Asian Journal of Information Technology 13 (10): 670-677, 2014
ISSN: 1682-3915
© Medwell Journals, 2014

Parallel Frequent Correlated Pattern Mining Using Time Series Data

[1]G.M. Karthik, [2]S. Karthik and [3]Ramachandra V. Pujeri
[1]Department of Computer Science and Engineering,
SACS MAVMM Engineering College, Madurai, Tamil Nadu, India
[2]Department of Computer Science and Engineering,
SNS College of Technology, Coimbatore, Tamil Nadu, India
[3]Department of Computer Science and Engineering,
KGiSL Institute of Technoloby, Coimbatore, Tamil Nadu, India

Abstract: Since, extraction of frequent itemsets from transaction database is crucial to several data mining task such as association rule generation, so frequent itemset mining is one of the most important concepts in data mining. Frequent pattern mining has been widely used for discovering association and correlation among real data sets. However, discovering interesting correlation among frequent periodic patterns is more complicated and majority of them are unnecessary or uninformative. Researchers designed an algorithm that uses FP-tree for finding periodicity and correlation among multiple longest common substrings in time series data. Researchers introduce a parallel version of the algorithm called Frequent Correlated Periodic Pattern Mining algorithm which takes O(kN) for finding periodicity and tested on a coarse-grained multi-computer (BSP/CGM) Model with synthetic and real data sets. The experiment results show that algorithm is noise resilient, efficient and scalable than existing techniques.

Key words: Frequent pattern mining, correlated pattern, parallel processing, time series, LCS

INTRODUCTION

Identifying repeating (periodic) patterns could reveal important observations about the behavior and future trends of the case represented by the time series (Hansen, 1994) and hence would lead to more effective decision making. The Multiple Longest Common Subsequences (MLCS) are an NP-hard problem (Maier, 1978) with vital application in bioinformatics and computational biology, mostly in DNA and protein sequence analysis. With the increase volume of biological data, researchers expect that MLCS algorithm will have a significant impact on computational biology methods and their applications. The association rule mining (Han and Pei, 2000), sequential pattern mining (Pei and Han, 2002), graph pattern mining (Yan and Han, 2002) etc. are the few common approaches used in it. The real complication occurs in terms of real data sets. The real challenge is gather similar useful pattern collected from a large volume of information that catches the researcher concentration (Al Hasan *et al*., 2007; Chen *et al*., 2008).

A time series is said to have three type of periodic pattern: symbol periodicity, sequence periodicity or partial periodic pattern and segment or full-cycle periodicity (Rasheed *et al*., 2011). Many Existing algorithms (Elfeky *et al*., 2005a, b; Han *et al*., 1998; Indyk *et al*., 2000) 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 (Rasheed *et al*., 2011). To find the periodic pattern in time series data, researchers propose a new and efficient pattern enumeration approach based on the ideas of frequent pattern mining techniques. First, researchers have developed an efficient parallel version of Frequent Correlated Periodic Pattern (FCPP) Mining algorithm for BSP/CGM Model with a near-linear speedup. A novel, compact Frequent Pattern tree (FP-tree) like TRIE data structure, called consensus tree is constructed which enables a highly parallelized search along the tree branches. The construction of consensus tree detects symbol, sequence and segment patterns without periodicity within subsection of the series. The growth of the tree is restrained by providing additional mining constraints. Using the strategy of Constraint-Based Mining (Lu and Lin, 1994; Han *et al*., 1999a, b; Ng *et al*., 1998; Lee and De Raedt, 2004), researchers restraint growth of FP-tree using user-specified constraints

Corresponding Author: G.M. Karthik, Department of Computer Science and Engineering, SACS MAVMM Engineering College, Madurai, Tamil Nadu, India

(Ng *et al.*, 1998) such as level constraint (Han *et al.*, 1999a, b) and rule constraint (Lee and De Raedt, 2004). 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. In time series, there are three types of periodic patterns (symbol/sequence/segment) can be detected (Rasheed *et al.*, 2011).

Thirdly, to mine item pairs of a particular node, represent a periodic pattern and determine the correlated relationship among item pairs. Researchers select appropriate measure to mine the task and demonstrate the outstanding performance of the algorithm based on correlated relationship in terms of both efficiency and effectiveness on datasets. The concept of Frequent Correlated Periodic Pattern mining (FCPP) used with time series data was handled efficiently in this study.

