I. J. Computer Network and Information Security, 2012, 10, 37-46
Published Online September 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2012.10.04

Constraint Based Periodicity Mining in Time Series Databases

Dr.Ramachandra.V.Pujeri, G.M.Karthik
Vice-Principal, KGiSL Institute of Technology, Saravanampatti, Coimbatore -641035, Tamil Nadu, INDIA
Assistant Professor, CSE Dept., SACS MAVMM Engineering College, Madurai -625301, Tamil Nadu, INDIA
sriramu.vp@gmail.com, gmkarthik16@gmail.com

Abstract — The search for the periodicity in time-series database has a number of application, is an interesting data mining problem. In real world dataset are mostly noisy and rarely a perfect periodicity, this problem is not trivial. Periodicity is very common practice in time series mining algorithms, since it is more likely trying to discover periodicity signal with no time limit. We propose an algorithm uses FP-tree for finding symbol, partial and full periodicity in time series. We designed the algorithm complexity as *O (kN)*, where *N* is the length of input sequence and *k* is length of periodic pattern. We have shown our algorithm is fixed parameter tractable with respect to fixed symbol set size and fixed length of input sequences. Experiment results on both synthetic and real data from different domains have shown our algorithms' time efficient and noise-resilient feature. A comparison with some current algorithms demonstrates the applicability and effectiveness of the proposed algorithm.

Index Terms — Data Mining, CBPM, FP-tree, Periodicity mining, Time series data, Noise resilient

I. Introduction

A collection of data are gathered and observed at uniform interval of time to reflect certain behavior of an entity. A time series is mostly discretized before it is analyzed [8], [9], [13], [18], and [19]. Several example of time series such as frequently sold products in a retail market, frequent regular interval pattern in DNA sequence, stock growth, power consumption, computer network fault analysis, transactions in a superstore, gene expression data analysis [7], [12], [22], [23] etc. In the above examples, we observe that the occurrence periodicity plays an important role in discovering some interesting frequent patterns in a wide variety of application areas. Identifying repeating (periodic) patterns could reveal important observations about the behavior and future trends of the case represented by the time series [35], and hence would lead to more effective decision making. The goal of time series analysis is to find whether and how frequent a periodic pattern (full or partial) is repeated within the data. In time series is said to have three types of periodic patterns (*symbol / Sequence / Segment*) can be detected [26]. For example, in time series contain the hourly number of transactions in retail store; the mapping different ranges of transactions (is referred as discretization process); a: {0} transactions, b: {1-300} transactions, c: {301-600} transactions, d: {601-1200} transactions, e: {>1200} transactions. Based on this mapping, the time series T' =0,212, 535, 0, 398, 178, 0, 78, 0, 0, 102, 423 can be discretized into *T = abdacbabaabc*. At least one symbol is repeated periodically in time series *T* is referred as *Symbol periodicity*. For example T = a bd a cb a ba a bc, symbol *'a'* is periodic with periodicity p=3, starting at position zero. *Sequence periodic or partial periodic pattern* consists of more than one symbol, maybe periodic in a time series. For example T = ab dacb ab aabc, symbol *'ab'* is periodic with p=5 starting at position zero. In whole time series, a repetition of pattern or segment is called *segment or full-cycle periodicity*. For example T = abdc abdc abdc has segment periodicity of p=5 starting at position zero.

Real time examples are mostly not characterized by perfect periodicity in time series. A time series is said to have three type of periodic pattern: 1) symbol periodicity, 2) sequence periodicity or partial periodic pattern, and 3) segment or full-cycle periodicity [26]. The degrees of perfection calculated by confidence, and are mostly characterized by the presence of noise in the data. Many existing algorithms [8], [9], [13], [17] detects periods that span through entire time series. Some algorithms detect all the above mentioned three type of periodicity, along with noise within subsection of time series, separately for each patterns [26]. Compared to this, we show that our *Constraint Based Periodicity Mining (CBPM)* technique is more efficient and flexible. We also demonstrate through empirical evaluation that *CBPM* is more scalable and faster than existing methods.

We propose a new efficient pattern enumeration approach on ideas of frequent pattern mining techniques. First, we construct a TRIE –like data structure called consensus tree which explores the space of all motifs, and enables a highly parallelized search along the tree motif. The growth of the tree is restrained by providing additional mining constraints. The consensus tree is fixed and anchored with symbol set size and length of input sequence. The construction of consensus tree detects symbol, sequence, and segment patterns without periodicity, within subsection of the series. The

additional constraint (namely user-specified *level* and *rule* constraint) will prune and eliminate redundant patterns. Secondly, the algorithm looks for all periods starting from all positions available in a particular node of consensus tree. All the node of the consensus tree exists based on confidence greater than or equal to the user-specified periodicity threshold.

We make the following contributions in this paper:

1. 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.
2. We describe a novel motif periodicity mining algorithm called *CBPM* (*Constraint Based Periodicity Mining*) that uses a concurrent traversal of frequent pattern trees to efficiently explore the space of all motifs.
3. We present a comparison of *CBPM* with several existing algorithms (*CONV* [8], *WARP* [9], *ParPer* [15] and *STNR* [26]). In fact, we show that *CBPM* is able to identify many periodicities in a single span that existing algorithms miss.
4. We show that our algorithm is scalable, accurate, and often faster than existing methods by more than an order of magnitude.
5. We show *CBPM* can be applicable to biological data sets and results are compared with existing algorithms like SMCA[16] in terms of produced results.
6. We also detailed algorithm analysis for time performance and space consumption.

The remainder of the paper is organized as follows: Section 2 presents related work and Section 3 is preliminaries. Section 4 describes our proposed algorithm. In Section 5 algorithm is analyzed for complexity and the utilized optimization strategies. Experimental results are reported in Section 6 using both real and synthetic data. Section 7 is conclusion and future research directions.

