International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-1 Issue-5, June 2012

Efficient Method for De-Duplication and Periodicity Mining in Time Series Databases

S. Drishya, I. Nancy Jeba Jingle.

Abstract— ***Periodic pattern mining or periodicity detection has a number of applications, such as prediction, forecasting, detection of unusual activities, etc. The problem is not trivial because the data to be analyzed are mostly noisy and different periodicity types(namely symbol, sequence, and segment) are to be investigated. Noise is the duplication of data from different databases when they are used for same purpose in different places. So it should be removed. Time series is a collection of data values gathered generally at uniform interval of time to reflect certain behavior of an entity. Real life has several examples of time series such as weather conditions of a particular location, transactions in a superstore, network delays, power consumption, earthquake prediction. A time series is mostly characterized by being composed of repeating cycles. Identifying repeating (periodic) patterns could reveal important observations about the behavior and future trends of the case represented by the time series, and hence would lead to more effective decision making. The goal of analyzing a time series is to find whether and how frequent a periodic pattern (full or partial) is repeated within the series. There is a need for a comprehensive approach capable of analyzing the whole time series or in a subsection of it to effectively handle different types of noise (to a certain degree) and at the same time is able to detect different types of periodic patterns; combining these under one umbrella is by itself a challenge. In this paper, we present an algorithm which can detect symbol, sequence (partial), and segment (full cycle) periodicity in time series. The algorithm is noise resilient; it has been successfully demonstrated to work with replacement, insertion, deletion, or a mixture of these types of noise.***

Index Terms— Time series, periodicity detection, suffix tree, symbol periodicity, segment periodicity, sequence periodicity, noise resilient.

I.INTRODUCTION

Time series do exist frequently in our daily life and their analysis could lead to valuable discoveries. In other words, we argue the need to develop a noise-resilient algorithm that could tackle the problem well by being capable of:

a) Identifying the three different types of periodic patterns (Symbol, Sequence and Segment Periodicities).
b) Handling asynchronous periodicity by locating periodic patterns that may drift from their Expected positions up to an allowable limit.
c) Investigating periodic patterns in the whole time series as well as in a subsection of the time series.
d) At each step in the development process, we conducted comprehensive testing of the algorithm before new features are incorporated.

In addition to these features, the algorithm proposed in this paper can detect periodic patterns found in subsections of the time series, prune redundant periods, and follow various optimization strategies; it is also applicable to biological data sets and has been analyzed for time performance and space consumption.

Periodicity Mining is difficult because of noisy and different periodicity types. Noise is an important factor for the data become not suitable for pattern mining. Noise refers to the duplication of the data. We efficiently avoid the Noise from database which will not affect the nature of data. Noise may present in the database with the periodicity data. This won't matches with the database periodicity. This will affect the periodicity of the time series database. To predict the duplicates accurately we need to have a distributed environment. This will affect the entire behavior of the time series databases.

II. PROBLEM DEFINITION

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.

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.

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.

Manuscript published on 30 June 2012.
***** Correspondence Author (s)
S.Drishya*, Dept Of CSE,Vins Christian College Of Engineering,Nagercoil,India,+919489796919(e-mail: drishya17@gamil.com).
I.Nancy Jeba Jingle, Dept Of CSE,Vins Christian College of Engineering.,Nagercoil,India

Formally, the confidence of pattern X with periodicity p starting at position stPos is defined as:

$$\text{conf}(p; \text{tops}; X) = \frac{\text{Actual Periodicity}(p; \text{stPos}; X)}{\text{Perfect Periodicity}(p; \text{tops}; X)}$$

where Perfect Periodicity(p; tops, X)=(|T|-stpos+1)/p; and Actual Periodicity (p; stPos;X) is computed by counting (starting at stPos 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 stPos = 0, p = 5, and conf(5; 0; ab) = ¾. Note that the confidence is 1 when perfect periodicity is achieved.

III. PROJECT OVERVIEW

A time series is a collection of data values gathered generally at uniform interval of time to reflect certain behavior of an entity. Real life has several examples of time series such as weather conditions of a particular location, spending patterns, stock growth, transactions in a superstore, network delays, power consumption, computer network fault analysis and security breach detection, earthquake prediction, gene expression data analysis etc. A time series is mostly characterized by being composed of repeating cycles. For instance, there is a traffic jam twice a day when the schools are open; number of transactions in a superstore is high at certain periods during the day, certain days during the week, and so on. Identifying repeating (periodic) patterns could reveal important observations about the behavior and future trends of the case represented by the time series and hence would lead to more effective decision making. In other words, periodicity detection is a process for finding temporal regularities within the time series, and the goal of analyzing a time series is to find whether and how frequent a periodic pattern (full or partial) is repeated within the series.

