Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 1, January 2017, pg.01 – 10

NOISE RESILIENT PERIODICITY MINING USING SUFFIX TREES

G. Surya Kala Eswari[1], P. Surya Prabhakar Rao[2], N.V.S. Sowjanya[3]

[1]IT & Jawaharlal Nehru Technological University Kakinada, India
[2]IT & Jawaharlal Nehru Technological University Kakinada, India
[3]IT & Jawaharlal Nehru Technological University Kakinada, India
[1] padmakala1@gmail.com; [2] suryaprabhakar.p@gmail.com; [3] sowjanya828@gmail.com

Abstract*— *One important aspect of the data mining is found out the interesting periodic patterns from time series databases. One area of research in time series databases is periodicity detection. Time series forecasting is the use of a model to predict future values based on previously observed values. Discovering periodic patterns from an entire time series has proven to be inadequate in applications where the periodic patterns occur only with in small segments of the time series. In this paper we proposed an algorithm to extract interesting periodicities (Symbol, Sequence, and Segment) of whole time series or subsection of the time series. This algorithm uses suffix tree as a data structure. The algorithm is noise resilient. It has been successfully demonstrated to work with replacement, Insertion, deletion, or a mixture of these types of noise. The worst case complexity of this algorithm is O (k.n^2) where k is the maximum length of the periodic pattern. n is the length of the analyzed portion (whole or subsection) of the time series. The algorithm is compared with DTW algorithm on different types of data size and periodicity on both real and synthetic data. The algorithm is time efficient and noise resilient than DTW algorithm.

Keywords*— *Time Series, suffix tree, periodicity detection, noise resilient, DTW

I. INTRODUCTION

A time-series database consists of sequences of values or events obtained over repeated measurements of time. The values are typically measured at equal time intervals (e.g., hourly, daily, weekly). Time-series databases are popular in many applications, such as stock market analysis, economic and sales forecasting, budgetary analysis, utility studies, inventory studies, yield projections, workload projections, process and quality control, observation of natural phenomena (such as atmosphere, temperature, wind, earthquake), scientific and engineering experiments, and medical treatments. Time series forecasting is the use of a model to predict future values based on previously observed values. Identifying periodic patterns could reveal that important observation s about the behavior and future trends of the case represented by the time series and hence would lead to more effective decision making. In general, three types of periodic patterns can be detected in a time series: 1) symbol periodicity, 2) sequence periodicity or partial periodic patterns, and 3) segment or full-cycle periodicity. A time series is said to have symbol periodicity if at least one symbol

is repeated periodically. For example, in time series periodicity p =3, starting at position zero (stPos= 0). Similarly, a pattern consisting of more than one symbol maybe periodic in a time series; and this leads to partial periodic patterns. For instance, in time series T =bbaa abbd abca abbc abcd, the sequence ab is periodic with periodicity p =4, starting at position 4 (stPos= 4); and the partial periodic pattern ab * * exists in T, where * denotes any symbol or don't care mark. Finally, if the whole time series can be mostly represented as a repetition of a pattern or segment, then this type of periodicity is called segment or full-cycle periodicity. For instance, the time series T = abcab abcab abcab has segment periodicity of (p =5) starting at the first position (stPos =0), i.e., T consists of only three occurrences of the segment abcab. It is not necessary to always have perfect periodicity in a time series as in the above three examples. Usually, the degree of perfection is represented by confidence, which is 100 percent in the above three examples. On the other hand, real-life examples are mostly characterized by the presence of noise in the data and this negatively affects the confidence level. The confidence of a pattern is defined as the ratio of its actual frequency in the series over its expected perfect frequency in the series. The actual and expected perfect frequency are both the same in the above three examples; however, in the time series T_x =abefd abcde acbcd abefa, the pattern ab starting at position 0 with p= 5 has four and five as its actual and expected perfect frequencies, respectively; so the confidence of ab is 4.In addition to detecting periodicity in the full time series, seeking periodicity only in a subsection of the series is also possible. For instance, in time series T=bcdbabababababccdaccab, the sequence ab is periodic starting from positions 5 to 12. This type of periodicity is very common in real life. For example, there might be a specific periodic pattern of road traffic during school days, but not during summer or winter vacation. Unfortunately most of the existing algorithms only detect periods that span through the Entire time series. There are many applications involving.