II. Related works

There is a vast amount of literature on mining databases for frequent pattern [6], [27], [34]. The problem of mining for subsequence was introduced in [1]. Subsequence mining has several applications, and many algorithms like [33], [36], and [38] 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 [4], [11], [32], [39]. The existing algorithm [2], [10], [13], [20], [37] requires the user to specify the period and patterns occurring with that period, otherwise which look for all possible periods in the time series. Some algorithms are classified based on the detection type of periodicity for symbol, sequence or segment. Another algorithm that finds frequent trends in time series data was proposed in [31]. 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 [8], [9], [17], [26], looks for all possible periods by considering the range. COVN [8] fails to perform well when the time series contains insertion and deletion noise. *WARP* [9] can detect segment periodicity; it cannot find symbol or sequence periodicity. Sheng et al. [29], [30] developed algorithm based on [15] *ParPer* to detect periodic patterns in a section of the time series; their algorithm requires the user to provide the expected period value. COVN, *WARP* and *ParPer* are augmented to look for all possible periods, and which last till the very end of the time series. Cheung [5] used suffix tree similar to *STNR* [26] which is not beneficial in terms of growth of tree. Huang and Chang [16] and *STNR* [26] presented their algorithm for finding periodic patterns, with allowable range along the time axis. Both finds all type of periodicity by utilizing the time tolerance window and could function when noise is present. *STNR* [26] can detect patterns which are periodic only in a subsection of the time series. Periodic 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 both structured motif extraction problem [22], [23] and periodicity 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 periodicity. In this paper, we present a flexible algorithm that handles general extended structured motif extraction problem and uses *CBPM* to build Consensus tree. *CBPM* is capable of reporting all types of periods 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 period, without requiring specific knowledge about the characteristics of the resulting motif.

III. Preliminaries

Suppose $\sum$ is a finite symbol set and $|\sum|$ its cardinality. For DNA, $|\sum|$ is 4, and the symbols are the 20 amino acids. Let $S = \{s_1, s_2, \dots, s_N\}$ of input time series sequences over a finite symbol set $\sum$ with$|\sum| = R$, such that$|s_i| = L, 1 \le i \le N$, and positive integer d and q such that $0 \le d \le L$ and$1 \le q \le N$. Here given parameters N and L are the number and length of given input sequence. Let a is called a pattern (center string) if each of at least q input sequence contains a substring in a's d-neighborhood. Find all center string $t \in \sum^l$ with any length l, $0 \le d < l \le L$ such that for each t, there are at least q sequences of S containing an *x-mutated* copy $(x \le d)$ of t. In real time, we have to investigate time series to identify repeating patterns along with outliers in time series. In this paper, we concentrate to develop an

algorithm capable of detecting patterns (center strings) of all possible length l and finding positions of all (degenerative/outliers) instances of these patterns. From generated patterns detect symbol, sequence and segment periodicity, which are formally defined in [26]. We define the confidence and specify the degree of confidence as constraints utilized during the construction of the consensus tree. We handled the degree of confidence into two ways. First, user may request to find pattern (symbol / segment / sequence) appearing in specific number of given input sequence, and each with of specific length and mutation. To achieve higher degree of accuracy for above said we use *rule* and *level* constraints, which in turn helps to prune the consensus tree effectively. Second, deals with the confidence of a periodic pattern occurring in time series sequence. Formally as mentioned in [26], the confidence of pattern X with periodicity p starting at position stops is ration between actual periodicity of pattern X with perfect periodicity of pattern X. We designed *CBPM* algorithm to mine periodicity from given input time series sequence in two phases. First phase, we construct TRIE like FP tree with or without mutation and in the second phase, we use the consensus tree to calculate the periodicity of various pattern in the time series.

IV. Constraint Based Periodicity Mining

A. First Phase – FP tree construct

In spite of the relatively lower complexity of Apriori compared with the brute force enumeration method, it may still raise a high cost when applied to complex data such as data with long patterns. In order to deal with such kind of data, another technique, the FP-tree (frequent pattern tree) technique, was proposed [15]. Generally, each frequent itemset is represented as a path of a tree, from root to some leaf node. A delicate technique is used to let itemsets share as many nodes as possible in their path representations, in a way such that more frequently occurring items will have better chance of sharing nodes than less frequently occurring items. Thus, as pointed in [16], "the FP-growth method is efficient and scalable for mining both long and short frequent patterns." FP –tree method has inspired us to consider the possibility and appropriateness of applying their basic ideas and techniques to solve the periodicity problem. The periodicity problem also asks for mining pattern by finding all their mutated copies limited by a distance$d \geq 0$. The mutated copies refer the handling asynchronous periodicity by locating periodic pattern that may alter by noise up to an allowable limit. Hence, using the mutation factor, our technique becomes *noise-resilient* technique. In this paper, the mutated copies of a pattern (sometimes referred as *consensus pattern*) that they occur quite often in the input time series sequences. Any sequence containing a consensus pattern is called an origin of the pattern. We take all sub-patterns of the input sequences as seeds and use them iteratively to find longer consensus pattern by *a level-wise search* strategy, which is also a *downward closure property*.

In a FP tree (referred as *consensus tree*) there are $|\sum|$ branches grown out from each nonleaf node n. We map each sub-patterns t of each time series sequence s to a path starting from the root of the tree. Each node n contains pointers to all substrings mapped to n, where a pointer (j, k, e) points to a substring starting at kth position of the jth sequence among given N input time series sequence with level of mutation $e \leq d$. The consensus tree is usually not full, because any node containing pointers pointing to less than q input sequences, not satisfying *rule* and *level* constraints will have no successors. The consensus tree growth is pre pruned based on the constraints which happens at each level like backward closure property. In *Constraint Based Periodicity Mining (CBPM)*, constraint are categorized into five constraints [14, 28] in which *CBPM* technique use *antimonotone, monotonic* and *succinct* constraints to restraint the growth of the consensus tree with reasonable complexity. The number of levels in the consensus tree is at most L of the sequences. Nodes with confidence value as $conf(b) = \big(N - sup(b)\big)/(N - q) < 1$ will be pruned; it is used as an *antimonotonic* constraint [28]. For each node support value calculated based on the number of sequence that do not contribute in production of consensus pattern. A node in the consensus tree will branch out only if a support value is $\leq q$ which is used as a *monotonic* constraint [28]. Each pointer in a consensus node has to satisfy degree of mutation $e>d$, otherwise it will be also pruned which sustains all position in consensus node like *succinct* constraint [28]. For each pointers without the mutation level $e>d$ will not participate in production of pointers in next consensus node.

