iWAP: A Single Pass Approach for Web Access Sequential Pattern Mining

Tarannum Shaila Zaman, Nafisah Islam, Chowdhury Farhan Ahmed, Byeong-Soo Jeong

Abstract— **With the explosive growth of data availability on the World Wide Web, web usage mining becomes very essential for improving designs of websites, analyzing system performance as well as network communications, understanding user reaction, motivation and building adaptive websites. Web Access Pattern mining (WAP-mine) is a sequential pattern mining technique for discovering frequent web log access sequences. It first stores the frequent part of original web access sequence database on a prefix tree called WAP-tree and mines the frequent sequences from that tree according to a user given minimum support threshold. Therefore, this method is not applicable for incremental and interactive mining. In this paper, we propose an algorithm, improved Web Access Pattern (iWAP) mining, to find web access patterns from web logs more efficiently than the WAP-mine algorithm. Our proposed approach can discover all web access sequential patterns with a single pass of web log databases. Moreover, it is applicable for interactive and incremental mining which are not provided by the earlier one. The experimental and performance studies show that the proposed algorithm is in general an order of magnitude faster than the existing WAP-mine algorithm.**

Index Terms—**Data mining, web mining, web access sequences, incremental mining.**

I. Introduction

Data mining [2, 3, 9, 10, 11, 12], the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Web mining is an application of data mining which discovers useful patterns from the Web data repository [4]. This can be defined as the integration of information gathered by traditional data mining methodologies and techniques with information gathered over the World Wide Web. The information gathered through Web mining is evaluated (sometimes with the aid of software graphing applications) by using traditional data mining parameters such as clustering and classification, association, and examination of sequential patterns. Web mining allows you to look for patterns in data through content mining, structure mining, and usage mining. Web usage mining [1], one of the broad categories of Web mining, analyzes frequent user access patterns and relationships from web usage data that can be stored in web server logs, proxy logs or browser logs. The discovered patterns are usually represented as collections of pages, objects, or re-sources that are frequently accessed by groups of users with common needs or interests. Behavior of all users on each web server can be extracted from the web log. An Example of a line of data in a web log is given below in the format:

host/ip user [date:time] ``request url" status bytes 137.207.76.120 -- [30/Aug/2001: 12:03:24 -- 0500] ``GET/jdk1.3/docs/relnotes/deprecatedlist.html HTTP/1.0" 200 2781

Preprocessing tasks can be applied to the original log files to obtain web access sequences after data cleaning, user identification, session identification, etc., for mining purposes.[6]

Sequential mining is the process of applying data mining techniques to a sequential database for the purposes of discovering the correlation relationships that exist among an ordered list of events. Essentially, a Web access pattern is a sequential pattern in a large set of pieces of Web logs, which is pursued frequently by users. [4]

Pei et al. [4] proposed a compressed data structure known as Web Access Pattern Tree (or WAP-tree), which facilitates the development of algorithms for mining web access patterns efficiently from web logs.[6] It stores the web log data in a prefix tree format similar to the frequent pattern tree[2] for non-sequential data.[5] An efficient recursive algorithm is proposed to enumerate access patterns from WAP-tree. Many other researches have been done on this algorithm.

Techniques for mining sequential patterns from web logs fall into Apriori or non-Apriori. The Apriori-like algorithms generate substantially huge sets of candidate patterns, especially when the sequential pattern is long. WAP-tree associates WAP-mine algorithm (Pei et al., 2000), a non-Apriori method which stores the web access patterns in a compact prefix tree, called WAP-tree, and avoids generating huge number of candidate sets . [5] This sequential pattern mining technique finds all sets of frequent sequences by first storing the frequent part of the

Tarannum Shaila Zaman is a student of the Department of Computer Science & Engineering at University of Dhaka, Dhaka, Bangladesh. (e-mail: imagine_prottasha@yahoo.com)

Nafisah Islam is a student of the Department of Computer Science & Engineering at University of Dhaka, Dhaka, Bangladesh.(e-mail:.nafisah001@yahoo.com)

