ISLAMIC UNIVERSITY OF TECHNOLOGY

Organization of Islamic Cooperation

A STUDY OF MOTIF DISCOVERY ALGORITHMS

A Thesis Submitted to the Academic Faculty in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING

Prepared by

Abdul Aziz (134428)

Md. Abu Taleb Nadim (134441)

Supervised by,

Tareque Mohmud Chowdhury

Assistant Professor

Department of Computer Science and Engineering,

Islamic University of Technology (IUT)

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

Gazipur, Bangladesh

2017

A STUDY OF MOTIF DISCOVERY ALGORITHMS

by

Abdul Aziz (134428)

Md. Abu Taleb Nadim (134441)

A Thesis Submitted to the Academic Faculty in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

Gazipur, Bangladesh

2017

A STUDY OF MOTIF DISCOVERY ALGORITHMS

Approved by:

_____

Tareque Mohmud Chowdhury

Supervisor and Assistant Professor,

Department of Computer Science and Engineering,

Islamic University of Technology (IUT)

Boardbazar, Gazipur-1704.

Date:_____

Candidates' Declaration

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Signature of the candidate

Signature of the candidate

ABDUL AZIZ
Student no. 134428
Department: CSE

MD. ABU TALEB NADIM
Student no. 134441
Department: CSE

Signature of the Supervisor

Tareque Mohmud Chowdhury
Assistant Professor
Department: CSE

ABSTRACT

The problem of identifying meaningful patterns (i.e., motifs) from biological data has been studied extensively due to its paramount importance. Motifs are short similar sequence elements in DNA with a common biological function. These recurring patterns in DNA can indicate sequence-specific binding sites for proteins and hence motif finding has become one of the most widely studied topics in bioinformatics. Motifs can be used to determine evolutionary and functional relationships. Recent advancements in the availability of DNA sequences and Gene-expression analysis technologies have greatly influenced the development of motif finding algorithms. Over the past few years, many motif discovery tools have been designed and make available to public. Motif discovery in unaligned DNA sequences is a challenging problem in computer science and molecular biology. Finding a cluster of numerous similar subsequences in a set of biopolymer sequences is evidence that the subsequences occur not by chance but because they share some biological function. In this report, we present an algorithm on motif discovery developed using Genetic Algorithm (GA). Our algorithm is originally based on a popular motif finding algorithm "Finding Motifs by Genetic Algorithm" (FMGA) developed by Falcon F.M Liu, with a handful of modifications to get better result. In our approach, we try to find potential Motifs from a group of promoter sequences of transcription start site (TSS). The Genetic operations such as mutation, crossover is performed using position weight matrix to reserve the completely conserved position. A rearrangement method is used to avoid the presence of a very stable local minimum. A preprocessing function is used to relate randomly generated initial motifs with the promoter sequences and a discursion function is used to minimize the computational time. We evaluated our result based on a fitness score and occurrence frequency of a candidate motif in a group of promoter sequence. We chose the original FMGA algorithm which showed superior result in comparison to two other Motif finding algorithm namely **Multiple Em for Motif Elicitation (MEME)** ***and*** **Gibbs Sampler.**

Keywords: DNA, Motifs, Motif Finding, Genetic Algorithm, Mutation, Crossover, SP-STAR, WINNOWER, Gibbs Sampling Algorithm, FMGA, MEME

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Overview:

Bioinformatics is the field of studying biological activities of macromolecules, such as carbohydrates, lipids, proteins and nucleic acids, using computational technologies. In general, there are three aims of bioinformatics: the first is to maintain a database (such as protein making HLA sequences) accessible for researchers to analyze it; the second is to develop tools that are helpful for analyzing these datasets and to understand the functions of macromolecules; and the third aim is to use these analysis tools for interpreting biologically meaningful information about the macromolecules.

During the first decade of this century, an early step in the simplification of the use of bioinformatics was the development of workflow management software that allows the integrations of multiple bioinformatics tools. Their implementation made automation and large scale handling of data processing possible.

DNA (Deoxyribo Nucleic Acid) consists of bases of *A (Adenine), T (Thymine), C (Cytosine), G (Guanine).* A long sequence of *A, T, C, G* is called a *DNA sequence*. Every individual in every species has its own unique DNA sequence. The nucleotide sequence of DNA is the most fundamental knowledge of genomic information of species and the way of determining this nucleotide pattern is called DNA sequencing [1].

Advancements in the field of life science have created massive amount of biological data which has ushered a new era of computational biology. Large amounts of data from biological sequencing projects have necessitated the development of efficient algorithms for multiple sequence comparison problems.

The gene is the fundamental unit of inherited information in DNA, and is defined as a section of base sequences that is used as a template for the copying process called transcription. The main

idea in gene expression is that every gene contains the information to produce a protein. Gene expression begins with binding of multiple protein factors, known as transcription factors, to enhancer and promoter sequences. Transcription factors regulate the gene expression by activating or inhibiting the transcription machinery. Unraveling the mechanisms that regulate gene expression is a major challenge in biology. An important task in this challenge is to identify regulatory elements, especially the binding sites in DNA for transcription factors. These binding sites are short DNA segments that are called motifs. Pattern discovery in DNA sequences is one of the most challenging issues in computer science and molecular biology nowadays because motifs exists in different sequences in different mutated forms.

Importance of Multiple Sequence Comparison:

Most multiple sequence comparison problems seek to extract commonalities from a given set of sequences and have become especially important in DNA sequence analysis. As outlined in [2], because DNA codes for the specific characteristics of all living organisms, similarities in multiple organisms' DNA are useful in deducing biological information. Commonalities in different species' DNA, for example, may expose key evolutionary relationships and resolve missing links in phylogenetic trees. Since the structure of biological molecules usually determines their function, and DNA sequences code for the amino acid sequences of proteins, the discovery of preserved DNA sequences facilitates protein function studies, protein categorization, and gene characterization.

Two universal approaches to multiple sequence comparison problems are outlined in [2]. For some problems, biological relationships are deduced once common subsequences are found computationally. Other problems seek to decipher the common molecular subsequences responsible for a known biological relationship.

Motivation:

A DNA motif is defined as a nucleic acid sequence pattern that has some biological significance such as being DNA binding sites for a regulatory protein, i.e., a transcription factor. Normally, the

pattern is fairly short (5 to 20 base pairs (bp) long) and is known to recur in different genes or several times within a gene. DNA motifs are often associated with structural motifs found in proteins. Motifs can occur on both strands of DNA. Transcription factors indeed bind directly on the double-stranded DNA. Sequences could have zero, one, or multiple copies of a Motif.

This first step of gene expression, 'transcription', is finely regulated by a number of different factors, among which 'transcription factors' (TFs) play a key role binding DNA near the transcription start site of genes (in the 'promoter' region). The actual DNA region interacting with and bound by a single TF (called TFBS) usually ranges in size from 8–10 to 16–20 bp. TFs bind the DNA in a sequence-specific fashion, that is, they recognize sequences that are similar but not identical, differing in a few nucleotides from one another. So, given a set of sequences, if we can find an unknown pattern of m letters that occurs frequently in a group of sequences, a simple enumeration of all m-letter patterns that appear in the sequences gives the solution. In the past, binding sites were typically determined through rigorous experiments in the laboratories. That way it was hard to find a potential candidate motif without any prior knowledge. But with the emerging of computational methods, if we can develop an efficient algorithm that can predict potential motif in DNA sequences without any prior knowledge rather by searching, we will be able to explore deeper into the hidden message in DNA.