Fig. 1 shows the structure of our consensus tree, and explains the dynamic level-wise growth process of the tree. When l=1, sub pattern *a, b* appears in all sequences. Then *aa, ab, ba,* and *bb* are produced to test whether they are consensus pattern at the second level of the tree. But, since *aa* appears in less than four sequences, (*a*) is a leaf node and (*aa*) is pruned by rule constraint. This has a consequence that any sub-pattern containing (*aa*) is definitely not a pattern and will be forbidden to grow in the next level according to the downward closure property. The next level does not exist if *level constraint* is not satisfied. *CBPM* does not produce all patterns. It produces only the longest pattern in order to save time. But, it is easy to modify it to get all patterns.

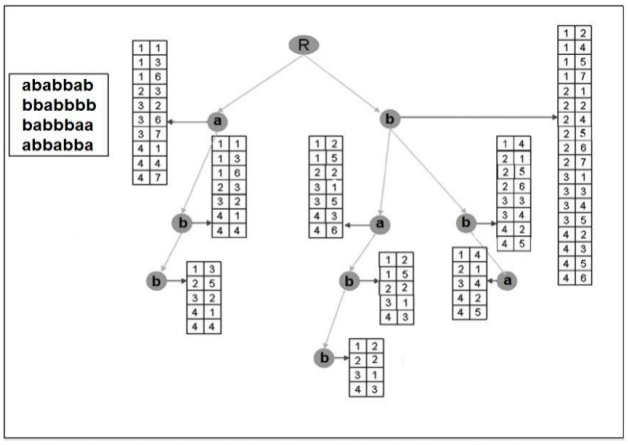


Figure 1: An example of consensus tree structure constructed using CBFP algorithm with d = 0, Σ = {a, b} and S = {(ababbab), (bbabbbb), (babbbaa), (abbabba)}, N = 4, L = 7, q = 4, and |∑| = 2.

Algorithm 1: FP-tree construction

1. **For** each string *j* of given input sequence *N* **do**
2. **For** each symbol *k* of input string *j* of length *L* **do**
3. **If** the *k*th symbol *i*th sequence is $b_1 \in \sum$ **do**
4. Put $(j, k, 0)$ in new node S_{b_1} , find $(j, k, 1)$ substring is in all $S_{b_1'}$ for $b_1' \neq b_1$ and j in $T_{b_1''}$ for each $b_1'' \in \sum$ if and only if $sup(b_i) > threshold$.
5. **For** each *i*th sequence from 1 to *L* **do**
6. **Loop(1):**
7. **For** each substring's $conf(b_1, b_2, b_3, \ldots, b_{i+1}) \geq 1$ **do**
8. **Loop(2):**
9. **For** each entry (j, k, e) in each nodes $S_{b_1,b_2,\ldots,b_i}$where $k < L - i + 1$ **do**
10. **Loop(3):**
11. **If** the $k + i$ th element of the *j*th sequence is $b_{i+1} \in \sum$ and $sup(b_{i+1}) < q$ **do**
12. **Begin(1):**
13. put (j, k, e) in $S_{b_1,b_2,\ldots b_i,b_{i+1}}$;
14. **if** $e < d$ then for all $b_{i+1}' \neq b_{i+1}$
15. put $(j, k, e + 1)$ in $S_{b_1,b_2,\ldots b_i,b_{i+1}}$ if and only if $conf(b_1, b_2, b_3, \ldots, b_{i+1}) \geq 1$;
16. **End Begin 4;**
17. **If** $conf(b_{i+1}) < 1$ **then** Remove $S_{b_{i+1}}$**;**
18. **End Loop 3;**
19. **For** each node $S_{b_1,b_2,\ldots,b_{i+1}} \neq \emptyset$ **do**
20. **For** each node in next level $S_{b_1',b_2',\ldots,b_i'}$ with $distance(b_i, b_i') \leq d$ **do**
21. **For** each $S_{b_1'',b_2'',\ldots,b_i''} \neq \emptyset$ and $conf\left(S_{b_1'',b_2'',\ldots b_i''}\right) \geq q$ along with $distance(b_i'', b_i') \leq d$ **do**
22. **Loop(5):**
23. **If** $conf(b_i'') < 1$ **then** Remove $S_{b_i''}$
24. Create a new level in consensus tree with $\boldsymbol{T_{b_1'',b_2'',\ldots b_i''} \leftarrow T_{b_1'',b_2'',\ldots b_i''} \cup S_{b_1'',b_2'',\ldots b_i''}}$
25. **If** no node exists in $\boldsymbol{T_{b_1'',b_2'',\ldots b_i''}}$ **then**
26. **Increment** *i* ; **End Loop2;**
27. **Else**
28. **Print** the output sequence $\left(b_i'', S_{b_i''}\right)$;
29. **End Loop 5;**
30. **If** all $S_{b_1,b_2,\ldots b_i,b_{i+1}}$ are removed then stop the program **else** output all pairs $\left(b_1, b_2, \ldots, b_{i+1}; S_{b_1,b_2,\ldots,b_i,b_{i+1}}\right)$
31. Remove all $S_{b_1,b_2,\ldots,b_i}$and $T_{b_1,b_2,\ldots,b_i}$;
32. **End Loop 2;**
33. $i = i + 1$**;**
34. **End Loop 1;**

CBPM performs many compare operations for calculating distance between two patterns. Mainly, complexity *O (i)* deals with comparing the symbols of the two strings one by one in *ith* level of the consensus tree. For each candidate center string in each node is $\sum_{j=0}^{min(d,i)} \binom{i}{j} (|\Sigma| - 1)^j$, calculated at most $N \times (L - i + 1)$ distances. The time complexity is roughly bound by $O(N \times L)$. The secondary storage used for running the CFPM is bound by $O\big(N \times (L - i + 1)\big)$. Therefore, the total space complexity is $O(N \times L)$. Testing all possibilities of patterns restrained by *rule* and *level* constraints this would raise the time complexity to $|\Sigma|^L$. We generate those candidates whose consensus strings satisfy the prescribed *rule* constraint. Therefore, using our strategy we raise time complexity by $O(N \times L)$ with quite low space complexity.

