until we reach a "best" profile; this is the motif. Select random starting positions. It create a profile **P** from the substrings at these starting positions and change the starting position to the starting , Then Find the **P**-most probable *l*-mer **a** in each sequence position of **a**. Later Compute a new profile based on the new starting positions after each iteration and proceed until we cannot increase the score anymore. Basic algorithms are given below –

```
GreedyProfileMotifSearch(DNA, t, n, l )
 1. Randomly select starting positions s=($s_1$,...,$s_t$) from DNA
 2. bestScore = 0
 3. while Score(s, DNA) > bestScore
 4.    Form profile P from s
 5. bestScore = Score(s, DNA)
 6. for $i$ = 1 to $t$
 7.    Find a P-most probable $l$-mer a from the $i$th sequence
 8.    $s_i$ = starting position of a
 9.  return bestScore
```

Figure 3.1: A simple greedy profile motif search algorithm

Limitations:

- Since we choose starting positions randomly, there is little chance that our guess will be close to an optimal motif, meaning it will take a very long time to find the optimal motif.
- It is unlikely that the random starting positions will lead us to the correct solution at all.
- In practice, this algorithm is run many times with the hope that random starting positions will be close to the optimum solution simply by chance

3.2 Gibbs Sampling

We can improve the previous algorithm by introducing **Gibbs Sampling**, an iterative procedure that discards one *l*-mer after each iteration and replaces it with a new one. Gibbs Sampling proceeds more slowly and chooses new *l*-mers at random increasing the odds that it will converge to the correct solution.

Procedure:

- Randomly choose starting positions = (s_1... s_t) and form the set of *l*-mers associated with these starting positions.
- Randomly choose one of the *t* sequences.
- Create a profile **P** from the other *t* -1 sequences.
- For each position in the removed sequence, calculate the probability that the *l*-mer starting at that position was generated by **P**.
- Choose a new starting position for the removed sequence at random based on the probabilities calculated in step 4.
- Repeat steps 2-5 until there is no improvement

Limitations:

- Gibbs sampling needs to be modified when applied to samples with unequal distributions of nucleotides (***relative entropy*** approach*)*.
- Gibbs sampling often converges to locally optimal motifs rather than globally optimal motifs.
- Needs to be run with many randomly chosen seeds to achieve good results.

3.3 Randomized Projection

It uses a random array of projecting position on a given string to find the motifs and store them in a Motif Bucket.

Procedure:

- Choose *k* positions in string of length *l*.
- Concatenate nucleotides at chosen *k* positions to form *k*-tuple.
- This can be viewed as a projection of *l*- dimensional space onto *k*-dimensional subspace.
- Select *k* out of *l* positions uniformly at random.
- For each *l*-tuple in input sequences, hash into bucket based on letters at *k* selected positions.
- Recover motif from enriched bucket that contain many *l*-tuples.
- Some projections will fail to detect motifs but if we try many of them the probability that one of the buckets fills in is increasing.

Limitations:

- It cannot estimate multi-way distances nor can it estimate 1-norm distances.
- Projection on only Boolean data and nearly independent data.

3.4 Micron Automata Processor

A Non-von Neumann architecture called the Micron automata processor that is a intergraded computer processor which works on the PCI slot of the motherboard and works only to solve problems related to DFA and NFA. [6]

- It can create simulation to solve the problems
- A Processor only to solve finite automaton problems
- It can solve the finite automaton problems faster than any existing algorithms
- It's a costly processor to work with

3.5 Significance score of Motif in biological sequence

Here the methodology provides e formula that computes from the NFA to match given sequence. It at first deletes the nodes that are not connected to the final state. It [4] then formulates an algorithm that counts the matched position

```
Require: $(A,Q, \sigma,F, \delta)$ be a (minimal) NFA whose language is A*M
1: $S \leftarrow \{\sigma\}$
2: for $i = 1 \dots \ell$ do
3:        $S \leftarrow \cup_{q \in S} \delta(q, X_i)$
4:        if $S \cap F \neq \emptyset$ then
5:            report i as a matching position
6:        end if

7: end for
```

Figure 3.2: NFA Pattern matching. Returns all matching positions of motif M in $X_{1:\ell}$.

Complexity is $O(|Q| * \ell)$.

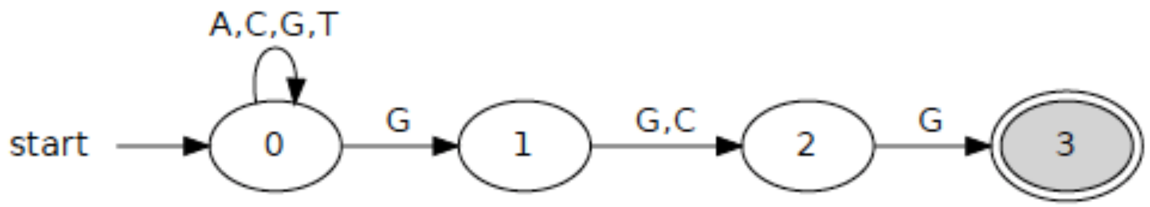


Figure 3.3: Minimal NFA whose language is (A|C|G|T)*G(G|C)G

Given the String AGCGGTGGGCGA. Now-

• i = 3, X_3 = C,S ←δ({0, 1}, C) = {0, 2};

