Shahjalal University of Science and Technology
Department of Computer Science and Engineering

Motif Finding Algorithm

Supervised By:

Mr. Sabir Ismail
Lecturer
Department of Computer Science & Engineering
Shahjalal University of Science & Technology

Submitted By:

Md. Imran Hossain Showrov
Reg. No: 2010331071

Priyam Pritim Paul
Reg. No: 2009331013

28th March, 2015

Motif Finding Algorithm

A Thesis submitted to the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering.

Supervised By:

Mr. Sabir Ismail
Lecturer
Department of Computer Science & Engineering
Shahjalal University of Science & Technology

Submitted By:

Md. Imran Hossain Showrov
Reg. No: 2010331071

Priyam Pritim Paul
Reg. No: 2009331013

28th March, 2015

Recommendation Letter from Thesis Supervisor

This Student, **Md. Imran Hossain Showrov** and **Priyam Pritim Paul**, whose thesis entitled **"Motif Finding Algorithm"**, is under my supervision and agree to submit for examination.

Advisor :
Date :

Qualification Form of Bachelor Degree

Student Name	:	Md. Imran Hossain Showrov Priyam Pritim Paul
Thesis Title	:	Motif Finding Algorithm

This is to certify that the thesis submitted by the student named above in March, 2015. It is qualified and approved by the Thesis Examination Committee.

Head of the Dept. Chairman, Thesis Committee Supervisor

Abstract

Finding Motifs in DNA is an important computational problem. It allows the discovery of patterns in biological sequences in order to better understand the structure and function of the molecules the sequences represent. Motifs can differ slightly from one gene to the next. When a sequence motif appears in the exon of a gene, it may encode the "structural motif" of a protein; that is a stereotypical element of the overall structure of the protein. Nevertheless, motifs need not be associated with a distinctive structure. If we find the motifs, we can easily differ from one gene to another from parents to its ancestors. Recent advances in genome sequence availability and in high-throughput gene expression analysis technologies have allowed for the development of computational methods for motif finding. We try to implement some algorithms and measure their performance.

Keywords: Motif, Word-based algorithms, Probabilistic algorithms, Genetic Algorithm, EM methods, Gibbs sampling methods, transcription factors, palindromic motifs, spaced dyad, Position Weight Matrix, Suffix tree, Sequence alignment.

Acknowledgement

This report is the result of the Course No- CSE 400 (Project) in Computer Science & Engineering at Shahjalal University of Science & Technology, Sylhet, Bangladesh. We would like to thank from the bottom of our heart, to the people who have encouraged us to be here. We are grateful to our supervisor Mr. Sabir Ismail, Lecturer, Dept. of CSE, SUST, who is very careful, resourceful and a dedicated advisor. His influence was constant and profound. We have learnt a lot of things from his direction & discussion.

Table of Contents

Page

List of Figures

Chapter 1

Introduction

Biological sequence motifs are short sequence of patterns. A lot of features of DNA, RNA and protein molecule can be well approximated by motif. Often they indicate sequence-specific binding sites for proteins such as nucleases and transcription factors (TF). Motifs can be used to determine evolutionary and functional relationships.

Discovery of motifs is an important because it allows the discovery of patterns in biological sequences for better understanding about the structure and the function of the molecules the sequences represent. 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. In addition to the common forms of DNA motifs two special types of DNA motifs are recognized: palindromic motifs and spaced dyad (gapped) motifs. A palindromic motif is a subsequence that is exactly the same as its own reverse complement. A spaced dyad motif consists of two smaller conserved sites separated by a spacer (gap).

Over the past few years, many motif discovery tools have been designed and make available to public. They differ each other mostly in their definition of what constitutes a motif, what constitutes statistical overrepresentation of a motif and what method has been used to find statistically overrepresented motifs.

A large number of algorithms for finding DNA motifs have been developed. Existing algorithms run in time linear with respect to the input size. Nevertheless, the output size can be very large due to the approximation. This often makes the output unreadable. Sometimes these algorithms detect overrepresented motifs and conserved motifs that might be good candidates for being transcription factor binding sites. Algorithms that detect overrepresented motifs perform not as well in higher organisms. To overcome this, some algorithms consider conserved motifs from orthologous species. Recent algorithms have also combined the two approaches to achieve improvement in motif finding.