Sequence data. Typical examples include customer shopping sequences, Web click streams, biological sequences, sequences of events in science and engineering, and in natural and social developments.

Contributions of our work can be summarized as follows:

1. The development of DTW (Dynamic Time Warping)algorithm that can detect only segment periodicity.
2. The development of suffix tree based comprehensive algorithm that can simultaneously detect symbol, sequence and segment periodicity.
3. Finding periodicity within subsection of the series.
4. Identifying and reporting only useful and no redundant periods by applying pruning techniques to eliminate redundant periods, if any.
5. Detailed algorithm analysis for time performance and space consumption by considering three cases, namely the worst case, the average case, and the best case.
6. The proposed algorithm is shown to be applicable to biological data sets such as DNA and protein sequences and the results are compared with those produced by other existing algorithms like DTW.
7. Various experiments have been conducted to demonstrate the time efficiency, robustness, scalability, and accuracy of the reported results by comparing the proposed algorithm with existing state-of-the-art algorithms like WARP.

The rest of the paper is organized as follows: Related work is presented in Section 2. The periodicity detection problem in time series is formulated in Section 3. The proposed algorithm is presented in Section 4 along with algorithm analysis and the utilized optimization strategies. Experimental results are reported in Section 5 using both real and synthetic data. Section 6 is conclusions and future research directions.

II. RELATED WORK

Existing literature on time series algorithms requires user has to specify the period value or find out the candidate period value. Finding the threshold value requires many database scans. Another way to classify existing algorithms is based on the type of periodicity they detect. Some detect symbol periodicity, some detect segment periodicity, and some detect only sequence periodicity.

Recently there are few algorithms which look for all possible periods by considering the range ($2 \leq p \leq n \div 2$).

DTW algorithm detects only segment periodicity. It also works well in the presence of insertion and deletion of noise. Our single algorithm detects all the three types of periodicity. It requires only single scan of the data base. Recently, Huang and Chang [13] presented their algorithm for finding asynchronous periodic patterns, where the periodic occurrences can be shifted in an allowable range along the time axis. This is somehow similar to how our algorithm deals with noisy data by utilizing the time tolerance window for periodic occurrences.

III. PROBLEM DEFINITION

Consider a time series T= e0; e1; e2; . . . ; en_1 of length n, where ei denotes the event recorded at time i; and let T be discretized into symbols taken from an alphabet set with enough symbols, i.e., || represents the total number of unique symbols used in the discretization process. In other words, in a systematic way T can be encoded as a string derived from P. For instance, the string abbcabcdcbab could represent one time series over the alphabet={a,b,c,d}.Researchers have mostly investigated time series to identify repeating patterns and some researchers studied exceptional patterns (outliers) in time series. In this paper, we concentrate on the first case; we need to develop an algorithm capable of detecting in an encoded time series (even in the presence of noise) symbol, sequence, and segment periodicity, which are formally defined next. We start by defining confidence because it is not always possible to achieve perfect periodicity and hence we need to specify the degree of confidence in the reported result.

Definition 1 (Perfect Periodicity). Consider a time series T, a pattern X is said to satisfy perfect periodicity in T with period p if starting from the first occurrence of X until the end of T every next occurrence of X exists p positions away from the current occurrence of X. It is possible to have some of the expected

Occurrences of X missing and this leads to imperfect periodicity.

Definition 2 (Confidence). The confidence of a periodic pattern X occurring in time series T is the ratio of its actual periodicity to its expected perfect periodicity. Formally, the confidence of pattern X with periodicity p starting at position stPos is defined as:

$$Conf(p, stPos, X) = \frac{\text{Actual Periodicity}(P, Stpos, X)}{\text{Perfect Periodicity}(P, stpos, X.)}$$

Where Perfect Periodicity(p,stops,X)=|T|-stPos+1

Actual Periodicity (p,stops,X) is computed by counting (Starting at stP os and repeatedly jumping by p positions) the number of occurrences of X in T.

For example, in T = abbcaabcdbaccdbabbca, the pattern ab is periodic with stP os = 0, p= 5, and conf(5; 0; ab) =3.Note that the confidence is 1 when perfect periodicity is achieved.