• i = 4, X_4 = G,S ←δ({0, 2}, G) = {0, 1, 3}, matching position;

• i = 5, X_5 = G,S ←δ({0, 1, 3}, G) = {0, 1, 2};

• i = 6, X_6 = T,S ←δ({0, 1, 2}, T) = {0};

• i = 7, X_7 = G,S ←δ({0}, G) = {0, 1};

• i = 8, X_8 = G,S ←δ({0, 1}, G) = {0, 1, 2};

• i = 9, X_9 = G,S ←δ({0, 1, 2}, G) = {0, 1, 2, 3}, matching position;

• i = 10, X_{10} = C,S ←δ({0, 1, 2, 3}, C) = {0, 2}.

• i = 11, X_{11} = G,S ←δ({0, 2}, G) = {0, 1, 3}, matching position;

• i = 12, X_{12} = A,S ←δ({0, 1, 3}, A) = {0}.

We hence return three matching positions: 4, 9 and 11.

Limitations:

- Only on exact match, no gapped match is established

3.6 Common Motifs with gaps using Finite Automata

It present an algorithm that uses finite automata to and the common motifs with gaps occurring in all strings belonging to a finite set S = {S_1,S_2,…….S_r}. In order to find these common motifs it first identify the factors that exist in each string. Therefore the algorithm begins by constructing a factor automaton for each string Si.
To find the common factors of all the strings, the algorithm needs to gather all the factors from the strings together in one data structure and this is achieved by computing an automaton that accepts the union of the above-mentioned automata. Using this automaton it is able to create a new factor alphabet. Based on this factor alphabet a finite automaton is created for each string S_i that accepts sequences of all non-overlapping factors residing in each string. The intersection of the latter automata produces the finite automaton which accepts all the common subsequences with gaps over the factor alphabet that are present in all the strings of the set S = {S_1,S_2…..S_r}. These common subsequences are the common motifs of the strings. [2]

As an example consider a set of strings S = {aabccddab, babbcdacd}. It finds common motifs in this set of strings bounded from parameters where p = 2; q = 3.

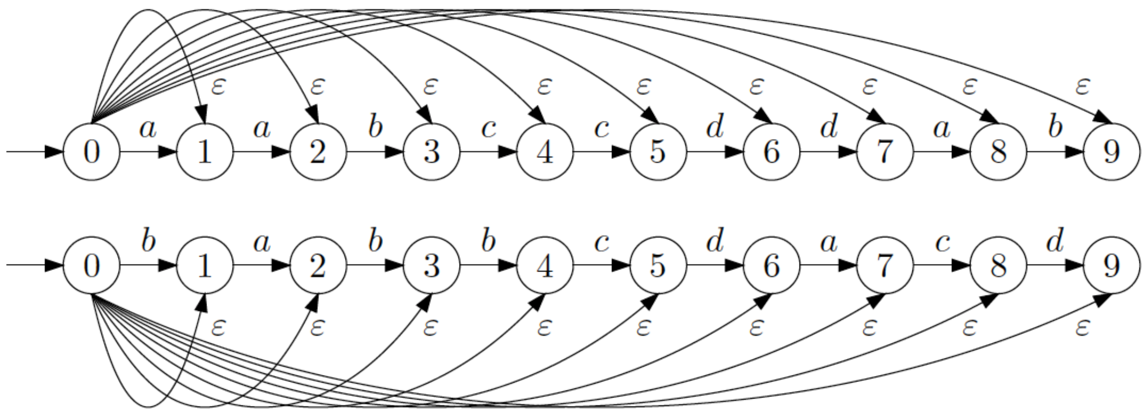


Figure 3.4: Transition diagrams of finite automata M_{1e} and M_{2e} for the set of strings

S = {aabccddab; babbcdacd} from the example and the output of this using automata

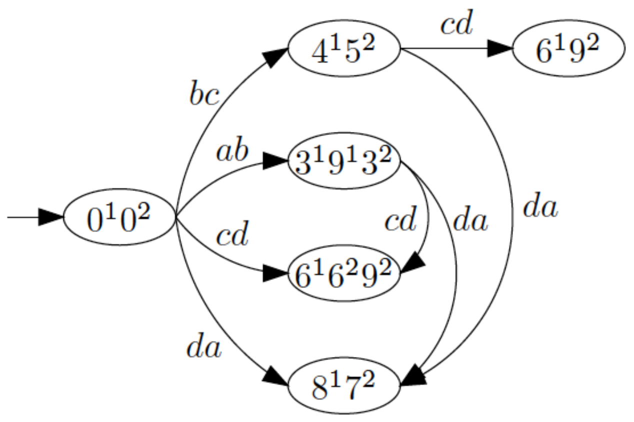


Figure 3.5: Transition diagram of finite automaton MS from the example

Limitations:

- Complexity O (n^k) when size increases as steps require intersection of automata

Proposed Method *Chapter 4*

4.1 Overall Concept

Our algorithm for finding motif starts with the given DNA sequences and a user defined l-mer with number of mismatches if any. Goal is to find the motif with highest probable pattern with l number in the given sequences. Now our work is on two scenarios where exact matches are computed and on other approximate are computed. Given the sequence of pattern, first according to our algorithm we are going to construct a NFA for each of the possible of the string according to the l-mer. According to the concept of NFA, when a next base is found on the sequences, it will create a new node with possible combination of l sequence of this base, also previous occurred base will append and look for pattern. In this way for each possible combination of pattern occurrence a path will be expanded. Whenever a l-mer pattern is found it keeps the track of that pattern on a hash table and search for the next occurrence. When we have read the entire input sequence file we will have a long list of possible motif occurrences of l-mer on the hash table containing each of the pattern with its frequency. Highest frequency of those sequences is most probable occurrence and each of the highest frequency is the candidate of the motifs of that sequences. Given the sequences of these candidate motifs we can analyze which factors are the reasons that are likely to occur in the sequences of these data sets.

Now, we are likely to have the candidate's motif that occurs in different sequences of DNA data set. Given the sequences of different species and their DNA set , we can find the pattern that occur in which places , it will be helpful if the pattern of same species has exact or approximate matches in a position.

4.2 Proposed Algorithm

Motif_By_NFA(S_{mn},k)

1. Given **n** number of String **S** each with **m** size in total S={S_1, S2, …… , S_n}, we need to find **l-mer** matched pattern in total n sequences where 6<=l<=20
2. If the value of **k is 0**
3. For each set of string sequences we construct a NFA N containing each of the l sequences pattern as accepted, so total **n-m+1** pattern
4. For the next string sequences, again same possible combination of l sequence will be checked
5. If a new pattern is found, its staring value and pattern will be saved in a storage as a counter for number of motifs and its index position
6. If a matched pattern is found, it will just increment the number of occurrence of the matched pattern on the storages
7. If **k is > 0**
8. Number of k defines the at most mismatches, so if a unmatched string is found it can still represent a pattern with at most k mismatches
9. If mismatched number is > k, that will not be considered as a pattern
10. This iteration will continue at all the sequence, then it will increment l value by 1, until l is less than or equal to 20
11. For exact matches, Return the table containing the most probable motif candidate that appears in sequences
12. For approximate matches, Return the motifs with at most k mismatches that counts and find the most probable of this pattern that occurs

4.3 Methodology

For the demonstration of the given algorithm, let's assume 2 string sequences of ATGCA & ATTGA. We have to find the l-mer pattern of occurrence. Lets assume here 2<=l<=3.

First for the l=2 , we construct NFA, where possible candidates are AT,TG,GC,CA and for 2nd string AT,TT,TG,GA. So AT and TG are the most occurred pattern here.

L=2

Pattern	AT	TG	GC	CA	TT	GA
Frequency	2	2	1	1	1	1

Table 4.1: Frequency of matches for Length(l=2)

For l=3, we construct NFA, where possible candidates are ATG,TGC,GCA and for 2nd string ATT,TTG,TGA. So each motif occurred once here.

L=3

Pattern	ATG	TGC	GCA	ATT	TTG	TGA
Frequency	1	1	1	1	1	1

Table 4.2: Frequency of matches for Length (l=3)

So in these scenario highest candidates of motif are AT and TG, with each probability of 20% of occurrences.

This sequences is calculated when k=0, so for exact matches this is the output of the motif candidates.

Now for k=1 and l=2, we first compute the same way and output candidates are same as without mismatches as the condition of being k and l is k<<l

So for k=1 and l=3, we get from 1st string ATGCA – A_TG, AT_G, T_GC, TG_C, G_CA, GC_A total 6 occurrences

And from 2nd string ATTGA – A_TT, AT_T, TT_G, T_TG, TG_A, T_GA total 6 occurrences.