B. Second Phase- Periodicity Detection Algorithm

As mentioned above, we utilize the consensus tree node with its pointer for periodicity 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. 2 presents how the *Diff_matrix* is derived from the position pointers of a particular node. From the matrix the periodicity 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). *CBPM* 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 periodic regard to the *StPos* and *K*. Algorithm 2 formally represent the formation of *Diff_matrix* form consensus node pointers.

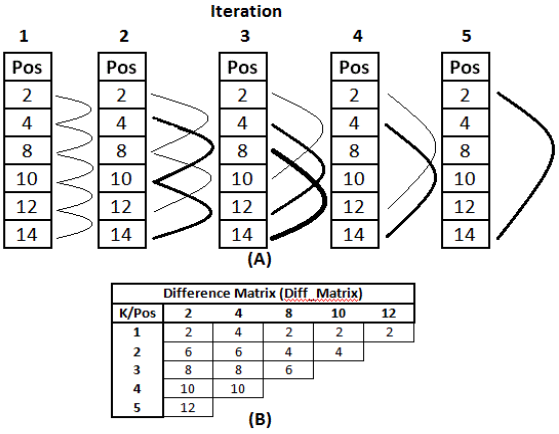


Figure 2: Difference Matrix calculation for 'ab' time series pattern from FP tree node pointers

The noise- resilient features in periodicity detection in presence of noise, is presented in [25] and [26]. Three types of noise generally considered in time series data are replacement, insertion, and deletion noise. In order to deal with this problem, [26] used the concept of time tolerance into the periodicity detection process. The idea is that periodic occurrence can be drifted within a specified limit called time tolerance (denoted as *tt*), which is utilized in *CBPM* algorithm. The *CBPM* algorithm with time tolerance is presented in Algorithm 3.

Algorithm 2: Difference Matrix (*Diff_matrix*) Algorithm

- **Input**: a time series (***S***) of size ***N*** contains position pointers **Pos;**
- **Output:** Difference Matrix (***A***) containing Difference vector;

1. **For** $i = 1\ to\ N-1$
2. **Begin Loop** 1:
3. **Assign** $j = 1$
4. **if** $(j < N - i)$
5. $A(j,i) = S_j - S_{j+i}$;
6. **if** $(j+1 \neq j+i)$
7. **Then**
8. $t = j + 1$;
9. **While**$(t < j + i - 1)$
10. **Begin Loop** 2:
11. $A(t,i) = S_t - S_{t+i}$;
12. $t = t + i$;
13. **End Loop** 2;
14. **Endif**;
15. $j = j + i$;
16. **Endif**;
17. **End Loop** 1;

Algorithm 3: Constraint Based Periodicity Mining Algorithm (*CBPM*)

- **Input**: *Diff_matrix* (**A**), and time tolerance value ***tt***;
- **Output**: position of periodic patterns **P**;

1. **For** $K = N - 1\ to\ 1$;
2. **Begin Loop** 1:
3. **For** $i = 1\ to\ N - k$;
4. **Begin Loop** 2:
5. **Assign** $j = i, c = 1$;
6. **if**$(j + K \leq N - K)$ **then**
7. **if** Difference $(A(j,K), A(j+K,K))$ is in between $(A(j,K) \pm tt)$
8. **then** $c++$;
9. **Endif**;
10. **if** $\exists P_{j,K+1}$ and Diff $(A(j,K), A(j,K+1))$ is in between $(A(j,K) \pm tt)$
11. **then** $c = c + P_{j,K+1}(q)$;
12. **Endif**;
13. $j = j + k$;
14. **Goto step** 6:
15. **Else**
16. **Assign** $stPos = j, endpos = j + k,\ p = k, q = c$;
17. Project Periodicity $P_{j,K}(S, p, stPos, endPos, q)$;
18. **If** $(i + K > N - K)$
19. **Break Loop** 2;
20. **Endif**;
21. **End Loop** 2;
22. **End loop** 1;

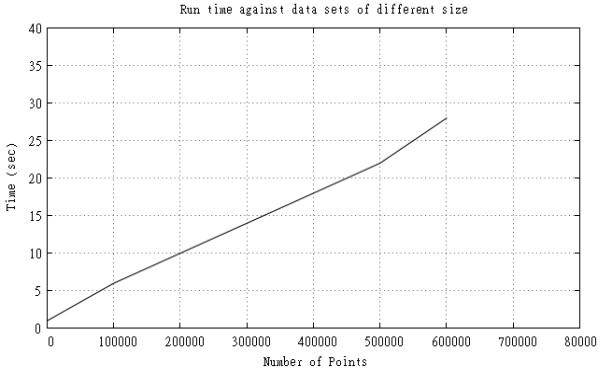
CBPM algorithm calculates all patterns which are periodic starting from any position and continues till the end of the time series or till the last occurrence of the pattern. Our algorithm can also find the periodic patterns within a sub section of the time series. FP tree node which contains pointers (*pos*) accessed as a continuous pattern for *Diff_matrix* calculation. Such types of periodicity calculation are very useful in real time DNA sequences and in regular time series. The existing algorithms [26] do not prune or prohibit the calculation of redundant periods; the immediate drawback is reporting a huge number of periods, which makes it more challenging to find the few useful and meaningful periodic patterns within the large pool of reported periods. Our algorithm reduces the number of comparison of pointers which are used for calculation periodicity. In 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 periodicity 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 periodicity by the mining algorithm itself.

