SEQUENTIAL PATTERN EXTRACTION FROM SERVER ACCESS LOGS USING PLPC-Tree ALGORITHM

Narayan Prasad Keer

Nitin Agrawal

ABSTRACT

This contribution concerns mining frequent sequential patterns on data stream data applications contend with many challenges such as limited memory for unlimited data. Existing work on mining frequent patterns on data streams are mostly for non-sequential patterns and mines the frequent sequences from the WAP-tree by recursively re-constructing intermediate trees, starting with suffix sequences and ending with prefix sequences. This paper proposes an algorithm that uses data structure PLPC-tree to handle the complexities of mining frequent sequential patterns in data streams by totally elimination of numerous re-construction of intermediate WAP-trees during mining. The proposed algorithm constructs the tree with finding frequent individual events and then builds the frequent header node links of the original WAP-tree in an ordered fashion and uses the position code of each node to identify the ancestor/descendant relationships between nodes of the tree. Then, finds each frequent sequential pattern, through progressive prefix sequence search, starting with its prefix subsequence event. Experiments show good performance gain over the WAP-tree technique.

Keywords: Web usage mining (WUM), WAP-tree, WUM methodology, PLPC-Tree Algorithm, sequential pattern mining.

1. Introduction : With the explosive growth of Internet use, mining frequent access patterns from huge datasets of web logs [1] has become important. The frequent web access patterns mined from web logs can help web masters and designers to understand the behaviors of web surfers on their web sites. This understanding and knowledge is essential for the improve the business of the companies and the design of the websites. The advance step of data preprocessing contains transfer the structured file containing visitor episodes to the relational database. After wards, we apply the data generalization at the request level (for URLs) and the aggregated data computation for episodes, visits and user sessions to completely fill the database. The WUM term was introduced by Cooley et al. 1997 [3] [11] [21], the term of "heuristic WUM" was introduced for referring to a WUM process where web access logs from particular Web sites are incrementally analyzed [11] [12]; we use this term for referring to a particular case of Web Usage Mining, when the WUM process applies to a Web site composed of several Web servers (several Web

Student M.Tech. (CSE) Deptt. of CSE, NIIST, Bhopal

Deptt. of CSE, NIIST, Bhopal

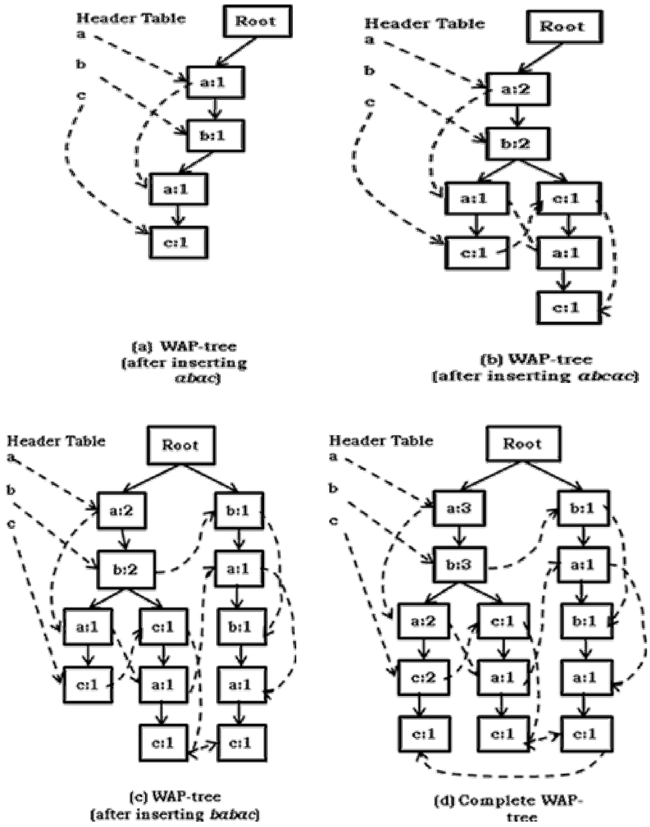
access logs). Web access pattern mining (or web log mining) is an example of general sequential pattern mining, where an event is an acc ess of a URL. The web access sequence database is a multiset of web access sequ ences, each of which, in turn, is a sequence of events of web accesses over a period of time. Another example of frequent pattern mining is the mining of biological sequence s where each sequence is a sequence of amino acids or nucleotides. Due to the importanc e of the problem and large number of applications, web access log mining has attracte d significant attention in the recent years [3, 23, 15, 13, 14, 2]. The most notable algorit hms of web access pattern mining include the apriori-based algorithm GSP [30] and a pattern-growth algorithm, the WAP-tree algorithm [23]. A WAP-tree is an aggregate e [31] that represents the web access sequence database. All nodes with the same label ar e linked when the tree is built [23]. The mining algorithm in both [23] uses recursive co nditional searching of projection databases to find frequent web access patterns. The difference is that the WAP-tree mining [23] grows the suffix of frequent patterns. s known that the mining time of the WAP-tree [23] algorithm is less than that of the Ap riori-based GSP [30] due to much candidate generation process.

