International Journal of Soft Computing 9 (1): 37-43, 2014
ISSN: 1816-9503

Multiplex Tree Pruning for Periodic Pattern Mining

[1]B. Sujatha and [2]S. Chenthur Pandian
[1]Department of CSE, Sengunthar Engineering College, Tiruchengode, Tamil Nadu, India
[2]Dr. Mahalingam College of Engineering and Technology, Pollachi, Tamil Nadu, India

Abstract: Discovering specified patterns in a time series database has expected much consideration and is nowadays a comparatively mature field. Existing Periodic Pattern Mining algorithms concentrates on mining which involves subsequences. However, huge portion of requests for example, genetic DNA and protein pattern mining requires estimated patterns that are adjacent in nature. The existing algorithms applied to discover such estimated pattern mining comprises of complicated problems such as deprived scalability and complexity while applying towards certain other applications. To overcome these limitations, a novel technique is presented that evolves a set of periodic pattern if the regularity of the occurrence changes from that estimated pattern. The technique is based on the combination of both suffix and prefix tree patterns, to develop a multiplex tree pruning, for an activity normalized time periodicity data sequences. The integrative sequence of prefix and suffix trees is based on the threshold factor of predominant data pattern occurrence rate. The conceptual model of multiplex tree pruning technique presented in this study, in combination with the prefix and suffix tree model for pruning items identifies the regularity of all observed patterns in an efficient manner. The detailed experimental study shows strong gains in periodic pattern mining, ensure fast storage of all the time series for a specified item. Empirical studies with varied time series data obtained from bank and car data set using UCI repositories is measured and evaluated in terms of time efficiency of pruning patterns of interest, sensitivity and accuracy.

Key words: Data mining, sequential pattern mining, periodic pattern mining, multiplex tree pruning technique, suffix tree, prefix tree

INTRODUCTION

Time series data constrains the development of a data over a period of time. Life comprises of numerous instances for varied time series data. The instances comprises of meteorological data holding pertaining to various dimensions, e.g., warmth and moist, stock exchange prices portrayed in economic market, utilization of power accounted in energy corporations and log files checked in computer networks. Periodicity mining is a method which assists in forecasting the activities of time series data. For instance, periodicity mining permits power group to determine the power utilization patterns and extract the periods of high and low custom utilized data so that appropriation towards scheduling might take place in an efficient manner.

Several research works pertaining to time series data mining concentrate on determining different types of prototypes ranging from chronological patterns, sequential patterns, cyclic organization rules, limited periodic prototypes and astonishing patterns to name a few. These periodicity mining methods requires the user to identify a period that resembles a change in data at which the time series is irregular. They assume that users, moreover, identify the timeliness and worth of the data that are previously disposed to different time for acceptable periodic patterns. As the mining procedure has to be implemented frequently to obtain finite results this trial-and-error system is obviously not sufficient. Even with the introduction of time series data there might be incomprehensible periods and as a result, appealing irregular patterns will not be exposed. The solution to these problems is to design methods for determining possible periods in time series data.

Sometimes using primary methods do not result into support for mainstream files in spite of its significance as one would process from one state to another. The first stage comprises of construction of suffix trees processed so as to resolve the so-called substring crisis. Some of the properties of suffix tree are given as:

- SF1: it contains merely some number of leaves numbered from 1 to S
- SF2: instead of the root, all internal nodes contain at least two children

Corresponding Author: B. Sujatha, Department of CSE, Sengunthar Engineering College, Tiruchengode, Tamil Nadu, India

- SF3: every edge is processed with a non-empty substring of S
- SF4: no two edges initiating out of a node can encompass string-labels starting with the similar character
- SF5: the string received by adding up all the string-labels originated on the path from the root to leaf i processes out suffix

A digital tree or prefix tree consists of a prearranged tree data construction that is employed to aggregate an active set or associative collection where the key comprises of regularly string types. It is of not in the forma of a binary explore tree, no node in the tree consists of the key connected with that node whereas the location in the tree helps to identify the location where the key actually is connected. All the leaf node of a tree encompasses a widespread prefix of the string related to that node and in turn the root node is connected with the empty string. The values are not connected to the node directly whereas the leaves and some internal nodes communicate each other.