LITERATURE REVIEW

Wang *et al.* (2011) have given basic definition of MLCS problem which is fixed parameter traceable with respect to the length of sequences. Existing techniques widely used dominant point approaches, applied to a case of two sequences (Apostolico *et al.*, 1992; Chin and Poon, 1991). Algorithm A uses three input sequence (Hakata and Imai, 1998). FAST-LCS (Chen *et al.*, 2006), Hakata and Imai (1998)'s algorithm and Wang *et al.* (2011)'s Quick DP algorithm works for arbitrary number of strings. To speed up the computation, parallel MLCS algorithms are developed (Wang *et al.*, 2011). PRAM algorithms for LCS and are presented by Wang *et al.* (2011) for two input strings. Lu and Lin (1994) proposed parallel algorithm with $0(\log^2 m+\log n)$ time complexity with $mn/\log n$ processors when $\log^2 m\log\ \log m\leq \log n$. Xu algorithm takes $0(mn/p)$ time where $1\leq p\leq \max\ (m, n)$. Wang *et al.* (2011) take $|\sum|\log^d n$ time complexity where d represents number of input strings.

The algorithms specified by Elfeky *et al.* (2005a, b), Indyk *et al.* (2000) and Rasheed *et al.* (2011), looks for all possible periods by considering the range. COVN (Elfeky *et al.*, 2005a) fails to perform well when the time series contains insertion and deletion noise. WARP (Elfeky *et al.*, 2005b) can detect segment periodicity; it cannot find symbol or sequence periodicity. Sheng *et al.* (2006) developed algorithm based on Han *et al.* (1999b). 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 *et al.* (2005) used suffix tree similar to sSTNR (Rasheed *et al.*, 2011) which is not beneficial in terms of growth of tree. Huang and Chang (2005) and STNR (Rasheed *et al.*, 2011) 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 (Rasheed *et al.*, 2011) 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 the algorithm is been reduced.

The study of correlation pattern mining focused on two important aspects. The first aspect is the significance of the patterns. More specifically, it is relevant to provide significance measures for the correlation of attribute sets and the correlation patterns. The second aspect is related to the computational cost of the proposed task. Fp-Growth Mining algorithm presented by Han and Kamber (2006), offers the better performance in mining null transactions for subsequent scanning of conditional databases. Omiecinski (2003) and Kim *et al.* (2004) which used to find correlated patterns satisfying gave minimum all-confidence. The concept of independent and correlated pattern was addressed to get the exact time correlated data from the time series data's was handled by Lu and Lin (1994) and Maier (1978).

PARALLEL FCPP MINING ALGORITHM

FP tree construct: Researchers will now present a parallel computing version of FCPP algorithm for MLCS problem. For simplicity, researchers consider the number of processors p to be a power of 2 and m to be a multiple of p, in which p-1 (slave/local) processor used for constructing consensus tree and one processor (master/global) used to find longest common subsequence. The parallel FCPP algorithm divides the number of input sequence into p-1 subsets of size $\lfloor m/p-1\rfloor$ that do not overload. In each p_i processor construct their own consensus tree T_{P_i} with given number of input sequences. All generated subsequence of T_P from each processor given to p_1 (master) to find the longest common subsequence and periodicity among them. Each processor construct consensus tree based on user specified rule and level constraints. FCPP algorithm uses the TRIE like structure (called consensus tree) for shared representation of all subsequence.

FCPP finds all subsequence $S_t\in\sum^t$ with any length l, $0\leq d<l\leq L$ such that for each S_t, there are at least q sequences of S containing an x-mutated copy ($x\leq d$) along with their instances. Each subsequence is mapped to sequence represent by a path from root to particular node (leaf/nonleaf node). Each node contains pointers to all subsequences mapped to st, where a pointer (j, k, e) points to a subsequence are starting at the k th position of the jth sequence and node containing pointers pointing to less than q input sequences with level of mutation $e\leq d$.