V. Algorithm Optimiczation and Analysis

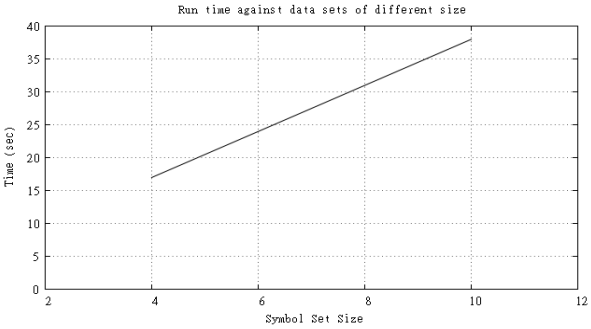
CBPM algorithm is optimized to improve the algorithm efficiency significantly. First, FP tree allowed reducing the redundant node creation by constraints. Each node creation done based on the values of the user defined constraints (*rule* and *level*). This is useful as we calculate the period in an encoded time series for symbol and sequence periodicity. The first level of the consensus tree nodes are used in Symbol periodicity detection and

other level consensus node are used in Sequence periodicity. Segment and Segment periodicity differs only with the mutation factor. Segment periodicity uses same consensus node as Sequence periodicity but considers only the position pointer with zero mutation factor value. Secondly we do not physically construct the *Diff_matrix* for each consensus node which results in huge number of redundant comparison of position pointers. Rather, a single list of periods (p) is maintained and starts producing periodicity considering every position pointers as starting position. The First level periodicity is generally avoided because experiments have shown that in most cases, the first level does not add any new period. This is based on the observation that in time series periodic patterns mostly consist of more than one symbol. Third, usage of *rule* and *level* constraints in FP tree which will prune unwanted nodes from the consensus tree. For example, if the node contains less than 1 percent child nodes, there is less chance to find a significant periodic pattern there. This leads to ignoring all nodes which would mostly leaf to multiples of existing periods. Finally, the collection of periods is maintained in separate index similar to [26]. This facilitates fast and efficient search of periods because we check the existing collection of periods a number of times.

The proposed algorithm requires only a single scan of the data in order to construct the suffix tree; and produces all patterns with length ≥ 2. *CBPM* 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 $O(N^2)$ 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 [26]. The cost of processing k levels would be $O(k.N)$ because each $k \leq N$, 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 suffix 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 $O(N \times L)$.



(a)



(b)

Figure 3: Run time against data sets of different size

VI. Experimental Evaluations

We tested our algorithm over a number of data sets. For real data experiments, we used supermarket data which contains sanitized data of timed sales transactions for Wal-Mart stores over a period of 15 months. Synthetic data taken from Machine Learning Repository [3] were also used. We tested how *CBPM* satisfies this on both synthetic and real data. Data generation is controlled by constraints for obtaining specific data distribution; based on symbol set size and amount of noise (replacement, insertion and deletion or any mixture) in the data. The algorithm can find all periodic patterns 100 percent. The size of the symbol set implies the number of computation of size of N required. The time shown on the Fig. 3 corresponds to synthetic control data set of N = 452378 time points. This is an important feature in using FP tree which guarantees identifying all repeating patterns.

A. Accuracy

In order to test the accuracy, we test the algorithm for various period sizes, distribution and time series length. We used synthetic data obtained from Machine Learning Repository [3], have been generated in the same done in [8]. Fig. 3 (a) shows the behavior of the algorithm against the number of the time points in the time series. Fig. 3 (b)b shows that the algorithm speeds up linearly to symbol set $|\sum|$ of different size. The size of the symbol set is FPT (fixed-parameter tractable) when the number of sequences N is fixed. *CBPM* checks the periodicity for all periods within synthetic data in absence of noise.

Table I. *CBPM* algorithm output for Wal-Mart data

Data	Threshold	No. of Periods	StPos	EndPos	Conf	Pattern
Store 1	0.8	4	109968	145081	0.42	AAA*******AAA******AA***
	0.7	9	134887	161412	0.4	AAABBBCCC**********AA*
	0.6	11	151141	194123	0.3	AABBBBCCCD******AAAA***
	0.5	16	213476	263129	0.32	AAAABBCCD***AADD********
	0.4	25	234980	280673	0.4	AAA***AAAAA*********AAA*

Store 2	0.8	6	180613	199457	0.44	AAA****BCCC* ****DD******
	0.7	7	164319	200312	0.47	AAB****AAABB BDDD****BB*
	0.6	13	229846	273422	0.37	AAAABBB***** **CC****DDD*
	0.5	17	215978	286421	0.4	AAAAACCCC** **BBBCC***DD
	0.4	20	283149	304231	0.41	AAAAAACCC** **BB*********
Store 3	0.8	5	147030	155044	0.42	AA****BBB***B CDDD*******
	0.7	8	152783	167329	0.3	AAAABBBC**** ***CCCCD****
	0.6	12	182390	186064	0.46	AAAAAACCCC CD**********
	0.5	14	177389	216892	0.47	AAAAABBBBBC CCCD*********
	0.4	27	258202	299582	0.49	AAAABBBBBC* ******BCCDDD

Table II. *CBPM* algorithm output for Wal-Mart data (Store 1)

Periodicity threshold	No. of periods (symbol)	No. of periods (segment)	No. of periods (sequence)
0.8	2123	6	4
0.7	2468	156	67
0.6	2741	671	89
0.5	3987	1033	101
0.4	4531	2152	138

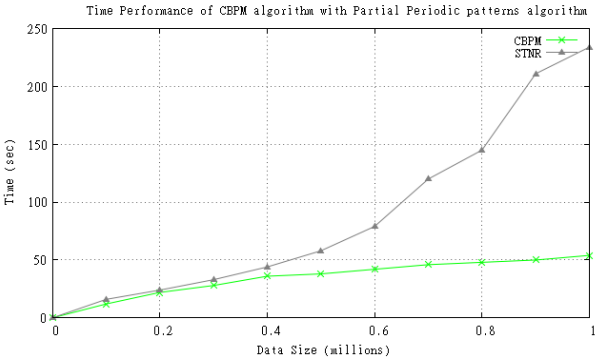


Figure 4. Time performance of CBPM with ParPer algorithm.