Definition 3 (Symbol Periodicity). A time series T is said t have symbol periodicity for a given symbol s with period p and starting position stP os if the periodicity of s in T is either perfect or imperfect with high confidence.

Definition 4 (Sequence Periodicity). A time series T is said to have sequence periodicity (or partial periodicity) for a pattern (partial periodic pattern) X starting at position stP os, if $|X| \geq 1$ and the periodicity of X in T is either perfect

or imperfect with high confidence, i.e., X occur in T at most of the positions specified by stops+i*p. where p is period and i≥0.

For example, in T= abbcaabcdbabcdbabbca, the
pattern ab is partially periodic starting from stops
=0 and period value of 5.

Definition 5 (Segment Periodicity). A time series T of length n is said to have segment periodicity for a pattern X with period p and starting position stP os if p = |X| and the periodicity of X in T is either perfect or imperfect with high confidence, i.e., X occurs in T at most of the positions specified by stP os þ+i* p, where integer i ≥ 0 takes consecutive values starting at 0.

Definition 6 (Periodicity in Subsection of aTime Series). A time series T possesses symbol, sequence, or segment periodicity with period p between positions stP os and endPos (where 0≤ stP os < endPos ≤|T|), if the investigated period satisfies Definition 3, Definition 4, or Definition 5, respectively, by considering only subsection [stP os,endPos] of T.

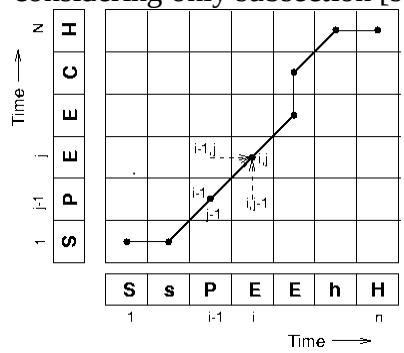

Fig.1 Alignment between the test and training pattern

For example, in T= abdabcaabcdbaccdbabbca, the pattern ab is periodic with p =5 in the subsection [3, 19] of T.

Definitions 3, 4, 5, and 6 will not be applicable in case the time series contains some insertion or deletion noise.

Unfortunately, it is not possible to avoid noise in real-life applications. Thus, to be able to tackle noisy time series, we introduce the concept of time tolerance, denoted tt, where a periodic occurrence is counted valid if it is found within ±tt positions of the expected position.

Definition 7 (Periodicity with Time Tolerance). Given a time series T which is not necessarily noise-free, a pattern X is periodic in subsection [stP os,endPos] of T with period p and time tolerance tt≥ 0 if X is found at positions stP os.stP os+ p± tt; stP os + 2p± tt; . . . endPos -p± tt.

For example, in time series T = abce dabc cabc aabcbabc c, freq(ab, 4, 0, 18, tt= 1)= 5 and conf (ab, 4, 0, 18,1)= 5, while based on Definitions 4 and 2, conf(ab, 4, 0, 18)= 0.2 and conf(ab, 4, 5, 18)= 1.

IV. PERIODICITY DETECTION ALGORITHMS

Dynamic Time Warping (DTW) is a pattern matching technique used to compare time series, not necessarily of the same length, based on their characteristic shapes. It is common practice to constrain the warping path in a global sense by limiting how far it may stray from the diagonal of the warping matrix. The matrix subset that the warping path is allowed to visit is generally called warping Window or band.

Noise resilient algorithm involves two phases. First build the suffix tree for the time series and in the second phase, we use the suffix tree to calculate the periodicity of various

patterns in the time series. One important aspect of our algorithm is redundant period pruning.

A. DTW Representation

Algorithm 1 calculates an optimal warping path between two time series. The algorithm starts with local distances Calculation between the elements of the two sequences using different types of distances. The most frequent used method for distance calculation is the absolute distance between the values of the two elements (Euclidian distance). DTW Algorithm is very useful for isolated words recognition in a limited dictionary. it is currently used in many areas: handwriting and online signature matching , sign language recognition and gestures recognition, data mining and time series clustering (time series databases search)