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 kth symbol ith 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 ith sequence from 1 to L do
6.  Loop(1):
7.  For each substring's conf($b_1, b_2, b_3, ..., b_{i+1}$)≥1 do
8.  Loop(2):
9.  For each entry (j, k, e) in each nodes $S_{b_1, b_2, ..., b_i}$ where k<L-i+1 do
10. Loop(3):
11. If the k+ith element of the jth sequence is $b_{i+1} \in \sum$ and sup($b_{i+1}$)<q do
12. Begin(1):
13. put (j, k, e) in $S_{b_1, b_2, ..., b_i, b_{i+1}}$;
14. if e<d then for all $b'_{i+1} \neq b_{i+1}$
15. put (j, k, e+1) in $S_{b_1, b_2, ..., b_i, b_{i+1}}$ if and only if conf($b_1, b_2, b_3, ..., b_{i+1}$)≥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 $S_{b_1, b_2, ..., b_i, b_{i+1}} \neq \phi$ do
20. For each node in next level $S_{b'_1, b'_2, ..., b'_i}$ with distance ($b_i$, $b'_i$)≤d do
21. For each $S_{b''_1, b''_2, ..., b''_i} \neq \varnothing$ and $conf(S_{b''_1, b'_2, ..., b''_i}) \geq q$ 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 $T_{b'_1, b'_2, ..., b'_i} \leftarrow T_{b'_1, b'_2, ..., b'_i} \cup S_{b'_1, b'_2, ..., b'_i}$
25. If no node exists in $T_{b'_i, b''_2, ..., b'_i}$ then
26. Increment i; end loop2;
27. Else
28. Print the output sequence $(b'_i, S'_{b_i})$;
29. End Loop 5;
30. If all $S_{b_1, b_2, ..., b_i, b_{i+1}}$ are removed then stop the program else output all
    pairs ($b_1, b_2, ..., b_{i+1}$; $S_{b_1, b_2, ..., b_i, b_{i+1}}$)
31. Remove all $S_{b_1, b_2, ..., b_i, b_{i+1}}$ and $T_{b_1, b_2, ..., b_i}$;
32. End Loop 2;
33. i = i+1;
34. End Loop 1;
```

Each node has $|\sum|$ branches only if nodes satisfy prescribed support and confidence value. The consensus tree's growth is restrained using rule and level constraints. The number of levels in the consensus tree is at most $L = \max\{L_1, L_2, ..., L_N\}$ of the sequences. Nodes with confidence value as:

$$conf(s_t) = \frac{(N\text{-}sup(s_t))}{N\text{-}q} < 1$$

will be pruned; used as antimonotonic constraint (Lee and De Raedt, 2004). The support value (s_t) of subsequence, stand for number of pointers in each input sequences. A node in the consensus tree will not branch out if a support value is ≤q, used similar to monotonic constraint (Lee and De Raedt, 2004). Each instance in a consensus node has to satisfy degree of mutation e≤d otherwise that particular instance is dropped and researchers used it like succinct constraint.

FCPP algorithm performs many comparisons between the subsequence using Hamming distance. Researchers use bitwise comparison with complexity $O((\ln_2|\sum|\times i)/w$.

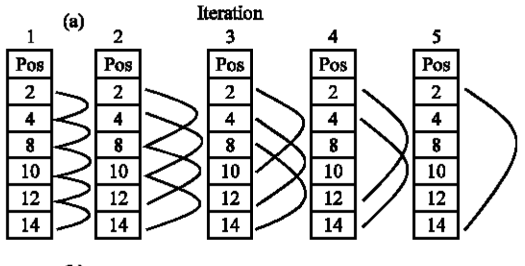
Bitwise comparison is better than Hamming distance calculation when N>2. In worst case the number of developed nodes is N(L-i+1) where each node can produce $\sum_{j=0}^{i}\binom{i}{j}(|\Sigma|-1)^j$ variations with mismatched. This makes space complexity O(N×L×f(d, l)) which is roughly bound by O(N×L).

Periodicity Detection algorithm: The algorithm is linear-distance-based; researchers 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. Figure 1 presents how the Diff_matrix is derived from the position pointers of a particular node.

Algorithm 2 (Difference Matrix (Diff_matrix):