Problem Statement:

With a view to solving this problem of motif discovery, a large number of algorithms are already developed and applied to various motif models over past decades. Most of these algorithms are designed to deduce motifs by considering the regulatory region (promoter) of several coregulated genes from a single genome. It is assumed that co-expression of genes arises mainly from transcriptional coregulation. As coregulated genes are known to share some similarities in their regulatory mechanism, possibly at transcriptional level, their promoter regions might contain some common motifs that are binding sites for transcription factors.

In this report, we look at some of the already available algorithms and finally we have tried to present a new algorithm "*Finding Motifs using Genetic Algorithms with Fixed Iterations*" (*nFMGA*) that will consider a group of promoter sequence of transcription start site of variable

length from 800 bp to over 11000 bp and create a number of candidate motifs randomly from those sequences. Then after passing the candidate motifs of first generation through a preprocessing, we will allow those candidate motifs to evolve through genetic operations like mutation, crossover for a number of generation and look for potential candidate motifs through all the generation of motifs that produced. To make it more related to biological meaning, we also introduce a deletion process of comparatively weak motifs that has no chance of further evolution. We also used the concept of two types of base pair (Purine (A, G) and Pyridine (T, C)) in the scoring factor as these two types of tends alter within themselves during biological processes fairly frequently.

Research Challenges:

Motif finding in biological sequences is very important as they help the biologists to better understand the structure and functions of the molecules represented by the sequences. Unfortunately, it is no easy task to determine what these motifs are. One cannot simply look under a microscope and see what each part of the DNA strand is doing. The best approach that is currently known is to analyze many strands of DNA and search for patterns because the patterns might be the motifs in question.

For computational biology Motifs carry special significance. Specific regions of DNA work as Transcription Factor Binding sites where a specific proteins known as Transcription Factors bind for subsequent protein and enzyme creations. These sites are often conserved in different sequences and recur in same sequence. Such binding sites are relatively short segments of DNA. Due to higher potential of degeneracy and mutations it is difficult to distinguish these sites from long sequences. Motif finding can be defined as the problem of discovering patterns sometimes without any prior knowledge of how the motifs look or the task of detecting overrepresented motifs as well as conserved motifs from orthologous sequences that are good candidates for being transcription factor binding sites.

In biological applications, it is mandatory to allow for some mismatches between different occurrences of the same motif. In fact, point mutations might have taken place, as well as errors in the sequencing procedure, so that molecules that have the same or related function(s), have no

longer identical sequences. This is what makes the problem difficult from the computational point of view.

As there are only 4 nucleotide bases, finding all possible combinations for a certain length of motif and then finding the matches for each of those combinations might seem relevant but is a very slow and impractical solution for larger length of motifs. It is a simple process, though by no means a quick process, to find all possible string patterns and then determine which pattern has the most identical matches in other DNA strings. However, this process fails to find motifs with small mutations that do not change their function but cause them to be distinct from one another in regards to nucleotide sequence. So we cannot simply search for identical patterns in the strings because the motifs will generally look similar, but not identical. Therefore, different motif finding algorithms have been developed over the years. Several methods have been proposed for identifying motifs in a set of biological sequences.

Scope:

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. When the concerned strings code biological macromolecules, i.e., sequences of DNA, RNA and some proteins, the extracted patterns offer to the biologists numerous tracks, unknown previously, to explore and allow them to face numerous difficult problems. Indeed a motif can help biologists to learn functions of the biological sequences and thus to understand mechanisms and biological processes in which these sequences are involved. In spite of numerous displayed efforts, the problem of searching for patterns stays a challenge, at the same time, for computer scientists and biologists alike. On one hand, the general version of this problem is NP-hard, on the other hand, our incomplete and vague understanding of a number of biological mechanisms does not help us to supply good models for this 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 within motifs, and the few that do, have other limitations. Thus there is still a lot of work available for interested ones.

Objective:

The main objective of this thesis is to develop an algorithm for motif finding which can predict potential motifs with higher occurrence frequency from a set of promoter sequence that can regulate gene expression and be a good candidate for transcription factor binding site.

RELATED WORK

In this section, we will give a short overview of a few methods developed in past two decades that successfully predicted candidate motifs.

Expectation-Maximization (EM) Methods

EM for motif finding was introduced by Lawrence and Reilly [3] and it was an extension of the greedy algorithm for motif finding by Hertz et [4]. It was primarily developed for protein motifs; however, it can also be applied for DNA motif finding. No alignment of the sites is required and the basic model assumption is that each sequence must contain at least one common site. The uncertainty in the location of the sites is handled by employing the missing information principle to develop an EM algorithm. This approach allows for the simultaneous identification of the sites and characterization of the binding motifs. The MEME algorithm by Bailey and Elkan [5] extended the EM algorithm for identifying motifs in unaligned biopolymer sequences. The aim of MEME is to discover new motifs in a set of biopolymer sequences where little is known in advance about any motifs that may be present. MEME incorporated three novel ideas for discovering motifs. First, subsequences that actually occur in the biopolymer sequences are used as starting points for the EM algorithm to increase the probability of finding globally optimum motifs. Second, the assumption that each sequence contains exactly one occurrence of the shared motif is removed. Third, a method for probabilistically erasing shared motifs after they are found is incorporated so that several distinct motifs can be found in the same set of sequences, both when different motifs appear in different sequences and when a single sequence may contain multiple motifs.

SP-STAR Algorithm [6]

One of the approximate algorithms for Planted Motif Search (PMS) is the SP-STAR algorithm which we are going to look at in detail.

Goal: Given a set of DNA sequences, find a set of *l-mers*, one from each sequence, which maximizes the consensus score.

Input: A $(t \times n)$ matrix of DNA, and l, the length of the pattern to find.

Output: An array of starting positions $s = (s_1, s_2, \ldots s_t)$ maximizing Score (s,DNA)

Defining some terms:

1. t : number of input sequences
2. n : length of each sequence
3. *DNA* - sample of DNA sequences $(t \times n)$ array
4. l : length of the motif (also known as the *l-mer)*
5. d : number of mismatch between two sequences
6. s_i : starting position of an *l-mer* in sequence i
7. $s = (s_1, s_2, \ldots\ldots s_t)$: array of motif's starting positions

The Algorithm process:

The algorithm proceeds via two steps.

Step 1 – Initial guessing:-

It uses a sum-of-pair scoring function to find the initial signals.

- The first *l-mer* is taken from the first position of the first sequence and that is considered as the current *l-mer*.
- The closest *l-mer* from the second sequence using the sum-of-pair scoring function for the current *l-mer* is found.
- The closest *l-mer* from the third sequence is found. This process is repeated up to the n^{th} sequence.