11 Related Work Web Usage Mining

Pei et al. (2000) [24] proposed an algorithm using WAP-tree, which stands for web access pattern tree. This approach is quite differe nt from the Apriori-like algorithms. The main steps involved in this technique are summarize d next. The WAP-tree stores the web log data in a prefix tree format similar to the fre quent pattern tree (Han et al. 2004) [5] (FP-tree) for non-sequential data. The algorithm f irst scans the web log once to find all frequent individual events. Secondly, it scans the web log again to construct WAP-tree over the set of frequent individual events of each transaction. Thirdly, it finds the conditional suffix patterns. In the fourth step, it constructs the intermediate conditional WAP-tree using the pattern found in previous step. Finally, it goes back to repeat Steps 3 and 4 until the constructed conditional WAP-tree ha s only one branch or is empty. The complete construction and deconstruction processes to find sequential patterns have been shown in (Pei et al. 2000) [24] similar as shown in the figure 1 & 2 by considering Table 2.

TID	Web Access Sequence	Frequent subsequence
100	abdac	Abac
200	eaebcac	Abcac
300	babfaec	Babac
400	afbacfc	Abacc

Table 2: Sample Web access sequence database for WAP Ptree



(a) WAP-tree (after inserting *abac*)

(b) WAP-tree (after inserting *abcac*)

(c) WAP-tree (after inserting *babac*)

(d) Complete WAP-tree

Figure 1: Construction of WAP-tree

1.2 Motivation and Contributions

Web usage mining is the application of established data mining techniques for analyzing web site usage. For an e-commerce company, this means detecting future customers likely to make a large number of purchases, or predicting which online visitors will click on what commercials or banners based on observation of prior visitors who have behaved either positively or negatively to the advertisement banners. The Apriori-like sequential mining algorithms generate huge sets of candidate patterns, especially when the patterns are long. WAP-tree algorithm has the drawback of recursively re-constructing intermediate WAP-trees during mining which is time-consuming. This paper proposes slightly different technique than [1] which stores the sequential data in an ordered linked PLPC tree. Each of this tree's nodes has a binary position code assigned for directly mining the sequential patterns without re-constructing the intermediate WAP trees. Unlike [1], this paper contributes the technique of constructing tree while finding frequent individual events then builds the frequent header links. The technique proposed for mining in this paper presents a much better performance than that achieved by the previous WAP-tree techniques especially for large databases with very low support

The proposed ordered linked PLPC-Tree Mining algorithm with the Tree Binary Code Assignment (TBCA) algorithm [1]. Section 3 presents an example sequential mining of a web log database with the PLPC-Tree algorithm. Section 4 discusses experimental performance analysis, while Section 5 presents conclusions and future work.

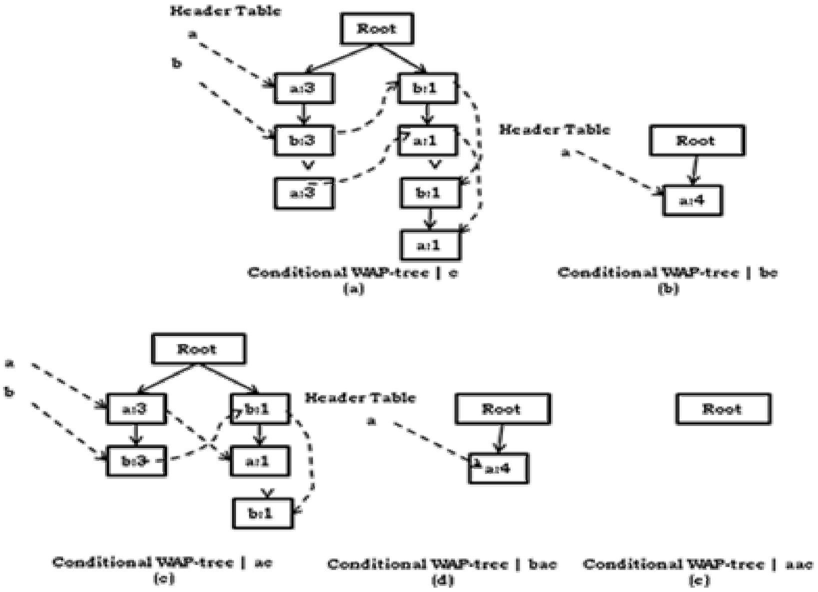


Figure 2 Reconstruction of WAP tree

II. PROPOSED WORK

In this paper the PLPC tree mining algorithm has been proposed to carry out mining of web access logs.