Chapter 2

Background Study

Different motif finding algorithms are being described. Based on the type of DNA sequence information employed by the algorithm to deduce the motifs, Modan K Das and Ho-Kwok Dai classify available motif finding algorithms into three major classes:

(1) Those that use promoter sequences from coregulated genes from a single genome,
(2) Those that use orthologous promoter sequences of a single gene from multiple species and
(3) Those that use promoter sequences of coregulated genes as well as phylogenetic Footprinting.

However, most of the earlier literature categorized motif finding algorithms into two major groups based on the combinatorial approach used in their design:

(1) word-based methods and
(2) Probabilistic sequence models.

Word-based methods are mostly rely on exhaustive enumeration , counting and comparing oligonucleotide frequencies and the model parameters of probabilistic sequence models are estimated using maximum-likelihood principle or Bayesian inference The word-based enumerative methods guarantee global optimality and they are appropriate for short motifs and are therefore useful for motif finding in eukaryotic genomes where motifs are generally shorter than prokaryotes. The word-based methods can also be very fast when implemented with optimized data structures such as suffix trees and is a good choice for finding totally constrained motifs.

However, for typical transcription factor motifs that often have several weakly constrained positions, word-based methods can be problematic and the result often needs to be post-processed with some clustering system. Word-based methods also suffer from the problem of producing too many spurious motifs. The probabilistic approach involves representation of the motif model by a position weight matrix. Position weight matrices are often visualized as a pictogram in which each position is represented by a stack of letters whose height is proportional to the information content of that position. Probabilistic methods have the advantage of requiring few search parameters but rely on probabilistic models of the regulatory regions, which can be very sensitive with respect to small changes in the input data.

Many of the algorithms developed from the probabilistic approach are designed to find longer or more general motifs than are required for transcription factor binding sites. Therefore, they are more appropriate for motif finding in prokaryotes, where the motifs are generally longer than eukaryotes. However, these algorithms are not guaranteed to find globally optimal solutions, since they employ some form of local search, such as Gibbs sampling, expectation maximization (EM) or greedy algorithms that may converge to a locally optimal solution.

A lot of scientists try to improve motif finding algorithm. Therefore, biologists and computer scientists have been very interested in identifying computational tools for motif finding. With the advent of availability of large scale genome sequencing and high-throughput gene expression analysis techniques, a large number of motif finding tools have been designed and implemented over the past decade.

Liu *et al*. developed the algorithm FMGA based on genetic algorithms (GAs) for finding potential motifs in the 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.

Thijs *et al*. developed the motif finding algorithm MotifSampler using a modification of the original Gibbs sampling algorithm. The two major modifications are

(1) the use of a probability distribution to estimate the number of copies of the motif in a sequence and
(2) the incorporation of a higher-order Markov-chain background model.

Shida developed the motif discovery algorithm GibbsST using the method of simulated tempering with Gibbs sampling. Gibbs sampling is one of the most promising pattern discovery methods in terms of its flexibility and wide range of application, however, it is known to be rather strongly affected by the local optima problem. Therefore, the Gibbs sampling method can be further improved by a search method in the solution space. In pattern discovery and bioinformatics in general, the simulated annealing method is mostly used for improvement of search methods in the solution space. However, satisfactory improvements were not obtained using this method. Simulated tempering is one of many proposals from the field of thermodynamics for the systematic avoidance of local optima in multivariate optimization problems and is quite useful for reducing the vulnerability of Gibbs sampling to local optima.

Kingsford *et al*. used a mathematical programming approach for DNA motif discovery that involved finding subsequences of a given length such that the sum of their pairwise distances was minimized. They used integer linear programming (ILP) that utilizes the discrete nature of the distance metric imposed on pairs of subsequences. Since finding a solution to the ILP is computationally difficult, the authors tightened the linear programming relaxation by adding an exponential set of constraints and used an efficient separation algorithm that can find violated constraints and thus having a polynomial time solution. The authors tested the effectiveness of their approach in identifying DNA motifs in *E. coli* and demonstrated that the performance of their method is competitive with some Gibbs sampling-based algorithms for motif finding.