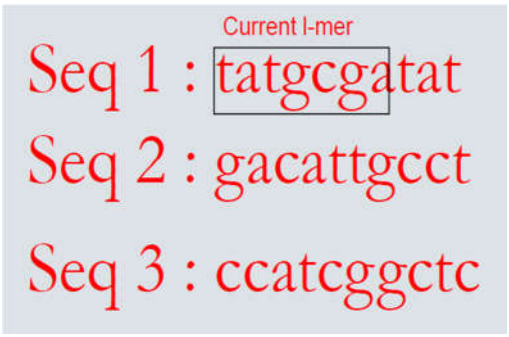


Figure 1: Choosing *l-mer*

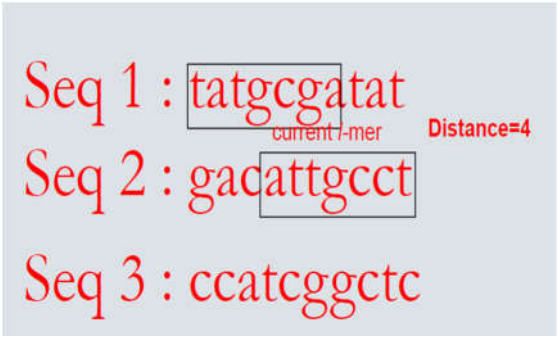


Figure 2: Choosing closest *l-mers*

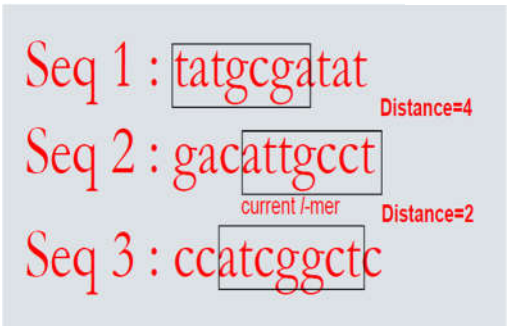


Figure 3: Choosing closest *l-mers*

- These *n* instances, one from each of the sequences forms the group 1 and the consensus and its score is calculated.

Figure 4: Closest matches from all sequences

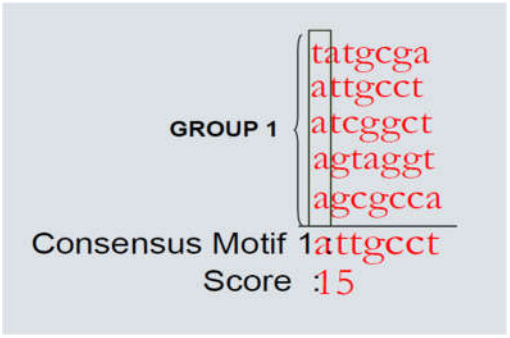


Figure 5: Calculating consensus score

- The above steps are repeated for the second *l-mer* of the first sequence.
- The same process is repeated for the third and fourth *l-mer* of the first sequence and their corresponding consensus and scores are recorded.

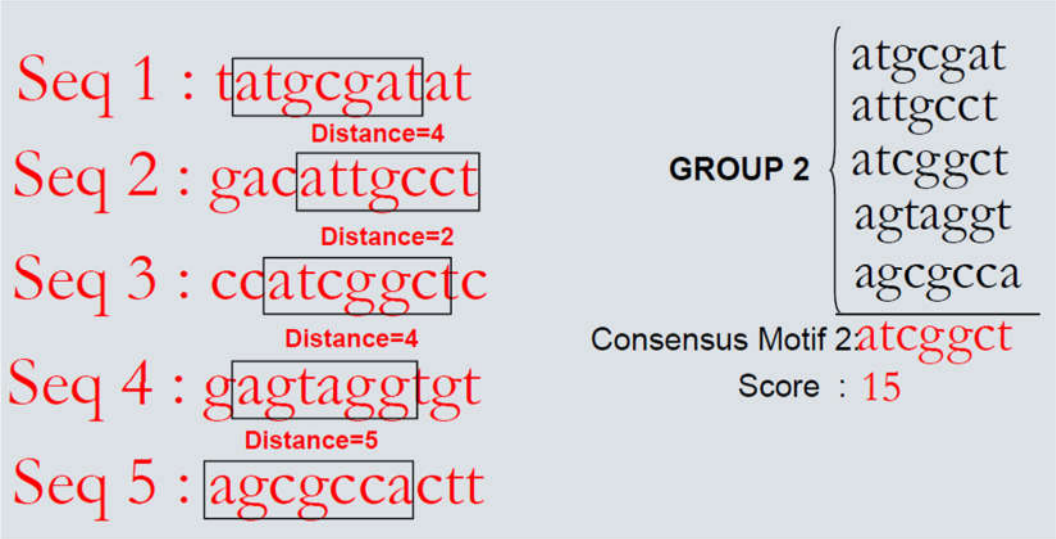


Figure 6: Repeating for all *l-mers*

Step 2 – Refinement step:-

In this step, the initially found signals are improved using a local improvement strategy.

- The best scoring motif from the initially guessed motifs is selected

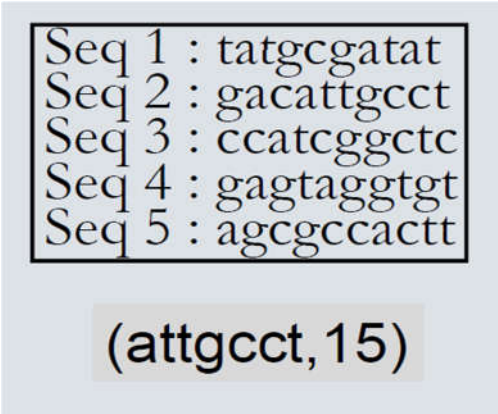
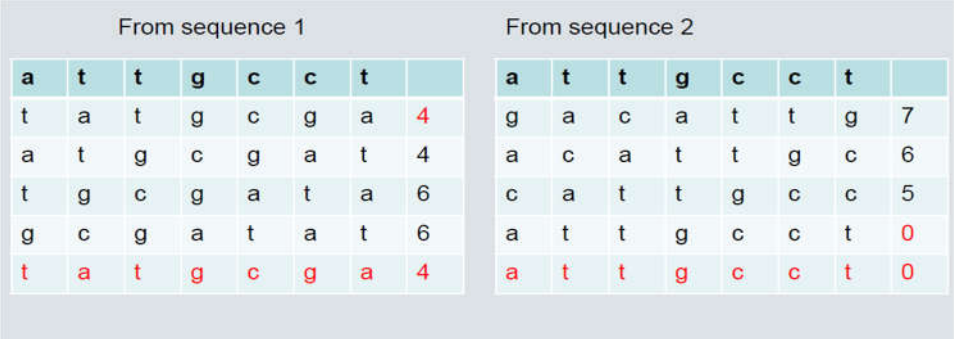


Figure 7: Best scoring motif

- The closest *l-mer* is taken from every sequence by calculating the distance/mismatch and these instances form a new group



a	t	t	g	c	c	t	
t	a	t	g	c	g	a	4
a	t	g	c	g	a	t	4
t	g	c	g	a	t	a	6
g	c	g	a	t	a	t	6
t	a	t	g	c	g	a	4

a	t	t	g	c	c	t	
g	a	c	a	t	t	g	7
a	c	a	t	t	g	c	6
c	a	t	t	g	c	c	5
a	t	t	g	c	c	t	0
a	t	t	g	c	c	t	0

Figure 8: Comparison with all other *l-mers* from each sequence

- The majority string (consensus) and corresponding score for this group is found.

t	a	t	g	c	g	a	4
a	t	t	g	c	c	t	0
a	t	c	g	g	c	t	2
g	t	a	g	g	t	g	5
a	g	c	g	c	c	a	3
a	t	c	g	g	c	t	14

Figure 9: Consensus of new group

- If there is an improvement in the score, we refine again using this new consensus for the next iteration.
- If there is no improvement, we stop and the consensus is the final planted motif.