```
• 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) = $S_j$-$S_{j+1}$;
6.  if (j+1 ≠ 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;
```

From the matrix the periodicity is represented by S, K, StPos, EndPos and c denoting the pattern, period value,



(b)

Difference matrix (Diff. matrix)					
K/Pos	2	4	8	10	12
1	2	4	2	2	2
2	6	6	4	4	
3	8	8	6		
4	10	10			
5	12				

Fig. 1: a, b) Difference matrix calculation for 'ab' time series pattern from FP-tree node pointers

starting position and ending position and number of occurrences respectively for a particular consensus node (which denote a pattern). 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 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. The existing algorithms (Kim *et al.*, 2004) 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.

In Algorithm 2, researchers 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. The 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.

Finding correlation in periodic patterns: Discrete Fourier Transform (DFT) is used to identify correlated item pairs in consensus node. The correlation coefficient of two item x and y can be reduced to the Euclidean distance between their normalized series such as corr (x, y) = $1/2md^2(x', y')$, where d(x', y') is the Euclidean distance between x' and y'. By reducing the correlation coefficient to Euclidean distance, applying the technique (Zhu and Shasha, 2003) to report the correlation between the item pairs exists in consensus node which is higher than a specific threshold. Few item pairs can be ignored for which $d_k = (X', Y') > \sqrt{2m(1-T)}$, since they cannot have correlation above a given threshold T. By ignoring such pairs, algorithm will get a set of likely correlated signal pairs. Conceptually, the algorithm produces a matrix like one shown in Fig. 2 where all pairs with correlation above a threshold and some pairs with correlation below the threshold are marked as 1 and all other pairs are marked as 0. Algorithm can call this a pruning matrix P and use it in subsequent steps.

In the technique, the pattern occurrence of item in a node is partition based on the capacity of cache. If the cache does not fit with all instance of the node, partition the instance of the node is done. Thus, computing correlation between signals in different batches incurs additional costs. Hence, existing F-M partitioning algorithm is chosen for partitioning instances of a node into equal size.

In the correlated output R, every non-zero element represents the total number of occurrences of the symbol starting from that position. In that, the first element

	1	2	3	4	5	6	7	8
1	■			■				■
2					■		■	
3			■			■		
4		■					■	
5					■			
6	■			■		■		■
7		■			■		■	
8			■			■		

Fig. 2: Computing a threshold correlation matrix

represents the total number of occurrences of the symbol. The index positions of the non-zero elements are derived from the matrix. From that index position, the perfect and imperfect periodic rates are computed. The equation used is as:

$$R_{xx}(j) = \sum x_n x'_{n-j}$$

Where:

R = The discrete autocorrelation
j = The lag for a discrete signal X_n

EXPERIMENTAL RESULTS