Periodicity mining is significantly used for identifying the preferences in time series data. Designing the time at which the time series is occurring has penetrated as an obstacle for absolutely predetermined periodicity mining. Time series data limits the improvement of data value in specified period of time. Periodicity mining is a masterpiece that supports in forecasting the measures of time series data. In this research, a novel technique termed multiplex tree pruning technique is presented for normalizing the time periodicity data sequences. The multiplex tree pruning technique combines suffix and prefix tree and the integrative viewpoint is processed based on the data pattern occurrence rate.

Literature review: In spite of the reality that in the precedent decade, researchers have practiced an abundance of time-series space measures and depictions, the greater part of them effort to differentiate the relationship among series based exclusively on shape. Nevertheless, it is fetching gradually more obvious that structural relationships can supply more instinctive series descriptions that stick on more strongly to human observation of comparison. Rasheed *et al.* (2011) presented an algorithm which can recognize symbol, sequence (fractional) and segment (full cycle) periodicity in time series. The algorithm uses suffix tree as the basic data organization this authorizes us to map the algorithm such that it's nearly all dreadful case complexity of the time series. The algorithm is clatter bendable; it has been efficiently recognized to try with deputy, calculation, crossing out or an arrangement of these sorts of noise.

Two sorts of periodicities are dissimilar and a scalable, computationally adept algorithm is considered for all types. Conventional sketch out growth-based techniques for in order pattern mining attain patterns sustained on the anticipated databases recursively. At all level of recursion, they unidirectional construct the period of perceived patterns by one all along the suffix of observed patterns which requests k phases of recursion to determine a prototype (Chen, 2010).

Sequential pattern mining, since its introduction has received considerable attention among the researchers with broad applications. The Sequential Pattern algorithms generally face problems when mining long sequential patterns or while using very low support threshold. One possible solution of such problems is by mining the closed sequential patterns which is a condensed representation of sequential patterns (Niranjan *et al*., 2010). Periodicity mining is used for predicting trends in time series data (Elfeky *et al*., 2005). Discovering the rate at which the time series is periodic has always been an obstacle for fully automated periodicity mining. Existing periodicity mining algorithms assume that the periodicity rate (or simply the period) is user-specified. The task of discovering frequent sequences is challenging because the algorithm needs to process a integrative explosive number of possible sequences (Vijayalakshmi *et al*., 2010).

Deciding closest Chronological Patterns (CSP) is an imperative trouble in web usage mining. Chen and Cook (2007) proposed a novel data structure, UpDown Tree for CSP mining. An UpDown Tree merges suffix tree and prefix tree for proficient storage space of all the series that hold a specified item. Existing Series Mining algorithms typically center on mining for subsequences. Floratou *et al.* (2011) presented a novel algorithm termed FLAME (Flexible and Accurate Motif Detector). FLAME is a flexible suffix tree supported algorithm that could be employed to discover recurrent patterns with a diversity of descriptions of motif (prototype) representations to legitimately exaggerate revision on iterative pattern mining, the researchers commence mining iterative originators, i.e., trifling patterns free of any sub-pattern surrounding the analogous support (Lee *et al*., 2011). Periodic pattern mining or periodicity detection has abundant requests for instance forecast, forecasting, recognition of abnormal events, etc. The episodic patterns are noticed in a time-series database depending on the time intervals (Lo *et al*., 2011).

The proposed approach by Rasheed and Alhajj (2010) suppose the periodicity of replacement substrings, besides permitting for vigorous gap to perceive the periodicity of certain designs of substrings. However, even free of nested prototypes, the argot is

significant enough to confine diverse forward-temporal provisions from abundant source requests (Lo *et al*., 2007; Lo and Maoz, 2008). In the Move Mine System (Li *et al*., 2011), a situation of usually engaged touching study mining principles are created and a user-friendly border line is offered to formulate possible interactive assessment of touching object data mining and flexible modification of the mining fetters and bounds.

Mining predominant periodic pattern mining can be processed under two constraints. At first, every element in a series encompasses only one item. Then, items emerging in a series that encloses a prototype must be closest as regards the primary order as formulated in the pattern. Most existing approaches do not deal with the trouble particularly but as a substitute they used a common constraint depiction structure to resolve the crisis which is ineffective owing to the large set of databases. In this study, a new data structure, multiplex pruning tree is applied for predominant periodic pattern mining. A multiplex pruning tree combines suffix tree and prefix tree for proficient storage of all the series that include a specified item. Experiments show the effectiveness of multiplex pruning tree for predominant periodic pattern mining.