There are other algorithms, for example, that combine the word-based methods and probabilistic approaches, like the MDScan algorithm. The TAMO algorithm runs multiple motif discovery algorithms (MEME, AlignACE and MDscan) and combines the results [20]. Other approaches are based on other machine learning techniques, neural networks, and clustering algorithms. Algorithms based on phylogenetic foot-printing has the advantage of the co-regulated gene approach is that co-regulated methods require a way for identifying co-regulated genes; phylogenetic foot-printing approach is possible to identify motifs specific to even a single gene as long as they are sufficiently conserved across the many orthologous sequence considered. There are also algorithms that are based on promoter sequences of co-regulated genes and phylogenetic foot-printing. These algorithms are often packaged into user-friendly interfaces, either on web servers or toolboxes. For example, the user-friendly interface, Toolbox of Motif Discovery (Tmod), integrates 12 widely used motif discovery programs: MDscan, BioProspector, AlignACE, Gibbs Motif Sampler, MEME, CONSENSUS, MotifRegressor, GLAM, MotifSampler, SeSiMCMC, Weeder and YMF.

Chapter 3

Methodology

For discovery of motifs, we have studied some algorithms and implemented them.

3.1 Position weight matrix

Position weight matrix (PWM) is a probabilistic algorithm. It is not only one of the most widely used bioinformatics methods, but also a key component in more advanced computational algorithms (e.g., Gibbs sampler) for characterizing and discovering motifs in nucleotide or amino acid sequences. However, few generally applicable statistical tests are available for evaluating the significance of site patterns, PWM, and PWM scores (PWMS) of putative motifs. Statistical significance tests of the PWM output, that is, site-specific frequencies, PWM itself, and PWMS, are in disparate sources and have never been collected in a single paper, with the consequence that many implementations of PWM do not include any significance test.

The multiple comparison problem associated with the test of site-specific frequencies is best handled by false discovery rate methods. The test of PWM, due to the use of pseudo counts, is best done by resampling methods. The test of individual PWMS for each sequence segment should be based on the extreme value distribution.

We have sequence set

For example, given the following DNA sequences:

```
GGTGTGGGGA
CAGGGGTGTG
GGGACAGGGG
TCTGGGGACA
GGGGTGTGGG
GACAGGGGTC
CTGGGGACAG
GGGTGTGGGG
ATAGGGGTGT
GGGGACAGGG
```

The corresponding PFM is:

$$M = \begin{array}{c} A \\ C \\ G \\ T \end{array} \begin{bmatrix} 1 & 2 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 6 & 5 & 6 & 7 & 6 & 7 & 6 & 7 & 6 & 6 \\ 1 & 2 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \end{bmatrix}.$$

Therefore the resulting PPM is:

$$M = \begin{array}{c} A \\ C \\ G \\ T \end{array} \begin{bmatrix} 0.1 & 0.2 & 0.1 & 0.2 & 0.1 & 0.1 & 0.2 & 0.1 & 0.1 & 0.2 \\ 0.2 & 0.1 & 0.1 & 0.0 & 0.1 & 0.1 & 0.0 & 0.1 & 0.1 & 0.1 \\ 0.6 & 0.5 & 0.6 & 0.7 & 0.6 & 0.7 & 0.6 & 0.7 & 0.6 & 0.6 \\ 0.1 & 0.2 & 0.2 & 0.1 & 0.2 & 0.1 & 0.2 & 0.1 & 0.2 & 0.1 \end{bmatrix}.$$

The probability of the sequence *S* = GGTGTGGGGA given the above PPM M can be calculated:

p(S | M) = 0.3 x 0.5 x 0.2 x 0.7 x 0.2 x 0.7 x 0.6 x 0.7 x 0.6 x 0.2 = 0.0002964

From this way we can find the motif. When the PWM elements are calculated using log likelihoods, the score of a sequence can be calculated by adding rather than multiplying the relevant values at each position in the PWM. The sequence score gives an indication of how different the sequence is from a random sequence. The score is 0 if the sequence has the same probability of being a functional site and of being a random site. The score is greater than 0 if it is more likely to be a functional site than a random site, and less than 0 if it is more likely to be a random site than a functional site. The sequence score can also be interpreted in a physical framework as the binding energy for that sequence.