Researchers designed and implemented the parallel version of FCPP algorithm for finding correlated periodic pattern in time series. The algorithm was implemented on the Message-Passing Interface (MPI) System and run on local IBM SP3 machine. Researchers have used Scalasca parallel processing tool which runs in Dell NVIDIA Linux Cluster System and aids in testing the parallel FCPP in Massively Parallel Processing (MPP) systems. The algorithms were tested on set of strings similar to the length of nucleotide and protein sequences ranging between 100 and 5000 with $|\sum| = 4$ and $|\sum| = 20$.

Analysis of FCPP algorithm to find MLCS: The parallel FCPP algorithm was implemented using Scalasca parallel processing tool. The reason is it supports MPP environment which provides efficient performance. The FCPP algorithm consist of master thread which runs on master processor and FP tree creation by slave thread which runs in other slave processors. Master thread divides the number of input sequence based on available slave processor and assigns the input sequence along with constraints to each slave processor. After all slave processor complete consensus tree creation, they generate all subsequence and report it to master processor. Then, the master thread performs bitwise comparison among the all reported subsequence and report the longest subsequence with/without mutation value. FCPP algorithm is compared with Hakata and Imai (1998)'s A and C algorithms, Quick-DP algorithm (Wang *et al*., 2011). The A algorithm is designed for three sequence and C algorithm work with any number of sequence S. Quick-DP has consistent speed up than Hakata and Imai's algorithm.

The algorithm takes more time than both Hakata and Imai's algorithm and Quick-DP because the implementation generates all subsequence of three random DNA sequences of various lengths. Figure 3 shows that FCPP compared with Quick-DP, Hakata and Imai's and FAST-LCS (Chen *et al*., 2006) for more than three random sequence of time series. From Table 1, it is clear Quick-DP has benchmark result than Hakata and Imai's algorithm and FAST-LCS algorithm for both alphabet size |∑| = 4 and |∑| = 20. FCPP algorithm has

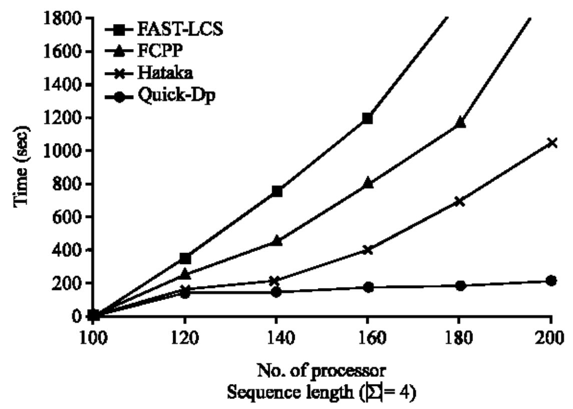


Fig. 3: The average running time of FCPP, Hataka and Imai's C algorithm, Quick-DP and FAST-LCS on MLCS protein of five random strings of different lengths

moderate running time on long sequences. FAST-LCS is fast enough than Hataka and Imai (1998)'s algorithm but compared to Quick-DP less time efficient. The FCPP algorithm's performance is far better than other existing technique. Time performance of FCPP remains same as it checks for all possible subsequence irrespective of the data set.

For real data experiments, researchers 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 (Hettich *et al*., 1998) were also used. Researchers tested how FCPP satisfies this on both synthetic and real data. The algorithm can find all periodic patterns 100% along with their correlation coefficient.

Researchers test the algorithm for various period sizes, distribution and time series length. The data contains the record of around 15 months of data with expected period value of 24. FCPP algorithm with periodicity threshold values ranging from 0.8-0.4 and observed: the number of periods captured by algorithm, StPos and EndPos of the sequence, confidence value and the Pattern shown in Table 2. The expected period 24 is captured at the threshold value 0.8.