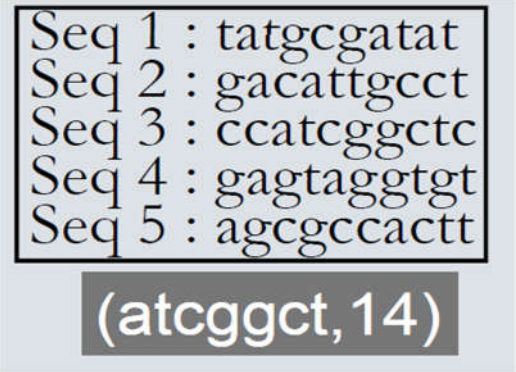


Figure 10: Final result after refinement

Gibbs Sampling For Motif Finding [7 - 8]

Gibbs sampling is a generalized probabilistic inference algorithm used to generate a sequence of samples from a joint probability distribution of two or more random variables. In the arena of bioinformatics, Gibbs sampling is one of several approaches to motif detection, including expectation-maximization approaches. The power of Gibbs sampling is that the joint distribution of the parameters will converge to the joint probability of the parameters given the observed data.

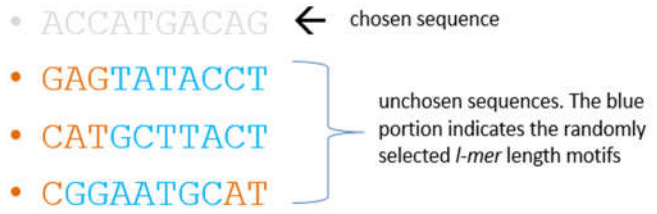
Goal: Given a set of DNA sequences, find a given motif from the set of given DNA sequences.
Input: A ($t \times n$) matrix of DNA, and the pattern to be found.
Output: An array of *l-mers* from each sequences, $s = (s_1, s_2, \ldots s_t)$.

Defining some terms:

1. N : number of sequences
2. $S_1, S_2, \ldots\ldots S_N$: set of sequences
3. l : length of the given motif which is to be searched
4. $c_{i,j,k}$: observed counts of residue j in position i of motif k
5. $q_{i,j}$: frequency of residue j occurring in position i of the motif
6. b_j : pseudocounts for each residue
7. B : summation of the pseudocounts. $B = \sum b_j$

The Algorithm process:

- Given a set of sequences and the required motif to be found from the set.
- A sequence is isolated from the others for sampling.
- From each of the unselected sequences, a random motif position of l length is chosen.
- The residue occurrences for each position for all unchosen sequences is counted and a table of count is created.



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

Table 1: Table of Counts

- Using the formula

$$q_{i,j} = \frac{c_{i,j} + b_j}{N - 1 + B}$$

we calculate frequency counts from the randomly selected motifs

	0	1	2	3	4	5	6	7
A		0.1	0.3	0.3	0.5	0.3	0.1	0.1
C		0.1	0.3	0.1	0.1	0.3	0.5	0.3
G		0.5	0.3	0.1	0.1	0.1	0.3	0.1
T		0.3	0.1	0.5	0.3	0.3	0.1	0.5

Table 2: Frequency Count

- Using the formula

$$q_{0,j} = \frac{c_{0,j} + b_j}{\sum_{k=1}^{j} c_{0,k} + B}$$

we calculate background frequency counts.

	0	1	2	3	4	5	6	7
A	0.31	0.1	0.3	0.3	0.5	0.3	0.1	0.1
C	0.23	0.1	0.3	0.1	0.1	0.3	0.5	0.3
G	0.23	0.5	0.3	0.1	0.1	0.1	0.3	0.1
T	0.23	0.3	0.1	0.5	0.3	0.3	0.1	0.5

Table 3: Background Frequency Count

- The weight for each possible motif position in the chosen sequence is calculated

$$A_1 = \frac{p_{1,\mathrm{A}} \cdot p_{2,\mathrm{C}} \cdot p_{3,\mathrm{C}} \cdot p_{4,\mathrm{A}} \cdot p_{5,\mathrm{T}} \cdot p_{6,\mathrm{G}} \cdot p_{7,\mathrm{A}}}{p_{0,\mathrm{A}} \cdot p_{0,\mathrm{C}} \cdot p_{0,\mathrm{C}} \cdot p_{0,\mathrm{A}} \cdot p_{0,\mathrm{T}} \cdot p_{0,\mathrm{G}} \cdot p_{0,\mathrm{A}}} \approx \frac{0.1 \cdot 0.3 \cdot 0.1 \cdot 0.5 \cdot 0.3 \cdot 0.3 \cdot 0.1}{0.31 \cdot 0.23 \cdot 0.23 \cdot 0.31 \cdot 0.23 \cdot 0.23 \cdot 0.31} \approx 0.16$$

Figure 11: Weight Calculation for the motif positions

- From these weights, we normalize the values and select a random sample from the distribution.

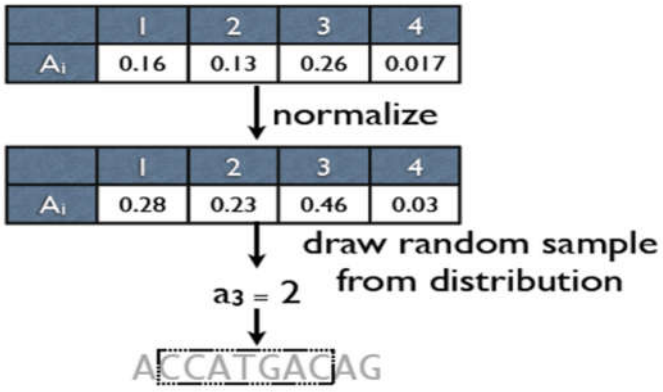


Figure 12: Normalizing weights

- The whole process is repeated using a new chosen sequence for sampling until all sequences have been chosen for sampling.
- Afterwards, the whole process is repeated from the first sequence until convergence.

Figure 13: Final Converged Motifs

WINNOWER Algorithm [6]

WINNOWER is a graph-theoretic approach which represents a motif as a large clique (complete sub graph) and attempt to solve the clique problem efficiently by filtering.

Problem Reduction

Given a set of sequences $S = \{S_1, S_2, \ldots\ldots, S_m\}$ and suppose we are looking for a *(l, d)*-motif. We construct a graph *G* as follows:

- Every vertex in *G* corresponds to a length-*l* word in *S*.
- Consider two words *x* and *y* appear in two different sequences in *S*. *x* and *y* are connected by an edge if their hamming distance is at most *2d*. *2d* is chosen to cater for all possible *(l, d)*-motif, that "sits" in the middle in terms of distance between the two vertex that has distance *2d*.

Notice that the problem of finding a *(l, d)*-motif corresponds to finding a clique of size *m*, where *m* is the number of strings. Thus, the problem of finding motifs is reduced to the problem of finding large cliques. However, finding cliques is a NP-complete problem. Therefore WINNOWER proposes a method to filter edges which definitely do not belong to any large cliques.

Defining some terms:

- A vertex *v* is a neighbor of a clique *C* if it is connected to every vertex in this clique (i.e., *C* U {*v*} is a clique).
- A clique is called extendable if it has at least one neighbor in every part of the multipartite graph *G*.
- An edge is called spurious if it does not belong to any extendable clique of size *k*.

WINNOWER is an iterative algorithm that converges to a collection of extendable cliques by filtering out spurious edges.

The Algorithm Process:

- Construct a graph *G* by:
  - Consider two words *x* and *y* appear in two different sequences in *S*. *x* and *y* are connected by an edge if their hamming distance is at most *2d*.
- Filtering of spurious edges:
  - Filtering weak vertices:
    - Vertices that are not supported by a neighbor in every part of *G* are filtered out.
  - Filtering weak edges:
    - Unsupported edges are removed.
  - Filtering weak triangles.
  - If the computation allow filter more.