MATERIALS AND METHODS

Multiplex tree pruning on periodic pattern mining: The proposed research, efficiently designed for identifying the exact attributes accessed from the time series dataset. In order to remove the noise is distributed data pattern distribution model is employed in the first module. The next module proceeds with the process of extracting the frequent set of items using multiplex pruning technique. The multiplex pruning tree technique integrates the suffix and prefix tree which efficiently encodes the frequent item set into the time series database. Once the prefix and suffix tree is constructed, the predominant value is assigned to each item set. Then the value to be pruned is evaluated at the point of integrative form. The architecture diagram of the proposed Multiplex Tree pruning for Periodic Pattern Mining (MTPPM) is shown in Fig. 1.

From the Fig. 1, it is being observed that the proposed periodic pattern mining is based on the multiplex tree pruning technique. At first, the multiplex tree pruning constructs both the suffix and prefix tree for the set of periodic patterns accessed from the time series database. Upon completion of tree construction, the pattern matching procedure takes place on the basis of the predominant values already assigned to the set of nodes in the tree. If the values matched for the suffix and

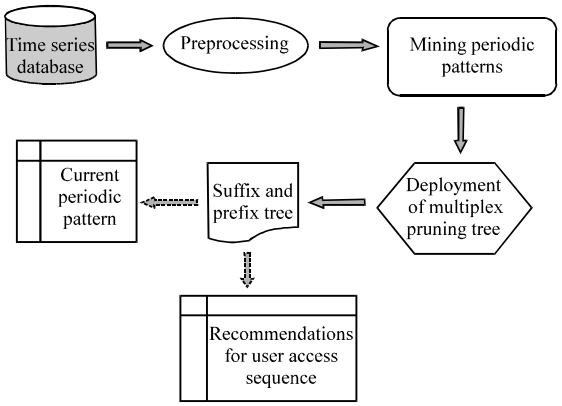

Fig. 1: Architecture diagram of the proposed MTPPM

prefix tree then the derived set of periodic patterns are stored. Or else, the patterns are again constructed with an appropriate value and then integrative function takes place at this juncture.

Construction of suffix tree for set of patterns: The approach used in MTPPM discovers all probable representations. In order to facilitate the examination in a proficient way, two suffix trees are created which are given as:

- A suffix tree on the real dataset (termed data suffix tree)
- A suffix tree on the probable representation strings (termed model suffix tree)

The first tree is categorized into data suffix tree whereas the second tree is termed as the model suffix tree. The model suffix tree assists in the examination of the representation space in a way that consists of unnecessary work. The data suffix tree assists us rapidly in calculation of the support of a representation string. Each node maintains the number of incidences of the string equivalent to that node. Subsequently, the data suffix tree merges the research with the model suffix tree to identify the support for representations like ABCDE and ABCDF (having a common prefix) and accomplishes it only once.

As the second suffix tree (constructed on all potential representation strings) can be tremendously huge, MTPPM does not really build this suffix tree. MTPPM then explores the representation space by negotiating theoretical representation suffix tree. Employing the suffix tree on the dataset, MTPPM calculates the support value at different nodes in the representation space and snips away huge portions of the representation space that are guaranteed not to create any

outcomes beneath the model. This vigilant pruning makes sure that MTPPM doesn't use any time for discovering models that do not contain sufficient support. The algorithm below describes the process of periodic pattern mining using suffix tree.

Procedure for construction of suffix tree:

```
MTPPM _suffix (mTree, dTree, l, d, k):
Step 1. model = mTree.FirstNode()
Step 2. Whilst (model ≠ mTree.LastModel())
Step 3. Assess Support (model, dTree)
Step 4. If (is Valid (model)) Print "establish Model: ", model
Step 5. Else If (model. support() <k)
Step 6. mTree. PruneAt (model)
Step 7. model = Next Node (model, m Tree)
Step 8. End While
Step 9. End
Evaluate Support (model, dTree):
Step 10. nsymbol = last symbol of model. String
Step 11. omatches = model. Parent(). Matches()
Step 12. nmatches = Empty Matches()
Step 13. If (model. Parent() == root)
Step 14. nmatches = enlarge Matches (root, new symbol, dTree)
Step 15. Else
Step 16. For every match x in omatches
Step 17. nmatches = nmatches U
Expand Matches (x, nsymbol, dTree)
Step 18. End For
Step 19. model. Set Matches (nmatches)
Step 20. Return
Expand Matches (x, nsymbol, dTree):
Step 21. Let Y = Set of all single character expansions of x.String in dTree
Step 22. For Each element b in Y
Step 23. If b's last symbol 6 = n symbol
Step 24. b. mismatches++
Step 25. If b. mismatches>max mismatches
Step 26. Remove b from Y
Step 27. End For Each
Step 28. Return Y
```