The representation of FCPP and Par Per algorithm and its impact over time series was shown in Fig. 4 and 5. The performance comparison of FCPP and ParPer was checked with the data size between 1 and 10 lakhs. As the result the FCPP shown better performance compared with WARP and STNR and projects less impact compared with CONV based on runtime. FCPP maximizes its performance over a period of time as there is persistent progress in data and period size were such combination affects the performance of ParPer (Han *et al*., 1999a, b) and WARP (Elfeky *et al*., 2005b) in terms of its data size.

Table 1: Average running time (seconds) of FCPP, FAST-LCS, Quick-DP and Hataka and Imai's C algorithm for random five sequence of different length

Sequence	Quick-DP		Hataka and Imai's C algorithm		FAST-LCS	FCPP	
length	|∑| = 4	|∑| = 20	|∑| = 4	|∑| = 20	∑| = 4	|∑| = 4	|∑| = 20
100	0.2	0.0	3.6	1.7	46.8	36	26
120	0.6	0.1	15.8	12.2	266.0	184	150
140	0.9	0.4	54.9	26.2	1430.0	1014	890
160	1.4	0.5	149.9	71.5	4801.0	2891	1450
180	2.2	0.8	426.0	203.0	17143.0	6434	2350
200	2.6	1.1	896.0	560.0	40262.0	9832	3122

Table 2: FCPP algorithm output for Wal-Mart data

Data	Periodicity threshold	No. of periods	StPos	EndPos	Conf	Pattern	Time correlated
Store 1	0.8	4	109968	145081	0.42	AAA*******AAA******AA***	AAA****AAA
	0.7	9	134887	161412	0.40	AAABBBCCC************AA*	AABBCC**AA
	0.6	11	151141	194123	0.30	AABBBBCCCD*******AAAA***	AABBBBCCCD
	0.5	16	213476	263129	0.32	AAAABBCCD***AADD*********	AABBCCDD*A
	0.4	25	234980	280673	0.40	AAA***AAAAA*********AAA*	AA*****BBAA

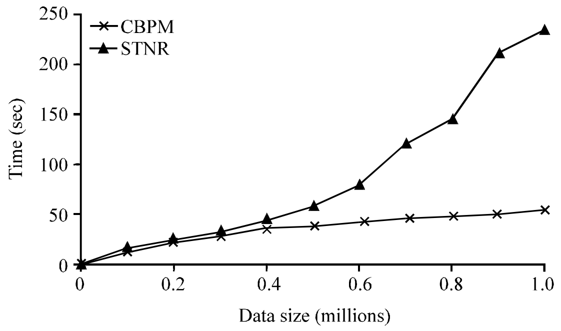
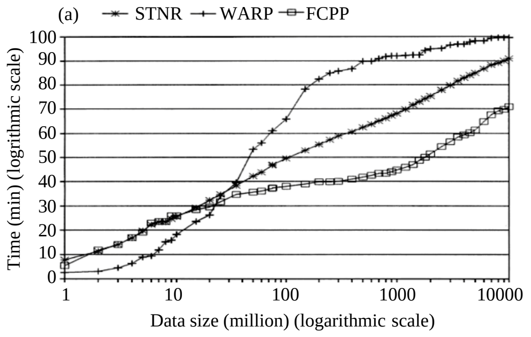


Fig. 4: Time performance of FCPP with ParPer algorithm



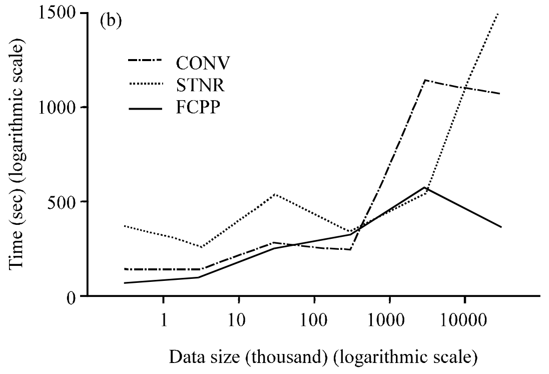


Fig. 5: Time performance of FCPP algorithm with STNR, CONV and WARP; a) time performance of FCPP Mining algorithm; b) time performance comparison of FCPP

Estimation of time correlation: To estimate the correlation and threshold based on wall mart data in Table 2 used to create the correlation matrices as shown in the Fig. 6 of several load balanced Wal-Mart cluster machines. To perform experiments on massive data sets,algorithm generates equal number of data sets with small amount of noise in Fig. 7. The data sets contain day to day transaction established by 10 servers in real data center. One measurement contains minimum 450 transactions; each dataset contains 1000 items sold details. Using FCPP mining algorithm to compute each leaf

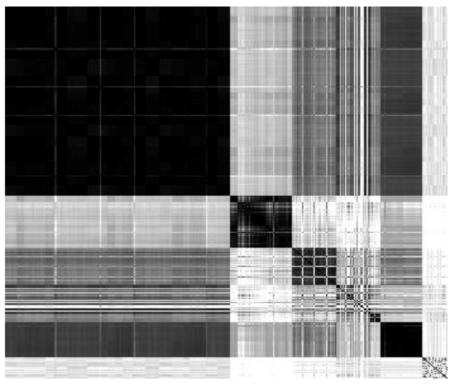


Fig. 6: Correlation matrix C^T

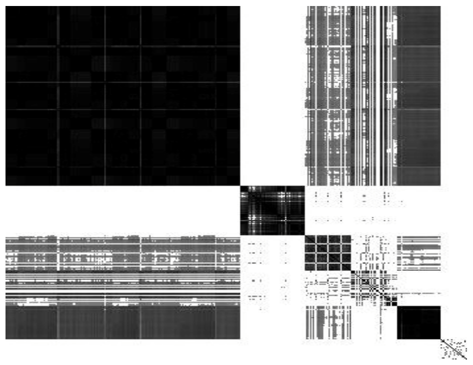


Fig. 7: Threshold correlation matrix C^T

node in consensus tree in each node contains all pair exact correlated items with threshold correlation matrix C^T. For different algorithms, the speedup factors are reported.

To compute C^T for given threshold, discrete fourier transform coefficients is used. The graph show that even with small error is significantly faster than existing algorithms for all real datasets. The small error above comes with significant benefits of speedup. Figure 8 shows the speedup performance of the algorithm to compute C^T than BRAID (Sakurai *et al.*, 2005) which is the state of the art algorithm for computing correlation. the algorithm optimal performance when compared to BRAID for different lagged correlation. Figure 8 shows that the algorithm is slower than BRAID (Sakurai *et al.*, 2005) for different data sets. This huge speedup comes because the algorithm works in finding all correlated patterns among domain and reusing DFT coefficients across different lags. Figure 9 shows the algorithm performance in terms of threshold value with it periods also by considering the factors of starting and ending position with its occurrence. The performance measures of proposed algorithm proven to be effective in terms of its threshold value towards its time series correlation.