2.1 PLPC-tree mining algorithm

PLPC-Tree algorithm is new sequential pattern mining algorithm for web logs, which is based on PLWAP tree [1] and WAP tree (Pei et al 2000) [24], which avoids recursively reconstructing intermediate WAP trees during mining of the original WAP tree for frequent patterns. PLPC-Tree algorithm is able to quickly determine the suffix trees or forests of any frequent pattern prefix under consideration by comparing the assigned binary position codes of nodes of the tree [1] .

[1] suggested few properties which are required for the PLPC tree. So this section also describes some properties and rules of [1]. According to common terms and concepts related to PLWAP tree based mining from the root to any node in the tree defines a frequent sequence. For any node labeled e_i in the WAP tree, all nodes in the path from root of the tree to this node (itself excluded) form a prefix sequence of e_i. The count of this node e_i is called the count of the prefix sequence. Any node in the prefix sequence of e_i is an ancestor of e_i. On the other hand, the nodes from e_i (itself excluded) to leaves form the suffix sequence of e_i.

Forexam plfabcdfequenpaterntobedisco veredThePLW A Ptreew ouldfindhe
prefxeventafithen,usinghesufixtees ofnodea,w findhenextprefx
subsequence ab and continuing w ithhesufixtree ofb,w findhenextprefx
subsequenceabcandfinaly,abcdThus,heideao fPLW A Pistousehesufixtees of
heleafequenteventinanm -prefxsequenceto recursivelyextendhesubsequenceto
m + 1sequencebyaddingafequenteventhatoccur redihesufixtees.
To assignpositioncodetoPLW A Ptreenodes,hefo lowingpropetiesaredefned.

Rule1 GivenaW A Ptreew ithsom enodes,hepositioncod eofeachnodecansim plybe
assignedfolow ingheulehatheroothasnul positioncode,andheleftm ostchildof
heroothascodeof1,buthecodeofanyother nodeisderivedbyappending1tohe
positioncodeofitsparent,ifthisnodeistheleft m ostchild,orappending10tohe
positioncodeoftheparentifthisnodeisthesec ondleftm ostchild,thethirdleftm ost
childhas100appendedetc.Ingeneral,fothent heleftm ostchild,thepositioncodeis
obtainedbyappendingthebinarynum berof2n−1to theparent'scode.

Property1 A node αisanancestorofanothernode βifandonlyifthepositioncodeof α
with"1"appendedtoitsend,equalsthefirstnu m berofbitsinthepositioncodeof β,
wherexisthe(num berofbitsinthepositioncod eof α)+1)

The PLPC-Tree algorithm is similar to the W A Ptree [24] and PLW A Ptree [1]
algorithm sintroducedearliersincebothePLW A P treeandPLPC-Treeareconstructed
hesam ew ay.H ow ever,thePLPC-Treeism oreeffi entsinceitconstructsthetree
w hilescanningthedatabasefofindingoutfequen individualevents,thenbuildsthe
fequentheadernodelinksoftheoriginalW A Ptree inanorderedfashionandusesthe
positioncodeofeachnodetoidentifytheancestor /descendantrelationshipsbetw een
nodesofthetree.ThePLPC-Treebetteralgorith m forlargedatabaseandw hensupport
isverylow isbasedonthefolow ingProperty 2w hichisderivedfrom theApriori
algorithm (A graw aland Srikant,1995)[33].Othera pplicable propeties are also
presented.

Property2 Ifisafequenteventw ithinthesetofsufixe sofasequentialpaternLin
thew ebaccesssequencedatabase(W A SD),thenthe sequenceLeisafrequentaccess
paternofW A SD.Forexam ple,aciisafequentp atern,andbisafrequentevent
w ithinthesufixsetofa,then,acbisafequen taccesspatern.

Property3 Thesupportcountofanodee$_i$inPLPC-Treeisgreaterthanorequaltothe
sum ofthecountsofthesufixsubtreesofe$_i$.

