Dhaka Univ. J. Eng. & Tech. Vol. 1(1) 45-49, 2010 (July)

Data Mining Based Motif Detection in Biological Sequences

Faisal Sikder and Abu Ahmed Ferdaus*

Masters Student, Department of Computer Science and Engineering, University of Dhaka
faisal.sikder@gmail.com
Assistant Professor, Department of Computer Science and Engineering, University of Dhaka
ferdaus1167@gmail.com

Abstract

This paper considers the problem of discovering motif in DNA and protein sequences. Motif finding problem has important applications in understanding gene regulation, protein family identification and determination of functionally and structurally important identities. Biological approaches for this problem are long-winded, complex and time-consuming. Here, we have developed a method based on data mining to detect frequent residue motifs. Our proposed method is based on FP-tree and FP-growth algorithms of frequent pattern mining techniques. The limitation of iterative nature of existing Apriori based method has been overcome in the developed PF-tree based method. Also we have developed a tool based on proposed method which can expeditiously detect novel motifs based on information content and shows better performance over the existing Apriori based method. Experimental results show that this new method successfully elucidates true motifs on real biological sequence datasets which support the effectiveness of the method.

Keywords: Motif Discovery, Frequent Pattern, PF-growth, DNA, Protein, Bioinformatics

1. Introduction

Motifs are short and preserved patterns that are part of a family of sequences. Motif can be used as protein function identification, gene regulation and many other essential task of sequence analysis. Sequence alignment may not identify closely related protein or genes of unknown structure but it can be found more accurately by the occurrence in its sequence of particular residue pattern, motif or fingerprint. Some regions are preserved because of specific requirements on the structure of particular region of protein which may be crucial.

Motif can be discovered as subsequences that are common to the family of sequences from sequence patterns (sub-sequences). Since the dramatic increase of genetic data, data mining have become essential to the analysis of protein and nucleic acid sequences. As a result, numbers of bio-data mining techniques have been developed. In this paper, we demonstrate how FP-tree structure and FP-growth algorithm of data mining technique can be used for motif prediction.

Existing motif finding algorithms can be classified into two groups such as combinational and probabilistic. PRATT [1] and TEIRESIAS [2] are example of combinational method. MEME [3] and Gibbs [4] search for motif using statistical approach and these are probabilistic methods. Over a long period of time there have been developed hundreds of methods but among them Gibbs and MEME are the most widely used methods. These algorithms successfully finds motif in biological sequences, but no algorithm works perfect as the motif discovery problem is complex and very difficult to solve. The limitation of MEME is that it takes a lot of time. Again Gibbs algorithm suffers from the random search behavior which means it generates different motifs in every execution. BioPM [5] is a very efficient protein sequential pattern mining algorithm based on prefix projected method. It uses a new data structure BioP-tree and its researcher claimed better performance over Apriori based method. But still there is a problem of choosing minimum support threshold. H. G. Ozer and William C. R. [6] proposed an algorithm called informative motifs to find frequent residue motifs that are high in information content and outside of the family consensus. It has modified classic Apriori algorithm to mine frequent residue pattern. CRMD [7] employs a flexible statistical motif model allowing a variable number of motifs and motif instances. It first uses a novel entropy-based clustering to find complete and good starting candidate motifs from the DNA sequences; then uses an effective greedy refinement to search for optimal motifs from the candidate motifs.

2. Basic Experimental Theory

Frequent-pattern growth or simply FP-growth [8] adopts a divide-and-conquer strategy which works as follows. First, it compresses the database into a frequent-pattern tree or FP-tree representing frequent items, which retains the itemset association information. It then divides the compressed database into a set of conditional database; each associated with one frequent item or "pattern fragment" and mines each such database separately.

FP-tree is constructed using prefix-tree structure with some extended in structure. It will be used for storing crucial, quantitative information about frequent patterns. Only frequent length-1 items will have nodes in the tree, and more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones.

We have explored motif discovery problem by applying the techniques of data mining. Many types of data mining approaches have been used in motif discovery. Apriori and prefix based tree are many of those existing approach. Frequent pattern growth algorithm (FP-growth) is very efficient than Apriori and prefix based tree method. It scans the database only twice to generate frequent patterns. This is why we have applied this method in our motif discovery approach along with some other initial and post mining filtering and selecting methods.