Let us assume that the dataset consists of series over the set of instances (A, B, C). The dataset and the standards of L, d and k are precised as input. All the cords of length L initiating with the symbol 'A' shape a separation of the representation for suffix tree. Figure 2 shows the process diagram of the suffix tree with the sample set of dataset.

The algorithm puts simultaneously the ideas described. MTPPM uses a suffix tree for the representation and a tot up suffix tree on the dataset. This is created by negotiating the nodes of the representation in depth first order. At every node in the representation suffix tree, the subroutine Evaluate Support() calculate the catalog of matches and the novel support. This practice uses the match catalog from the parent node to derive the calculation. The routine expand matches makes sure that the number of mismatches to the representation string does not go beyond a certain limit d. At some node, if MTPPM determines that the support is lesser than a certain limit k, it snips away that sub-tree in the representation suffix tree and persists its traversal. If it discovers a representation of length L with the necessary support, it merely outputs the result.

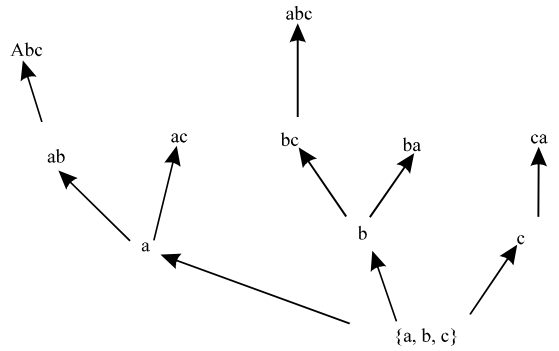

Fig. 2: Representation of suffix tree

Construction of prefix tree for set of patterns: In this study, the procedures explain the definition of key terms that clarify the notions of recurrent pattern mining more effectively. Let L = {i1, i2,..., in} be a position of literals, termed items that have been employed as in turn units in an appliance domain. A set P = {i1, ..., ik}⊆L where, k∈(1, n) is termed as a pattern (or an itemset) or a k-itemset if it holds k items. An operation t = (tid, Y) is a tuple where tid is a transaction-id and Y is a prototype. If P⊆Y it is believed that t holds P or P happens in t. A transaction database DB over L is a position of transactions, i.e., the total number of communication in DB. The support of a prototype P in a DB, indicated as Sup (P) is the number of communication in DB that enclose P. A pattern is termed as a recurrent pattern if its support is no less than a user specified smallest amount support threshold.

A prefix tree is a prearranged tree construction which can symbolize a transaction database in a vastly compacted form. The prefix tree is built by communication being generated one at a time with a pre-arranged item order and planning every operation onto a path in the prefix-tree. As diverse transactions contain numerous items in general their paths might overlap. Given two prearranged patterns $p_1 = \{i_{a1}, i_{a2}, ..., i_{am}\}$ and $p_2 = \{i_{b1}, i_{b2}, ..., i_{bk}\}$ then p_1 is termed as a prefix of p_2. If k = m+1 then p_1 is termed as an instantaneous prefix of p_2. The origin of the prefix-tree symbolizes the empty set. Every node in the tree symbolizes a prototype (or an itemset). There is a boundary among a pattern and its instant prefix. All nodes at point k (the root might be at level 0 and its offspring at point 1 and so on) symbolizes a k-itemset. The representation of prefix tree is shown in Fig. 3.

