Pattern Matching Based Technique to Solve Motif-Finding Problem

Pankaj Agarwal[1] and **Dr. S.A.M. Rizvi**[2]

Abstract - ***The problem of finding motifs from multiple molecular sequences is considered to be a difficult problem in molecular biology. In fact it is considered to be a Non-Deterministic Polynomial (NP)-complete problem and constant research is been carried out to solve the problem using some deterministic algorithm in polynomial time. Finding motifs from a set of DNA sequences is a critical step for understanding the gene regulatory network. This paper is an attempt to solve the motif problem using a deterministic method in polynomial time. The proposed method is not an exact algorithm but the probability of success is quite high by using it. Significance of the technique is its simplicity and time efficiency. The proposed technique is implemented as one of the modules in our general-purpose tool by the name "Sequence Comparison and Analysis Tool" for solving a number of sequence comparison problems encountered in the field of bioinformatics.***

1. INTRODUCTION

One important problem in bioinformatics is to understand how genes function in a gene regulatory network. Related to this is a sub problem of finding motifs for co-regulatory genes. A *gene* (protein coding gene) is a segment of DNA that codes for a specific protein. Genes seldom work alone. In most cases, genes cooperate to produce different proteins to provide particular functions. Understanding how the gene regulatory network works is important in molecular biology. In order to start the decoding process (*gene expression*), a molecule called *transcription factor* binds to a short region (*binding site*) preceding the gene. A *transcription factor* is a protein that regulates the activation of transcription in the eukaryotic DNA. Transcription factors localize the regions of promoter and enhancer sequence elements either through direct binding to DNA or through binding other DNA-bound proteins.

Transcription factor can bind to the binding sites of several genes to cause these genes to co-express. These binding sites have similar patterns called *motifs* [1]. Finding motifs from a set of DNA sequences is a critical step for understanding the gene regulatory network. In general by ``motifs'', we refer to specific regions within larger DNA sequences that have some specific function. For example restriction sites are an example of a short sequence within a DNA molecule that has the function of being recognized by restriction enzymes. Motifs are generally short patterns (and usually but not always ungapped) and may be defined for DNA, RNA or Protein sequences.

[1]*Asst. Professor, Krishna Institute of Engineering and Technology, Department of Computer Science and Engineering, Ghaziabad, U.P.*

[2]*Head, Department of Computer Science, Jamia Millia Islamia Central University, New Delhi.*

The discovery of motifs will allow the biologist to understand the varied and complex mechanism that regulates gene expression [2]. The objective of this paper is to devise a simple & effective methodology for determining motifs of any size from multiple molecular sequences.

2. RELATED WORK

Solving motif-finding problem has always been one of the key areas of interest for the researchers in the field of bioinformatics. Number of methods, algorithms and tools have come up in the recent years. Few of the common methods have been considered and discussed here.

CONSENSUS [2, 3] is a greedy algorithm requires no additional prior information other than the size of the desired motif. Generally, it works by extracting all possible subsequences of the correct length that are
found in the sequences. Then it iteratively combines these subsequences together and calculates the positional weight matrix (PWM) for each set, keeping the best ones at each step.

Gibbs sampling [4] approach starts with a guess for where a motif is located in each input sequence and then uses those guesses to make more informed guesses. It chooses motif locations in a semi-random fashion, so it is not a greedy algorithm, but it is affected by where the initial guesses are located.

Expectation Maximization (EM) [5] is a term for a class of algorithms that estimates the values of some set of unknowns based on a set of parameters (the so-called "Expectation step"), and then uses those estimated values to refine the parameters (the "Maximization step"), over several iterations.

SP-STAR Combinatorial approach [6] was proposed by **Pevzner and Sze, 2000**. First, it chooses a suitable scoring function to access the goodness of a motif. Then, for each *l*-mer appearing in the sample, it finds the best instance in each sequence and collects these instances together to form an initial motif. It then employs a local improvement heuristic to improve each initial motif.

In recent times many new methods and algorithms have been proposed [7, 8, 9, 10]. A recent comparison of 13 current motif-Finding tools has been made available on the web page http://bio.cs.washington.edu/assessment

3. PROPOSED METHOD

This work concentrates only on *planted motif problem*, which is defined as:

Let $S = \{S_1, S_2 \ldots S_m\}$ be a set of sequences. For a length l pattern M, define $\delta(S_i, M)$ to be the minimum number of substitutions between S_i and M. Define *score* $(M) = \sum \delta(S_i, M)$. For example, suppose $S = \{S_1, S_2, S3, S4\}$, where

