International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 5 Issue 10 ǁ Oct. 2017 ǁ PP.78-86

Experimental Analysis of the Frequent Periodic Cryptic Sequence Mining in Biological Data

Nedunchezhian. T[1], Archana Panda[2]
[*1] *Department of Computer Science & Engineering, Gandhi Engineering College, Odisha, India*
[2] *Department of Computer Science & Engineering, Gandhi Institute For Technology, Odisha, India*

ABSTRACT: *Amino acid sequences are known to constantly mutate and diverge unless there is a limiting condition that makes such a change deleterious. The few existing algorithms that can be applied to find such contiguous approximate pattern mining have drawbacks like poor scalability, lack of guarantees in finding the pattern, and difficulty in adapting to other applications. In this paper, we present a new algorithm called Constraint Based Frequent Motif Mining (CBFMM). CBFMM is a flexible Frequent Pattern-tree-based algorithm that can be used to find frequent patterns with a variety of definitions of motif (pattern) models. They can play an active role in protein and nucleotide pattern mining, which ensure in identification of potentiating malfunction and disease. Therefore, insights into any aspect of the repeats – be it structure, function or evolution – would prove to be of some importance. This study aims to address the relationship between protein sequence and its three- dimensional structure, by examining if large cryptic sequence repeats have the same structure. We have tested the proposed algorithm on biological domains. The conducted comparative study demonstrates the applicability and effectiveness of the proposed algorithm.*
Keywords: *Motif, FP mining, FP tree, sequence mining, Repetition detection, data mining.*

I. INTRODUCTION

The approximate subsequence mining problem is of particular importance in computational biology, where the challenge is to detect short sequences, usually of length 6- 15, that occur frequently in a given set of DNA or protein sequences. These short sequences can provide clues regarding the locations of so called "regulatory regions," which are important repeated patterns along the biological sequence. The repeated occurrences of these short sequences are not always identical, and some copies of these sequences may differ from others in a few positions. A repeat is defined as two or more contiguous segments of amino acid (three or more) residues with identical and similar sequence. When such repeats are in high-complexity regions, they are called 'cryptic' [9]. Although low-complexity repeats are essential for evolutionary analysis and comprise a large section of the eukaryotic genome, high-complexity repeats are usually associated with a particular structure or function. This study considers large cryptic repeats comprising eight or more residues, as [26] fixed the length of a moderate-sized repeat as being between five and eight amino acids. The study of repeats is crucial because all but 5–6% of the high eukaryotic genome is repetitive [25]. Internal protein repeats are observed to be associated with structural motifs or domains. It is evolutionarily more 'economical' to evolve complex structures such as multiple domains by using 'modular plug-ins' [22] to fulfill a specific function. Furthermore, longer repeats normally act to enhance the stability of the native fold of the protein and, while small repeats interact with each other, larger repeats may either interact or remain isolated like beads on a string [22]. Three prominent reviews on repeats are those of [22], [11] and [33], and they concentrate on the relationship between structural repeats and their primary structure along with the characteristics of protein families. In [33] discuss the evolution of repeats as modules in the proteins. It is mentioned that the number of repeats in a protein can vary between proteins, implying that the loss or gain of repeats is very rapid in evolution.

The remainder of the paper is organized as follows: Section 2 presents related works and Section 3 describes our model. In section 4, we present optimization strategy for our model and in Section 5 contains our experimental results. Section 6 contains our conclusions.

II. PREVIOUS WORK

There is a vast amount of literature on mining databases for frequent pattern [30], [17], [47]. The problem of mining for subsequence was introduced in [29]. Subsequence mining has several applications, and many algorithms like [23], [48], and [36] have been proposed to find patterns in the presence of noise. However, they primarily focus on subsequence mining, while we focus on contiguous patterns. A host of techniques have been developed have been developed to find sequence in a time series database that are similar to a given query sequence [29], [3], [31], [49]. The existing algorithm [5], [14], [20], [42], [24] requires the user to specify the repetition and patterns occurring with that repetition, otherwise which look for all possible repetitions in the

time series. Some algorithms are classified based on the detection type of repetition for symbol, sequence or segment. Another algorithm that finds frequent trends in time series data was proposed in [1]. However, this algorithm is also limited to a simple mismatch based noise model. In addition, this is a probabilistic algorithm, and is not always guaranteed to find all existing patterns. The algorithms specified in [34], [35], [39], [13], looks for all possible repetitions by considering the range. COVN [34] fails to perform well when the time series contains insertion and deletion noise. WARP [35] can detect segment repetition; it cannot find symbol or sequence repetition. Sheng et al. [6], [8] developed algorithm based on ParPer [21] to detect repeated patterns in a section of the time series; their algorithm requires the user to provide the expected repetition value. COVN, WARP and ParPer are augmented to look for all possible repetitions, and which last till the very end of the time series. Cheung [7] used FP tree similar to STNR [13] which is not beneficial in terms of growth of tree. Huang and Chang [28] and STNR [13] presented their algorithm for finding repeated patterns, with allowable range along the time axis. Both finds all type of repetition by utilizing the time tolerance window and could function when noise is present. STNR [13] can detect patterns which are repeated only in a subsection of the time series. Repeated check in STNR last for all the positions of a particular pattern, which in our algorithm is been reduced.