Algorithm 1: Dynamic Time Warping Algorithm
Input: Two sequences of time series.
Output: Warping path of two sequences.

```
1:path[]   new array
2: i = rows(dtw)
3: j = columns(dtw)
4: while (i > 1) & (j > 1) do
5: if i == 1 then
6: j = j - 1
7: else if j == 1 then
8: i = i - 1
9: else
10: if dtw(i-1; j) == min fdtw(i - 1; j); dtw(i; j- 1);
        dtw(i - 1; j- 1)    then
11: i = i - 1
12: else if dtw(i; j-1) == min{dtw(i - 1; j), dtw(i, j -
            1), dtw(i – 1, j- 1) then
13: j = j - 1
14: else
15: i = i - 1; j = j - 1
16: end if
17: path:add((i; j))
18: end if
19: end while
20: return path
```

The Dynamic Programming part of DTW algorithm uses the DTW distance function

$$DTW(X,Y) = cp^*(X,Y) = \min\{cp(X, Y), p \in P^{N*M}$$

where P^{N*M} is the set of all possible warping paths and builds the accumulated cost matrix or global cost matrix D which defined as follows:

1. First row: $D(1,j) = k=1jc(x1,yk)\ j \in [1,M]$.
2. First column: $D(I, 1) = k=1ic(xk,y1)\ i \in [1,N]$.
3. All other elements: $D(i, j) = \min \{D(i- 1, j- 1), D(i – 1, j), D(i, j- 1)+c(xi, yj), i \in [1,N], j \in [1,M]\}$.

To reduce the computational cost associated to the mining task, we consider the lower bounding technique introduced by Keogh et al (LB_Keogh). This technique aims at reducing the required number of DTW by the adoption of a lower-bounding measure.

B. Suffix Tree Based Representation

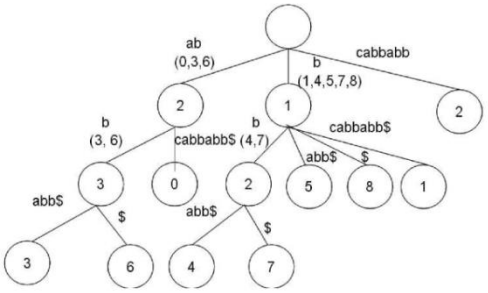

Fig.2 The suffix tree for the string abcabbabb$.

occur_vec	diff_vec
0	3
3	9
12	4
16	5
21	3
24	3
27	11
38	7
45	3
48	

Fig.3 An Example Occurrence Vector and Its Corresponding Difference Vector

Suffix tree (also called PAT tree) is a data structure that presents the suffixes of a given string in a way that allows for a particularly fast implementation of in string processing . Suffix tree is a famous data structure [9] that has been proven to be very useful in string processing [9], [11], [15], [19]. It can be efficiently used to find a substring in the original string, to find the frequent substring g and other string matching problems. A suffix 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 suffix tree. Given that a time series is encoded as a string, the most important aspect of the suffix tree, related to our work, is its capability to very efficiently capture and highlight the repetitions of substrings within a string. Fig. 1shows a suffix tree for the string abcabbabb$, where $ denotes end marker for the string; it is a unique symbol that does not appear anywhere in the string. The path from the root to any leaf represents a suffix for the string. Since a string of length n can have exactly n suffixes, the suffix tree for a string also contains exactly n leaves. Each edge is labelled 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 edge), which is repeated at least twice in the original string. There are algorithms for handling the disk based suffix tree. Suffix tree can be extended as new symbols are added to the string. Depth of the suffix tree is of order log(n).

C. Periodicity Detection in Presence of Noise

Three types of noise generally considered in time series data are replacement, insertion, and deletion noise. In replacement noise, some symbols in the discretized time series are replaced at random with other symbols. In case of insertion and deletion n noise, some symbols are inserted or deleted, respectively, randomly at different positions (or time values). Noise can

also be a mixture of these three types; for instance, RI type noise means the uniform mixture of replacement (R) and insertion (I) noise.. This is because insertion and deletion noise expand or contract the time axis leading to shift of the original time series values. For example, time series T = abcabcabc after inserting symbol b at positions 2 and 6 would be T= abbcabcabbc. The occurrence vector for symbol a in T is(0, 3, 6), while it is (0, 4, 7) in T0. It is very clear that when the time series is distorted by insertion and/ or deletion noise, linear-distance-based algorithms do not perform well. Actually, this is not only the case with linear distance- based algorithms, it is also true for periodicity detection algorithms in general [7].In order to deal with this problem, we introduce the concept of time tolerance (see Definition 7) into the periodicity detection process. The idea is that periodic occurrences can be drifted (shifted ahead or back) within a specified limit called time tolerance (denoted as tt in the algorithm). This means that if the periodic occurrence is expected to be found at positions x, x + p, x +2p, . . . , then with time tolerance, occurrences at x, x + p± tt; x + 2p ±tt; . . . would also be considered valid. For example, in the occurrence vector for pattern X = ab is(0,5,11, 17, 21).