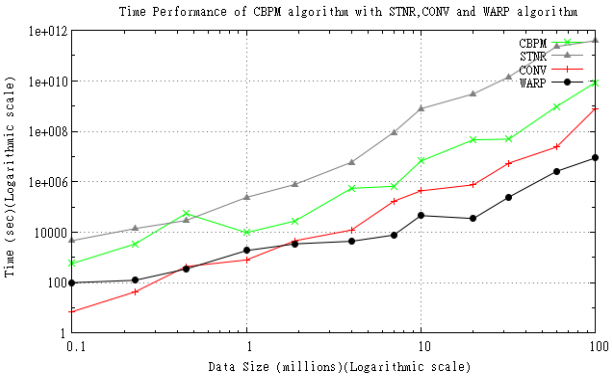
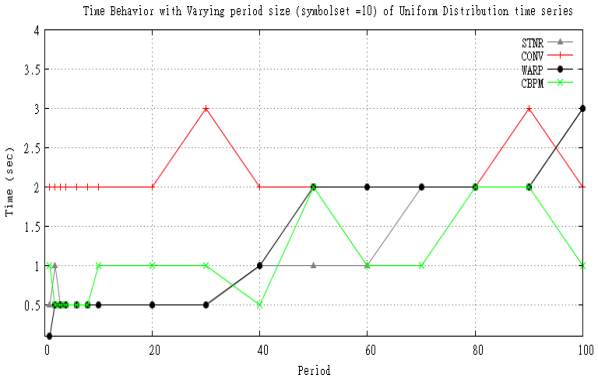
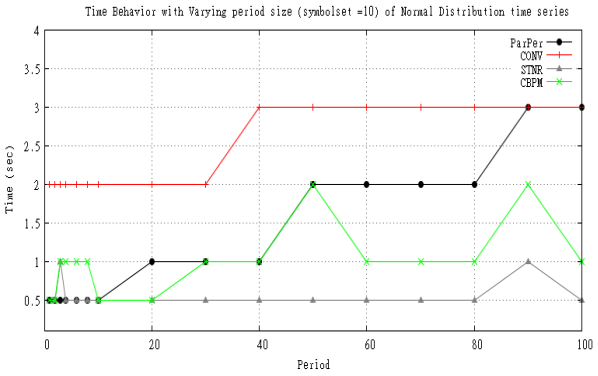


Figure 5: Time performance of CBPM algorithm with STNR, CONV and WARP



(a)



(b)

Figure 6: Time behavior with varying period size

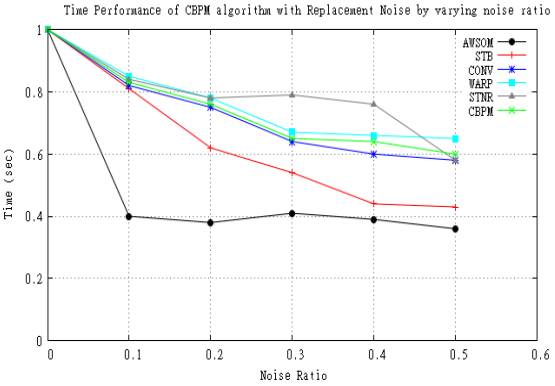
B. *Real data analysis*

For real data experiments, we used the Wal-Mart data which contains hourly based records of all transaction performed at a Supermarket. The data contains the record of around 15 months of data with expected period value of 24. The Wal-Mart data are discretized into five regions; very low (0 transaction), low(less than 250 transaction), medium (between 250 to 450 transaction), medium (between 450 to 650 transaction), high (between 650 to 850) and very high (above 850 transaction) mapped respectively to symbols a, b, c, d and e. We run our *CBPM* algorithm with periodicity threshold values ranging from 0.8 to 0.4 and observed: the number of periods captured by algorithm, *StPos* and *EndPos* of the sequence, confidence value and the Pattern shown in Table 1. The expected period 24 is captured at the threshold value 0.8. We observed from the above results that algorithm never filled the don't-care symbol (*) in the sequence. The patterns are periodic mostly weekly, which is captured in our results at the period. Periodic pattern obtained less in number but accurate, useful and meaningful. Table 2 presents the number of symbol, segment and sequence periodic patterns. It shows that initial and the closing hours generally have the least number of transactions. The number of transaction increases as the day progresses, which is also evident. Table 1 and 2 demonstrates that how periodic pattern are obtained without redundant period. *CBPM* algorithm does not calculated redundant period, because which are supper-pattern has already been found periodic with same

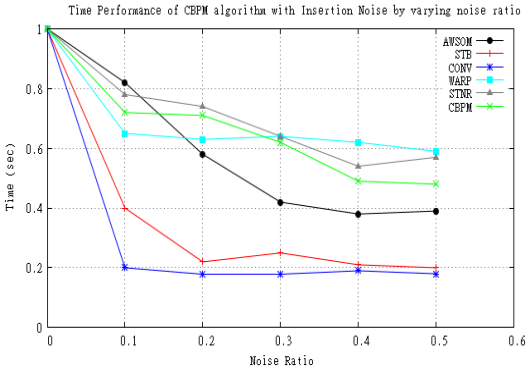
period value using *Diff_matrix*. Periodicity is calculated using *Diff_matrix* from bottom to top, hence algorithm does not check the redundant periods.

C. Time Performance

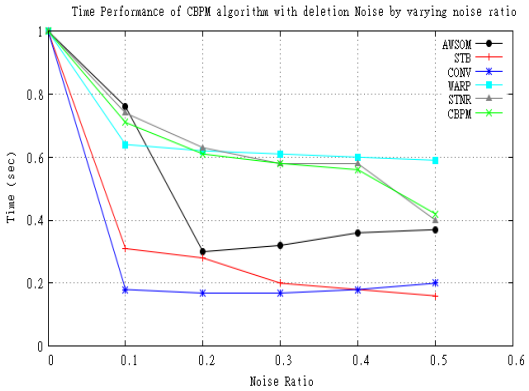
The time performance of *CBPM* compared to *ParPer*, *CONV*, *WARP* and *STNR* in three perspectives: varying data size, period size and noise ratio. First we compare *CBPM* performance against *ParPer* [15], with synthetic data with varying data size from 1,00,000 to 10,00,000. The results are shown in Fig.4. *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 periodicity 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* [26], *CONV* [8] and *WARP* [9] are compared with size of the series varied from 1,00,000 to 10,00,00,000. Fig.5 shows *CBPM* 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 $O(nlogn)$. *CBPM* finds the periodicity for all patterns in continuous or subsection of a time series even in the presence of noise. *CBPM* can find singular events if exists in time series. *CBPM* performs better than *WARP* and *STNR* because *CBPM* 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 period, we fixed the time series length and symbol set size. *CBPM* performance is shown in Fig. 6 with varying period size from 5 to 100. *ParPer* [15] and *WARP* [9] get affected as the period size increased. Time performance of *CBPM*, *CONV* and *STNR* [26] remains same as it checks for all possible periods irrespective of the data set.