Several approaches described in the literature handle structured motif extraction problem [3], [2] and repetition among subsection of the time series. However, our approach described in this paper is capable of handling both motif extraction and reporting all type of repetition. In this paper, we present a flexible algorithm that handles general extended structured motif extraction problem and uses CBFMM to build Consensus tree. CBFMM is capable of reporting all types of repetitions with or without the presence of noise in the data up to a certain level. We believe that this is an interesting problem since it allows mining for useful motif patterns with all type of repetition, without requiring specific knowledge about the characteristics of the resulting motif. In this paper, we present a new model that is very general and applicable in many emerging applications. We demonstrate the power and flexibility of this model by applying it to data sets from several real applications. We describe a novel motif mining algorithm called CBFMM that uses a concurrent traversal of FP trees to efficiently explore the space of all motifs. We present a comparison of CBFMM with several existing algorithms (COVN [34], WARP [35], STNR [13], ParPer[21]). CBFMM never misses any matches (as opposed to some of these methods that apply heuristics). In fact, we show that CBFMM is able to identify many true biological motifs that existing algorithms miss. We show that our algorithm is scalable, accurate, and often faster than existing methods by more than an order of magnitude. We present an algorithm that uses CBFMM as a building block and can mine combinations of simple approximate motifs under relaxed constraints.

III. CONSTRAINT BASED FREQUENT MOTIF MINING (CBFMM)

Our algorithm involves two phases. In the first phase, we build the Frequent Pattern (FP) tree for the biological data and in the second phase, we use the FP tree to calculate the repeat of various patterns in the biological data. One important aspect of our algorithm is redundant period pruning, i.e., we ignore a redundant period ahead of time. As immediate benefit of redundant period pruning, the algorithm does not waste time to investigate a repeat which has already been identified as redundant. This saves considerable time and also results in reporting fewer but more useful repeats. This is the primary reason why our algorithm, intentionally, reports significantly fewer numbers of repeats without missing any existing repeats during the pruning process.

A FP tree for a string represents all its suffixes; for each suffix of the string there is a distinguished path from the root to a corresponding leaf node in the FP tree. Given that a time series is encoded as a string, the most important aspect of the FP tree, related to our work, is its capability to very efficiently capture and highlight the repetitions of substrings within a string. The path from the root to any leaf represents a FP for the string. Since a string of length n can have exactly n suffixes, the FP tree for a string also contains exactly n leaves. Each edge is labeled by the string that it represents. Each leaf node holds a number that represents the starting position of the suffix yield when traversing from the root to that leaf. Each intermediate node holds a number which is the length of the substring read when traversing from the root to that intermediate node. Each intermediate edge reads a string (from the root to that edge), which is repeated at least twice in the original string. These intermediate edges form the basis of our algorithm presented in Appendix section.

The approach we take in CBFMM explores the space of all possible models. In order

to carry out this exploration in an efficient way, we first construct FP trees: a FP tree on the actual data set that contains counts in each node (called the data TRIE tree), this set is typically the set of all strings of length L over the alphabet. As we describe below, the model FP tree helps guide the exploration of the model space in a way that avoids redundant work. The data FP tree helps us quickly compute the support of a model string. Recall that a FP tree with counts is merely a FP tree in which every node contains the number of leaves in the sub tree rooted at that node. In other words, every node contains the number of occurrences of the string corresponding to that node. CBFMM then explores the model space by traversing this (conceptual) model FP tree. Using the FP tree on the data set, CBFMM computes support at various nodes in the model space and

prunes away large portions of the model space that are guaranteed not to produce any results under the model. This careful pruning ensures that CBFMM does not waste any time exploring models that do not have enough support. The CBFMM algorithm simply stops when it has finished traversing the model FP tree and outputs the model strings that had sufficient support.

As mentioned above, we utilize the consensus tree node with its pointer for repetition detection algorithm. Our algorithm is linear-distance-based; we take the difference between any two successive position pointers leading to Difference vector, represented in Difference Matrix (Diff_matrix). Diff_matrix is not kept in the memory but this is considered only for the sake of explanation. Fig. 1 presents how the Diff_matrix is derived from the position pointers of a particular node. From the matrix the repetition is represented by (S, K, StPos, EndPos, c), denoting the pattern, period value, starting position and ending position, and number of occurrences respectively for a particular consensus node (which denote a pattern). CBFMM algorithm scans the difference vector starting from its corresponding position (Pos), and increases the frequency count of the period (K) if and only if the difference vector value is repeated regard to the StPos and K. Algorithm 1 in Appendix section formally represent the formation of Diff_matrix form consensus node pointers.