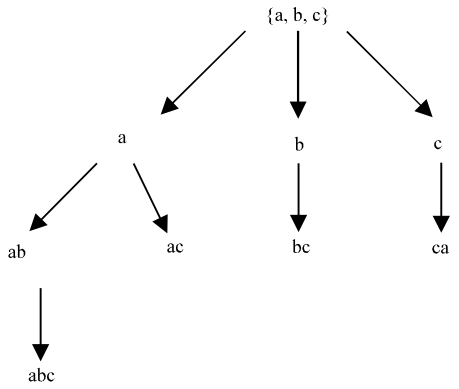

Fig. 3: Representation of prefix tree

The dimension of a prefix tree depends on the extent of prefix sharing amongst all prototypes in the tree. As stated earlier, the superior the prefix distribution is the denser the tree structure could be. The quantity of prefix division also, depends on how specifically the tree explore space is discovered, i.e., the sort of items. For example, if the most recurrent items are rearranged at the better area of the prefix-tree then the probability of reaching towards the superior prefix sharing and compression amplify as a result. Subsequently, a prefix-tree in a canonical order (e.g., lexicographical order) loses its probability to ensure compactness although it might be easier to construct the tree. The dimension of such tree might be improperly huge when there are vast items with small frequency in the database.

Integrative function of prefix and suffix tree represntation: The main obejctive is to discover a data structure that provides efficient periodic pattern mining in terms of both sensitvity and time. For this purpose in this study, MTPPM is presented a special data structure, multiplex pruning tree. Since, multiplex pruning tree has the same root they are integrated by taking a depth first order to recognize periodic pattern mining. The algorithm below describes the process of the proposed MTPPM:

Input/output: Multiplex pruning Tree of x_i/All the patterns that contain x_i
Step 1: Form an empty set for x_i. For every node k in the suffix Tree in depth first order,
Step 2: Form an empty leaf set and obtain the ending nodes of all series in the Id set of k in the prefix Tree. For every ending node, put the Ids of all the series that proceed at it in its endIdSet;
Step 3: Enqueue every ending node into a preference queue supported on the descending order of its stage in the Down Tree;
Step 4: Dequeue nodes in the preference queue still the queue is empty. For every dequeued node m if the dimension of its Set is no fewer than minSup, add m to leaf set, merge to its root.
Step 5: For every node j in the leaf set, form a pattern by adding up the nodes in the path from k to j and unite the prototypes to adjacent set;
Step 6: For the series that are not added up the support of patterns, unite their Ids to the Id set of k's parent node.

Experimental evaluation: In this study, the outcome of numerous experiments that have been processed using both bank and car dataset gathered from UCI repository, account the outcome of trying different individuality of the proposed MTPPM algorithm against other existing algorithms like STNR (Suffix-Tree-based Noise-Resilient algorithm). As mentioned in study, two kinds of algorithms explained in the literature: algorithms in the initial group discover periodic patterns for a precise period value and those in the other group ensure time series for each and every time. The first set of tests is committed to reveal the comprehensiveness of MTPPM in the sense that it would be capable to discover a time once it exists in the time series. Researchers check how MTPPM suits this on both synthetic and real data.

The synthetic data have been produced in the similar way. The parameters collected through data creation are data allocation (uniform or normal), alphabet extent (number of exclusive signs in the data), dimension of the data (quantity of symbols in the data), time period size and the style and quantity of noise in the data. A datum might hold substitution, addition and removal noise or any combination of these kinds of noise.

The MTPPM algorithm discover all the periods in the intense periodic section, generally at superior poise level. An increase in the confidence level of the periodicity patterns results in the performance improvement of the method MTPPM when compared to the existing STNR technique. The performance of the proposed pattern distribution model for noise distributed time series database (MTPPM) is measured in terms of:

- Time efficiency of pruning interested patterns
- Sensitivity
- Selectivity

RESULTS AND DISCUSSION

In this research, researchers have seen how the noise has been removed from the set of periodic patterns in the time series dataset. At first, periodic pattern distribution model is applied to time series database which remove the redundant and unwanted noisy patterns present in it. The next research proceeds with the application of multiplex tree pruning technique in order to form a pattern with respect to prefix and suffix tree representations. The pattern matching problem is accomplished based on assigning a predominant values assigned to it. Experiments have been conducted using bank and car data set and the performance has also been evaluated. Figure 4 describes the performance of the proposed periodic pattern mining using multiplex tree pruning and

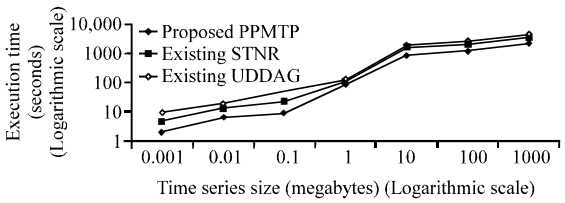

Fig. 4: Time series size vs. execution time

a detailed comparison has been done with the existing STNR technique (Rasheed *et al*., 2011) and Up-Down Directed Acyclic Graph (UDDAG), for efficient sequential pattern mining (Jinline *et al*., 2010).