Example:

An example is illustrated in the figures below:

Figure 14: Cliques in a string

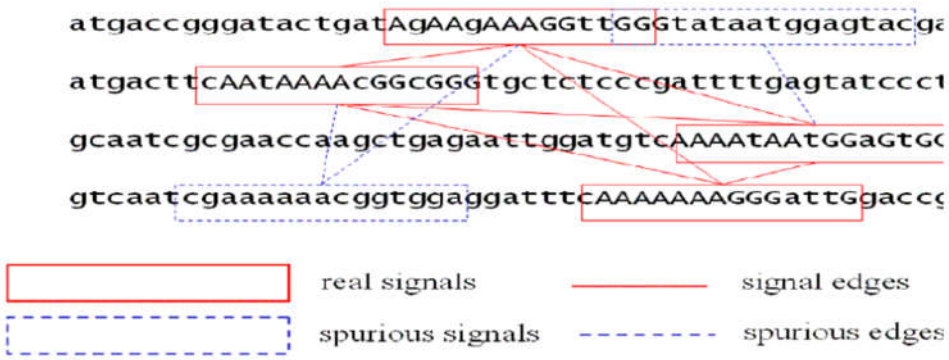


Figure 15: After Filtering

Comparisons among the Different Algorithms [11]

Comparison of performance of the various algorithms for samples with implanted (15, 4)-signals. The sequence length *n* varies from 100 to 1000. Every entry in the table represents the average performance over samples each containing 20 sequences of length *n*.

SEQUENCE LENGTH	**100**	**200**	**300**	**400**	**500**	**800**	**1000**
SP-STAR	0.98s	0.98s	1.00s	0.96s	0.96s	0.69s	0.23s
GIBBS SAMPLING	0.93s	0.96s	0.51s	0.46s	0.29s	0.34s	0.12s
WINNOWER (*t=2)*	0.98s	0.98s	0.97s	0.95s	0.97s	0.02s	0.02s
WINNOWER (*t=3)*	0.98s	0.98s	0.97s	0.94s	0.97s	0.93s	0.88s

Table 4: Time comparison of the algorithms

The following table compares the space consumption and the highest length of motifs that can be found using the algorithms studied [11]:

ALGORITHM	**Highest Length of Motif that can be found**	**Space Consumption**	**Approximate/Exact Algorithm**
SP-STAR	(15,4)	Low	Approximate
GIBBS SAMPLING	N/A as motif is provided	Moderate	Approximate
WINNOWER	(15,4)	Moderate	Approximate

Table 5: Space comparison and limitations of the algorithms

The Genetic Algorithm

In computer science and operations research, a genetic algorithm (GA) [9] is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA) [10]. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection. In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found. The general architecture of GA is illustrated in the figure below. The common operators of GA are as follows:

1. Selection –

   During each successive generation, a portion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming. The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent.

2. Mutation –

   Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to a better solution by using mutation. Mutation occurs during evolution according

to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

For different genome types, different mutation types are suitable. Some types of mutation are as follows:

Bit string mutation: The mutation of bit strings ensues through bit flips at random positions.

Boundary mutation: This mutation operator replaces the genome with either lower or upper bound randomly. This can be used for integer and float genes.

Non-Uniform mutation: The probability that amount of mutation will go to 0 with the next generation is increased by using non-uniform mutation operator. It keeps the population from stagnating in the early stages of the evolution. It tunes solution in later stages of evolution. This mutation operator can only be used for integer and float genes.

Uniform mutation: This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

Gaussian mutation: This operator adds a unit Gaussian distributed random value to the chosen gene. If it falls outside of the user-specified lower or upper bounds for that gene, the new gene value is clipped. This mutation operator can only be used for integer and float genes.

3. Crossover –

In genetic algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Cross over is a process of taking more than one parent solutions and producing a child solution from them. There are methods for selection of the chromosomes. Two types of common crossover are as follows:

Single-point crossover: A single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children.

Two-point crossover: Two-point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms.

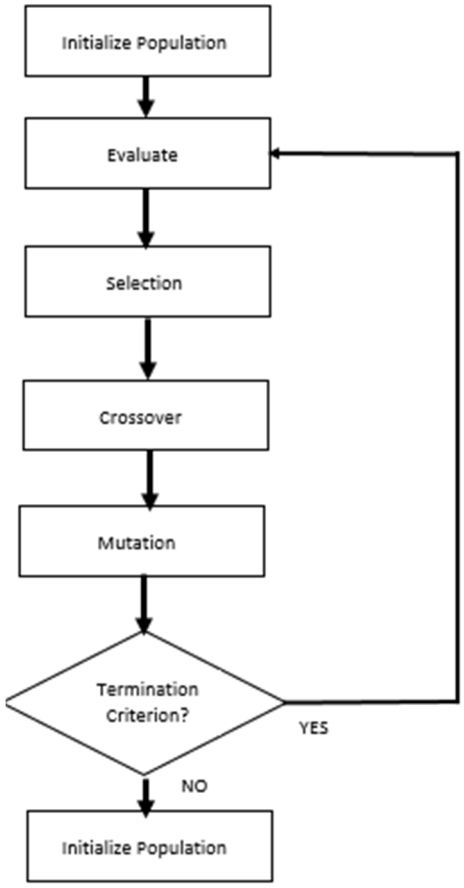


Figure 16: General Architecture of Genetic Algorithm

Finding Motifs using Genetic Algorithm (FMGA)

Liu [12] developed the algorithm FMGA based on genetic algorithms (GAs) for finding potential motifs in the regions located from the -2000 bp upstream to +1000 bp downstream of the transcription start site. The mutation in GA is performed by using position weight matrices to reserve the completely conserved positions. The crossover is implemented with specially designed gap penalties to produce the optimal child pattern. This algorithm also uses a rearrangement method based on position weight matrices to avoid the presence of a very stable local minimum,

which may make it quite difficult for the other operators to generate the optimal pattern. The authors reported that FMGA performs better in comparison to MEME and Gibbs sampler algorithms. As we mentioned earlier, our algorithm is developed based on FMGA with some major modification.

Among these methods, Gibbs sampler has the advantage of spending lower computation time. MEME is superior to the other methods by its prediction accuracy, but has the drawback of taking enormous computation time. In this paper, we propose a new approach based on genetic algorithm to predict motifs. The predicted results obtained by using our approach are more accurate than that of Gibbs sampler and spend less computation time than MEME. But FMGA operate on a limited range of base pair (-2000 upstream to +1000 downstream) which make its scope limited to find optimal motif with better fitness score and occurrence frequency. It has also a limitation of operating on a smaller group of datasets. The flowchart of the original FMGA algorithm is given below:

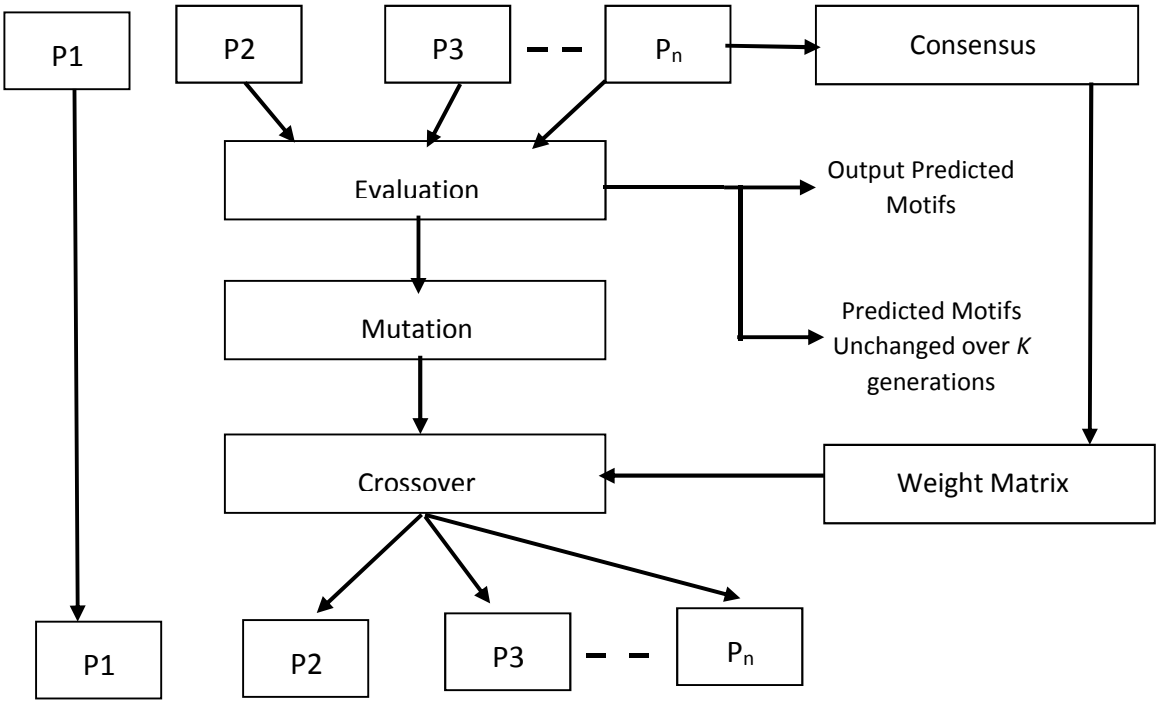


Figure 17: Flow Chart of FMGA

PROPOSED ALGORITHM

Method

Our method to predict motifs is to use a total fitness score function and occurrence frequency to find the optimal motif using genetic algorithm. Apart from the original FMGA algorithm, we introduced a new parameter in calculating the total fitness score on the basis of two types of base pair purine and pyridine. Like the original FMGA, we also use the general genetic algorithm framework and operators to serve as our basic architecture as sequence alignment by genetics algorithm (SAGA) [13] did. In the original FMGA, they only considered IUPAC ambiguity base pair M, R, W, S, K, Y, N to calculate ambiguity code penalty. In the new algorithm, we also include other ambiguity base pairs that includes V, H, D, B to calculate ambiguity codes penalties to charge the scores of consensus sequences so that optimal motifs can be predicted more efficiently. We also introduce a change in the crossover operator where we keep both the child to be evaluated in the next generation. As a result, to keep the increasing number of candidate motif in check, we introduced a discursion function which eliminate comparatively weak motifs from each generation. And there by we removed the risk of a child that could be a better candidate motif, to be eliminated. This is one the major region of getting superior result than the original FMGA.

Evaluation Criteria

We evaluated out proposed algorithm on the basis of two parameters. Total fitness score and occurrence frequency. They are defined as follows:

1. **Fitness Score Function:**

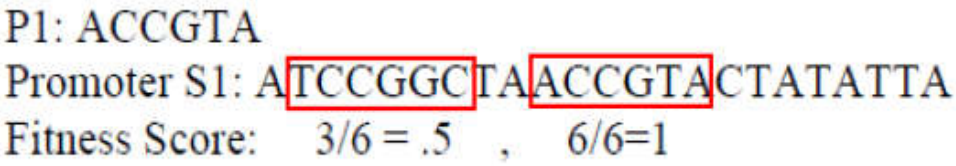
   First let us consider the fitness score for a candidate motif in a single sequence. Given a motif pattern, there may have several regions in the sequence that match the motif pattern and each has a fitness score according to the fitness score function defined as follows:

$$\text{FS}\left(S_{mj,}P_n\right) = \sum_{i=0}^{k} match(\frac{S_{mji,}P_{ni}}{k})$$

Where,

$$\text{Match}\left(S_{mji,}P_{ni}\right) = \begin{cases} 1 \ if \ S_{mji} = P_{ni} \\ 0 \ if \ S_{mji} \neq P_{ni} \end{cases}$$

Here, *m* is the index of sequences, *i* is the position within the motif, *n* is the index of motif patterns, *k* is the length of motif pattern, *j* is number of matched regions in the sequence. For example, the fitness score calculation of a motif P_1 in a promoter sequence S_1 is illustrated in the following figure:



In this case, we will take the fitness score of highest value. So finally, the fitness score function for a single sequence can be given by

$$\text{FS}\left(S_{mj,}P_n\right) = max\left\{\sum_{i=0}^{k} match\left(\frac{S_{mji,}P_{ni}}{k}\right)\right\}$$

In our new algorithm, we take into consideration the two kind of base pair purine and pyridine as they tend to convert into each other frequently. So, these two types of base are represented by ambiguity base pair R (A, G) and Y (T, C). We also included the reaming ambiguity base pairs V, H, B, D that were grouped with N in the original FMGA with a small weightage. Giving R, Y more importance, we redefined out fitness score function as follows:

$$\text{Match}\left(S_{mji,}P_{ni}\right) = \begin{cases} 1 \ if \ S_{mji} = P_{ni} \ for \ P_{ni} \in (A,T,G,C) \\ 0.5 \ if \ S_{mji} = P_{ni} \ for \ P_{ni} \in (R,Y) \\ 0.2 \ if \ S_{mji} = P_{ni} \ for \ P_{ni} \in (M,W,S,K) \\ 0.1 \ if \ S_{mji} = P_{ni} \ for \ P_{ni} \in (B,V,H,D) \\ 0 \ if \ S_{mji} \neq P_{ni} \end{cases}$$

2. Total Fitness Score:

Total fitness score of a motif is the summation of fitness score of best match from all the sequences. It represents the score of motif in a particular generation. The total fitness score is defined as follows:

$$\text{TFS}\left(S, P_n\right) = \sum_{m=1}^{L} FS(S_m, P_n)$$

For example, if we consider a candidate motif P1 and 5 promoter sequences S1 ~ S5 then the calculation of total fitness score of the motif is illustrated in the following figure:

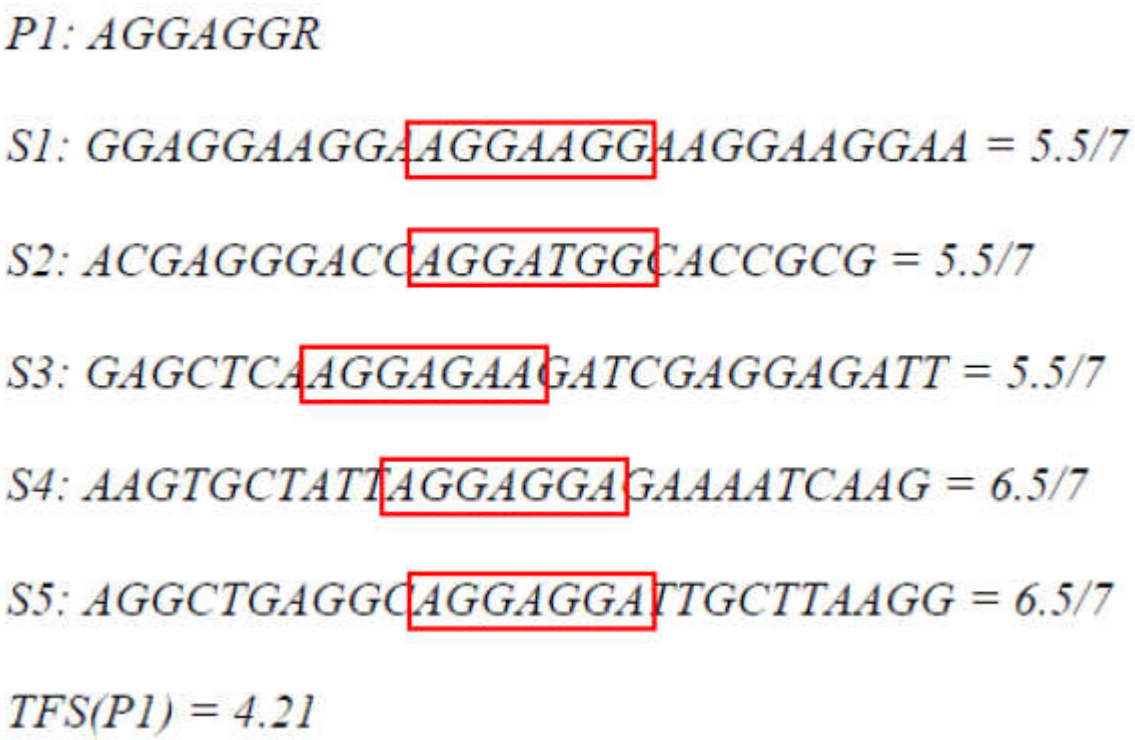


Figure 18: Calculating Total Fitness Score

Proposed Algorithm

We introduce our new algorithm *nFMGA* in this section. Data is selected with an 800 bp length to over 11000 bp length from Transcription Start sites of different genome. In *nFMGA*, the initial motif patterns with the same pattern length are created randomly. The users can set the pattern length. All the motif patterns will be different from the initial patterns after N-generation evolution. Preprocessing the first generation make the randomly generated motifs more suitable for evolution.

Despite of starting with some randomly generated motifs, we preprocess the motifs and convert into worst possible motifs and start evolving. Mutation using weight matrix speed up the time to find potential motifs. During crossover unlike the original FMGA algorithm, we keep both the child to be evaluated on the next generation with other candidate motifs. This is to avoid the possibility of losing a globally potential motif. To check the increasing number of candidate motif, we use the discursion function with a variable discursion factor. The flowchart of *nFMGA* is illustrated in figure:

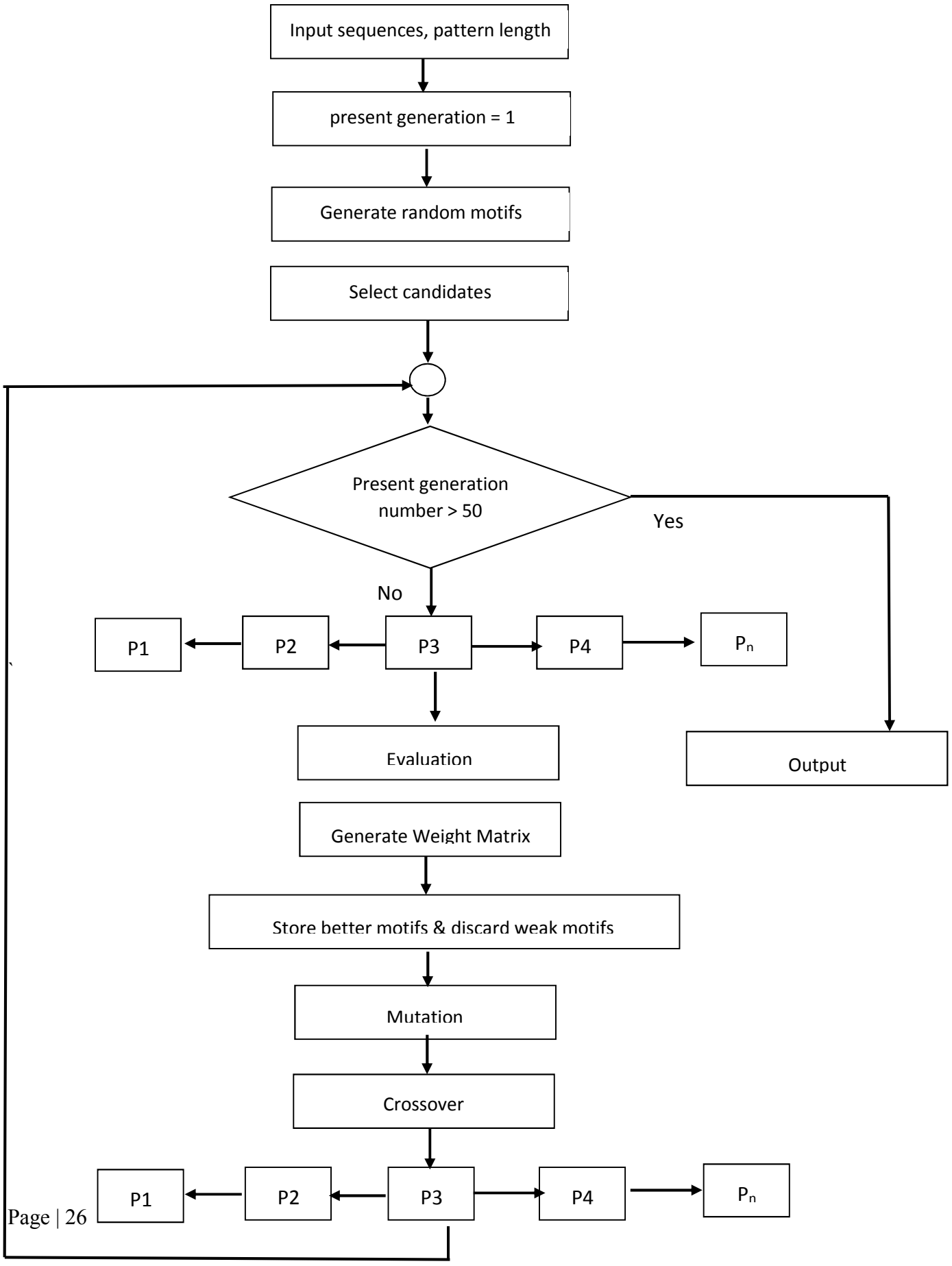


Figure 19: Flowchart of proposed algorithm (*nFMGA*)

Operations

The essential operations in *nFMGA* are Mutation, Crossover and Discursion that can speed up the algorithm at the same time predict better motif. We introduce the operations as follows:
The weight matrix is generated from the best matched pattern from each sequence cross ponding to a particular candidate motif. An illustration of creating a weight matrix is shown in figure 8.
Let us consider 5 best matched pattern of a candidate motif P1 is given by, TGACGCA, TGACGCA, AGACGCA, TGACACA and AGACGCA. The score in weight matrix is calculated as the ratio of occurrences of corresponding base and the numbers of matched motifs. For example, in column 1 of the table, the numbers of occurrences of A are 2, the numbers of matched motif patterns are 5. So, the value is equal to 2/5 = 0.4.

	1	2	3	4	5	6	7
A	0.4	0	1	0	0.2	0	1
T	0.6	0	0	0	0	0	0
G	0	1	0	0	0.8	0	0
C	0	0	0	1	0	1	0

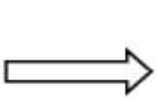
Table 6: Weight Matrix

After generating the weight matrix, we insert ambiguity code on the basis of values in the weight matrix. The column with value 1 remain unchanged and the remaining base pair is changed into ambiguity code based on occurrence of the cross ponding base pairs.