T = abcde abdcec abcdca abdd abbca
01234 567890 123456 7890 12345;

According to Algorithm 1, conf(ab, p = 5, 0, 21)= 2/5. But, Conf (ab, 5, 0, 21, tt = 1)= 5/5. By considering time tolerance of 1. The occurrences at positions 11 and 17 are counted because they are +1 position away from the expected positions 10 and 16, respectively, and the occurrence at position 21 is counted because it is -1 position away from the expected position 22. It is important to note that our algorithm maintains the moving reference, i.e., when the occurrence at position 11 is counted with p= 5, the next expected occurrences are taken as 16+ 5* i. This is the reason why the occurrence at position 17 is counted because it is +1 position away from 16 and the next reference is set at position 17; this propagates into the next positions to be checked, i.e., the next expected occurrences are checked at positions 22 + 5* i. The modified algorithm with time tolerance taken into consideration is presented in Algorithm 2

Algorithm 2 . Noise Resilient Periodicity Detection

Algorithm

Input: a time series of size n and time tolerance value tt.

Output: positions of periodic patterns;

1. for each occurrence vector occur_vec of size k for pattern X, repeat
1.1 for j = 0;j < n; j++;
1.1.1 p= occur vec[j+ 1]- occur vec[j]
1.1.2 StPos =occur vec[j],endPos= occur vec[k];
1.1.3 for i = j;i < k; i++;
1.1.3.1 A = occur vec[i]- currStP os;
1.1.3.2 B= Round(A÷p);
1.1.3.3 C= A –(p * B);
1.1.3.4 if ((-tt ≤ C≤+tt) AND
(Round((preOccur –currSTPos)p) ≠B))
1.1.3.4.1 currStPos =occur vec[i];
1.1.3.4.2 preOccur = occur vec[i];
1.1.3.4.3 increment count(p);
1.1.3.4.4 sumPer+=(p + C);
1.1.4 endfor
1.1.5 mean p = sumPer- p/ (count(p)-1) ;
1.1.6 conf(p)= count(p)/Perfect Periodicity (p,stops,X) ;

End Algorithm

Algorithm 2 contains three new variables, namely A, B, and C. A represents the distance between the current occurrence and the current reference starting position;

B represents the number of periodic values that must be passed from the current reference starting position to reach the current occurrence. Variable C represents the distance between the current occurrence and the expected occurrence. For instance, assume the occurrence vector of the last example, namely (0, 5, 11, 17, 21) is modified into(0, 5, 16,22, 26). For this, when the current occurrence (occur vec[i]) is 16, A = 16-_ 5 = 11, B = 11 /5 = 2, C = 11- 5* 2= 1. The second condition at line 1.1.3.4 of Algorithm 2 is also very interesting. It checks to see if the current occurrence is the repetition of an already counted periodic value. Consider how the occurrence vector (0, 5, 15, 16, 22, 26) would have been affected if this condition was not included in Algorithm 2. Count(p = 5, stops= 0) would be 6 as the occurrences at positions 15 and 16 are within _1 range of the expected position 15; so they would be counted as valid occurrences; this scenario is avoided by the condition in line 1.1.3.4 of Algorithm 2. Finally, Algorithm 2 calculates the average period value as in the presence of insertion and/or deletion noise, the first difference or the candidate period might be misleading. Suppose the occurrence vector is (0, 5, 11, 17, 23, 29, 35, 42) based on Algorithm 2, the period value would be p = 5, but the average period value is P = 5.85 ~ 6, which reflects more realistic information.

V. RESULTS

A. Time Performance

This section reports the results of the experiments that measure the time performance of STNR compared to WARP. We test the time behavior of the compared algorithms by considering two perspectives of varying data size. Data distribution.