Property4 Ifthereisnodeinane$_i$currentsufixsubtree,w hichisalsolabeledede$_i$,the
supportcountofthefirste$_i$istheonethatcontributestothetotalsupport countofe$_i$,
w hilethecountofanyothere$_i$nodeinthisam esubtreeisignored.

Property5 Thesupportcountofanodee$_i$,inthecurrente$_i$sufixtrees(alsocalled
currentconditionalPLPC-trees)readytobeminned isthesum ofallthe$_i$nodesinal

suffix trees of e$_i$, or the suffix tree of the Root if the event of a frequent pattern is being mined.

Property 6 e_i is the next frequent event in the mined prefix sub sequence of the node e_i in the current suffix tree of e_i has a support count greater than or equal to the minimum support threshold.

Property 7 For any frequent event e_i, all frequent subsequences containing e_i can be visited by following the e_i linkage starting from the last visited e_i record in the PLPC tree being mined.

Property 8 For any access sequence in an access sequence database WASD, there exists a unique path in the PLPC tree starting from the root such that all labels of the nodes in the path in order are exactly the same as the events in the sequence.

Algorithm 1 (PLPC-Tree: Mines Web Log Sequences with Ordered Linked WAP Tree)

Algorithm PLPC_TREE(WASD, MS, F$_i$)

This algorithm mines web log sequences with ordered linked WAP-Tree, accepts WASD: Web access sequence database, MS: minimum support $(0 < MS \le 1)$ and produces F$_i$: A complete set of frequent pattern in WASD

```
Begin
1)   Scan WASD once, find frequent individual events .
2)   Call PLPC-CONSTRUCT(WASD, MS, L, T)
     //construct a PLPC tree over the set of individual frequent events using algorithm
     PLPC-Construct (Algorithm 2)//
3)   Call PLPC_TREE(WASD, MS, F_i)
     //Recursively mine the PLPC tree using common prefix pattern search algorithm ,
     PLPC-Mine (Algorithm 3)//
End//PLPC-Tree//
```

Algorithm 2 (PLPC-Construct: Construct the PLPC-Tree Mining)

Algorithm PLPC_CONSTRUCT(WASD, MS, L, T)

This algorithm accepts WASD: Web Access Sequence Database, MS: MinSupport (0< MS ≤1), L: 1-frequent events (stored in the header linkage table) and produces a PLPC-Tree T (Ordered Linked Position Coded) and removes the unlinked node. This algorithm also uses two local variables NODE_FOUND and CURRENT_NODE (point)

```
Begin
1.  GET_NODE (ROOT) Set ROOT.POSITION_CODE=NULL; Set
    ROOT.COUNT=0.
2.  For each access sequence S in the sequence database WASD, do
    2.1  Extract frequent subsequence, S'(S_1, S_2, .. , S_n) from S by removing all the
          events in S that are not in L.
    2.2  Set CURRENT_NODE=ROOT.LEFT_CHILD //of Tree T//
    2.3  For K = 1 to N (length of sequence S') do
         2.3a    If CURRENT_NODE=NULL then
```

```
            GET_NODE(NEW); NEW.LABLE=$S_k$; NEW.COUNT=1
            /*child node ($S_k$)*/
            NEW.POSITION_CODE=PARENT.POSITION_CODE+"1"
            Else If CURRENT_NODE.LABEL=$S_k$ Then
            Set NODE_FOUND=True
            Else Set CURRENT_NODE=
            CURRENT_NODE.SIBLING, and keep checking whether
            CURRENT_NODE.LABEL=$S_k$ and search is not done till no sibling of $S_k$ is found
    2.3b If NODE_FOUND Then
    NEW.COUNT=NEW.COUNT+1
    CURRENT_NODE=NEW
    (current_node points to $S_k$)
    Else GET_NODE(NEW); NEW.LABEL=$S_k$; NEW.COUNT=1
    NEW.POSITION_CODE=
    CURRENT_NODE.POSITION_CODE+"0"
    CURRENT_NODE=NEW
3.  PREORDER(T, ROOT) /*visit root, then left and then right subtrees and add all
    nodes to appropriate linkage queue*/
4.  Return(T) with linkage header table L.

End /*PLPC-Construct*/
```

Algorithm-3 (PLPC-Mine: Mining the Ordered Linked PLPC Tree)

Algorithm PLPC-Tree_Mine(T, L, MS, FM, R, F')

This algorithm mines ordered linked PLPC-Tree and accepts T: PLPC-Tree, L: Header linkage table, MS: Min Support ($0 < MS \leq 1$), FM: Frequent m-sequence, R: Suffix root set (R includes root and FM is empty when algorithm is invoked)