This work is done for 100 length sequence. PWM works well for small dataset. But if the dataset is larger, then the complexity becomes higher.

3.2 Suffix Tree

It’s a word based algorithm. A suffix tree is a data structure that exposes the interval structure of a string in a very deep meaningful way. A suffix tree τ for a string x = x1, . . . , xn is a rooted directed tree with exactly n leaves numbered from 1 to n.

Suffix tree is another way to find sequence motifs in DNA.

3.2.1 Algorithm of Suffix tree

Algorithm 1: constructing the pseudo suffix tree algorithm.

```
Procedure CheckPattern($v$, $u$, $d$, $\varepsilon$, $depth$, $\ell$, $p$)
Var $i$, $j$, $r$ : Integer ; $b$ : Character ; $y$ : TreeNode ;
Begin
    If ($u \neq v$) Then Begin
        If ($d < \varepsilon \times depth$) or ($d = 0$) Then Begin
            If $depth < \ell$ Then Begin
                $b = p[depth]$ ;
                For each child $y$ of node $u$ Do Begin
                    If ($label(y) = b$) Then
                        CheckPattern($v$, $y$, $d$, $\varepsilon$, $depth + 1$, $\ell$, $p$); ;
                    Else
                        CheckPattern($v$, $y$, $d + 1$, $\varepsilon$, $depth + 1$, $\ell$, $p$); ;
                End ;
            End
            Else Begin
                $mis\text{-}postvector[v] = mis\text{-}postvector[v] \cup self\text{-}postvector[u]$ ;
                $mis\text{-}postvector[u] = mis\text{-}postvector[u] \cup self\text{-}postvector[v]$ ;
            End ;
        End ;
    End ;
End ;
```

Algorithm 2: Potential motif searching algorithm.

```
Procedure FindPotentialMotif($v$, $depth$, $\ell$)
Var $p$ : String ; $y$ : TreeNode ;
Begin
    if ($depth < \ell$) Then Begin
        For each child $y$ of node $v$ Do
            FindPotentialMotif($y$, $depth + 1$, $\ell$) ;
    End ;
    $p$ = A subsequence from $root$ to $v$ ;
    CheckPattern($v$, $root$, 0, $\varepsilon$, 0, $\ell$, $p$);
    CheckPattern($v$, $root$, 0, $\varepsilon$, 0, $\ell$, $p^{rc}$);
End ;
```

Algorithm 3: The checking pattern algorithm.

```
Procedure CheckPattern($v$, $u$, $d$, $\varepsilon$, $depth$, $\ell$, $p$)
Var $i$, $j$, $r$ : Integer ; $b$ : Character ; $y$ : TreeNode ;
Begin
    If ($u \neq v$) Then Begin
        If ($d < \varepsilon \times depth$) or ($d = 0$) Then Begin
            If $depth < \ell$ Then Begin
                $b = p[depth]$ ;
                For each child $y$ of node $u$ Do Begin
                    If ($label(y) = b$) Then
                        CheckPattern($v$, $y$, $d$, $\varepsilon$, $depth + 1$, $\ell$, $p$); ;
                    Else
                        CheckPattern($v$, $y$, $d + 1$, $\varepsilon$, $depth + 1$, $\ell$, $p$); ;
                End ;
            End
            Else Begin
                $mis\text{-}postvector[v] = mis\text{-}postvector[v] \cup self\text{-}postvector[u]$ ;
                $mis\text{-}postvector[u] = mis\text{-}postvector[u] \cup self\text{-}postvector[v]$ ;
            End ;
        End ;
    End ;
End ;
```