First, we compare the performance of STNR against

The WARP. Which finds the periodic patterns for a specific period with uniform distribution containing 10 unique symbols and embedded period size of 25 and 32. The size of the series has been varied from 100,000 to 100,000,000 (100 millions). The results of STNR are compared with those reported by WARP [7] the curves are plotted in Fig. 3, where it can be seen that STNR performs better than WARP. Second compare the performance of STNR against the WARP with different data distribution. The complexity of WARP is $O(n^2)$, STNR performs better than WARP because STNR applies various optimization strategies, most notably the adopted redundant period pruning techniques. From Fig.5 and fig. 6 shows warp doesn't depend on the data distribution. Whereas STNR depends on data distribution. WARP performs well in the small time series. But STNR performs well large time series with insertion, deletion and replacement of noise than WARP.

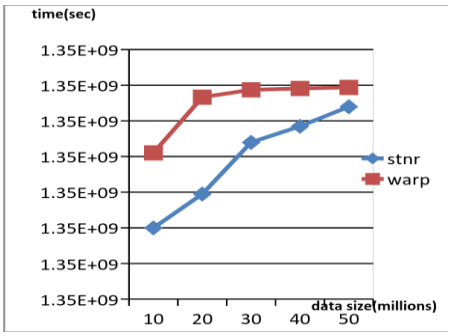

Fig.4 Time performance STNR compared with WARP

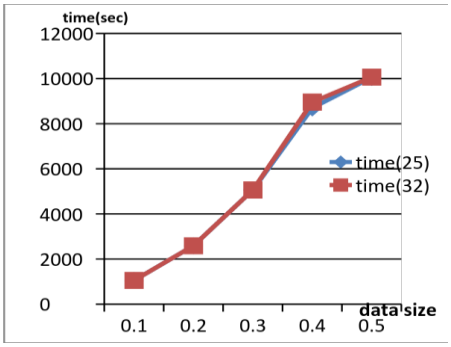

Fig.5 Time performance of different data distribution

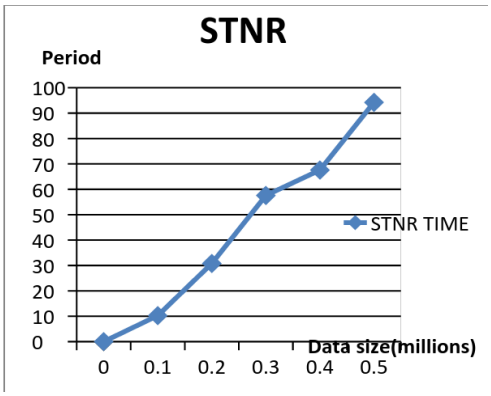

Fig.6 WARP under uniform distribution

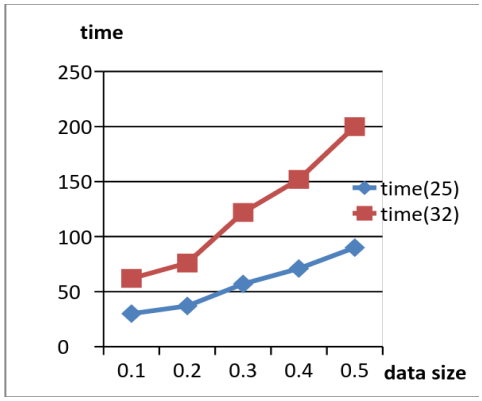

Fig.7 STNR under uniform distribution

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented DTW Algorithm. Using Fourier transform, auto correlation functions to reduce the time complexity of the distance functions. and STNR algorithm is a suffix-tree-based algorithm for periodicity detection in time series data. Our algorithm is noise-resilient and run in $O(k.n^2)$ in the worst case. The single algorithm can find symbol, sequence (partial periodic), and segment (full cycle) periodicity in the time series. It can also find the periodicity within a subsection of the time series. We performed several experiments to show the time behavior, accuracy, and noise resilience characteristics of the data. We run the algorithm on both real and synthetic data. The reported results demonstrated the power of the employed pruning strategies. Currently, we are working on online periodicity detection where the suffix tree is constructed online, i.e., extended while the algorithm is in operation. This type of stream mining has a large number of applications especially in sensor networks, ubiquitous computing, and DNA sequence mining. We are also implementing the disk-based solution of the suffix tree to extend capabilities beyond the virtual memory into the available disk space. Using Fourier transform to reduce the complexity and wavelet transform is used to solve the problem in linear time.