This algorithm produces F': Frequent (m + 1)-sequence as an output and uses local variables S: Stores whether node is ancestor of the following nodes in the queue, C: Stores the total number of event e_i in the suffix trees.

```
Begin
1.      If R=NULL Then Return;
2.      For each event $e_i$ in L, FIND($e_e$, $e_i$ ($e_i \in e_e$| suffix tree), do)
        2a.     Save event $e_i$ queue to S
        2b.     Perform for $e_i$-queue
                If event $e_i$ is the descendent of any event in R, and is not descendent of S
                Then
                INSERT(SUFFIX_TREE_HDR, R')
                C.COUNT=C.COUNT+$e_i$.COUNT
                REPLACE(S, $e_i$)
                If C>MS Then
                Append $e_i$ after FM to F' and output F'
                Call PLPC_Mine(T, L, MS, F', R', F')

End /*PLPC-Mine*/
```

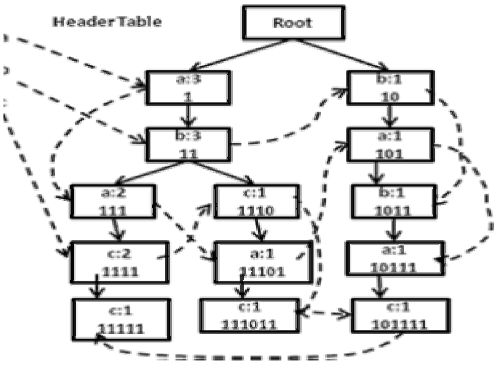
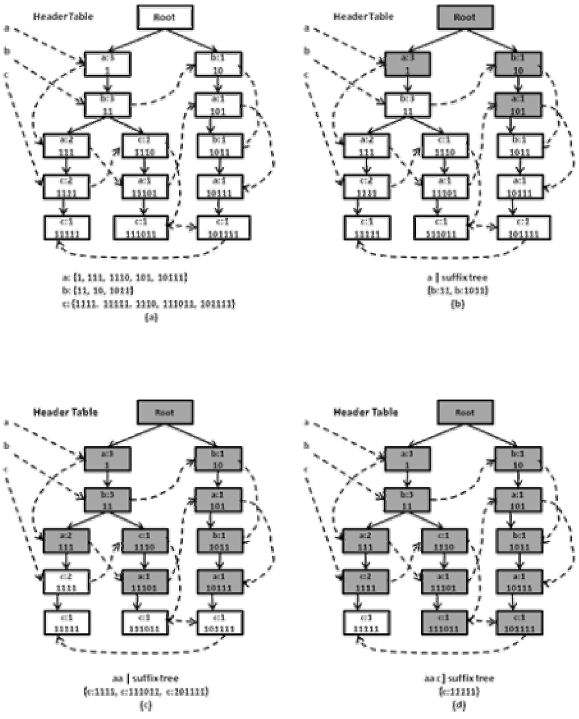


Figure3: Construction of PLPC tree using pre-order traversal

All these three algorithms shown here are used to construct PLPC tree and mine sequential web access pattern. The process of construction PLPC tree is shown in Figure 3 using pre-order traversal and position code assignment suggested in [1].

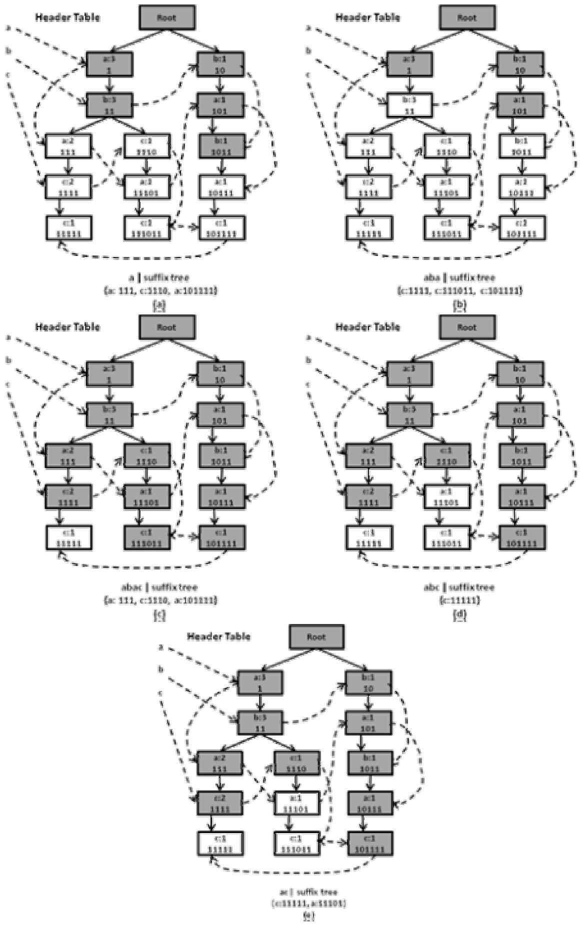

a: {1, 111, 1110, 101, 10111}
b: {11, 10, 1011}
c:{1111. 11111. 1110, 111011, 101111}
(a)