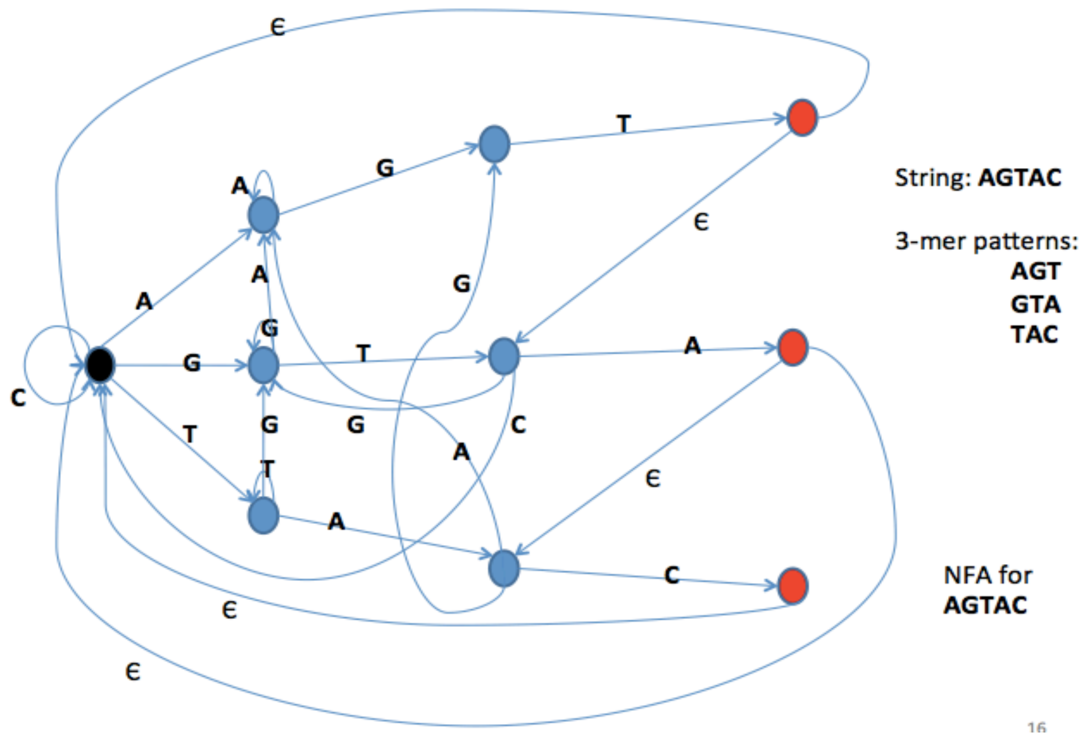


Figure 4.1: NFA for AGTAC. 3-mer patterns of AGT, GTA, TAC

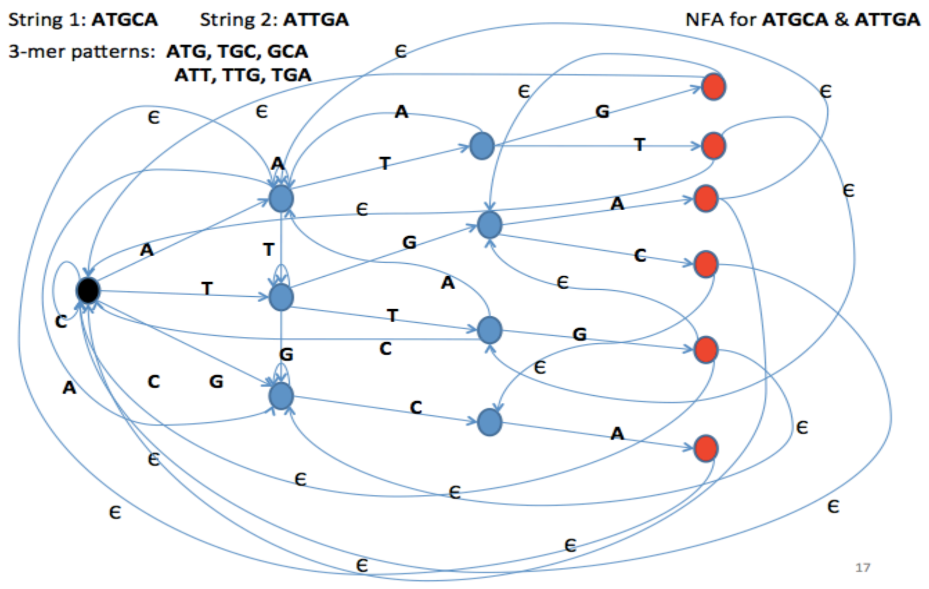

String 1: ATGCA String 2: ATTGA
3-mer patterns: ATG, TGC, GCA
ATT, TTG, TGA
NFA for ATGCA & ATTGA

Figure 4.2: NFA for ATGCA & ATTGA. Total 6 unique 3-mer patterns

So with Gapped match each probability of the sequences are equally distributed among these examples.

4.4 Experimental Data And Analysis

In this paper, we have used *Homo sapiens, Sus Scrofa, Aedes aegyapti* these three species of data set. The data was taken from GeneBank database. This data set has sequences among 829 bases to 850 bases. Table shows the dataset we have used for experimental purpose.

4.4.1 Data set:

Homo Sapiens	*Sus Scrofa*	*Aedes aegyapti*
NM_001166002.2	BW970508	KJ736826
NM_001166004.2	BW971304	KJ736827
NM_001166003.2	BW972295	KJ736825
NM_181773.4	BW973679	KJ736824

Table 4.3: List of different used Gene Sequences

4.4.2 Results:

Here we have considered variable length motif discovery with computing both as exact match and gapped match with 1 or 2 gaps at most each of the sequences. As a motif length of 6 without mutation (6,0), total number of potential motif presents in the DNA sequence is 824. For each set of different motif target sets we compute the candidates one and find the most occurred one. After each occurrence of the potential motif in a sequence of either 0,1 or 2 mismatches we find the potential motif in that sequences. Considering the scenario of this motif data we can find the probabilistic of the highest occurred motif in that scenario of l-mer. Now with this methodology we can consider each of l values motif and find the most occurred one in this consent.