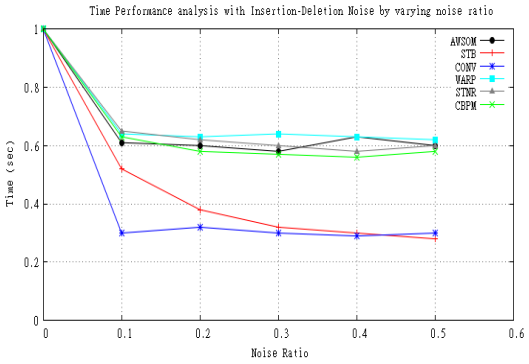
(a) Replacement Noise



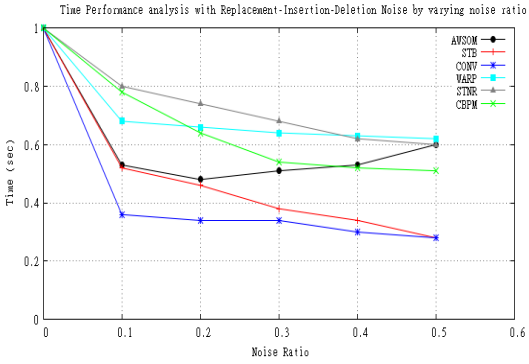
(b) Insertion noise



(c) Deletion Noise



(d) Insertion- Deletion Noise



(e) Replacement-Insertion-Deletion Noise

Figure 7: Time performance of CBPM compared with STNR, CONV, ParPer, WARP, AWSOM, STB.

D. Noise resilience

In the case of noise ratio, we used a synthetic time series of length 10,000 containing 4 symbols with embedded period size of 10. Symbols are uniformly distributed and the time series is generated in the same way as done in [8]. 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 period of 10 is detected. The time tolerance for all the experiments is ±2. Fig. 7 show that our algorithm compares well with *WARP* [9], *STNR*[26] and performs better than *AWSOM*[21], *CONV*[8], and *STB*[24]. For most of the combination of noise, the algorithm detects the period at the confidence higher than 0.5. The worst results are found with deletion noise, which disturbs the actual periodicity. *CBPM* shows consistent superiority because we consider asynchronous periodic occurrences which drift from the expected position within an allowable limit. This turns our algorithm a better choice in detecting different types of periodicity.

VII. Conclusions

In this paper, we have presented a novel algorithm that uses FP tree as underlying structure. The algorithm can detect symbol, sequence and segment periodicity as well as present the patterns that are periodic. It can also find periodicity within a subsection of the time series. It can detect the redundant period which are pruned; before calculating confidence which in turn saves a significant amount of time. We tested the algorithm on both real and synthetic data in order to test its accuracy, effectiveness of reported results, and the noise resilience characteristics. Our algorithm runs in *O (k. N)* in the worst case. In future, we are trying to extend our algorithm's working on online periodicity detection. The algorithm to be experimented with streaming data using disk based tree [26]. There are disk based implementations of the suffix tree construction [5], [26], which might be used to devise an online algorithm that can detect periodicity in very large time series database.

References

[1] Agrawal, R. and Srikant, R., Mining Sequential Patterns. In: Proceedings of 11th IEEE Int'l Conf. Data Eng. (ICDE). 1995. p. 3-14,

[2] Berberidis, C., Aref, W., Atallah, M., Vlahavas, I. and Elmagarmid, A., Multiple and Partial Periodicity Mining in Time Series Databases. In Proceedings of European Conf. Artificial Intelligence, July 2002.

[3] Blake, C.L. and Merz, C.J., UCI Repository of Machine Learning Databases, University of California, Department of Information and Computer Science. 1998.

[4] Chen, L., Tamer Ozsu, M. and Oria, V., Robust and Fast Similarity Search for Moving Object Trajectories. In: Proceedings of ACM SIGMOD. 2005. p. 491-502

[5] Cheung, C.F., Yu, J.X., and Lu, H., Constructing Suffix Tree for Gigabyte Sequences with Megabyte Memory. IEEE Trans. Knowledge and Data Eng. January 2005. 17(1): p. 90-105,

[6] Das, M. and Dai, H.K., A Survey of DNA Motif Finding Algorithms. BMC Bioinformatics. 2007. 8: p. S21-S33

[7] Dubiner, M. et al., Faster Tree Pattern Matching. Journal of ACM. 1994. 14:205-213

[8] Elfeky, M.G., Aref, W.G. and Elmagarmid, A.K., Periodicity Detection in Time Series Databases. IEEE Trans. Knowledge and Data Eng. 2005. 17(7): p. 875-887

[9] Elfeky, M.G., Aref, W.G. and Elmagarmid, A.K., *WARP*: Time *WARP*ing for Periodicity Detection. In: Proceedings of Fifth IEEE Int'l Conf. Data Mining, November 2005

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

[11] Fu, A.W.C., Keogh, E.J., Lau, L.Y.H. and Ratanamahatana, C.A., Scaling and Time *WARP*ing in Time Series Querying. In: Proceedings of Int'l Conf. Very Large Data Bases (VLDB). 2005. p. 649-660

[12] Glynn, E.F., Chen, J. and Mushegian, A.R., Detecting Periodic Patterns in Unevenly Spaced Gene Expression Time Series Using Lomb-Scargle Periodograms. Bioinformatics, February 2006. 22(3): p. 310-316

[13] Han, J., Gong, W. and Yin, Y., Mining Segment-Wise Periodic Patterns in Time Related Databases. In: Proceedings of ACM Int'l Conf. Knowledge Discovery and Data Mining. 1998. p. 214-218

[14] Han, J., Lakshmanan, L.V.S. and Raymond, T.N., Constraint-Based Multidimensional Data Mining. IEEE Computer. 1999. 32(8): p.46- 50

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

[16] Huang, K.Y. and Chang, C.H., SMCA: A General Model for Mining Asynchronous Periodic Patterns in Temporal Databases. IEEE Trans. Knowledge and Data Eng. June 2005. 17(6): p. 774-785

[17] Indyk, P., Koudas, N. and Muthukrishnan, S., Identifying Representative Trends in Massive Time Series Data Sets Using Sketches. In: Proceedings of Int'l Conf. Very Large Data Bases, September 2000.