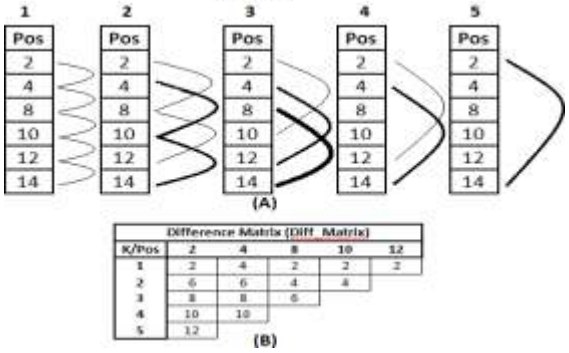


Fig.1. Difference Matrix calculation for 'ab' time series pattern from suffix tree node pointers

The noise- resilient features in repetition detection in presence of noise, is presented in [12] and [13]. Three types of noise generally considered in time series data are replacement, insertion, and deletion noise. In order to deal with this problem, [13] used the concept of time tolerance into the repetition detection process. The idea is that repeated occurrence can be drifted within a specified limit called time tolerance (denoted as tt), which is utilized in CBFMM algorithm. The CBFMM algorithm with time tolerance is presented in Appendix.

CBFMM algorithm calculates all patterns which are repeated starting from any position and continues till the end of the time series or till the last occurrence of the pattern. Our algorithm can also find the repeated patterns within a sub section of the time series. FP tree node which contains pointers (pos) accessed as a continuous pattern for Diff_matrix calculation. Such types of repetition calculation are very useful in real time DNA sequences and in regular time series. The existing algorithms [13] do not prune or prohibit the calculation of redundant repeats; the immediate drawback is reporting a huge number of repeats, which makes it more challenging to find the few useful and meaningful repeated patterns within the large pool of reported periods. Our algorithm reduces the number of comparison of pointers which are used for calculation periodicity presented in Appendix(Algorithm 2). We empowered to use p periods only one time for each and every position pointers from that Diff_matrix is calculated. Diff_matrix is able to assist in finding repetition for every starting position with different p periods. Our algorithm not only saves the time of the users observing the produces results, but also saves the time for computing the repetition by the mining algorithm itself.

The first paragraph under each heading or subheading should be flush left, and subsequent paragraphs should have a five-space indentation. A colon is inserted before an equation is presented, but there is no punctuation following the equation. All equations are numbered and referred to in the text solely by a number enclosed in a round bracket (i.e., (3) reads as "equation 3"). Ensure that any miscellaneous numbering system you use in your paper cannot be confused with a reference [4] or an equation (3) designation.

IV. OPTIMIZING STRATEGIES

There are some optimization strategies that we selected for the efficient implementation of the algorithm. These strategies, though simple, have improved the algorithm efficiency significantly. We do not include these into the algorithm pseudo code so as to keep it simple and more understandable. Some of these strategies are briefly mentioned in the text below.

1. Recall that each edge connecting parent node v to child node u has its own occurrence vector that contains values from leaf nodes present in the sub tree rooted at u. Accordingly, edges are sorted based on the

number of values they have to carry in their occurrence vectors; and edges that qualify to be processed in each step of the algorithm are visited in descending order based on the size of their occurrence vectors. This is beneficial as we calculate the repeats at each intermediate node, and we need to sort the occurrence vector at each intermediate node. Processing edges (indirectly) connected to the largest number of leaf nodes first will lead to better performance because this will mostly decrease the amount of work required to sort newly formed occurrence vectors.

2. We do not physically construct the occurrence vector for each intermediate edge as that would mostly result in huge number of redundant sub vectors. Rather, a single list of values is maintained and each intermediate edge keeps the starting and ending index positions of its occurrence list, and sorts only its concerned portion of the list. As the sorting might also require a huge number of shifting of elements, we maintain the globally unified occurrence vector as a linked list of integers so that the insertion and deletion of values do not disturb large part of the list.
3. Repeated subsequence detection at the first level (for edges directly connected to the leaves) is generally avoided because experiments have shown that in most cases, the first level does not add any new repeat. This is based on the observation that in time series repetitions patterns mostly consist of more than one symbol. This step alone improves the algorithm efficiency significantly.
4. Intermediate edges (directly or indirectly) connected to significantly small number of leaves can also be ignored. For example, if the sub tree rooted at the child node connected to an edge contains less than 1 percent leaves, there is less chance to find a significant repetitions pattern there. This leads to ignoring all intermediate edges (present at deeper levels) which would mostly lead to multiples of existing repeats. Experiments have shown that, in general, this strategy does not affect the output of the Subsequence detection algorithm.
5. Very small repeats (say less than five symbols) may also be ignored. Repeats which are larger than 30 percent (or 50 percent) of the series length are also ignored so that the infrequent patterns do not pollute the output. We also ignore repeats which start from or after index position n/2 , where n is the length of the time series. Similarly, intermediate edges which represent the string of length >n/2 are also ignored; that is, we only calculate repetitions for the sequences which are smaller than or equal to half the length of the series.
6. similarly, repeats smaller than edge value (the length of the sequence so far) are ignored. For example, in the series abababab$, if the edge value is 4 then we do not consider the repeat of size 2, which would otherwise mean that abab is repeated with p = 2, st = 0; this does not make much sense; rather the case should be expressed like abab is repeated with p = 4, st = 0.
7. The collection of repeats is maintained with two-levels indexing; separate index is maintained on repeat values and starting positions. This facilitates fast and efficient search of repeats because we check the existing collection of repeats a number of times. It is worth mentioning that the strategies outlined in points 3 and 4 are optional and can be activated according to the nature of the data, its distribution, and the sensitivity of the results.