Our values of these three sequences are computed here as three condition (6,0), (6,1) and (6,2) where 6 is the motif length and 0, 1 or 2 is the at most mismatches and considered as a probable motif.

The occurrence of candidates motif in three sequences are given below :

Species / l-k	(6,0)	(6,1)	(6,2)
Homo Sapiens	AAAAAG ACCCTG AGAGGC ….	GTCCAG TCCAGC	GCAAAA GTCAGC
Sus Scrofa	CATCAT	CCGCGC CTGTGG	GCCGCG CAAAGG
Aedes aegyapti	AAATTT AATTTA CAATTT …	AATTTA	TTATCA AATTCA

Table 4.4: Matched Sequences for Different Configuration (l,k)

From the given data scenario we can predict the candidate motif for this cases, with (6,0) exact matches of motif are computed from the species and from (6,1) and (6,2) approximate matches are computed from the data scenario.

Frequency comparison among the species motifs difference are given on the graph on here.

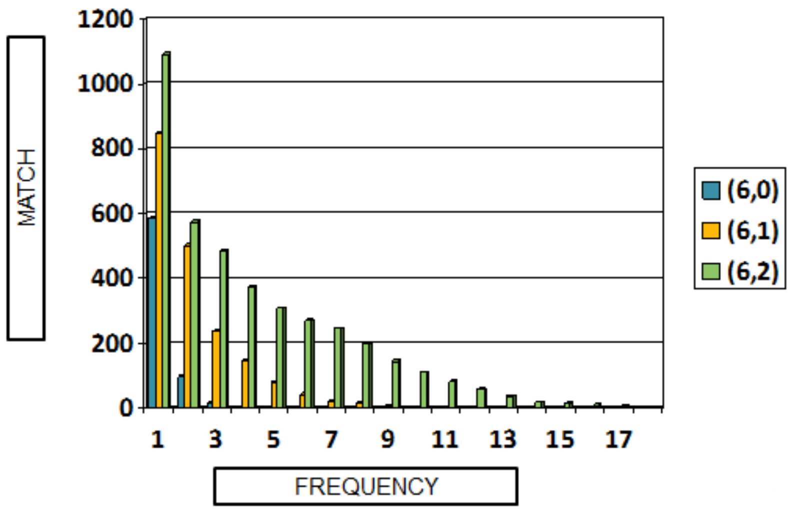


Figure 4.3: Match Frequency of ***Sus scrofa***

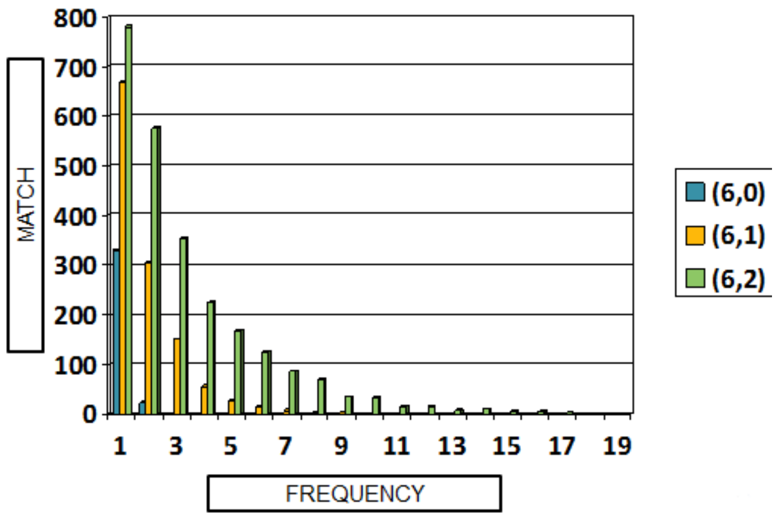
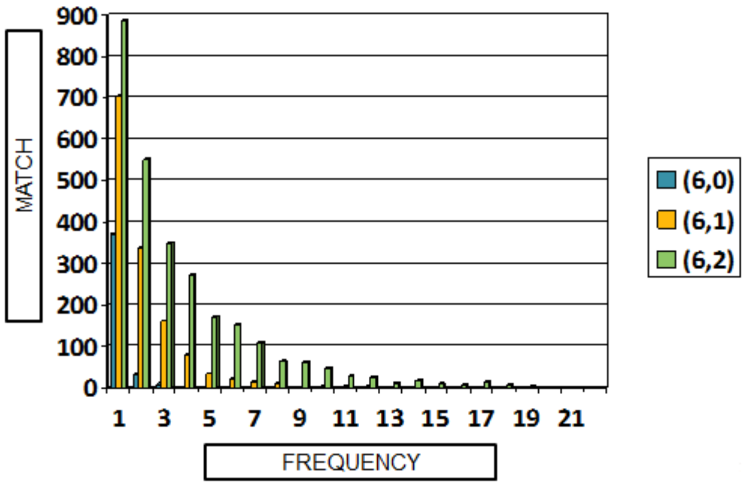