To sum up, time series do exist frequently in our daily life and their analysis could lead to valuable discoveries. However, the problem is not trivial and none of the approaches described in the literature is comprehensive enough to handle all the related aspects. We present an efficient algorithm that uses suffix tree as the underlying data structure to detect all the above mentioned three types of periodicity in a single run. The algorithm is noise-resilient and works with replacement, insertion, deletion, or any mixture of these types of noise. It can also detect periodicity within a subsection of a time series and applies various redundant period pruning techniques to output a small number of useful periods by removing most of the redundant periods. Contributions of our work can be summarized as follows:

1) The development of suffix-tree-based comprehensive algorithm that can simultaneously detect symbol, sequence, and segment periodicity.
2) Finding periodicity within subsection of the series.
3) Identifying and reporting only useful and no redundant periods by applying pruning techniques to eliminate redundant periods, if any;
4) Detailed algorithm analysis for time performance and space consumption by considering three cases, namely the worst case, the average case, and the best case;

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 SMCA.

IV. NOISE REMOVAL

Periodicity Mining is difficult because of noisy and different periodicity types. Noise is an important factor for the data become not suitable for pattern mining. Noise refers to the duplication of the data. We efficiently avoids the Noise from database which will not affect the nature of data. Noise may present in the database with the periodicity data. This won't matches with the database periodicity. This will affect the periodicity of the time series database. To predict the duplicates accurately we need to have a distributed environment. This will affect the entire behavior of the time series databases. So this is important because exact periodicity prediction.

V.PATTERN MINING

Time series databases we have data is in the form of patterns. The patterns can be mined in order to predict the future data. For example, consider the time series containing the hourly number of transactions in a superstore; the discretization process may define the following mapping by considering different possible ranges of transactions; {0} transactions: a, {1-200} transactions: b, {201-400} transactions: c, {401-600} transactions: d, f >600g transactions: e. Based on this mapping, the time series T= 243; 267; 355; 511; 120; 0; 0; 197 can be discretized into T= cccdbaab.

Matching may be done in 3 types.

A. Symbol Periodicity : A time series is said to have symbol periodicity if at least one symbol is repeated periodically.
B. Sequence Periodicity: A time series is said to have sequence periodicity if more than one symbol is repeated periodically.
C. Segment Periodicity: A time series is said to have sequence periodicity if more than one periodic data is repeated periodically.

VI. SUFFIX TREE BASED REPRESENTATION

Suffix tree is a famous data structure that has been proven to be very useful in string processing. It can be efficiently used to find a substring in the original string, to find the frequent substring 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. 1 shows 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.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

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 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.

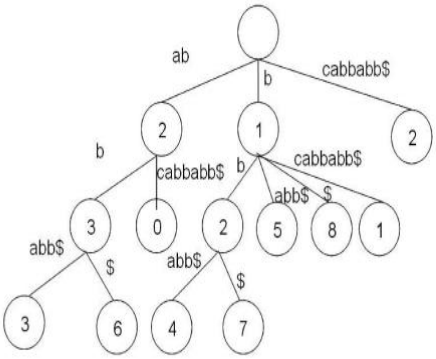

Fig. 1. The suffix tree for the string *abcabbabb$*.

Ukonen's algorithm works online, i.e., a suffix tree can be extended as new symbols are added to the string. A suffix tree for a string of length n can have at most 2n nodes .It is not necessary to always keep a suffix tree in memory. There are algorithms for handling disk-based suffix tree, making it a very preferred choice for processing very large sized strings such as time series and DNA sequences which grow in billions. Once the tree as presented in Fig. 1 is constructed, we traverse the tree in bottom-up order to construct what we call occurrence vector for each edge connecting an internal node to its parent. We start with nodes having only leaf nodes as children; each such node passes the values of its children (leaf nodes) to the edge connecting it to its parent node. The values are used by the latter edge to create its occurrence vector (denoted occur vec in the algorithm).

The occurrence vector of edge e contains index positions at which the substring from the root to edge e exist in the original string. Second, we consider each node v having a mixture of leaf and nonleaf nodes as children.