a | suffix tree
{b:11, b:1011}
(b)

aa | suffix tree
{c:1111, c:111011, c:101111}
(c)

aa c| suffix tree
{c:11111}
(d)

Figure4: Mining PLPC tree to find frequent sequences starting with a.



a | suffix tree
{a: 111, c:1110, a:101111}
(a)

aba | suffix tree
{c:1111, c:111011, c:101111}
(b)

abac | suffix tree
{a: 111, c:1110, a:101111}
(c)

abc | suffix tree
{c:11111}
(d)

ac | suffix tree
{c:11111, a:11101}
(e)

Figure 5. Mining PLPC tree to find frequent sequences starting with a b or ac.

Mining PLPC tree to find sequential web access pattern with different starting sequences are shown in figure 4 and 5 stepwise.

III. PERFORMANCE ANALYSIS AND EXPERIMENTAL EVALUATION

To analyze performance and to evaluation experiment results, we use synthetic datasets. For mining PLPC tree we have used synthetic datasets generated by program developed in C++.

3.1 PLPC Mining Experiments

This section we will see the experimental performance of PLPC algorithms. The PLPC algorithms are implemented in C++ language running under Borland C++ environment. All experiments are performed on 2.20 GHz Intel(R) Pentium(R) Dual CPU machine with 1 GB memory. The operating system is Windows Xp. Synthetic datasets are used. The datasets consist of sequences of events, where each event represents an accessed web page. The parameters shown below are used to generate the datasets.

D: Number of sequences in the database
C: Average length of the sequences
S: Average length of maximal potentially frequent sequence
N: Number of events

For example, C10S5N2000D60K means that C=10, S=5, N=2000, and D=60K. It represents a group of data which have average length of the sequences as 10, the average length of maximal potentially frequent sequences is 5, the number of individual events in the database are 2000, and the total number of sequences in database is 60 thousand. The datasets with different parameters test different aspects of the algorithms. Basically, if the number of these four parameters become larger, the execution time become longer.

3.2 Experiment 1: Execution time for different support

This experiment uses fixed size database and different minimum support to compare the performance of PLPC algorithms with WAP algorithm. The algorithms are tested with minimum supports between 0.2% to 15% against the 100 thousand (100K) database. From Table 5.1 and figure 5.1(a) and 5.1(b) can be seen that the execution time of every algorithm decreases as the minimum support increases. This is because when the minimum support increases, the number of candidate sequence decreases. Thus, the algorithms need less time to find the frequent sequences.

Algorithm	Execution time (sec) at different support							
MinSupport (%)	0.2	0.4	0.6	0.8	1	5	10	15
FP	109949	27692	14101	9124	6473	621	186	95
WAP-Tree	228	52	28	21	14	4	3	1
PLPC-Tree	38	9	5	3	3	1	1	0

Table 5.1: Execution times for dataset (100K) at different minimum supports

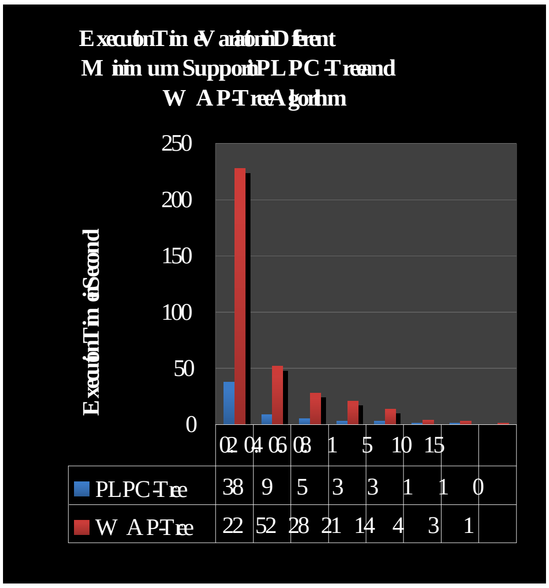


Figure 5.1(a) Execution time variation with different minimum support in PLPC-Tree Algorithm and WAP-Tree Algorithm