The execution time taken to identify the set of pruning patterns based on employing diverse approaches is illustrated.

To evaluate the performance of execution time the Proposed Periodic Pattern Mining Algorithms, Fig. 4 exhibits the time behavior of both the proposed and existing algorithms with respect to the time series length. Real and synthetic transactions data is used in diverse section lengths of rules of 2 up to 128 megabytes. Figure 4 shows that the execution time is linearly comparative to the time series extent. The graph also shows that the proposed MTPPM periodic pattern mining takes less execution time than the existing STNR and UDDAG algorithms. The experiment also proves the newest examination using synthetic data and mounting the number of iterations for the periodic pattern mining. In the proposed MTPPM, the pattern mining is done on basis of the segment periodicity mining using multiplex pruning tree. The constructed tree helps the user to identify the interesting patterns in the dataset without any redundancy and noise. Furthermore, Fig. 4 shows that the proposed pattern periodic mining using multiplex tree pruning technique outperforms the periodic trends algorithm of with respect to the execution time.

The sensitivity rate for the bank dataset is discussed in the Fig. 5 for different set of algorithms including both the proposed and existing techniques.

To evaluate the performance of the interesting patterns using bank dataset, the measure of sensitivity rate is shown in the Fig. 5. Sensitivity rate denotes the level of detecting truly interesting patterns from a time series dataset. The research evolved the result using bank dataset collected from the UCI repository. Compared to all other existing research like STNR and UDDAG, the proposed MTPPM provides a better sensitivity rate. The proposed MTPPM constructs both the prefix and suffix trees for a set of patterns for the bank dataset. After constructing a tree, integrative viewpoint is situated with a threshold value to match the sort of patterns

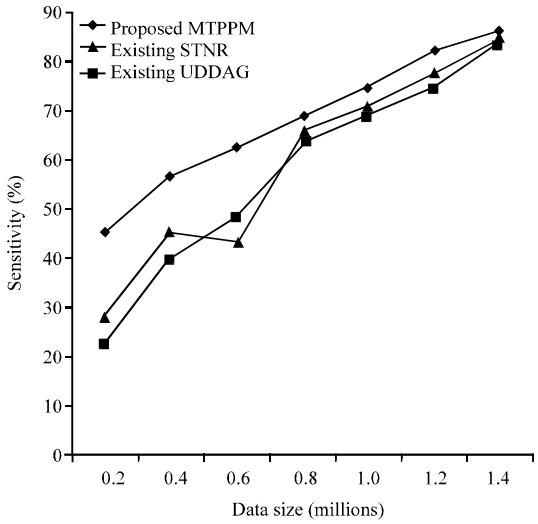

Fig. 5: Data size vs. sensitivity using bank dataset

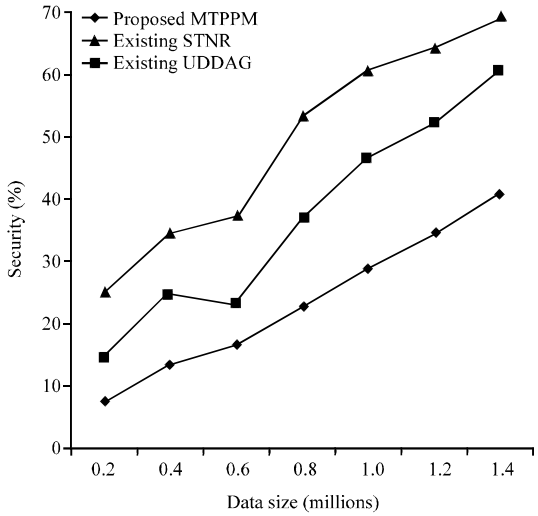

Fig. 6: Data size vs. selectivity using car dataset

constructed by multiplex tree pruning technique. With the values obtained, the patterns are identified and constructed in an appropriate manner. Rather then using the symbol periodicity detection for pattern mining, the proposed MTPPM outperforms well and the variance is 50-60% high in it.

The selectivity rate for the car dataset is discussed in the Fig. 6 for different set of algorithms including both the proposed and existing techniques.