The occurrence vector of the edge connecting v to its parent node is constructed by combining the occurrence vector(s) of the edge(s) connecting v to its non leaf child node(s) and the value(s) coming from its leaf child node(s).

Finally, until we reach all direct children of the root, we recursively consider each node u having only non leaf children. The occurrence vector of the edge connecting u to its parent node is constructed by combining the occurrence vector(s) of the edge(s) connecting u to its child node(s).Applying this bottom-up traversal process on the suffix tree shown in Fig. 1 will produce the occurrence vectors reported in Fig. 2.

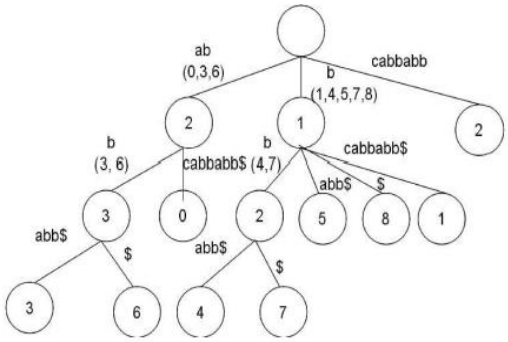

Fig. 2. Suffix tree for string *abcabbabb$* after bottom-up traversal.

The periodicity detection algorithm uses the occurrence vector of each intermediate edge (an edge that leads to a nonleaf node) to check whether the string represented by the edge is periodic.

The tree traversal process is implemented using the noN recursive explicit stack-based algorithm, which prevents the program from throwing the stack-overflow-exception.

VII. PERIODICITY MINING

From the generated suffix trees we are performing periodicity mining. By this method, we take the difference between any two successive occurrence vector elements leading another vector called the difference vector. For Mining periodicity we need to remove the noise from the data sequence. Here we need to remove the redundancy also. This supposed give the perfect periodic mining.

A. Perioditicity Detection

Our algorithm involves two phases. In the first phase, we 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, 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 period which has already been identified as redundant.

This saves considerable time and also results in reporting fewer but more useful periods. This is the primary reason why our algorithm, intentionally, reports significantly fewer number of periods without missing any existing periods during the pruning process.

B. *Periodicity Detection Algorithm*

We apply the periodicity detection algorithm at each intermediate edge using its occurrence vector.Our algorithm is lineardistance- based, where we take the difference between any two successive occurrence vector elements leading to another vector called the difference vector.It is important to understand that we actually do not keep any such vector in the memory but this is considered only for the sake of explanation.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

TABLE 1
An Example Occurrence Vector and Its Corresponding Difference Vector

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

Table1 presents example occurrence and difference vectors .Each value in the difference vector is a candidate period starting from the corresponding occurrence vector value (the value of occur vec in the same row).

Algorithm 1. Periodicity Detection Algorithm

Input: a time series of size n;

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/2 ; 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 if (stP os mod p = =occur_vec[i] mod p) increment

Count(p);

1.1.4 end for

1.1.5 conf(p)= count(p) / (Perfect Periodicity(p;stPos;X);

1.1.6 if (conf(p)<= threshold) add p to the period list;

1.2 end for

2 end for

EndAlgorithm

Recall that each period can be represented by 5-tuple (X; p; stPos;endPos; conf), denoting the pattern, period value, starting position, ending position, and confidence, respectively.

C. Periodicity Detection in Presence of Noise

We have already presented the noise-resilient features of the algorithm, but we briefly review them here for the sake of completeness. Three types of noise generally considered in time series data are replacement, insertion, and deletion noise.

a) In replacement noise, some symbols in the discretized time series are replaced at random with other symbols.
b) In case of insertion and deletion noise, some symbols are inserted or deleted, respectively, randomly at different positions (or time values).

TABLE 2
An Example Result of a Periodicity Detection Algorithm

Pattern(X)	Period(p)	Starting Position(stops)
ab	5	0
ab	10	0
ab	25	0
a	5	0
a	15	0
b	5	1
b	10	1
a	5	10
ab	5	20

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. ***Algorithm 1*** performs well when the time series is either perfectly periodic or contains only replacement noise and performs poorly in the presence of insertion or deletion 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.