3. Methodology

Given a large dataset of N biological sequence $S_1, S_2, \ldots, S_N$, our goal is to identify the conserved regions that represent this dataset. We have followed a specific procedure to identify these conserved regions. Our algorithm is designed to proceed in the following pathway-

I. Select initial set of fit subsequences.
II. Align fit subsequences.
III. Generate transactions from subsequences.

IV. Construct frequent pattern tree (FP-tree) from transactions.

V. Mine patterns using frequent pattern growth (FP-growth) algorithm.

VI. Filter and select motifs from patterns.

3.1 Select Initial Set of Fit Subsequences

A set of initial subsequences is collected which are patterns of a fixed length l and presented in certain number of input sequences. Subsequences should be statistically significant because motifs are more frequent than random patterns. To measure statistical significance of a pattern we have used second order Markov chain model. Let x be a biological sequence e.g. $x = x_1x_2...x_n$. The probability of x for a given second order Markov model M is:

$$P_M(x) = \Pi^l_{i=1} P(x_i|x_{i-2}x_{i-1})$$

Where $P(x_1|x_{-1}x_0) = P(x_1)$ and $P(x_2|x_1x_0) = P(x_2|x_1)$ if x_0 and x_{-1} are not available. The probability of x for a given random model R is $P_R(x)= \Pi^l_{i=1} P(x_i)$. Then the log-odd score of the sequence x, denoted $E(x)$, is defined as $E(x) = log(P_M(x)/P_R(x))$. We have only picked those subsequences whose log-odd score is greater then threshold T i.e $E(x)>T$. Proposed algorithm for selecting initial fit subsequences:

Input: A set of sequences (S)

Output: A set of fit subsequences (Q_S)

```
SelectFitSub(){
        $Q_S = \emptyset$                //qualified subsequence set
        $Q_r = \emptyset$                //subsequence set of $Q_S=\Phi$
        while($|Q_r|$ < number of sequences){
                $Q_S$= fit_subsequences($l, T, S-Q_r$) $U$ $Q_S$
        }
}
fit_subsequences($l, T, S'$){
        Find $P_q$ where $E(x) > T$ and length is $l$ in sequence set $S'$;
        Rank $S'$ according to the number of $Q_S$ in each sequence;
        $S''$ = Substring ($S'/2$);
        for each qualified subsequence $Q$ in $S''$
                $Q_S = Q_S U \{Q\}$;
        return $S''$
}
```

3.2 Align Fit Subsequences

In this step we have aligned those fit subsequences. Scoring or weight matrix is a very good method for representing the variation in a set of sequence patterns in a multiple sequence alignment and as a tool for finding additional sequences in database search. Odd score can be used to find the probability of each subsequence location. We have used BLOSUM62 scoring matrix to align those subsequences which was aligned by pair wise alignment.

3.3 Generate Transactions from Subsequences

The information content of a candidate motif is calculated by generating position weight matrix (PWM) from it. In DNA sequences the matrix has four columns representing 4 possible nucleotides (A, T, G, and C) and for protein sequences number of columns will be 20 representing 20 different amino acids. Number of rows will be equal to the length of the subsequences. For example, we have got 6 nucleotide subsequences shown in Table 1 and corresponding PWM in Table 2.

Table 1: subsequences

GTTCCAGCT
GTTCGAGGT
TTTCCAGCT
GATCCACCA
GTTCCTCGA
TATTCACCT

Table 2: Corresponding PWM of the given subsequences

N	1	2	3	4	5	6	7	8	9
A	0.00	0.33	0.00	0.00	0.00	0.83	0.00	0.00	0.33
T	0.33	0.67	1.00	0.17	0.00	0.17	0.00	0.00	0.67
G	0.67	0.00	0.00	0.00	0.17	0.00	0.50	0.33	0.00
C	0.00	0.00	0.00	0.83	0.83	0.00	0.50	0.67	0.00

In general, the average amount of information in bits per residue for column c of the PWM is given by,

$$I_c = -\sum P_{ic} \log_2 P_{ic}$$

Where P_{ic} is the frequency of amino acid/Nucleotide i in column c and is estimated by the frequency of occurrence of each amino acid/Nucleotide. Since we want to extract motifs which are special type of patterns, we eliminated residues with probabilities smaller than 0.2 to avoid unnecessary computations. Then, each subsequence with its remaining residues is recorded as a transaction into the transaction database Table 3.