[18] Keogh, E., Lin, J. and Fu, A., HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In Proceedings of Fifth IEEE Int'l Conf. Data Mining. 2005. p. 226-233

[19] Kumar, N., Lolla, N., Keogh, E., Lonardi, S., Ratanamahatana, C.A. and Wei, L., Time-Series Bitmaps: A Practical Visualization Tool for Working with Large Time Series Databases. In: Proceedings of SIAM Int'l Conf. Data Mining. 2005. p. 531-535

[20] Ma, S. and Hellerstein, J., Mining Partially Periodic Event Patterns with Unknown Periods. In:Proceedings of 17th IEEE Int'l Conf. Data Eng. April 2001
[21] Papadimitriou, S., Brockwell, A. and Faloutsos, C., Adaptive, Hands Off-Stream Mining. In: Proceedings of Int'l Conf. Very Large Data Bases (VLDB). 2003. p. 560-571
[22] Ptitsyn, A.A., Zvonic, S. and Gimble, J.M., Permutation test for periodicity in short time series data. BMC Bioinformatics (BMCBI). 2006. 7:(S-2)
[23] Ptitsyn, A.A., Zvonic, S. and Gimble, J.M., Permutation test for periodicity in short time series data. BMC Bioinformatics (BMCBI). 2007. vol 8
[24] Rasheed, F. and Alhajj, R., Using Suffix Trees for Periodicity Detection in Time Series Databases. In: Proceedings of IEEE Int'l Conf. Intelligent Systems, September 2008
[25] Rasheed, F. and Alhajj, R., *STNR*: A Suffix Tree Based Noise Resilient Algorithm for Periodicity Detection in Time Series Databases. Applied Intelligence. 2010. 32(3): 267-278
[26] Rasheed, F., Al-Shalalfa, M. and Alhajj, R., Efficient Periodicity Mining in Time Series Databases Using Suffix Trees. IEEE Trans. Knowl. Data Eng. (TKDE). 2011. 23(1):79-94
[27] Sandve, G.K. and Drablos, F., A Survey of Motif Discovery Methods in an Integrated Framework. Biology Direct. 2006. 1: p. 11-26
[28] Sau Dan Lee and Luc De Raedt, Constraint Based Mining of First Order Sequences in SeqLog. Database Support for Data Mining Applications. 2004. p. 154-173
[29] Sheng, C., Hsu, W. and Lee, M.L., Efficient Mining of Dense Periodic Patterns in Time Series. Technical report, Nat'l Univ. of Singapore. 2005.
[30] Sheng, C., Hsu, W. and Lee, M.L., Mining Dense Periodic Patterns in Time Series Data. In: Proceedings of 22nd IEEE Int'l Conf. Data Eng. 2005. p. 115
[31] Udechukwu, A., Barker, K. and Alhajj, R., Discovering all Frequent Trends in Time Series. In: Proceedings of Winter Int'l Symp. Information and Comm. Technologies. 2004. 58: p. 1-6
[32] Vlachos, M., Kollios, G. and Gunopulos, D., Discovering Similar Multidimensional Trajectories. In: Proceedings of 18th IEEE Int'l Conf. Data Eng. (ICDE). 2002. p. 673-684
[33] Wang, J. and Han, J., BIDE: Efficient Mining of Frequent Closed Sequences. In: Proceedings of 20th IEEE Int'l Conf. Data Eng. (ICDE). 2004. p. 79-90
[34] Wang, W. and Yang, J., Mining Sequential Patterns from Large Datasets. Springer-Verlag. 2005. vol 28
[35] Weigend, A. and Gershenfeld, N., Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley. 1994.
[36] Yan, X., Han, J. and Afshar, R., CloSpan: Mining Closed Sequential Patterns in Large Datasets. In: Proceedings of SIAM Int'l Conf. Data Mining (SDM). 2003
[37] Yang, J., Wang, W. and Yu, P., InfoMiner: Mining Partial Periodic Patterns with Gap Penalties. In: Proceedings of Second IEEE Int'l Conf. Data Mining. December 2002
[38] Zaki, M.J., SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning. 2001. 42(1): 31-60
[39] Zhu, Y. and Shasha, D., *WARP*ing Indexes with Envelope Transforms for Query by Humming. In: Proceedings of ACM SIGMOD. 2003. p. 181-192

G.M. Karthik, Born in Madurai, Tamil Nadu state in India, in 1981, received the B.E. in Computer Science and Engineering from SACS MAVMM Engineering College, Madurai, M.E. in Computer Science and Engineering from PSNA College of Engineering and Technology, Dindugal, in 2003 and 2005 respectively. He is having 8 years of teaching experience in more than two engineering colleges in India. This paper was written while he was working on the project on Data Mining techniques for real time issues as a Research scholar at Anna University of Technology, Coimbatore, India. His primary research interests are related to Data Mining and Web Mining. Currently, he is working as Assistant Professor of Computer Science Engineering Department of SACS MAVMM Engineering College, Madurai, India.

Dr.Ramachandra.V.Pujeri, Born in Bijapur, Karnataka state in India, in 1973, received the B E in Electronics and Communication Engineering from Karnataka University, Dharwad, ME in Computer Science and Engg from PSG College of Technology, Coimbatore, Ph.D in Information and Communication Engineering from Anna University, Chennai, MBA in Human Resource Management, from Pondicherry University, Pondicherry, in 1996, 2002, 2007 and 2008 respectively. He is active life member of ISTE, SSI, MIE, ACS and IEE. His has written three textbooks. He is having around 18 years of teaching experience in the various top ten engineering colleges in India. He is an active expert committee member of AICTE, NBA, DoEACC, NACC and various Universities in India. Currently, under him ten research scholars pursuing their Ph.D. His research interests lie in the areas of Computer Networking, Operating System, Software Engineering, Software Reliability, Modeling and Simulation, Quality of Services and Data Mining. Currently, he is working as Vice-Principal of KGiSL Institute of Technology, Coimbatore.