Chowdhury Farhan Ahmed is an Associate Professor of the Department of Computer Science & Engineering at University of Dhaka, Dhaka, Bangladesh. (e-mail: farhan@cse.univdhaka.edu)

Byeong-Soo Jeong is a Professor of the Department of Computer Engineering, Kyung Hee University,1 Seochun-dong, Kihung-gu, Youngin-si, Kyunggi-do, 446-701, Republic of Korea.(e-mail: jeong@khu.ac.kr)

DOI: 10.5176_2010-2283_2.1.129

original web access sequence database on a WAP-tree and then mining that tree according to a given minimum support threshold. The main steps involved in this technique are summarized next.

The algorithm first scans the web log once to find all frequent individual events. Secondly, it scans the web log again to construct a 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 has only one branch or is empty. Thus, with the WAP-tree algorithm, finding all frequent events in the web log entails constructing the WAP-tree and mining the access patterns from the WAP tree. [5]

WAP-tree algorithm scans the original database only twice and thus mining efficiency is improved sharply. But the main drawback is that it recursively constructs large numbers of intermediate WAP-trees during mining and this entails storing intermediate patterns, which are still time consuming operations which lead to many other researches.

A. Related Work

WAP-mine algorithm is obviously a novel data structure in the field of discovering interesting and frequent user access patterns. However, WAP-mine requires re-constructing large numbers of intermediate condition WAP-trees during mining, which is also very costly. So, many researchers proposed many efficient WAP-mine algorithms in different times.

CS-Mine

CS-Mine is based directly on the initial conditional sequence base of each frequent event and eliminates the need for re-constructing intermediate conditional WAP-trees. This can improve significantly on efficiency comparing with WAP-mine, especially when the support threshold becomes smaller and the size of database gets larger.[6]

FS-Mine

The FS-Miner is an incremental sequence mining system. FS-miner efficient strategy for discovering frequent patterns in sequence databases that requires only two scans of the database. Incremental and interactive mining functionalities are also facilitated by the FS-tree. [7]

mWAP:

The modified Web Access Pattern approach is based on WAP-tree, but avoids recursively re-constructing intermediate WAP-trees during mining of the original WAP tree for frequent patterns. The modified WAP algorithm is able to quickly determine the suffix of any frequent pattern prefix under consideration by comparing the assigned binary position codes of nodes of the tree. The tree data structure, similar to WAP-tree, is used to store access sequences in the database, and the corresponding counts of frequent events compactly, so that the tedious support counting is avoided during mining.[8]

PL-WAP Mine:

PL-WAP Mine is a more efficient approach for using the WAP-tree to mine frequent sequences, which totally eliminates the need to engage in numerous re-constructions of intermediate WAP-trees during mining. The proposed algorithm builds the frequent header node links of the original WAP-tree in a pre-order fashion and uses the position code of each node to identify the ancestor/descendant relationships between nodes of the tree. It then, finds each frequent sequential pattern, through progressive prefix sequence search, starting with its first prefix subsequence event.[5]

B. Motivations

Data mining has attracted a great deal of attention in the information industry and in society as a whole in recent years. With the explosive growth of data available on the World Wide Web, discovery and analysis of useful information from the World Wide Web becomes a practical necessity. Many algorithms are improved for sequential pattern mining. The Web access pattern tree stores highly compressed, critical information for access pattern mining and facilitates the development of novel algorithms for mining access patterns in large set of log pieces. Experimental results have shown that WAP-mine is obviously faster than traditional sequential pattern mining techniques.[6] But this algorithm does not support incremental and interactive mining techniques. That is, mining a fixed size web access sequence database for different support threshold needs to build complete WAP-tree for individual threshold. Again, mining with a fixed support threshold for different size of datasets also needs to construct the whole tree. This is because here only frequent part of the database plays role in the construction of WAP-tree. This motivates us to study alternative methods for Web access pattern mining. The key consideration is minimizing time requirement to build WAP-tree and also how to facilitate dynamic mining techniques. So, we propose an improved WAP-tree structure named iWAP-tree which surely minimizes elapsed time for tree construction. Based on iWAP-tree structure, an efficient single pass web access pattern mining algorithm, named as iWAP (improved Web Access Pattern) has been proposed. To support interactive and incremental mining techniques, this algorithm eliminate finding frequent events from database and record the whole web access sequence database in WAP-tree.