$S1$=TAGTACTAGGTCGGACTCGCGTCTTGCCGC
$S2$=CAAGGTCCGGCTCTCATATTCAACGGTTCG
$S3$=TACGCGCCAAAGGCGGGGCTCGCATCCGGC

*S*4=CCTCTGTGACGTCTCAGGTCGGGCTCTCAA
Here *M* = AGGTCGGGCTCGCAT.
Then we have $\delta(S_1, M) = 2$ as sequence S_1 and *M* differ in their respective positions at two places Similarly $\delta(S_2, M) = 2$, $\delta(S_3, M) = 2$ and $\delta(S_4, M) = 2$. Thus, *score* $(M) = 2 + 2 + 2 + 2 = 8$.
A set S of sequences each of length 'n' and two integers L & D with D<L<n acts as input to the proposed algorithm. The output is a pattern M such that every sequence in S contains a length L sub-string that can be transformed to M using at most D substitutions. For explanation of the algorithm, following set of four sequences have been considered
*S*1=TAGTACTAGGTCGGACTCGCGTCTTGCCGC
*S*2=CAAGGTCCGGCTCTCATATTCAACGGTTCG
*S*3=TACGCGCCAAAGGCGGGGCTCGCATCCGGC
*S*4=CCTCTGTGACGTCTCAGGTCGGGCTCTCAA

3.1 ALGORITHM

Step1: As a first step all the sub-strings corresponding to window size L from the sequence S_1 are stored. Sub-strings are obtained by considering a window of size L and shifting the window by one from left to right till all the sub-strings are collected in a table as depicted below in the Table 1.

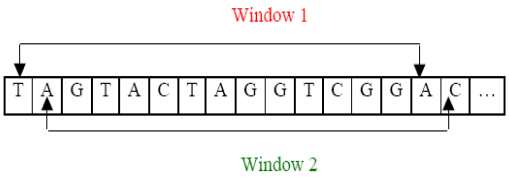

Figure 1: Two windows of size L=15 are shown.

For the considered sequence S_1 following sub-strings of window size L=15 can be obtained
Step 2: Now using the divide & conquer strategy FAT Tree is constructed for each of the obtained sub-strings corresponding to windows. Value for either height or level of the tree is also stored. A sample of FAT Tree is depicted below in the figure 2.
Step 3: Searching process begins here. For each of the obtained sub-sequence of window size L, it is searched in rest of the sequences. For a sample subsequence searching is carried out in the following manner:

a) Pattern at rood node is first searched in all the sequences (here S2, S3, S4). If found (exact pattern) it is stored along with the sequence number and calculated percentage of matched characters (here obviously 100%) in a table.
b) If the pattern is not found, counter value associated with the present sequence is incremented by one and then search is carried out starting with left node (here node associated with pattern 'TAGTACT'). If pattern at left node is found, then we search for the remaining part of the sequence by following the link part (here link between the left and right node is depicted in the figure 2 by dotted arrows). It is to be remembered that remaining part of the pattern should be searched immediately after the position where its first part was matched in the sequence under consideration.

Window 1	TAGTACTAGGTCGGA
Window 2	AGTACTAGGTCGGAC
Window 3	GTACTAGGTCGGACT
Window 4	TACTAGGTCGGACTC
Window 5	ACTAGGTCGGACTCG
Window 6	CTAGGTCGGACTCGC
Window 7	TAGGTCGGACTCGCG
Window 8	AGGTCGGACTCGCGT
Window 9	GGTCGGACTCGCGTC
Window 10	GTCGGACTCGCGTCT
Window 11	TCGGACTCGCGTCTT
Window 12	CGGACTCGCGTCTTG
Window 13	GGACTCGCGTCTTGC
Window 14	GACTCGCGTCTTGCC
Window 15	ACTCGCGTCTTGCCG
Window 16	CTCGCGTCTTGCCGC

Table 1: A list all the sub-strings corresponding to window size 15 and window shift of 1

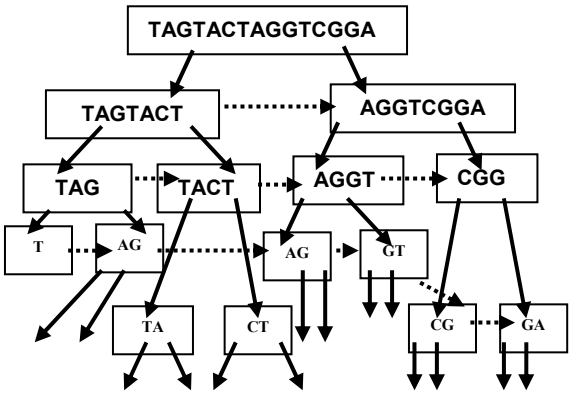

Figure 2: A FAT-Tree corresponding to pattern "TAGTACTAGGTCG"

c) Now if suppose pattern associated with right node is not found resulting in the increase of counter value further by one, then both the child nodes (here nodes with patterns 'AGGT' and 'CGGA') are exploited starting from left node.
d) While searching if at any instance of time counter value exceeds the value given as $2^{lvl}/2$ where 'lvl' refers to the level of the tree, then algorithm assumes that the considered sequence should be ignored from the process.