3.2.2 Scoring and motif selection

Assume the set M = {U1, U2,....., Um} includes all motif instance. The scoring schema for scoring these motif instance sets are based on information content (IC), this means that each motif instance set is scored separately with regard to the background (sequence set S). Thus, for the motif instance set Ui = {ui,1,..., ui,q}, the corresponding position frequency matrix Wi is constructed. Later the IC value corresponding to Wi is computed as follows:

$$IC(W_i) = \sum_{\alpha \in \{A,C,G,T\}} \sum_{j=1}^{\ell} W_i[\alpha, j] \log(\frac{W_i[\alpha, j]}{\omega[\alpha]}).$$

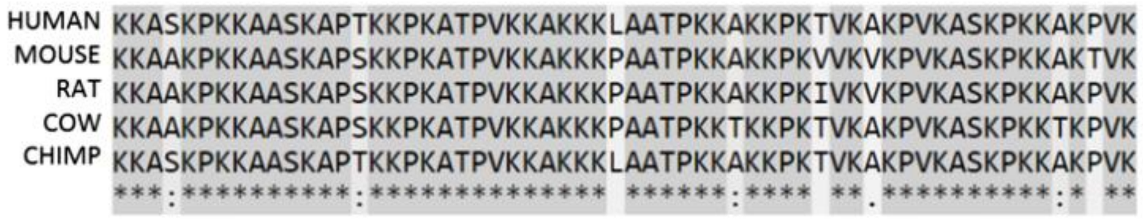
where Wi[α, j] is equal to the occurrence probability of character α ∈ {A, C, G, T} in j-th position of the motif instance set Ui, and ω[α] is the occurrence probability of character α in the set S. All the obtained motif instance sets are sorted based on this score. Finally the top 10 consensus motifs between the obtained motif instance sets are investigated for reporting as motifs.

3.2.3 Limitations

If we do more than a single comparison in an iteration then max (l, r) grows by 1 for each comparison on O(log n + m) time. For large sequence and if we want to find more than 1 motifs then the complexity is too high.

3.3 Sequence alignment

Sequential alignment means the way of arranging the sequence of DNA, RNA or proteins to identify the regions of similarity.



```
HUMAN KKASKPKKAASKAPTKKPKATPVKKAKKKLAATPKKAKKPKTVKAKPVKASKPKKAKPVK
MOUSE KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKAKKPKVVKVKPVKASKPKKAKTVK
  RAT KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKAKKPKIVKVKPVKASKPKKAKPVK
  COW KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKTKKPKTVKAKPVKASKPKKTKPVK
CHIMP KKASKPKKAASKAPTKKPKATPVKKAKKKLAATPKKAKKPKTVKAKPVKASKPKKAKPVK
      ***:**********:************** ******:**** **.**********:* **
```

Figure 1: Sequential Alignment

If two sequences in an alignment share a common ancestor, we can interpret mismatches as point mutations and gaps as introduced in one or both lineages in the time since they diverged from one another. In sequence alignments of proteins, the degree of similarity between amino acids occupying a particular position in the sequence can be interpreted as a rough measure of how conserved a particular region or sequence motif is among lineages. DNA and RNA nucleotide bases are more similar to each other than are amino acids; the conservation of base pairs can indicate a similar functional or structural role.

The main problem of sequential alignment is its complexity. It takes more time to discover motifs in a large sequence. The complexity of sequential alignment is almost same as general word based algorithm.

3.4 Genetic Algorithm

Genetic algorithm is a machine learning approach. Genetic algorithms use Evolutionary techniques to find good approximate solutions. They use survival of the fittest techniques and have self-repair, self-guidance, and reproduction methods. They are highly randomized and are ideal for search and optimization problems.

There are four major steps in genetic algorithm. The first challenge is to determine how the initial population will be created. Each individual in the population is a possible solution to the problem and should be generated randomly. The biggest challenge in a genetic algorithm is to determine a good fitness function. The fitness function measures the “goodness” of a solution versus other solutions. There is a reproduction phase which is based on both the fitness value and chance. There is a crossover phase where two parents individuals are Chosen and two new children individuals are created by mixing the traits of the two parent individuals. There is a mutation phase where one parent individual is selected and a mutation is done. A Mutation on an Individual occurs at a very low rate.