Table 3: Transactions from subsequences using PWM

TID	Transaction Items	TID	Transaction Items
T100	G1 T2 T3 C4 C5 A6 G7 C8 T9	T400	G1 A2 T3 C4 C5 A6 C7 C8 A9
T200	G1 T2 T3 C4 A6 G7 G8 T9	T500	G1 T2 T3 C4 C5 C7 G8 A9
T300	T1 T2 T3 C4 C5 A6 G7 C8 T9	T600	T1 A2 T3 C5 A6 C7 C8 T9

3.4 Construct Frequent Pattern Tree (FP-tree) from Transaction

In this step we have constructed a special tree like structure called frequent pattern tree (FP-tree) (Figure 1) from the transaction database. This is the compact structure which only stores the frequent patterns depending on support count. The FP-tree construction method includes following steps:

Step I: Scan the transactions once and collect the set of frequent items (1-itemsets) based on minimum support and their support count.

Step II: Now sort frequent 1-itemsets according to their support count in descending order.

Step III: Create the root of an FP-tree and label it as "null".

Step IV: For each transaction; select and sort the frequent items according to the descending order of the support count.

Step V: Now call the *fp_tree_insert([p|P], T)* where *[p|P]* is transaction and *T* is the FP-tree.

Step VI: For any remaining transaction in database go to *Step IV* to insert it into FP-tree *T*.

```
Method fp_tree_insert(([p|P], T){
        If (T has a child N such that N.item-name=p.item-name) Then
        Increment N's count by 1;
        Else Then
                Create a new node N, and let its count be 1; Its parent link is linked to T;
                link to the nodes with the same item-name
        via the node-link structure.
        End-If
        If (P is nonempty)
                Call fp_tree_insert(P,N);
}
```

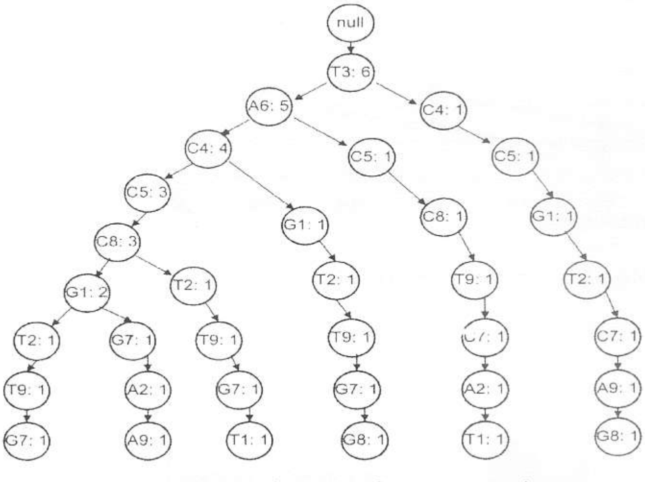

Fig. 1: Constructed FP-tree from Transactions.

3.5 Mine Patterns using FP-growth Algorithm

The mining of the FP-tree proceeds as follows. Starting from each frequent length-1 pattern; first its conditional pattern base (a sub database which consists of the set of prefix paths in the FP-tree co-occurring with the suffix pattern) is constructed, then mine frequent patterns recursively using conditional pattern tree. FP-growth uses the least frequent items as a suffix, offering good selectivity. The method substantially reduces the search costs. This FP-growth algorithm is given bellow:

```
Procedure FP-growth (Tree,$\alpha$)
{
    if Tree contains a single prefix path then {
    for each combination (S) of the nodes in the path P do
            generate pattern $S \bigcup a$ with support = minimum support of nodes in S;
    } else for each item ai in Q do {
            generate pattern $\beta = a_i \bigcup a$ with support = $a_i$.support;
            construct $\beta$'s conditional pattern-base then $\beta$'s conditional FP-tree Tree$\beta$ ;

            if Tree$\beta$ $\neq \emptyset$
    then call FP-growth(Tree$\beta$ ,$\beta$);
    }
}
```

Our FP-growth algorithm has some differences with existing FP-growth algorithm. In the modified algorithm all the items of the conditional pattern base remain in the conditional pattern tree and we only accept the largest frequent pattern generated by the conditional FP-tree.

3.6 Filter and Select Motifs from Patterns

We have to convert the frequent pattern into a DNA or protein sequence by using their column number associated with them. We further filter our frequent pattern to get our expected motif. We have used similarity scores to get the true motif. This filtering process can be done in different ways. First, we may get several same length frequent patterns. Then we have to rearrange and check whether all of them are a single motif or not. Such as if we get three frequent patterns like ACGCGT, ACGCGT, and ACTCGA; they are not two different motifs but same and consensus sequence is ACGCGT. Second, we may miss any column from transaction because of mutation in that column. We have to get the help of PWM to get that column and generate motif by using that column.