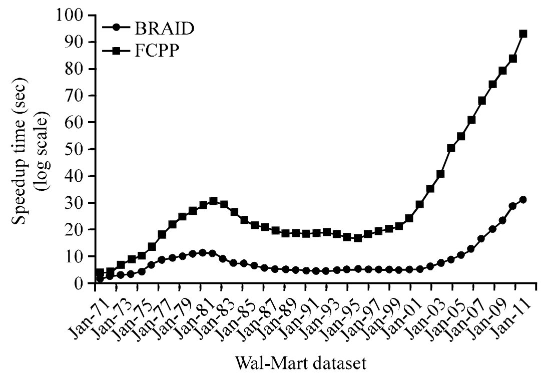


Fig. 8: Speedup performance of FCPP and BRAIT for lagged correlation (Wal-Mart dataset)

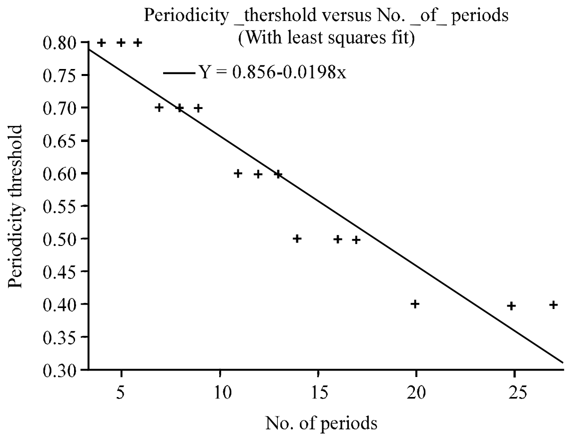


Fig. 9: FCPP comaraison with thresholds and periods

CONCLUSION

In FCPP, a highly parallel TRIE-like structure, the consensus tree and fast level-wise search strategy based on downward closure property help to increase the efficiency. In this research, it addresses the impact of time correlation over time series data. Researchers tested the algorithm on both real and synthetic data in order to test its accuracy, effectiveness of reported results and the noise resilience characteristics. The algorithm was implemented on the Message-Passing Interface (MPI) system and run on local IBM SP3 machine. The parallel version of FCPP algorithm was implemented in Scalasca parallel processing tool. The comparison between the threshold and the period provides maximum versatile result compared with the existing algorithm that addresses time series data. The correlation approach which was addressed in the research gives the result for periodic threshold at par with threshold and correlation all together. A Novel algorithm is proposed based on discrete fourier transform, to reduce the time in finding all-pair correlation patterns. The algorithm has strict error guarantees which make them as useful as corresponding exact solution for many real time applications.

REFERENCES

Al Hasan, M., V. Chaoji, S. Salem, J. Besson and M.J. Zaki, 2007. Origami: Mining representative orthogonal graph patterns. Proceedings of the 7th IEEE International Conference on Data Mining, October 28-31, 2007, Omaha, Nebraska, USA., pp: 153-162.

Apostolico, A., S. Browne and C. Guerra, 1992. Fast linear-space computations of longest common subsequences. Theor. Comput. Sci., 92: 3-17.

Chen, C., C.X. Lin, X. Yan and J. Han, 2008. On effective presentation of graph patterns: A structural representative approach. Proceedings of the 17th ACM Conference on Information and Knowledge Management, October 26-30, 2008, Napa Valley, CA., USA., pp: 299-308.

Chen, Y., A. Wan and W. Liu, 2006. A fast parallel algorithm for finding the longest common sequence of multiple biosequences. BMC Bioinformat., Vol. 7. 10.1186/1471-2105-7-S4-S4.

Cheung, C.F., J.X. Yu and H. Lu, 2005. Constructing suffix tree for gigabyte sequences with megabyte memory. IEEE Trans. Knowl. Data Eng., 17: 90-105.

Chin, F.Y. and C.K. Poon, 1991. A fast algorithm for computing longest common subsequences of small alphabet size. J. Inform. Process., 13: 463-469.

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

Elfeky, M.G., W.G. Aref and A.K. Elmagarmid, 2005b. WARP: Time warping for periodicity detection. Proceedings of the 5th IEEE International Conference on Data Mining, November 27-30, 2005, Houston, Texas, USA., pp: 138-145.

Hakata, K. and H. Imai, 1998. Algorithms for the longest common subsequence problem for multiple strings based on geometric maxima. Optim. Methods Software, 10: 233-260.

Han, J. and J. Pei, 2000. Mining frequent patterns by pattern-growth: Methodology and implications. ACM Sigkdd Explor. Newslett., 2: 14-20.

Han, J. and M. Kamber, 2006. Data Mining: Concepts and Techniques. 2nd Edn., Morgan Kaufmann Publishers, San Francisco, CA., USA., ISBN-13: 9781558609013, Pages: 800.