The last step in the genetic algorithm is to determine when to terminate. This is typically done in one of two ways. The user can either specify the maximum number of rounds that the genetic algorithm should run, or they can specify a maximum number of rounds to run where the highest fitness value hasn't changed. These are the basics of genetic algorithms. A flow-chart for genetic algorithms is shown below.

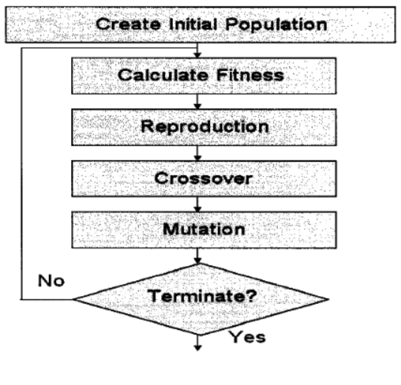


Figure 2: Flow-Chart of a Genetic Algorithm

In every generation, a new set of artificial individuals is created using bits and pieces from the fittest individuals of the population and an occasional new part that is tried for good measure. Although genetic algorithms are randomized, they efficiently exploit historical information to speculate on new search points with expected improved performance.

3.4.1 Steps

Initial Population:
Genetic algorithms require that there is a mapping between the real world representation of a problem and a data structure which can be as simple as a string. The string can be just a sequence of 1s and 0s where parts of the string represent pieces of the real world. In the genetic algorithms that will be introduced in this thesis, the individuals will be alignments of DNA sequences, sets of gene predictions for a DNA sequence, and pairs of alleles making up a trait.

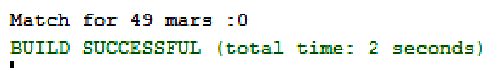
Reproduction:
Reproduction is the first process in a genetic algorithm where the individual strings in the population are evaluated according to their optimality which is determined by an objective function. Biologists call this the fitness function. The fitness function is just a way to measure the profit, utility, or goodness that is trying to be maximized. Selecting strings according to their fitness values means that strings with a higher fitness value have a higher probability of contributing one or more offspring to the next generation. This correlates to natural selection - survival of the fittest.

Crossover:
From the new population, pairs of strings are randomly chosen for crossover. Crossover is just a swapping of parts of the strings. A random crossover point is generated, and two new strings are generated by combining the first part of Individual 1 up to the crossover point with the last part of Individual 2 after the crossover point. The other new string will have the first part of Individual 2 up to the crossover point and the last part of Individual 1 after the crossover point.

Mutation:
Mutation is the last step in the process. The chance that mutation occurs is very low.

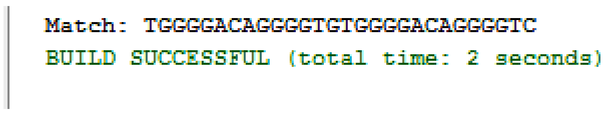
Chapter 4

Result Analysis



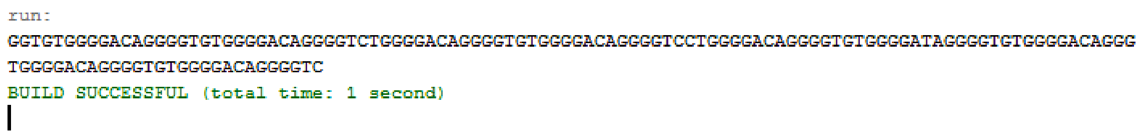
```
Match for 49 mars :0
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figure 3: 100 length test in general case.



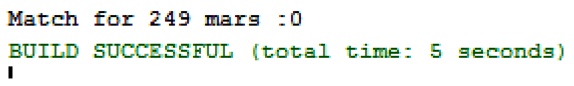
```
Match: TGGGGACAGGGGTGTGGGGACAGGGGTC
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figure 4: 100 lengths test in Sequential analysis.