The proposed algorithm requires only a single scan of the data in order to construct the FP tree; and produces all patterns with length ≥ 2. CBFMM performs many comparisons for comparison of two Diff_matrix values. The complexity of processing a Diff_matrix vector of length n would be O (N^2). The proposed algorithm depicts the Order of Growth is O ($N^{2)}$ complexity. The length of periodic patterns is independent of the size of the time series. The length of frequent patterns is independent of the time series length [13]. The cost of processing k levels would be O (k. N) because each k ≤ N, hence the sum of the size of all comparison in Diff_matrix, and the worst case complexity if processing a level is O (N^2). To analyze the space complexity of the FP tree will be gained because we produce only one copy of pattern at each time, the maximal number of generated nodes at ith level will not surpass N(L-i+1). The auxiliary storage used for running the subroutine is bound by O (N(L- i+1)) as well. Therefore, the total space complexity of FP tree is O (N×L).

V. EXPERIMENTAL EVALUATION

A. Experiment with Biological Data

The human genome sequence was downloaded from the National Centre of Biotechnology Information (NCBI) ftp site. To identify the corresponding three-dimensional protein structures of the human genome available in the Protein Data Bank (PDB), every sequence of the NCBI dataset was used as a query sequence against all the protein sequences available in the PDB using PSI-BLAST [44]. A 90% sequence cut-off was used. Using this procedure, a total of 3136 non-redundant structures from Homo sapiens was obtained, which comprised 5796 protein chains. This study makes extensive (and exclusive) use of the algorithm CBFMM to find internal sequence repeats. CBFMM was developed to find internal repeats within a sequence; it aligns the sequence on the X and Y axes. Next, it finds the suboptimal alignments and, finally, it displays the repeat along with the location after weeding out repeats that are merely subsets of larger repeats. After the repeats were

found, a web server, three-dimensional structural superposition (3dss) [27] was used to superimpose the three-dimensional structures of the repeats and obtain the structural alignment. Information about the protein was obtained from the Protein Structure Analysis Package (PSAP) [4]; and necessary three-dimensional atomic coordinates for the protein molecules used in the present study were obtained from the anonymous FTP server maintained at the Bioinformatics Centre, Indian Institute of Science, Bangalore, India. Further calculations and necessary analyses were carried out using locally developed Perl scripts.

CBFMM find all Cryptic repeats comprising eight or more amino acid residues are included. Out of the entire dataset, only 19 proteins were found to have 38 identical sequence repeats (Table 1). Out of the 38 large cryptic identical repeats found (Table 1), only two did not superimpose (from PDB-ids 1JBQ and 1FYH) since the atomic coordinates are missing in their PDB file. In fact, it is intriguing that although large-module repeats ought to exist in the proteins, none apart from the one in interferon-γ (PDB-id 1FYH) have remained so highly conserved with respect to the sequence. It is likely that identical and similar repeats serve some useful biological function, such as activity or scaffolding. This is supported by the fact that the amino acid sequence is highly conserved only in the case of some exacting function of the structure of the protein. An example of this can be seen in ice-binding β-sheets of insect anti-freeze protein [50]. Proteins with repeats conserved across species are under strong purifying selection [26]. Thus, large conserved repeats have properties of selectively conserved rather than neutral sequences.

Table 1 Large identical repeats from the non-redundant dataset of Homo sapiens proteins.

Sample Internal Sequence repeats in protein structures retrieved using CBFMM from PDB

PDB-id and chain	Name of protein	Repeat lengthS	Repeat	Location n	Location	STAMP score
1LARA	LAR protein	8 (H)	MVQTEDQY	258–265	549–556	9.792
1CZA N	Hexokinase I	10 (E)	GFTFSFPCQQ	151–160	599–608	9.781
2CMR A	Transmembrane glycoprotein	42(H)	QLLSGIVQQQNNLLRAI EAQQ HLLQLTVWGIKQLQARI LAGG	2–43	178–219	9.140
1OZ2 A	Lethal brain Tumour - like protein	8 (E)	MKLEAVDR	153–160	257–264	9.759
1OZN A	Reticulon 4 receptor	8 (E)	FLHGNRIS	38–45	159–166	9.786
1N11 A	Ankyrin	8 (H)	GLTPLHVA	405–412	569–576	9.785
1KI0 A	Angiostatin	8 (E)	ENYCRNPD	50–57	222–229	9.774

Table 2: Repeated Pattern Found in P14593

Per	stpos	StPosMod	EndPos	confidence	pattern	repetitions
4	312	2	345	0.66	d	5
4	321	0	469	1	gn	66
4	344	3	489	0.33	agn	64
4	367	2	480	0.85	aagn	58