Figure 4.4: Match Frequency of ***Homo sapiens***



\

Figure 4.5: Match Frequency of ***Aedes aegepty***

Here in figure three species and there motif frequency comparison are given on the basis of exact and 1 or 2 mismatches. The motif with exact matches has less number of match count whereas motif with mismatch has more of a match count

4.4.3 Time Complexity

According to the construction and searching of the algorithm NFA usually take a specific amount of build time and search through the NFA a computational time. So time complexity of a NFA of m node is O(m) at average case and $O(2^m)$ at worst case and running time for an NFA is $O(m^2n)$ because it is non deterministic and computer checks for every possible path for the current character string. So at average case its total complexity will be $O(m+m^2n)$ which is far better than [2] which having polynomial time as $O(m^n)$ as l>3.

Here a run time computation of three different species is considered as it takes the program to execute and finish. The more it counts with the mismatches the run time exceeds more.

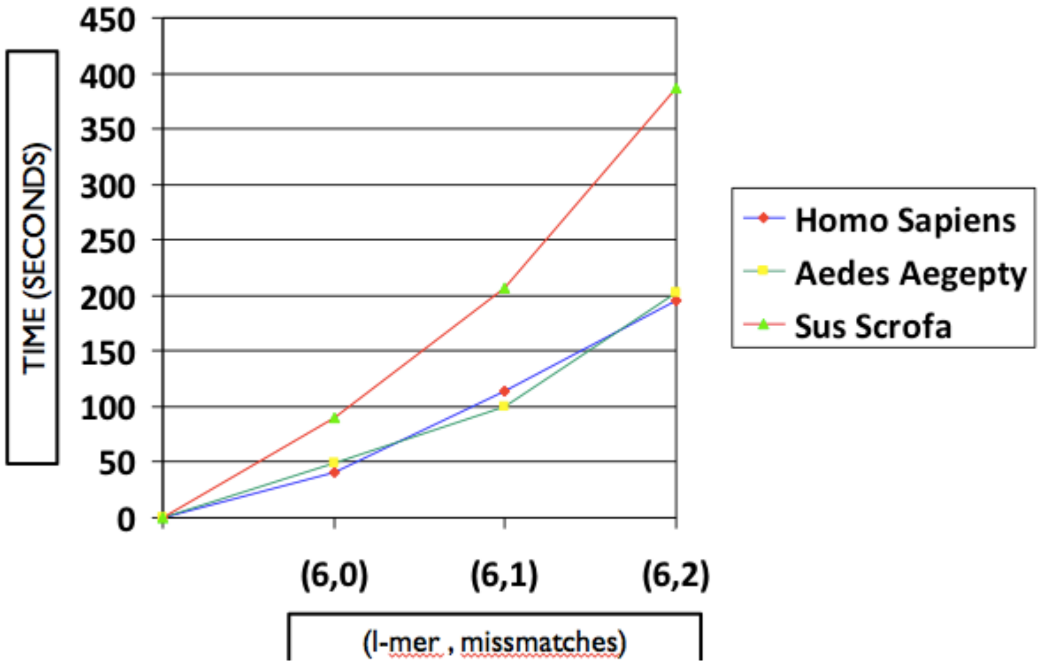


Figure 4.6: Run time for three species sequence on different motif configuration

Run time comparison stats show at the time approximate under 50s we get two results for (6,0) for *Aedes Aegepty* and *Sus Scrofa* , but for *Homo Sapiens* it took almost 100s , double of that. Again for (6,1) the comparison is almost same, as first two took time as 100s and the later one about 200s, almost double of last results. It grows exponentially for (6,2) first two species it took 200s and for *Homo Sapiens* its almost 400s, so it follows a binary exponential approach as time complexity was discussed.

4.4.4 Time Comparison

A time comparison of our proposed algorithm is computed against a formal hash table approach and our run time is computed against that algorithm, and by far its performance is getting better the more length and mutation increases.