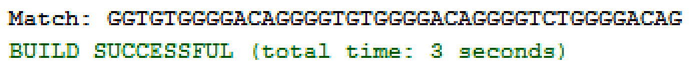
```
run:
GGTGTGGGGACAGGGGTGTGGGGACAGGGGTCTGGGGACAGGGGTGTGGGGACAGGGGTCCTGGGGACAGGGGTGTGGGGATAGGGGTGTGGGGACAGGG
TGGGGACAGGGGTGTGGGGACAGGGGTC
BUILD SUCCESSFUL (total time: 1 second)
```

Figure 5: 100 lengths test in Suffix tree.



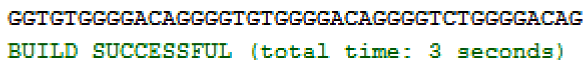
```
Match for 249 mars :0
BUILD SUCCESSFUL (total time: 5 seconds)
```

Figure 6: 500 length test in general case.



```
Match: GGTGTGGGGACAGGGGTGTGGGGACAGGGGTCTGGGGACAG
BUILD SUCCESSFUL (total time: 3 seconds)
```

Figure 7: 500 lengths test in Sequential analysis.



```
GGTGTGGGGACAGGGGTGTGGGGACAGGGGTCTGGGGACAG
BUILD SUCCESSFUL (total time: 3 seconds)
```

Figure 8: 500 lengths test in Suffix tree.

Chapter 5

Proposals

The mechanics of genetic algorithms are highly randomized, yet this is one of the main sources of their power. At first it seems surprising that chance should play such a fundamental role, but if some "idea" could be represented as a string, where substrings represent certain "notions" about the idea, then it can be clearly seen that the populations are not just stringent solutions to a problem, but rather "ideas" containing some helpful "notions" and some less helpful "notions" with the goal being to find the optimal "idea".

Genetic algorithms keep combining the helpful "notions" from one "idea" with other helpful "notions" from other "ideas" until an optimal "idea" is found.

Chapter 6

Conclusion

As transcription factors bind to DNA motifs and gene expression, identification of motifs in promoter region of genes will help understand the regulation of gene expression. So biologists and computer scientists are interested to construct computational methods for motif finding. With the advent of availability of large scale of genome sequencing and high throughput gene expression analysis techniques, many algorithms have been designed and implemented past decade. Each has its advantage and disadvantage. Diverse approach including combinatorial enumeration, probabilistic modeling, mathematical programing, neural networks and genetic algorithms, have been used. Very beginning algorithms relied on co-expressed gene and searched for overrepresented motifs.

Now-a-days latest algorithms are designed based on the overrepresentation and conservation among orthogonal sequences. From these algorithms of motif finding, user should be careful to choose the best one though the assessment of performance of algorithms is a difficult task. For this reason we do not have clear understanding of the biology of regulatory mechanisms, therefore, we lack an absolute standard against which correctness of algorithms can be measured. Each of the algorithms performs better in lower organism as Yeast with the comparison to the higher organisms. Recent algorithms that integrate the motif overrepresentation and cross-species conservation have proven to perform better in higher including human. So, experts should use a few complementary tools in combination rather relying on a single one.

References

[1] Besemer, J., & Borodovsky, M. (1999). Heuristic approach to deriving models for gene finding. *Nucleic Acids Research*, 27(19), 3911-3920

[2] A Genetic Algorithm with Clustering for Finding Regulatory Motifs in DNA Sequences by Shripal Vijayvargiya & Pratyoosh Shukla

[3] A Review: Applying Genetic Algorithms for Motif Discovery By Rahul Chauhan & Dr. Pankaj Agarwal

[4] A survey of DNA motif finding algorithms by Modan K Das and Ho-Kwok Dai, *BMC Bioinformatics* , 1471-2105

[5] Liu, X., D.L. Brutlag, and J.S. Liu, *BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes.*

[6] Application of genetic Algorithms in Bioinformatics by Amie Judith Radenbaugh

[7] Discovering Transcription Factor Binding Motif Sequences by I Lin

[8] Optimizing genetic algorithm for motif discovery by Hongwei Huo ,Zhenhua Zhao, Vojislav Stojkovic & Lifang Liu

[9] Sinha, S., *Discriminative motifs.* J Comput Biol

[10] http://en.wikipedia.org/wiki/Sequence_motif