Mutation

The aim of mutation is to create two parent motifs from a candidate motif to speed up the process of finding potential motif. It helps the crossover operation to create two child motifs for further evolution. The mutation is done based on the weight matrix of a candidate pattern. First, we keep the base pair with value of 1 in a column unchanged and the rest of the base pairs are changed randomly. In our algorithm, for the first parent we use the base pair cross ponding to maximum value in a column. For the 2nd parent we use the base pair cross ponding to 2nd highest value in the column. An illustration of mutation is as follows: Consider a weight matrix of a candidate pattern P1:

	0	1	2	3	4	5	6
A	.4	0	1	0	.2	0	1
T	.6	0	0	0	0	0	0
G	0	1	0	0	.8	0	0
C	0	0	0	1	0	1	0



Parent 1: TGACGCA

Parent 2: AGACACA

Figure 20: Mutation

Crossover

The aim of crossover is to create two child motifs from two parent motifs produced in mutation. There are different kinds of crossover in biology. Here we will implement single point crossover. First, we cut the two parent at the middle position. Then join left side of 1st parent with the right side of 2nd parent to create the 1st child motif. Then the right side of 1st parent is joined with the left side of 2nd parent to create the 2nd child motif. The illustration of crossover is shown in the following figure:

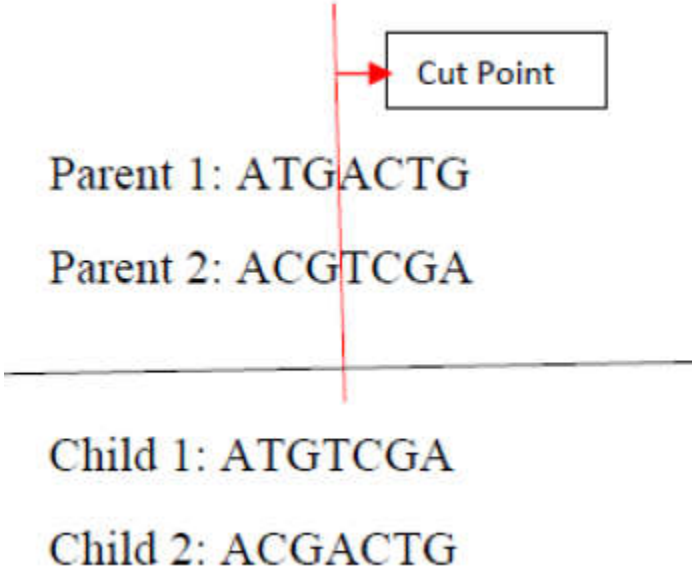


Figure 21: Crossover

Discursion

As we are keeping both the child produced by crossover, the number of candidate motif will keep multiplying in every generation. So, if we don't take any measure in reducing the number of candidate motif somehow, the calculation time of each generation will increase exponentially. To minimize the time complexity, after keeping as much motifs as possible, we introduce a discursion function associated by a discursion factor, in our algorithm. Initially the function will discard a little amount of candidate motif that has a low TFS value and less occurrence frequency from each generation. With the increase of candidate motif in each generation, the discursion factor will adjust itself to keep the number of motifs low. At the least the discursion function will never let the number of motifs to increase more than 4 times than the initial number of candidate motifs. To avoid discarding any potential candidate motif, we avoid discarding any candidate motif that has further chance of evolution through mutation and crossover.

CONCLUDING REMARKS

We had shown that *nFMGA* predicts better motif patterns than FMGA which itself is superior to two other popular algorithm MM and Gibbs Sampler. FMGA can predict more potential motifs than the other algorithms because the patterns are generated randomly during the operation processes of GA. This characteristic is used to overcome the possible problems of local minimum. But as the motifs are generated completely at random, so they have a very little chance of being related to the given sequences. As a result, the algorithm loses efficiency. In our algorithm, after generating random motifs, we preprocessed them on the basis of given sequences and make them related to given sequences. So, it can predict motif more accurately. In FMGA they also removed one child during the process of crossover. Now if both the children of a single motif are weaker, on the other hand if both the children of another motif is stronger, in FMGA, it will lose a strong child to a weak one. To come over this limitation, in our algorithm we keep both the children of all the patterns of a single generation and evaluate them as a whole in the next generation. Due to this reason, the number of candidate motifs increases in every generation which will increase the computation time of *nFMGA*. To overcome that problem, we introduced a discursion function to keep the increasing number of candidate motifs in check. The discursion function is associated with a discursion factor. Tuning of discursion factor affect the final result to a great extent. If we keep the discursion factor high, that will increase the predicting accuracy of *nFMGA* but will increase the computation time proportionally. As for the initial number of randomly generated candidate motifs, if we increase the number, that will give better result as well but will have the same drawback. Genetic algorithm solves the optimal problem based on the biological characteristics. It uses a simple way to cope with complex problems. In this paper, we had proposed a new approach to predict motifs based on the genetic algorithm. A lot of biological messages are hidden in promoter, and motif is one of the important messages. The motifs have the possibilities to be the binding sites of transcription factors. If the motifs can be predicted accurately, the biologists can then explore which transcription factors activate genes. In future *nFMGA* can contribute a lot in this sector.

FUTURE PROSPECT

- We will try to implement this in the future and test our algorithms with others in real life.
- We will try to improve the computation time and accuracy.
- In the future, we will try to implement MDGA in a distributed parallel computing system to overcome the problem of a huge computation time. In that way, we will be able to extract even better result from MDGA.
- In the process of crossover, we used the primary single point crossover. In future, we will try to implement double point and uniform crossover to get even better result.

REFERENCES

1. F. Sanger, F. Nicklen and A. R. Coulson, "*DNA sequencing with chain terminating inhibitors*", Proc. Natl. Acad. Sci. 74:5463-5467, December, 1977.
2. D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. New York, NY: Press Syndicate of the University of Cambridge, 1999. pp. 212 – 214, 217 – 223, 332 – 336.
3. Lawrence CE, Reilly AA: An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. Proteins 1990, 7:41-51.
4. Hertz GZ, Hartzell GW, Stormo GD: Identification of consensus patterns in unaligned DNA sequences known to be functionally related. Comput Appl Biosci 1990, 6:81-92.
5. Bailey TL, Elkan C: Unsupervised learning of multiple motifs in biopolymers using expectation maximization. Machine Learning 1995, 21:51-80.
6. P. Pevzner and S.-H. Sze, *Combinatorial approaches to finding subtle signals in DNA sequences*, Proc. Eighth International Conference on Intelligent Systems for Molecular Biology, 2000, pp. 269-278.
7. http://www.cs.uoi.gr/~kblekas/courses/bioinformatics/gibbs-example.pdf
8. Rouchka, E. (2008). Bioinformatics review: A brief overview of Gibbs sampling (No. TR-ULBL-2008-02). University of Louisville Bioinformatics Laboratory Technical Report Series. University of Louisville. Retrieved from http://bioinformatics.louisville.edu/lab/localresources/papers/TR-ULBL-2008-02.pdf
9. https://en.wikipedia.org/wiki/Genetic_algorithm
10. https://en.wikipedia.org/wiki/Evolutionary_algorithm
11. Faisal Bin Ashraf and Ali Imam Abir, "*DNA Motif Finding Algorithm*", Islamic University of Technology (IUT), 2016. pp. 23.
12. Liu FFM, Tsai JJP, Chen RM, Chen SN, Shih SH: FMGA: finding motifs by genetic algorithm. Fourth IEEE Symposium on Bioinformatics and Bioengineering 2004:459.
13. Cedric Notredame and Desmond G. Higgins, "SAGA: Sequence alignment by genetic algorithm," J. Nucleic Acids Research, 24, pp. 1515-1524, 1996.