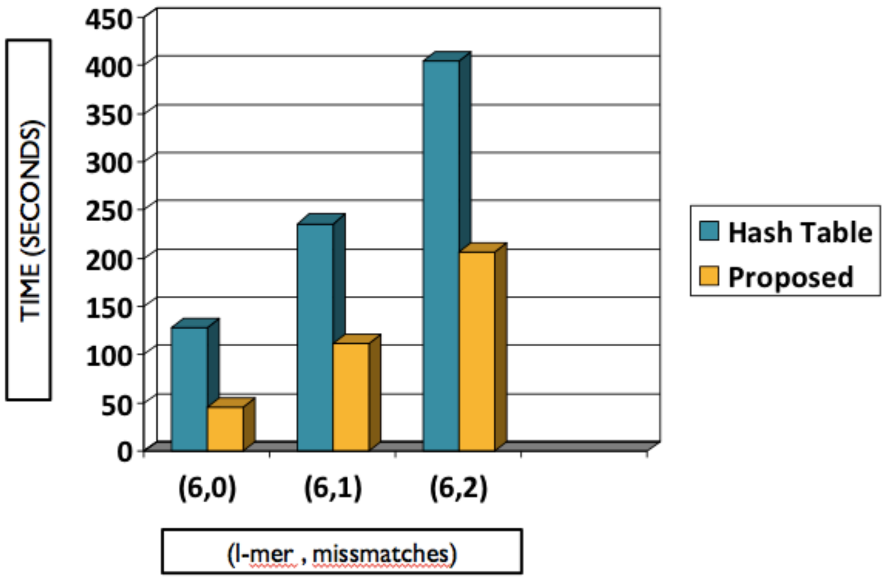


Figure 4.7: Run time comparison with Hash Table

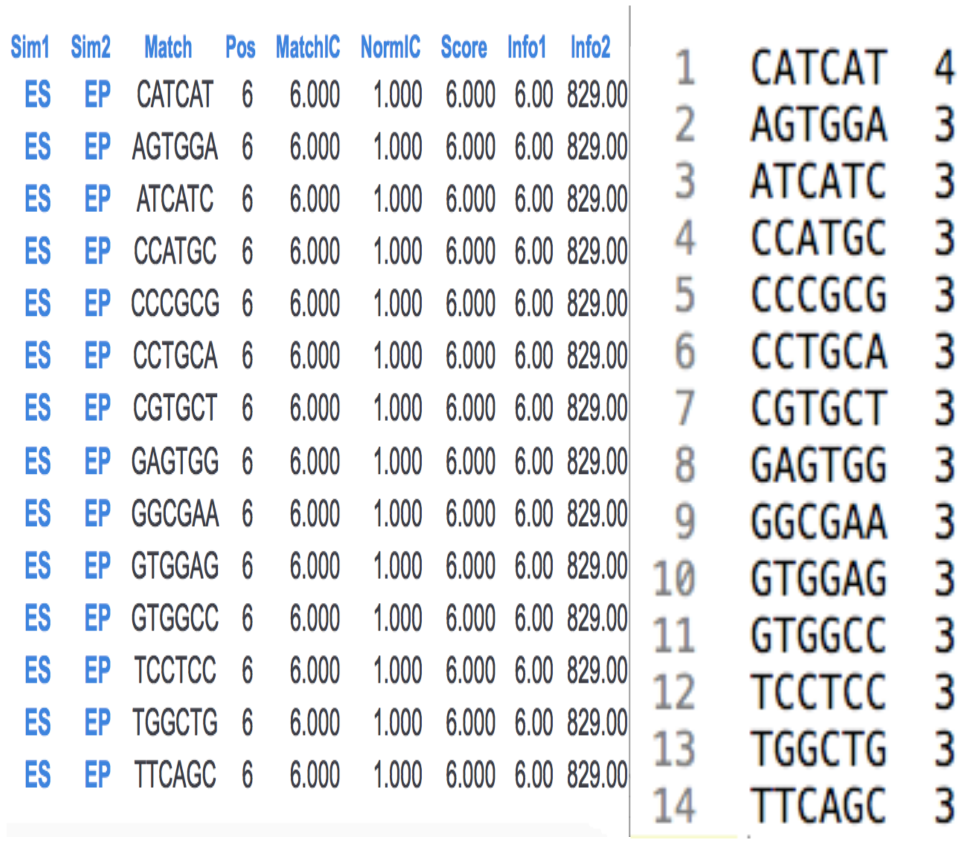
Here on the hash table algorithm the time takes 130,240 and 405 secods for 6-mer with exact,1 or 2 mismatches, but in our proposed algorithm it takes 47,105 and 207 seconds for this same sequence motif. So gradually its providing better performance than these approach.

4.4.5 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

1.Compare Motif	2. Proposed System
• Motif file • 1st DNA sequence from *Sus Scrofa* dataset • IC Value 1.0 (exact match)	• Motif File • 1st DNA sequence from *Sus Scrofa* dataset • Exact match



Sim1	Sim2	Match	Pos	MatchIC	NormIC	Score	Info1	Info2
ES	EP	CATCAT	6	6.000	1.000	6.000	6.00	829.00
ES	EP	AGTGGA	6	6.000	1.000	6.000	6.00	829.00
ES	EP	ATCATC	6	6.000	1.000	6.000	6.00	829.00
ES	EP	CCATGC	6	6.000	1.000	6.000	6.00	829.00
ES	EP	CCCGCG	6	6.000	1.000	6.000	6.00	829.00
ES	EP	CCTGCA	6	6.000	1.000	6.000	6.00	829.00
ES	EP	CGTGCT	6	6.000	1.000	6.000	6.00	829.00
ES	EP	GAGTGG	6	6.000	1.000	6.000	6.00	829.00
ES	EP	GGCGAA	6	6.000	1.000	6.000	6.00	829.00
ES	EP	GTGGAG	6	6.000	1.000	6.000	6.00	829.00
ES	EP	GTGGCC	6	6.000	1.000	6.000	6.00	829.00
ES	EP	TCCTCC	6	6.000	1.000	6.000	6.00	829.00
ES	EP	TGGCTG	6	6.000	1.000	6.000	6.00	829.00
ES	EP	TTCAGC	6	6.000	1.000	6.000	6.00	829.00