REFERENCES

[1] M. Ahdesma¨ki, H. La¨hdesma¨ki, R. Pearson, H. Huttunen, and O. Yli-Harja, *"Robust Detection of eriodic Time Series Measured from Biological Systems,"* BMC Bioinformatics, vol. 6, no. 117, 2005.

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

[3] H. Brown et al., *"Sequence Variation in S-Antigen Genes of Plasmodium falciparum,"* Molecular Biology and Medicine, vol. 4, no. 6, pp. 365-376, Dec. 1987.

[4] C.-F. Cheung, J.X. Yu, and H. Lu, "*Constructing Suffix Tree for Gigabyte Sequences with Megabyte Memory,"* IEEE Trans. Knowledge and Data Eng., vol. 17, no. 1, pp. 90-105, Jan. 2005.

[5] M. Dubiner et al., *"Faster Tree Pattern Matching,"* J. ACM, vol. 14, pp. 205-213, 1994.

[6] M.G. Elfeky, W.G. Aref, and A.K. Elmagarmid, *"Periodicity Detection in Time Series Databases,"* IEEE Trans. Knowledge and Data Eng., vol. 17, no. 7, pp. 875-887, July 2005.

[7] M.G. Elfeky, W.G. Aref, and A.K. Elmagarmid, *"WARP: Time Warping for Periodicity Detection,"* Proc. Fifth IEEE Int'l Conf. Data Mining, Nov. 2005.

[8] J. Fayolle and M.D. Ward, *"Analysis of the Average Depth in a Suffix Tree under a Markov Model,"* Proc. Int'l Conf. Analysis of Algorithms, pp. 95-104, 2005.

[9] D. Gusfield, *Algorithms on Strings, Trees, and Sequences.* Cambridge Univ. Press, 1997.

[10] E.F. Glynn, J. Chen, and A.R. Mushegian, *"Detecting Periodic Patterns in Unevenly Spaced Gene Expression Time Series Using Lomb-Scargle Periodograms,"* Bioinformatics, vol. 22, no. 3 pp. 310-316, Feb. 2006.
[11] R. Grossi and G.F. Italiano, *"Suffix Trees and Their Applications in String Algorithms,"* Proc. South Am. Workshop String Processing, pp. 57-76, Sept. 1993.
[12] J. Han, Y. Yin, and G. Dong, *"Efficient Mining of Partial Periodic Patterns in Time Series Database,"* Proc. 15th IEEE Int'l Conf. Data Eng., p. 106, 1999.
[13] K.-Y. Huang and C.-H. Chang, *"SMCA: A General Model for Mining Asynchronous Periodic Patterns in Temporal Databases,"* IEEE Trans. Knowledge and Data Eng., vol. 17, no. 6, pp. 774-785, June 2005.
[14] J. Han, W. Gong, and Y. Yin, *"Mining Segment-Wise Periodic Patterns in Time Related Databases,"* Proc. ACM Int'l Conf. Knowledge Discovery and Data Mining, pp. 214-218, 1998.
[15] E. Hunt, R.W. Irving, and M.P. Atkinson, *"Persistent Suffix Trees and Suffix Binary Search Trees as DNA Sequence Indexes,"* Technical Report TR2000-63, Univ. of Glasgow, Dept. of Computting Science, 2000.
[16] P. Indyk, N. Koudas, and S. Muthukrishnan, *"Identifying Representative Trends in Massive Time Series Data Sets Using Sketches,"* Proc. Int'l Conf. Very Large Data Bases, Sept. 2000.
[17] 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, vol. 9, no. 6, pp. 1203-1209, 2000.
[18] E. Keogh, J. Lin, and A. Fu, *"HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence,"* Proc. Fifth IEEE Int'l Conf. Data Mining, pp. 226-233, 2005.
[19] R. Kolpakov and G. Kucherov, *"Finding Maximal Repetitions in a Word in Linear Time,"* Proc. Ann. Symp. Foundations of Computer Science, pp. 596-604, 1999.