4. Experiments

In order to evaluate the correctness and efficiency of our proposed algorithm, we have developed a tool using java and tested that tool on collections of various DNA and protein sequences. DNA sequences were taken from TRANSFAC database and protein sequences from PROSITE database. To compare our algorithm against other established motif-discovery algorithms, we have used Gibbs motif sampling algorithm [9] and MEME [10] on the same test set. Standard parameters for proposed method are Threshold value *T*=0.05 and *Min support count*=30%. Runtime comparison with MEME algorithm is shown in Figure 2, 3 & 4. The average sensitivity reported by the three algorithms is plotted in Figure 5.

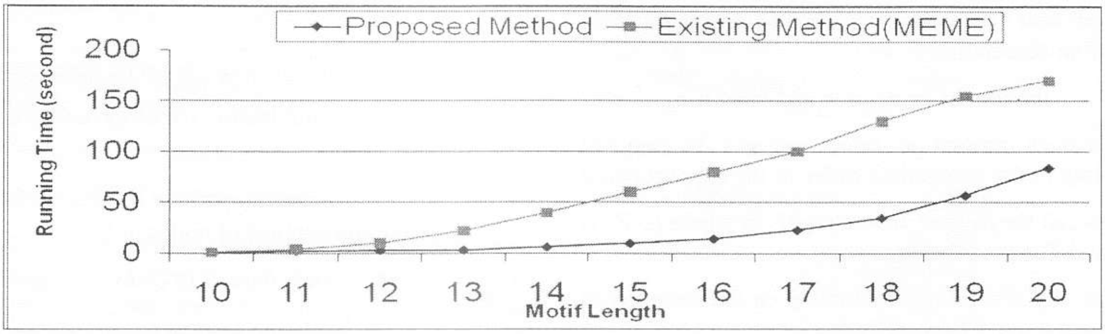

Fig. 2: Runtime Comparison with MEME Algorithm for yst04r DNA Dataset

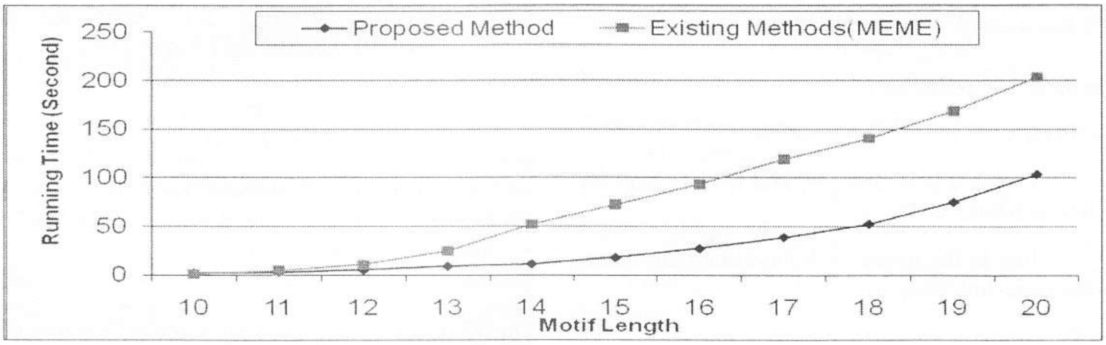

Fig. 3: Runtime Comparison with MEME Algorithm for yst08r DNA Dataset

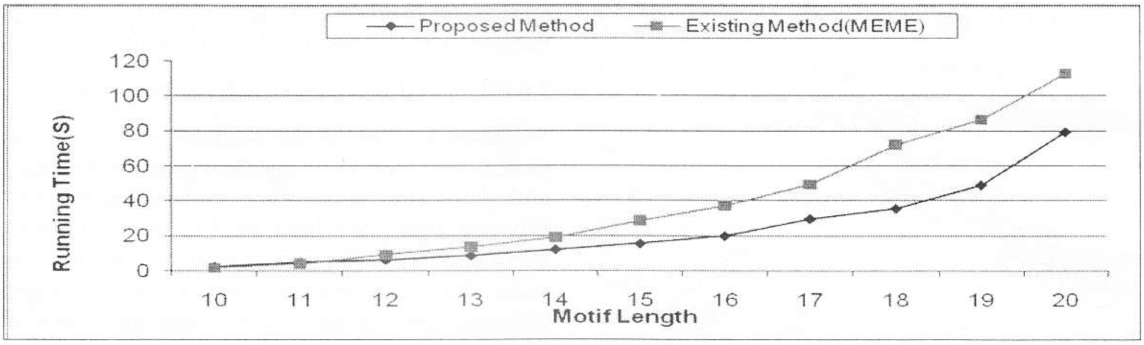

Fig. 4: Runtime Comparison with MEME Algorithm for Protein Dataset

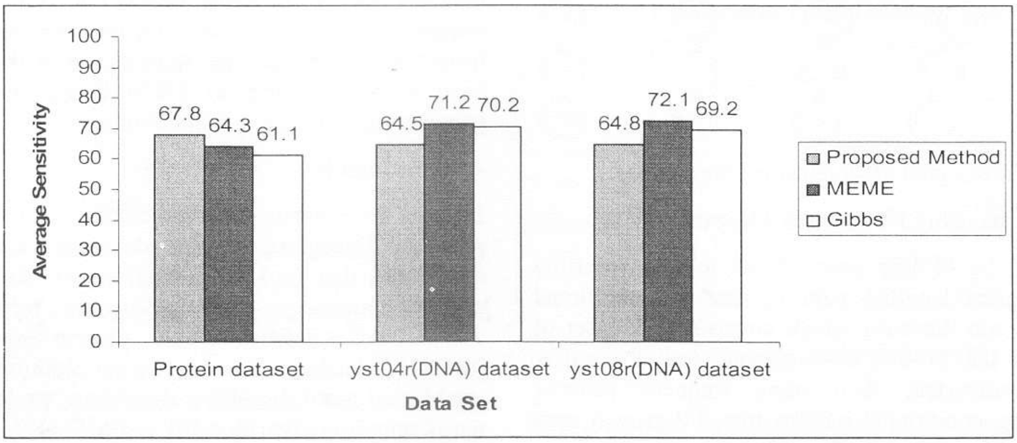

Fig. 5: Average Sensitivity Reported by the Three Algorithms

5. Discussion

As is evident from the graphs, the outcome of the three algorithms is comparable. Our proposed method has better runtime performance in all cases with compare to MEME algorithm. In case of sensitivity, our proposed algorithm has better performance in protein dataset but in DNA dataset MEME and Gibbs have better average sensitivity. Another advantage of our proposed algorithm is that it needs fewer parameters. However, we also emphasize that our method does not need to know number of motifs expected this is a quite advantage when there is no information available about dataset. To calculate runtime we run both MEME and our proposed algorithm in an Intel Core 2 Duo Linux machine having 4GB of memory.

6. Conclusion