1	CATCAT	4
2	AGTGGA	3
3	ATCATC	3
4	CCATGC	3
5	CCCGCG	3
6	CCTGCA	3
7	CGTGCT	3
8	GAGTGG	3
9	GGCGAA	3
10	GTGGAG	3
11	GTGGCC	3
12	TCCTCC	3
13	TGGCTG	3
14	TTCAGC	3

Figure 4.8: Result comparison from the proposed system and online de novo motif finding tools.

Here it shows the motif comparison between two algorithms. One is proposed one and according to our results we can check from de novo tools the frequency of these motif according to their system.

Conclusion *Chapter 5*

5.1 Summary

In this thesis we try to find an efficient algorithm for motif where both the exact match and gapped match are computed using automata. As the basic of automata contains the parallelism of different possible of the path, we are try finding every possible motif in a sequence in a way where we can get the most probable motif with highest frequency in this method.

As with the comparison with other paper, our main focus was to compute with variable length motif for different DNA sequence of Homo sapiens, Mus Musculus With similar data set of some papers. Our goal was to compute the variable motif possibility of different length and find out the most probable one using NFA Approach.

5.2 Future Work

Though our search process in this methods are in binary exponential search, we have tried to minimize it from the exhaustive search of [2], but our main future focus would be reduce the search process using index table so it can store dynamically that minimizes the result.

Bibliography

[1] Yazmín Magallanes, Ivan Olmos : Motifs Recognition in DNA Sequences Comparing the Motif Finding Automaton Algorithm against a Traditional Approach, Universidad de las Américas, Puebla, 2010

[2] Pavlos Antoniou, Jan Holub : Finding Common Motifs with Gaps using Finite Automata, Dept. of Computer Science, King's College London, London, 2006

[3] Tobias Marschall: Construction of minimal DFAs from biological motifs, Bioinformatics for High-Throughput Technologies, Computer Science XI, Dortmund, Germany, December 9, 2010

[4] Grégory Nuel: Significance Score of Motifs in Biological Sequences, Institute for Mathematical Sciences (INSMI), CNRS, Paris, November 2, 2011

[5] Markus Holzer: Nondeterministic Finite Automata-Recent Results on the Descriptional and Computational Complexity, Institut für Informatik Gießen, Hesse, Germany, January 2008

[6] Christopher Sabotta: Advantages and challenges of programming the Micron Automata Processor, Iowa State University, 2013

[7] Tobias Marschall: Algorithms and Statistical Methods for Exact Motif Discovery, Bioinformatics for High-Throughput Technologies, Computer Science XI, Dortmund, Germany, 2014

[8] Indranil Roy and Srinivas Aluru: Finding Motifs in Biological Sequences using the Micron Automata Processor, School of Computational Science and Engineering Georgia Institute of Technology, Atlanta, GA 30332, 2014

[9] L’ubomíra Išto ˇnová: Descriptional Complexity of Finite State Automata , Pavol Jozef Šafárik University in Košice, September 9, 2010

[10] Kayleigh Hyde: NONDETERMINISTIC FINITE STATE COMPLEXITY, University of Hawai’i, Manoa, April 2013

[11] Adnan Ferdous Ashrafi , A.K.M Iqtidar Newaz :A Modified Algorithm For DNA Motif Finding Considering Mutation, Department of Computer Science and Engineering (CSE) Islamic University Of Technology(IUT), 2014

[12] Sabine Broda, Ant_onio Machiavelo: Average Case Complexity of NFA's, Centro de Matem_atica da Universidade do Porto, 2008

[13] Nadia Pisanti_z, Alexandra M. Carvalho: RISOTTO: Fast extraction of motifs with mismatches, Dipartimento di Informatica, Universit_a di Pisa, Italy ,2002

[14] Paolo Ribeca∗ and Emanuele Raineri: Faster exact Markovian probability functions for motif occurrences: a DFA-only approach, Bioinformatics and Genomics Unit, Center for Genomic Regulation, C/ Dr.Aiguader 88, E08003 Barcelona, Spain, page 2839-2848, October 9, 2008

[15] Maxime Crochemore and Christophe Hancart. Automata for matching patterns. In G. Rozenberg and A. Salomaa, editors, Handbook of Formal Languages, volume2, Linear Modeling: Background and Application, chapter 9, pages 399–462. SpringerVerlag, 1997.