C. Outline of the paper

In Section II, we introduce our proposed iWAP-mine algorithm and compare it with existing WAP-mine algorithm. Section III shows the experimental results for both algorithm. Finally, the conclusion is given in Section IV.

II. PROPOSED SINGLE PASS APPROACH

We proposed an efficient algorithm named improved WAP (iWAP) works by scanning the whole database once. It builds its tree named iWAP-tree while scanning the database and counting the occurrence of individual events at the same time.

That means time elapsed for finding frequent event is dropped out.

Algorithm iWAP: mining access patterns in Web access sequence database.

Input: Web access sequence database WAS and support threshold € (0<€ <1).

Output: the complete set of €-patterns in WAS.

Method:

1. Scan WAS, construct a iWAP-tree over the set of individual event.
2. Recursively mine the iWAP-tree using conditional search.

A. *iWAP-tree construction*

iWAP algorithm tried to improve execution time for constructing WAP-tree of WAP-mine algorithm[4]. Main difference between these two algorithms lies in WAP-tree construction. In this subsection, an illustration of how tree construction is done in both algorithm and how they differ are stated using an example.

User ID	WAS	Frequent Access Sequence
100	abdac	abac
200	abcac	abcac
300	babac	babac
400	abacc	abacc

Table. 1 A database of Web access sequences(WAS) with Frequent patterns.

Suppose given threshold is 30% for both algorithm. In case of WAP mine algorithm it means an access sequence, s should have a count of 2 out of 4 records in our example database of table 1, to be considered frequent. Constructing WAP-tree, entails first scanning database once, this derives the set of frequent events *a, b, c* which is shown in the 3rd column of table 1 and stores the frequent items as header nodes.

Using iWAP algorithm finding frequent subsequence is omitted. After inserting a virtual root (Root) each event of individual sequence is inserted as a node with count 1 from Root if that node type does not yet exist. But the count of the node is increased by 1 if the node type already exists. Count is updated whenever the corresponding event is found in a sequence. In this way this header count corresponds to the total occurrence of the event in the database. This variable is useful to determine the frequent event. For example, as shown in Fig. 1(a)., to insert the first sequence *abdac* of transaction ID 100 of the example database, which is a direct child of the Root, a left child of Root is created, with label *a* and count 1.

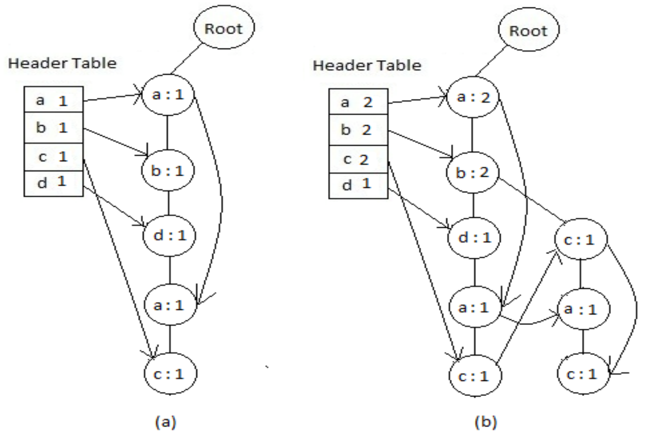

Fig. 1. Construction of iWAP-tree for sequence (a) abdac, (b) abcac

Then, the header link node for event *a* is connected to this inserted a node from the *a* header node. Count of a header node is updated to 1 as it was initially 0. The next event *b* is inserted as the left child of node *a* with a count of 1 and linked to header node *b* with updating the count to 1. The third event d is inserted as the left child of the node *b* having a count of 1, and the *d* link is connected to this node from header node of *d*. The fourth event is *a* and it is inserted as the left child of the d on this branch with a count of 1 and a connection to this node is made from last inserted node *a*. But this time count of a node is not updated because the header count for this event has been updated previously for the same sequence. The fifth and last event of this sequence is *c* and it is inserted as the left child of the second *a* on this branch with a count of 1 and a connection from *c* header node to this node is made.