The proposed method is a hybrid method which combines two methods of different area into bioinformatics problem. Previously some researchers have used data mining techniques in motif discovery and our proposed method performs better than those. The existing methods which used Apriori based algorithm need a lot of database scans where our proposed method needs only two database scans and it removes the major drawback of the Apriori based method. This is a significant advantage over existing motif discovery algorithms that it finds motif without being instructed on how many motifs should be discovered.

References

[1] Jonassen I., 1997, "Efficient discovery of conserved patterns using pattern graph", CABIOS, 13, pp 509-522.

[2] Rigoutsos I. and A. Floratos, 1998, "Combinatorial pattern discovery in biological sequences: The teiresias algorithm", Bioinformatics, 14, pp 55-67.

[3] Bailey T. and C. Elkan, 1995, "Unsupervised learning of multiple motifs in biopolymers using em", Machine Learning, 21(1-2), pp 51-80.

[4] Lawrence C., Altschul S., Bogouski M., Liu J., Neuwald A. and J. Wooten, 1993, "Detecting sequence signals: A gibbs sampling strategy for multiple alignment", Science, 262, pp 208-214.

[5] Yun X. and Z. Yangyong, 2007, "BioPM: An Efficient Algorithm for Protein Motif Mining," ICBBE-07, pp 394–397.

[6] Ozer H. and W. Ray, 2007, "Informative Motifs in Protein Family Alignments," Lecture Notes in Computer Science, 4645(1), pp 161-170.

[7] Chan G. Li. T., Leung L., and K. Lee, 2009 "A cluster refinement algorithm for motif discovery". Computational Biology and Bioinformatics, 99(1), pp 555-564.

[8] Han J., Pei J., Yin Y. and R. Mao, 2004, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," Data Mining and Knowledge Discovery, 8(1), pp 53–87.

[9] "Online free software tool for gibbs sampler motif search", Available at:
http://bayesweb.wadsworth.org/gibbs/gibbs.html.

[10] "Meme suite free software for motif search in dna and protein sequences", Available at:

http://meme.nbcr.net/downloads/meme 4.3.0.tar.gz.