Step 4: A table is constructed with four values namely sequence number, window number, sub-sequence obtained from the first taken sequence, pattern found in the searched sequence and it's associated score in percentage calculated as Score=(number of matched characters in the taken sub-sequence/L) *100 as depicted in table 2. Now table can be scanned to find the entries with maximum percentage for each set of considered sequences (here S1,S2, S3 & S4).

Step 5: All the maximal sets of patterns obtained are arranged in another table with separate rows for each pattern. Characters with maximum frequency at each position are then collected to give the final motif of length L as depicted in table 3

Sequence	Window	Window Pattern	Found Pattern	Score
S2	8	AGGTCGGACTCGCGT	AGGTCCGGCTCTCAT	73.3%
S3	8	AGGTCGGACTCGCGT	AGGCGGGGCTCGCAT	73.3%
S4	8	AGGTCGGACTCGCGT	AGGTCGGGCTCTCAA	73.3%

Table 2: The window patterns and associated found patterns with maximum score

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	G	G	T	C	G	G	A	C	T	C	G	C	G	T
A	G	G	T	C	C	G	G	C	T	C	T	C	A	T
A	G	G	C	G	G	G	G	C	T	C	G	C	A	T
A	G	G	T	C	G	G	G	C	T	C	T	C	A	A
A	G	G	T	C	G	G	G	C	T	C	TG	C	A	T

Table 3: Characters with maximum frequency at each position

Thus the final motif can be represented as
M=AGGTCGGGCTCTCAT or
AGGTCGGGCTCGCAT
The described method is based on dynamic programming approach, which is a well-known technique to solve optimization problem. Here the problem of finding motifs from given molecular sequences have been considered as an optimization problem. Since every possible sub solution (sub sequences) is considered and a matching algorithm is used to determine the degree of match in each iteration where the scores of match are stored within a table followed by final scanning of the table to give the most optimal match; the chances of failure is quite low.

3.2 Time Complexity Analysis

The above-presented proposed method is carried out basically in five steps:

***Extraction*:** All the sub-strings equivalent to window size L are extracted from one of the considered input sequence. This process will take at most O(n) worst-case time-complexity considering that the sequence has 'n' characters.

FAT-tree Construction: construction of the FAT-Tree corresponding to each of the extracted patterns in above step will take O(k.lg(L)) worst-case time complexity where k=number of extracted patterns each of length L with k<n.

Searching and Table construction: Searching each of the extracted patterns in all the remaining input sequences of length 'n' and 'm' being the number of such sequences will take O(k.m.n.lg(L)) time complexity.

***Table Scanning*:** this should take atmost O(m.k)
Frequency Calculation: this step will take O(m.n)+O(n) complexity

The final time complexity can thus be given as
$T(n)= O(n)+ O(k.lg(L))+ O(k.m.n.lg(L))+ O(m.k)+ O(m.n)+O(n)=O(k.m.n.lg(L))$ which can be further given as $O(k.n^2.lg(L))$ provided n=m.
As already stated that motif-finding problem is considered as NP-Complete problem and solving the problem for a given small set sequences by using some deterministic algorithm in polynomial time is always significant.

3.3 IMPLEMENTATION

The above described method is implemented as one of the modules in our general purpose computational tool by the name "*Sequence Comparison and Analysis Tool*" for solving various sequence comparison problems encountered in the filed of bioinformatics. It is implemented using Visual Basic-6 package. It has the following functions

a) **SequenceEntry (*QrySeq as string*):** adds the input sequence through interface to he database table 'MotifDB'.
b) **PatternExtractor(*QrySeq as string, WindowSize as integer)* as string**: It extracts all the pattern of size given by window size and stores them in database.
c) **Generate_FATtree(*SeqId as* string, *PatternRS as* Recordset)**: generates the FAT tree corresponding to all the patterns of a given sequence.
d) **PatternSearch(*PatternID* as string, *PatternTreeCode as string) as* long**: searches for all the patterns in first sequence in all the remaining set of input sequences.
e) **GenerateMotif *(frequencyDB as* recordset*) as* string**: it generates the required motifs from he frequency table constructed during pattern search phase.