Secondly, insert the sequence *abcac* of the next transaction with ID 200, starting from the virtual Root. Since the root has a child labeled *a*, the count of node *a* and corresponding header count is increased by 1 to obtain *(a:2)* now. Similarly, *(b:2)* is also in the tree and header count also holds 2. The next event , c does not match the next existing node a, and new node *c:1* is created and inserted as another child of *b* node. In same way, next *a* and *c* are inserted in newly created branch of the tree, as shown in Fig. 1(b). The third sequence *babac* and fourth sequence *abacc* are inserted next to obtain complete iWAP-tree Fig. 2. In this way, iWAP-tree construction takes place in single pass.

On the other hand, WAP-mine algorithm constructs the WAP tree with frequent sub-sequences (column 3 of table 2).It finds the frequent subsequence in the first pass and in the second pass as the same way as iWAP does it constructs the WAP tree(Fig. 3).No count variable is maintained in header table.

Once the sequential data is stored on the complete iWAP-tree Fig. 2 or WAP-tree Fig. 3, the tree is mined for frequent patterns starting with the lowest frequent event in the header list. Mining procedure of iWAP is same as the WAP-mine algorithm except it does not create conditional sequence base as well as conditional iWAP-trees for all events in the header table. Count variable in the header table helps to do mining for those events that meets with minimum support threshold.

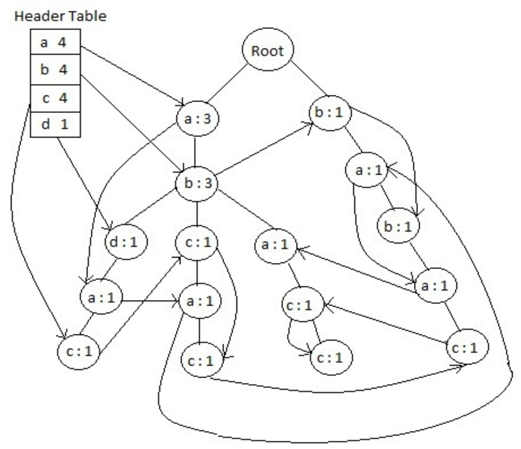

Fig. 2. Complete iWAP-tree based on table 1

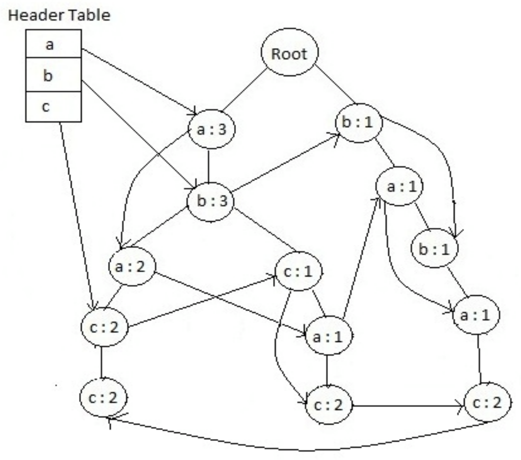

Fig. 3. Complete WAP-tree based on table 1

B. Mining Web access sequences

As does not meet support threshold we start from frequent event *c*. From the iWAP-tree of Fig 2, it first computes prefix sequence of the base *c* or the conditional sequence base of *c* as : {*abda:1; ab:1;abca:1; ab:-1;baba:1; abac:1}.*