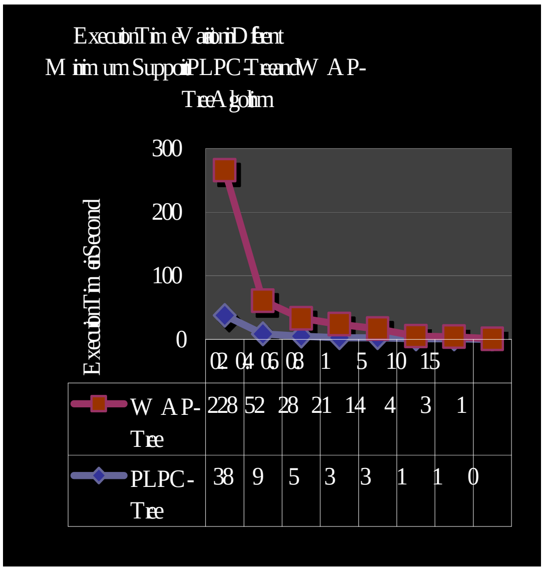


Figure 5.1(b) Execution time variation with different minimum support in PLPC-Tree Algorithm and WAP-Tree Algorithm .

3.3 Experiment 2: Execution time for different Data sizes

This experiment uses different sizes from 20K to 100K database and fixed minimum support 2% to compare the performance of PLPC algorithms with WAP algorithm. Table 5.2 and figure 5.2(a) and figure 5.2(b) can be seen that the execution time of every algorithm increases as the data size increases. This is because when the data sizes increases, the number of candidate sequence increases. Thus, the PLPC algorithm need less time of finding the frequent sequences.

	Different Changed Transaction size					
Algorithm time in Seconds	20K	40K	60K	80K	100K	
WAP	6	7	9	11	13	
PLPC	0	1	1	1	2	

Table 5.2: Execution time for different datasets with fixed minimum support 2%

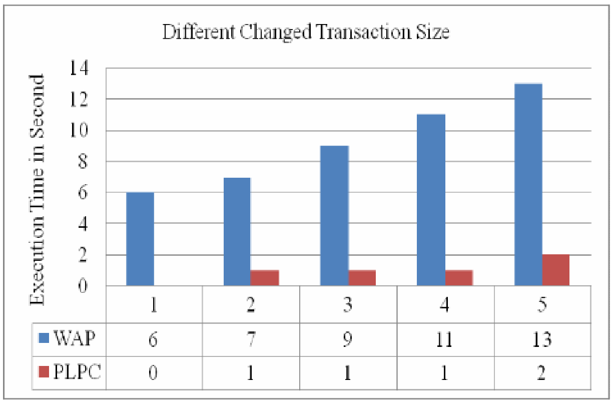


Figure 5.2(a): Execution time for different datasets with fixed minimum support 2%

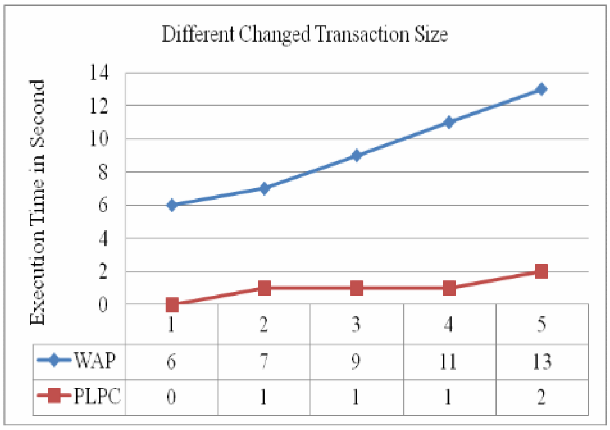


Figure 5.2(b) Execution times for different datase ts fixed minimum support of 2%