The two protein sequences namely P09593 and P14593 can be retrieved from the Expert Protein Analysis System (ExPASy) database server (www.expasy.org).The protein sequence P09593 is S antigens protein in Plasmodium falciparum v1 strain. S antigens are soluble heat-stable proteins present in the sera of some infected individuals. The sequence of S antigen protein is different among different strains [5]. Diversity in S antigen is mainly due to polymorphism in the repetitive regions. It has been shown that the repeated sequence is ggpgsegpkgt with repetition of 11 amino acids, and this pattern repeats itself 19 times [37]. Some of the repeated patterns with repeat value of 11, found using CBFMM are presented in Table 2. Note that the index position starts from zero, as is the case with all the examples and experimental results reported in this paper. The second pattern is just a rotated version of the original pattern ggpgsegpkgt with 18 repeats while the first pattern starts (StPos = 104) just before the second pattern (StPos = 105) with 19 repeats. Thus, our algorithm finds not only the expected pattern of ggpgsegpkgt, but also shows that its shifted version gpgsegpkgtg is also repeated. The pattern exists in the middle of the series and is repeated contiguously. We have discovered 18 repeats of shifted pattern instead of 19, and the reason is that the repetition which starts at

position 282 is gpgsespkgtg; it is different from the expected pattern by one amino acid at position 6, where it has s instead of g. Since CBFMM does consider alternatives, we were able to detect that pattern. The first result also hints this by showing the pattern ggpgse repeating 19 times. The shifted pattern gpgsegpkgtg, detected by CBFMM, is an interesting result that has not been reported in any protein database or in any other studies, we are investigating the practical significance of this discovery.

B. Time Performance

The time performance of CBFMM compared to ParPer, CONV, WARP and STNR in three perspectives: varying data size, repetition size and noise ratio. First, we compare CBFMM performance against ParPer [21], with synthetic data with varying data size from 1,00,000 to 10,00,000. The results are shown in Fig.2. ParPer only finds partial periodic patterns in the data namely symbol, segment and sequence patterns, and their complexity is O (N^2). ParPer is not able to find repetition within subsection of a time series. ParPer show poor performance when the time series contain insertion and deletion noise; and which might be prevalent in the time series. STNR [13], CONV [34] and WARP [35] are compared with size of the series varied from 1,00,000 to 10,00,00,000. Fig.3 shows CBFMM performs better than WARP and STNR, but worse than CONV. The run time complexity of STNR and WARP is O (N^2), but for CONV is O (nlogn). CBFMM finds the repetition for all patterns in continuous or subsection of a time series even in the presence of noise. CBFMM can find singular events if exists in time series. CBFMM performs better than WARP and STNR because CBFMM applies optimization strategies, mostly reduced the redundant comparison. This supports our algorithm that time complexity does not grow along with the size of time series. In case of varying repetition, we fixed the time series length and symbol set size. CBFMM performance is shown in Fig.4 with varying repetition size from 5 to 100. ParPer
[21] and WARP [35] get affected as the repetition size increased. Time performance of CBFMM, CONV and STNR [13] remains same as it checks for all possible repetitions irrespective of the data set.

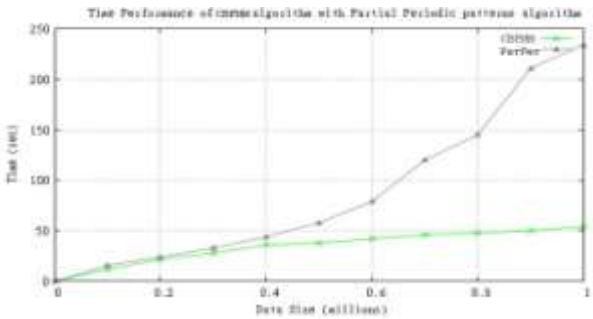


Fig.2 Time performance of CBFMM with ParPer algorithm.

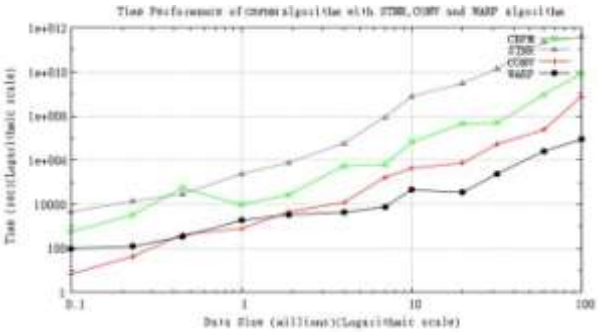
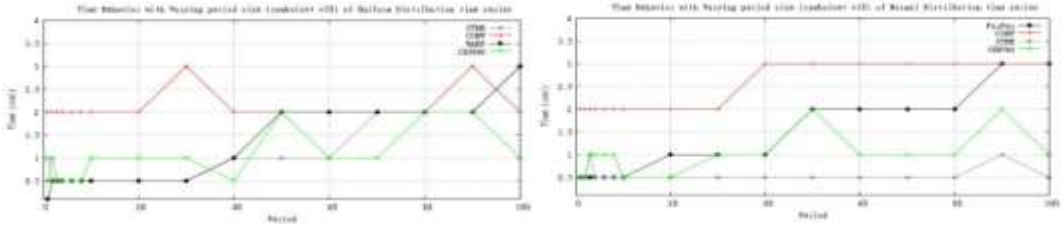
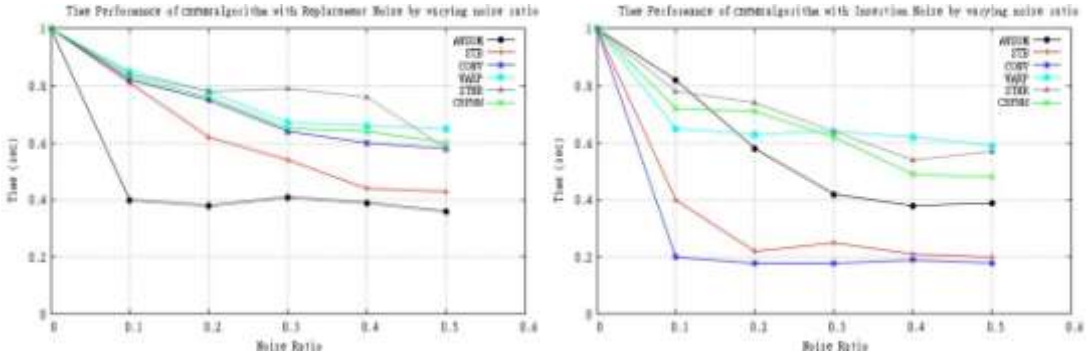