The count for each conditional base path is the same as the count on the suffix node itself. The first sequence in the list above, *abda* represents the path to the first *c* node in the iWAP-tree. When a conditional sequence in a branch of a iWAP-tree, has a prefix subsequence that is also a conditional sequence of a node of the same base, the count of this new subsequence is subtracted because it has contributed before. Thus, the conditional sequence list above has one *ab* with count -1. With these sequences next conditional iWAP-tree is constructed shown in Fig. 4. Next conditional sequence base

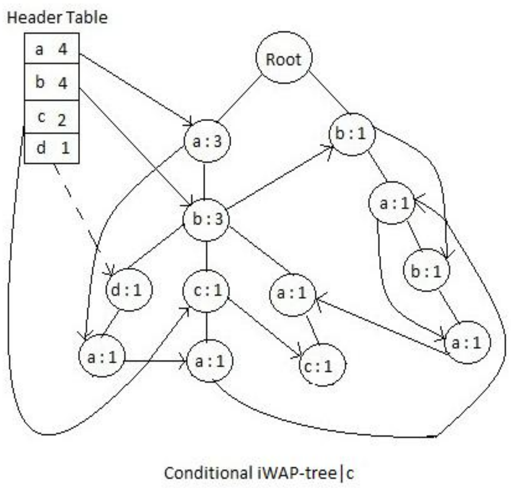

Fig. 4. Conditional iWAP-tree during mining

for suffix *cc* as ab:1,aba:1 and corresponding conditional iWAP-tree |cc is shown in Fig. 5(a). After building the next conditional iWAP-tree |bcc (Fig. 5(b)), this ends the re-construction of iWAP-trees for event *c* and the frequent

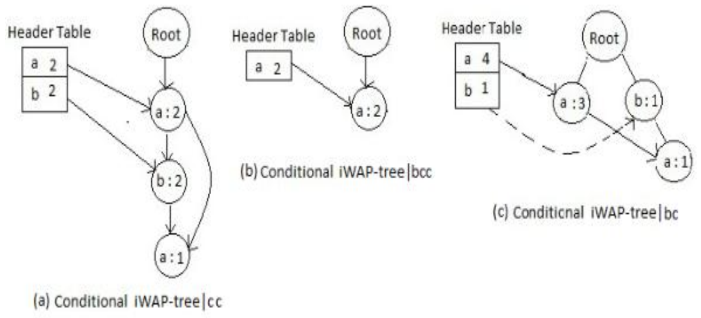

Fig. 5. Conditional iWAP-trees during mining

patterns found along this line are *cc, bcc, abcc, acc*. The recursion continues with the suffix path |c, |bc, |ac, |bac. The algorithm keeps running, finding the conditional sequence

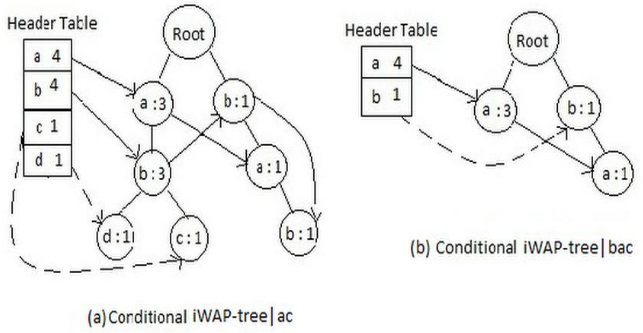

Fig. 6. Conditional iWAP-trees during mining

bases of *ac* and building the conditional iWAP-tree |ac Fig. 6(a). This has header count of 4, 4, 1, 1 for event a, b, c, d respectively.

Since the count of c & d is less than the minimum support threshold, it is discarded in future mining process. The conditional search of c is now finished. The search for frequent patterns that have the suffix of other header frequent events starting with suffix base |b and then |a are also mined in the same way the mining for patterns with suffix c is done above. Discovered frequent pattern set for c is: *{c, cc, acc, bcc, abcc, aac, bac, abac, ac, abc, bc}*.

After mining the whole tree, discovered frequent pattern set is: {*c, cc, acc, bcc, abcc, aac, bac, abac, ac, abc, bc, b, ab, a, aa, ba, aba*}.