To evaluate the presentation of extracting the non-interesting patterns derived from the car dataset, selectivity rate is measured which is shown in the Fig. 5. The selectivity rate is the rate at which the amount of non-interesting patterns derived by the respective

algorithm is measured. Compared to all other existing researches like STNR and UDDAG, the proposed MTPPM provides less selectivity rate. The proposed MTPPM constructs both the prefix and suffix trees for a set of patterns using car dataset. After constructing a tree, integrative viewpoint is situated with a predefined threshold value to match the sort of patterns constructed by multiplex tree pruning technique. With the predefined threshold values, the patterns are identified and constructed in an appropriate manner. Rather then using the symbol periodicity detection for pattern mining, the proposed MTPPM outperforms well and the variance is 45-65% less in it.

Henceforth, new data structure, multiplex pruning tree scheme is explored to solve the problem of periodic pattern mining using predefined threshold value. Experiment results show that multiplex pruning tree based approach performs much better in terms of both execution time and sensitivity comparing to STNR and UDDAG, the best existing approaches for periodic pattern mining.

CONCLUSION

Researchers propose an efficient Multiple Tree Pruning for Periodic pattern Mining (MTPPM) that enable scalability and efficiency. The systematic evaluation shows that the proposed MTPPM mines the complete set of patterns and its efficiency is proved in executing and runs faster than when compared to two other methods, STNR and UDDAG. The proposed MTPPM uses pre-order connecting of legend nodes in order to accumulate all patterns in the same suffix and prefix tree closely together in the relation, making the search space more efficient. Experimental evaluation shows that the proposed MTPPM proves to be a flexible technique which suits towards numerous real time series dataset for mining purposes. The proposed MTPPM outperforms existing time series mining algorithms in terms of execution time, sensitivity and selectivity. Compared to the existing research like STNR and UUDAG, the proposed MTPPM efficiently outperforms the pattern search easier and achieves 70-80% efficiency in its performance.

REFERENCES

Chen, J. and T. Cook, 2007. Mining contiguous sequential patterns from web logs. Proceedings of the 16th International Conference on World Wide Web, May 8-12, 2007, Banff, Canada, pp: 1177-1178.

Chen, J., 2010. An updown directed acyclic graph approach for sequential pattern mining. IEEE Trans. Knowledge Data Eng., 22: 913-928.

Elfeky, M.G., W.G. Aref and A.K. Elmagarmid, 2005. Periodicity detection in time series databases IEEE Trans. Knowl. Data Eng., 17: 875-887.

Floratou, A., S. Tata and J.M. Patel, 2011. Efficient and accurate discovery of patterns in sequence data sets. IEEE Trans. Knowledge Data Eng., 23: 1154-1168.

Lee, J.G., J. Han, X. Li and H. Cheng, 2011. Mining discriminative patterns for classifying trajectories on road networks. IEEE Trans. Knowledge Data Eng., 23: 713-726.

Li, Z., J. Han, M. Ji, L.A. Tang and Y. Yu *et al.*, 2011. MoveMine: Mining moving object data for discovery of animal movement patterns. ACM Trans. Intell. Syst. Technol., 2: 37-37.

Lo, D. and S. Maoz, 2008. Mining scenario-based triggers and effects. Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering, September 15-19, 2008, L'Aquila, pp: 109-118.

Lo, D., J. Li, I. Wong and S.C. Khoo, 2011. Mining iterative generators and representative rules for software specification discovery. IEEE Trans. Knowl. Data Eng., 23: 282-296.

Lo, D., S. Maoz and S.C. Khoo, 2007. Mining modal scenario-based specifications from execution traces of reactive systems. Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering, March 11-15, 2007, Seoul, Korea.

Niranjan, U., R.B.V. Subramanyam and V. Khana, 2010. An efficient system based on closed sequential patterns for web recommendations. Int. J. Comput. Sci., 7: 26-26.

Rasheed, F. and R. Alhajj, 2010. STNR: A suffix tree based noise resilient algorithm for periodicity detection in time series databases. Applied Intell., 32: 267-278.

Rasheed, F., M. Alshalalfa and R. Alhajj, 2011. Efficient periodicity mining in time series databases using suffix trees. IEEE Trans. Knowledge Data Eng., 23: 79-94.

Vijayalakshmi, S., V. Mohan and S. Suresh Raja, 2010. Mining of users' access behaviour for frequent sequential pattern from web logs. Int. J. Database Manage. Syst., 2: 31-45.