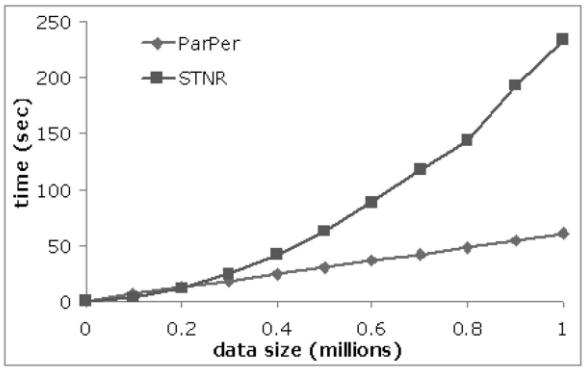

Fig 3 : Time performance of STNR compared with partial periodic patterns algorithm.

The occurrence vector for symbol a in T is (0; 3; 6), while it is (0; 4; 7) in T`. 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 lineardistance- based algorithms, it is also true for periodicity detection algorithms in general .In order to deal with this problem. we introduce the concept of time tolerance 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.

D. Redundant Period Pruning Techniques

Periodicity detection algorithms generally 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. Assume the periods reported by an algorithm are as presented in Table 2.

Looking carefully at this result, it can be easily seen that all these periods can be replaced by just one period, namely (X=ab; p= 5; stpos= 0); and all other periods maybe considered as just the mere repetition of period X.

Empowered by redundant period pruning, our algorithm not only saves the time of the users observing the produced results, but it also saves the time for computing the periodicity by the mining algorithm itself.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

TABLE 3
Periodic Patterns in Packet Data

Pattern	Period	StPos	EndPos	Confidence
aa	2	146698	155081	0.42
aa	2	180136	186061	0.42
aa	2	297362	304371	0.47
aaa	3	146772	155064	0.44
aaa	3	180897	186065	0.43
aaa	3	297525	304367	0.47
aaa*a	5	147030	155044	0.44
aaaa*	5	182390	186064	0.36
aaa**	5	297480	304324	0.44
****aa*	7	147203	155042	0.46
aaa	7	148841	155042	0.46
aa*****	7	149394	155070	0.46
a**aaaa	7	182091	186059	0.4
a*a*aa*	7	299362	304324	0.49

Table 3 presents some of the results produced by STNR when run on the Packet data.We implemented the redundant period pruning techniques as prohibitive steps which prohibit the algorithm from handling redundant periods.Some of the redundant period pruning approaches are outlined next.

I. If (X; p; stPos) exists then (X; k <p; stPos) would never be addressed, where k > 0 is an integer, e.g., if (a, 3, 0) exists then (a, 6, 0) and (a, 9, 0) are redundant.

II. If X $\subseteq$ Y , and (Y ; p; stpos) exists then (X; p; stpos) would never be calculated, e.g., if (abc; 5; 0) exists then (bc; 5; 0) and (bc; 5; 1) are redundant.

III. If X$\subseteq$ Y , and (Y ; p; stPos) exists then (X; kp; stPos) would never be calculated, where k > 0 is an integer, e.g., if (abc; 5; 0) exists then (ab*; 15; 0) and (ab; 15; 0) are redundant.

IV. If (X; p; stPos) exists then (X; p; k * stPos) would never be calculated, where k > 0 is an integer, e.g., if (a; 3; 0) exists then (a; 3; 9) and (a; 3; 27) are redundant.

VIII. CONCLUSION

I have implemented the concept of periodicity mining from the generated time series database. This will make the prediction process in the time series database to be more efficient. This database contains information about the data which can be duplicated for a particular time period. Periodicity detection is a process for finding temporal regularities within the time series, and the goal of analyzing a time series is to find whether and how frequent a periodic pattern (full or partial) is repeated within the series. Suffix tree formation is useful to find the repeated patterns. Existing work mainly focused on predicting future data based pattern only which is not adopted in all types of data. To avoid it I have implemented Suffix tree formation for predicting future data in Time Series Databases.

REFERENCES

1. M. Ahdesma¨ki, H. La¨hdesma¨ki, R. Pearson, H. Huttunen, and O. Yli-Harja, “Robust Detection of Periodic Time Series Measured from Biological Systems,” BMC Bioinformatics, vol. 6, no. 117,
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
3. 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.
4. Y. Tian, S. Tata, R.A. Hankins, and J.M. Patel, “Practical Methods for Constructing Suffix Trees,” VLDB J., vol. 14, no. 3, pp. 281-299, Sept. 2005.
5. 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.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)