Conclusion and perspectives: This paper presents a new algorithm (PLPC-Tree) for efficiently mining sequential patterns from web log .The PLPC algorithm adapts the WAP-tree structure for storing frequent sequential patterns to be mined. However, to improve on mining efficiency, the project proposes to find common prefix patterns instead of suffix patterns as done by WAP-tree mini ng. Moreover, in order to avoid recursively re-constructing intermediate WAP-trees, pre-order frequent header node linkages and position codes are proposed. While the pre-order linkage provides a way to traverse the event queue without going backwards, position codes are used to identify the position of nodes in the PLPC tree. With these two methods, the next frequent event in each suffix tree is found without traversing the whole WAP-tree. Thus, it avoids re-constructing WAP-tree recursively. The experiments show that mining web log using PLPC algorithm is much more efficient than with WAP tree and GSP algorithms, especially when the average frequent sequence becom es longer and the original database becomes larger. For mining sequential patterns from web logs, the following aspects may be considered for future work. The procedure for transforming the web log to database is time-consuming and could be improved upon for web log mining. The PLPC algorithm could be extended to handle sequential pa ttern mining in large traditional databases other than web log and any other order ca n be considered for improvement pre-order linkage. Efficient web usage mining could ben efit from relating usage to the content of web pages. Other areas of interest for f uture work include distributed mining with PLPC trees and applying these techniques to in cremental mining of web logs and sequential patterns.

References

1. Jen-Chih Ou, Chang-Hung Lee, and Ming-Syan Chen. Web log mining with adaptive support thresholds. In Proceedings of 2005 International World Wide Web Conference, pp. 1188-1189, 2005.
2. J. Wang and J. Han. BIDE: Efficient mining of frequ ent closed sequences. In Proceedings of the 20 ^{h} International Conference on Data Engineering (ICDE 04), pp. 79-90, 2004.
3. Han, J., Pei, J., Yin, Y., and Mao, R. Mining frequ ent patterns without candidate generation: A frequent-pattern tree approach. Inter national Journal of Data Mining and Knowledge Discovery, Kluwer Academic Publishers, 8(1), 53–87, 2004.
4. Q. Yang and H. H. Zhang. Web-log mining for predict ive web caching. IEEE Transactions on Knowledge and Data Engineering, 15(4), 1050-1053, 2003.
5. D. Oberle, B. Berendt, A. Hotho, and G onzalez. C onceptual User Tracking. In Proceedings of Atlantic Web Intelligence Conference (AWIC 03), volume 2663 of LNAI, pages 155{164. Springer, 2003.
6. Nanopoulos, A. and Manolopoulos, Y. Mining patterns from graph traversals. Data and Knowledge Engineering, 37(3), 243–266, 2001 .

7. A. Maedche and S. Staab. Ontology Learning for the Semantic Web. IEEE Intelligent Systems, 16(2):72{79, 2001.

8. Pei, J., Han, J., Mortazavi-Asl, B., and Pinto, H. 2001. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In Proceedings of the 2001 International Conference on Data Engineering (ICDE'01), Germany, Heidelberg, pp. 215–224, 2001.

9. F. Bonchi, F. Giannotti, C. Gozzi, G. Manco, M. Nanni, D. Pedreschi, C. Renso, and S. Ruggieri. Web Log Data Warehousing and Mining for Intelligent Web Caching. Data Knowledge Engineering, 39(2):165{189, 2001.

10. R. Kosala and H. Blockeel. Web Mining Research: A Survey. SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM, 2(1):1{15, 2000.

11. Jian Pei, Jiawei Han, Behzad Mortazavi-asl, and Hua Zhu. Mining access patterns efficiently from web logs. In Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00), pp. 396-407. Lecture Notes in Computer Science Vol. 1805, 2000.

12. Berendt, B. and Spiliopoulou, M. Analyzing navigation behaviour in web sites integrating multiple information systems. VLDB Journal, Special Issue on Databases and the Web, 9(1):56–75, 2000.

13. Han, FreeSpan: Frequent pattern-projected sequential pattern mining. In Proceedings of the 2000 Int. Conference on Knowledge Discovery and Data Mining (KDD'00), Boston, MA, U.S.A., pp. 355–359, 2000.

14. Nanopoulos, A. and Manolopoulos, Y. Finding generalized path patterns for web log data mining. Data and Knowledge Engineering, 37(3):243–266, 2000.

15. Han, J. and Kamber, M. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. 2000.

16. Srivastava, J., Cooley, R., Deshpande, M., and Tan, P. Web usage mining: Discovery and applications of usage patterns from web data. SIGKDD Explorations, Vol. 1. Shaffer, C. A. A Practical Introduction to Data Structures and Algorithm Analysis. Prentice Hall, 2000.

17. Spiliopoulou, M. The laborious way from data mining to web mining. Journal of Computer Systems Science and Engineering, Special Issue on Semantics of the Web, 14:113–126, 1999.

18. M. Spiliopoulou, L. C. Faulstich, and K. Winkler. A Data Miner Analyzing the Navigational Behaviour of Web Users. In Proceedings of the Workshop on Machine Learning in User Modelling of the ACAI'99 International Conference, Creta, Greece, July 1999.