Fig.3 Time performance of CBFMM algorithm with STNR, CONV and WARP

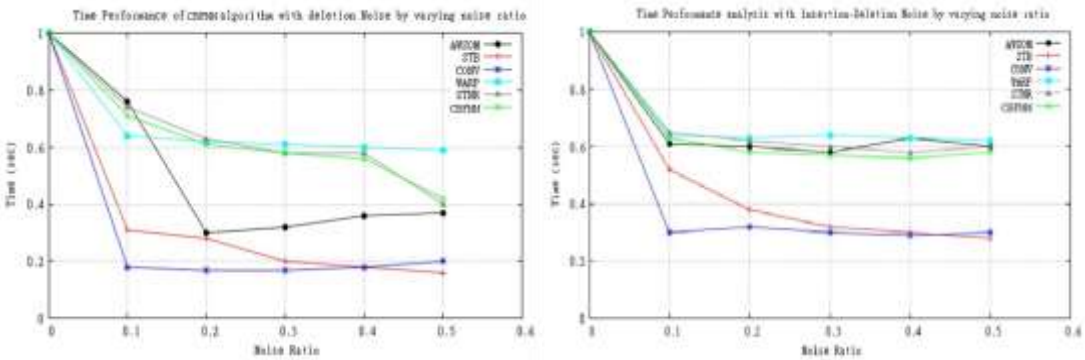


(a) (b)

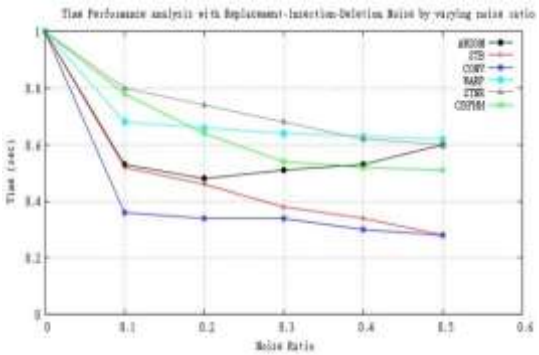
Fig.4 Time behavior with varying repetition size



(a) Replacement Noise (b) Insertion noise



(c) Deletion Noise (d) Insertion- Deletion Noise



(e) Replacement-Insertion-Deletion Noise

Fig.5 Time performance of CBFMM compared with STNR, CONV, ParPer, WARP, AWSOM, STB.

C. Noise Resilience

In the case of noise ratio, we used a synthetic time series of length 10,000 containing 4 symbols with embedded repetition size of 10. Symbols are uniformly distributed and the time series is generated in the same way as done in [35]. We used 5 combination of noise, i.e., replacement, insertion, deletion, insertion-deletion, and replacement-insertion-deletion. By gradually increased the noise ration from 0.0 to 0.5, the confidence at repetition of 10 is detected. The time tolerance for all the experiments is ±2. Fig.5 show that our algorithm compares well with WARP [35], STNR[13] and performs better than AWSOM[43], CONV[34], and STB[12]. For most of the combination of noise, the algorithm detects the repetition at the confidence higher than 0.5. The worst results are found with deletion noise, which disturbs the actual repetition. CBFMM shows consistent superiority because we consider asynchronous repeated occurrences which drift from the expected position within an allowable limit. This turns our algorithm a better choice in detecting different types of repetition.

VI. CONCLUSION

In this paper, we have presented a novel algorithm that uses FP tree as underlying structure. The algorithm can detect symbol, sequence and segment repetition as well as present the patterns that are repeated. It can also find repetition within a subsection of the biological data. It can detect the redundant repetitions which are pruned; before calculating confidence which in turn saves a significant amount of time. We took an initial step towards and understanding the constraints in the conservation of amino acid sequences by analyzing large cryptic identical and similar repeats. CBFMM is also superior to motif finding algorithms used in computational biology. We also presented experiments which show that CBFMM can scale to handle motif mining tasks that are much larger than attempted before. Our algorithm runs in O (k. N) in the worst case. In future, we are trying to extend our algorithm's working on online repetition detection. The algorithm to be experimented with streaming data using disk based tree [25].