Han, J., L.V. Lakshmanan and R.T. Ng, 1999a. Constraint-based, multidimensional data mining. Computer, 32: 46-50.

Han, J., G. Dong and Y. Yin, 1999b. Efficient mining of partial periodic patterns in time series database. Proceedings of the 1999 International Conference on Data Engineering, March 23-26, 1999, Sydney, Australia, pp: 106-115.

Han, J., W. Gong and Y. Yin, 1998. Mining segment-wise periodic patterns in time related databases. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, August 27-31, 1998, New York, USA., pp: 214-218.

Hansen, L.K., 1994. Book review: Time series prediction: Forecasting the future and understanding the past, Eds. Andreas S. Weigend and Neil A. Gershenfeld. Int. J. Neural Syst., 5: 157-158.

Hettich, S., C.L. Blake and C.J. Merz, 1998. UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, CA., USA.

Huang, K.Y. and C.H. Chang, 2005. SMCA: A general model for mining asynchronous periodic patterns in temporal databases. IEEE Trans. Knowl. Data Eng., 17: 774-785.

Indyk, P., N. Koudas and S. Muthukrishnan, 2000. Identifying representative trends in massive time series data sets using sketches. Proceedings of the 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt, pp: 363-372.

Kim, W.Y., Y.K. Lee and J. Han, 2004. Ccmine: Efficient mining of confidence-closed correlated patterns. Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, May 26-28, 2004, Sydney, Australia, pp: 569-579.

Lee, S.D. and L. De Raedt, 2004. Constraint Based Mining of First Order Sequences in SeqLog. Database Support for Data Mining Applications, Meo, R., P.L. Lanzi and M. Klemettinen (Eds.). Springer, Berlin, Germany, ISBN-13: 9783540224792, pp: 154-173.

Lu, M. and H. Lin, 1994. Parallel algorithms for the longest common subsequence problem. IEEE Trans. Parallel Distrib. Syst., 5: 835-848.

Maier, D., 1978. The complexity of some problems on subsequences and supersequences. J. ACM, 25: 322-336.

Ng, R., L.V.S. Lakshmanan, J. Han and A. Pang, 1998. Exploratory mining and pruning optimizations of constrained associations rules. ACM SIGMOD Rec., 27: 13-24.

Omiecinski, E.R., 2003. Alternative interest measures for mining associations in databases. IEEE Trans. Knowl. Data Eng., 15: 57-69.

Pei, J. and J. Han, 2002. Constrained frequent pattern mining: A pattern-growth view. ACM SIGKDD Explorat., 4: 31-39.

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.

Sakurai, Y., S. Papadimitriou and C. Faloutsos, 2005. BRAID: Stream mining through group lag correlations. Proceedings of the ACM SIGMOD International Conference on Management of Data, June 13-17, 2005, Baltimore, MD., USA., pp: 599-610.

Sheng, C., W. Hsu and M.L. Lee, 2006. Mining dense periodic patterns in time series data. Proceedings of the 22nd International Conference on Data Engineering, April 3-7, 2006, Atlanta, GA., USA., pp: 115-115.

Wang, Q., D. Korkin and Y. Shang, 2011. A fast Multiple Longest Common Subsequence (MLCS) algorithm. IEEE Trans. Knowl. Data Eng., 23: 321-334.

Yan, X. and J. Han, 2002. gSpan: Graph-based substructure pattern mining. Proceedings of the 2002 IEEE International Conference on Data Mining, December 9-12, 2002, Maebashi, Japan, pp: 721-724.

Zhu, Y. and D. Shasha, 2003. Warping indexes with envelope transforms for query by humming. Proceedings of the ACM SIGMOD International Conference on Management of Data, June 9-12, 2003, San Diego, CA., USA., pp: 181-192.