4. Alternative Search Approach

As an alternative approach to search method employed we can extract L-length patterns from all the input sequences in the same manner it is done for first sequence and then store these patterns in relational format. Now we can make use of SELECT…FROM… WHERE pattern LIKE constructs embedded within a procedural code to match most similar patterns corresponding to all the sequences and finally evaluate the frequency at each position as stated in the proposed method to give an approximate motif. The success of this method will depend on the effectiveness of the heuristic employed.

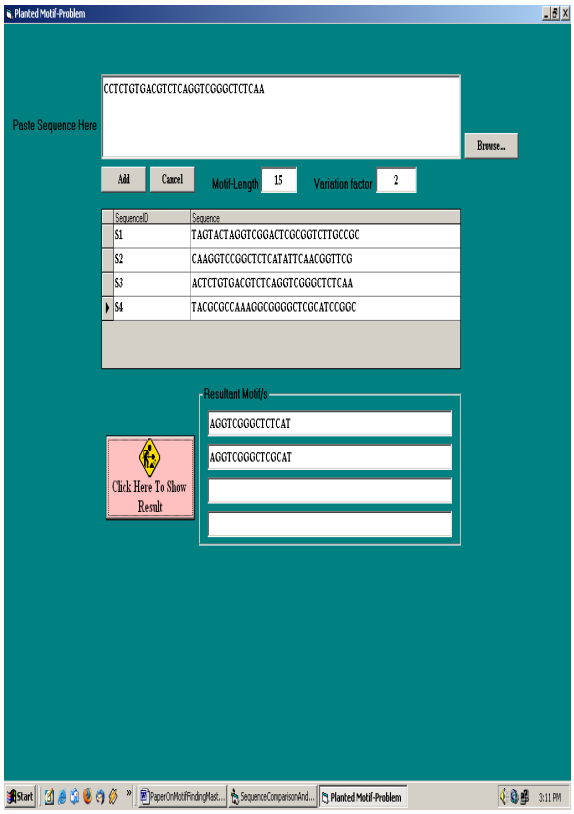

Figure 3: Interface that captures the input details and shows the output.

5. CONCLUSION AND FUTURE STUDY

Most of the existing methods require some additional information other than the sequences themselves like *SP-STAR, Expectation Maximization methods* or makes some assumption before hand like *Gibbs sampling*. The proposed method makes no prior assumptions or requires additional information about the sequences; it uses a simple algorithm based on dynamic programming approach to determine the motifs from any given number of input sequences of any length. Unlike CONSENSUM method, which is based on greedy algorithm, ours is based on dynamic programming model. As we all know chances of failure in applying greedy algorithms is always quite high in comparison to dynamic programming method.

But we still believe that a more efficient technique can be devised to solve the planted-motif problem using some deterministic algorithm in polynomial time. The use of pipelines in the context of parallel processing can be very handy for providing the solution to the above stated problem. We are already working in this direction and hope to come with a better solution using pipelines.

REFERENCES

[1] H. Leung and F. Chin. Algorithms for Challenging motif problems. *JBCB*, 43—58, 2005.

[2] Stormo, G. DNA binding sites: representation and discovery. Bioinformatics, 16:16{23, 2000.

[3] G.Z. Hertz and G.D. Stormo. Identification of consensus patterns in unaligned dna and protein sequences: a large-deviation statistical basis for penalizing gaps. The Third International Conference on Bioinformatics and Genome Research, 201—216, 1995

[4] Lawrence C E, Altschul S F, Boguski M S, Liu J S, Neuwald A F and Wootton J C. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262, 208-214, 1993.

[5] Cardon, L.R. and Stormo, G.D. (1992) An Expectation-Maximization (EM) Algorithm for Identifying Protein-Binding Sites with Variable Lengths from Unaligned DNA Fragments. J. Mol. Biol. 223:159-170.

[6] Pevzner P A and Sze S H. Combinatorial approaches to finding subtle signals in DNA sequences. Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB 2000), 269-278, 2000.

[7] Makino, K. and Uno, T. (2004) 'New algorithms for enumerating all maximal cliques', *Proceedings of Scandinavian Workshop on Algorithm Theory*, pp.260–272.

[8] Rajasekaran, S., Balla, S. and Huang, C.H. (2005) 'Exact algorithm for planted motif challenge problems', *Proceedings of Asia-Pacific Bioinformatics Conference*, pp.249–259

[9] Styczynski, M.P., Jensen, K.L., Rigoutsos, I. and Stephanopoulos, G.N. (2004) 'An extension and novel solution to the (*l*,*d*)-motif challenge problem', *Genome Informatics*, Vol. 15, pp.63–71

[10] Sze S H, Lu S and Chen J. Integrating sample-driven and pattern- driven approaches in motif finding. Lecture Notes in Computer Science/Lecture Notes in Bioinformatics (WABI 2004), 438-449, 2004