REFERENCES

[1] A. Udechukwu, K. Barker and R. Alhajj, Maintaining Knowledge-Bases of Navigational Patterns from Streams of Navigational Sequences, RIDE, 2005, 37-44.

[2] A.A. Ptitsyn, Computational analysis of gene expression space associated with metastatic cancer, BMC Bioinformatics (BMCBI), 10(S-11), 2009, 6.

[3] A. W.C. Fu, J. Li and P. Fahey, Efficient discovery of risk patterns in medical data, Artificial Intelligence in Medicine, 45(1), 2009, 77-89.

[4] B. Balamurugan et al., PSAP: protein structure analysis package, J. Appl. Crystalogr., 2007, 40773–777.

[5] C. Berberidis and G. Tzanis, Mining for Mutually Exclusive Items in Transaction Databases, Database Technologies: Concepts, Methodologies, Tools, and Applications, 2009, 2192-2203

[6] C. Sheng, W. Hsu, M. L. Lee, J. C. Tong and S. K. Ng, Mining mutation chains in biological sequences, ICDE, 2010. 473-484

[7] C.F. Cheung, J.X. Yu and H. Lu, Constructing FP Tree for Gigabyte Sequences with Megabyte Memory, IEEE Trans. Knowledge and Data Engg., 17(1), 2005, 90-105.

[8] C.Sheng, W. Hsu and M.L. Lee, Mining Dense Periodic Patterns in Time Series Data, In Proceedings of 22nd IEEE Int'l Conf. Data Eng. (ICDE), Atlanta, Georgia, USA, 2006, 115.

[9] D. Tantz, D. Trick and G.A. Dover, Cryptic simplicity in DNA is a major source of genetic variation, Nature (London), 1986, 322652–656.

[10] E. Eskin, N. A. Furlotte, H. M. Kang and C. Ye, Mixed-model coexpression: calculating gene coexpression while accounting for expression heterogeneity, Bioinformatics, 27(13), 2011, 288-294.

[11] E. M. Marcotte and Y. Park, Revisiting the negative example sampling problem for predicting protein-protein interactions, Bioinformatics, 27(21). 2011, 3024-3028.

[12] F. Rasheed and R. Alhajj, Using FP Trees for Periodicity Detection in Time Series Databases, In Proceedings of IEEE Int'l Conf. Intelligent Systems, September 2008.

[13] F.Rasheed, M. Al-Shalalfa and R. Alhajj, Efficient Periodicity Mining in Time Series Databases Using FP Trees, IEEE Trans. Knowl. Data Engg. (TKDE), 23(1), 2011, 79-94.

[14] Fei Chen, Jie Yuan and Fusheng Yu, Finding periodicity in pseudo periodic time series and forecasting, GrC 2006, 2006, 534-537.

[15] G. Das , C. A. Ratanamahatana, J. Lin, D. Gunopulos, E. J. Keogh and M. Vlachos, Mining Time Series Data, Data Mining and Knowledge Discovery Handbook, 2010,1049-1077

[16] G. Pavesi, P. Mereghetti, G. Mauri and G. Pesole, Weeder Web: Discovery of Transcription Factor Binding Sites in a Set of Sequences From Co-Regulated Gene, Nucleic Acids Research, 32(2004), W199-W203.

[17] G.K. Sandve and F. Drablos, A Survey of Motif Discovery Methods in an Integrated Framework, Biology Direct, 1, 2006, 11-26.

[18] H. Wu, B. Salzberg, G.C. Sharp, S.B. Jiang, H. Shirato, and D. Kaeli, Subsequence Matching on Structured Time Series Data, Proc. ACM SIGMOD, 2005, 682-693.

[19] J. Buhler and M. Tompa, Finding Motifs Using Random Projections, J. Computational Biology, 9(2), 2002, 225-242.

[20] J. Han, W. Gong and Y. Yin, Mining Segment-Wise Periodic Patterns in Time Related Databases, In Proc. of ACM Int'l Conf. Knowledge Discovery and Data Mining, New York City, New York, USA, 1998, 214-218.

[21] J. Han, Y. Yin and G. Dong, Efficient Mining of Partial Periodic Patterns in Time Series Database, In Proc. of 15th IEEE International Conference in Data Engineering, Sydney, Australia, 1999, 106.

[22] J. Heringa and Bernd W. Brandt, Protein analysis tools and services at IBIVU, Journal Integrative Bioinformatics (JIB), 8(2), 2011.

[23] J. Wang and J. Han, BIDE: Efficient Mining of Frequent Closed Sequences, In Proceedings of 20th IEEE Int'l Conf. Data Eng. (ICDE), 2004, 79-90.
[24] J. Yang, W. Wang and P. Yu, InfoMiner: Mining Partial Periodic Patterns with Gap Penalties, In Proceedings of Second IEEE Int'l Conf. Data Mining, Maebashi City, Japan, 2002.
[25] J.M. Hancock and M. Simon, Simple sequence repeats in proteins and their significance for network evolution, Gene, 2005, 345113–118.
[26] J.M. Hancock, E.A. Worthey and M.F. Santibanez-Koref, A role for selection in regulating the evolutionary emergence of disease-causing and other coding CAG repeats in human and mice, Mol. Biol. Evol., 2001, 181014–1023.
[27] K. Sumathi, P. Ananthalakshmi, M.N.A.M. Roshan and K. Sekar, 3dss: 3-dimensional structural superposition, Nucleic Acids Res., 2006, 34W128–W134.
[28] K.Y. Huang and C.H. Chang, SMCA: A General Model for Mining Asynchronous Periodic Patterns in Temporal Databases, IEEE Trans. Knowledge and Data Engg., June 17(6), 2005, 774-785.
[29] L. Chen, M. Tamer Ozsu and V. Oria, Robust and Fast Similarity Search for Moving Object Trajectories, In Proceedings of ACM SIGMOD, 2005, 491-502.
[30] M. Das and H.K. Dai, A Survey of DNA Motif Finding Algorithms, BMC Bioinformatics, 8, 2007, S21-S33.
[31] M. Vlachos, G. Kollios and D. Gunopulos, Discovering Similar Multidimensional Trajectories, In Proceedings of 18th IEEE Int'l Conf. Data Eng. (ICDE), San Jose, California, USA, 2002, 673-684.
[32] M.A. Andrade and P. Bork, Heat repeats in the Huntington's disease protein, Nat. Genet, 1995, 11115–116.
[33] M.A. Andrade, C. Perez-Iratxeta and C.P. Ponting, Protein repeats: structure, functions and evolution, J. Struct. Biol., 2001, 134117–131.
[34] M.G. Elfeky, W.G. Aref and A.K. Elmagarmid, Periodicity Detection in Time Series Databases, IEEE Trans. Knowledge and Data Eng, 17(7), 2005, 875-887.
[35] M.G. Elfeky, W.G. Aref and A.K. Elmagarmid, WARP: Time WARPing for Periodicity Detection, In Proceedings of Fifth IEEE Int'l Conf. Data Mining, Houston, Texas, USA, November 2005.
[36] M.J. Zaki, SPADE: An Efficient Algorithm for Mining Frequent Sequences, Machine Learning, 42(1), 2001, 31-60.
[37] M.V. Katti, R. Sami-subbu, P.K. Rajekar and V.S. Gupta, Amino Acid Repeat Patterns in Protein Sequences: Their Diversity and Structural-Function Implications, Protein Science, 9(6), 2000, 1203-1209.
[38] P. Djian, Evolution of simple repeats in DNA and their relation to human disease, Cell, 1998, 94155–160.
[39] P. Indyk, N. Koudas and S. Muthukrishnan, Identifying Representative Trends in Massive Time Series Data Sets Using Sketches, In Proceedings of Int'l Conf. Very Large Data Bases, Cairo, Egypt, 2000.
[40] P.Patel, E. Koegh, J. Lin and S. Lonardi, Mining Motifs in Massive Time Series Databases, Proc. IEEE Int'l Conf. Data Mining (ICDM), Maebashi City, Japan, 2002, 370-377.
[41] S. Hoppner, Discovery of Temporal Patterns - Learning Rules about the Qualitative Behaviour of Time Series, In Proc. Fifth European Conf. Principle and Practice of Knowledge Discovery in Databases, Freiburg, Germany, 2001, 192-203.
[42] S. Ma and J. Hellerstein, Mining Partially Periodic Event Patterns with Unknown Periods, In Proceedings of 17th IEEE Int'l Conf. Data Engg., Heidelberg, Germany, 2001.
[43] S. Papadimitriou, A. Brockwell and C. Faloutsos, Adaptive, Hands Off-Stream Mining, In Proceedings of Int'l Conf. Very Large Data Bases (VLDB), Berlin, Germany, 2003, 560-571.
[44] S.F. Altschul, T.L. Madden, A.A. Schaer, J. Zhang, Z. Zhang, W. Miller and D.J. Lipman, Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, Nucleic Acids Res., 1997, 253389–253402.
[45] S.Sinha and M. Tompa, YMF: A Program for Discovery of Novel Transcription Factor Binding Sites by Statistical Over representation, Nucleic Acids Research, 31(13), 2003, 3586- 3588.
[46] T.F. Smith, C.G. Gaitatzes, K. Saxena and E.J. Neer, The WD-repeat: a common architecture for diverse functions, Trends Biochem. Sci., 1999, 24181–185.
[47] W. Wang and J. Yang, Mining Sequential Patterns from Large Datasets. Springer-Verlag, 28, 2005.
[48] X.Yan, J. Han and R. Afshar, CloSpan: Mining Closed Sequential Patterns in Large Datasets, In Proceedings of SIAM Int'l Conf. Data Mining (SDM), San Francisco, CA, USA, 2003.
[49] Y. Zhu and D. Shasha, WARPing Indexes with Envelope Transforms for Query by Humming, In Proceedings of ACM SIGMOD, 2003, 181-192.
[50] Y.C. Liou, A. Tocilj, P.L. Davies and Z. Jia, Mimicry of ice structure by surface hydroxyls and water of a beta-helix antifreeze protein, Nature (London), 2000, 406322–324.