Though WAP-tree is an effective structure facilitating Web access pattern mining, and WAP-mine outperforms GSP based solution in a wide margin, it has still some drawbacks. Many researches have been done to improve this algorithm. Our strategy is to improve WAP-tree construction which leads to dynamic mining facility.

III. EXPERIMENTAL RESULTS

Testing is a very essential part of research. To demonstrate the effectiveness of our algorithm here we have shown some experimental results from a number of test cases with performance analysis. The two algorithms WAP-mine and iWAP are implemented in C language. All experiments are performed on Intel Core 2 Duo desktop processor of 2.53GHz having 4GB DDR 2 RAM with a Windows XP operating system.

Used Datasets: Experimental datasets are generated by using our data generation program. Three parameters are used to generate the data sets.

|D|: Number of sequences in the database.

|C|: Average length of the sequences.

|N|: number of events.

For example, C10.N50.D10 K means that |C| = 10, |N| = 50, and |D| = 10K.It represents a group of data with average length of the sequences as 10, the numbers of individual events in the database are 50, and the total number of sequences in database is 10 thousand. The datasets with different parameters test different aspects of the algorithms. Basically, if the number of these three parameters becomes larger, the execution time becomes longer.[5]

A. *Experiment 1*

Fixed size database and different minimum support is used in this experiment to compare the performance of WAP-mine and iWAP algorithms. The datasets are described as C15. N50. D5 K, and algorithms are tested with different minimum support threshold (between 5%-50%) against the 5 thousand (5 K) database. It shows, our algorithm is much efficient for small support threshold which is shown in Fig . 7.

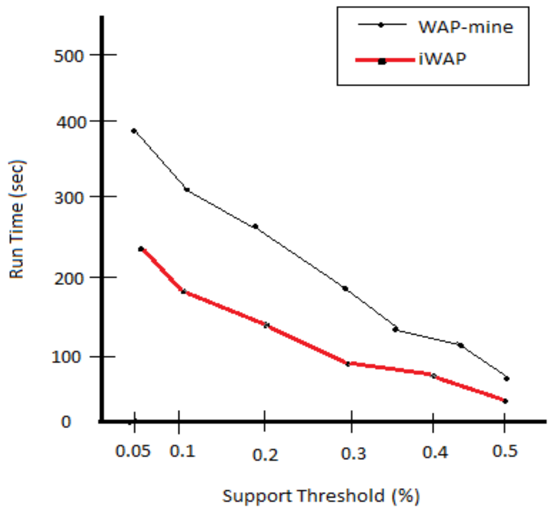

Fig. 7. Scalability measurement with different support thresholds

B. *Experiment 2*

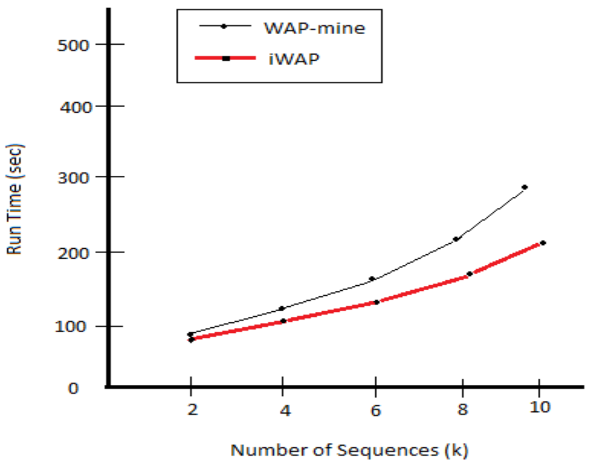

Fig. 8 Scalability measurement with different number of input sequences

Databases with different sizes from 2 K to 10 K with the fixed minimum support of 30% were used in this experiment. The five datasets are C10. N40. D2 K, C10. N40. D4 K, C10. N40.D6 K, C10. N40. D8 K and C10. N40. D10 K. Performance measurement of the WAP and Proposed algorithms were compared and the results of this experiment are presented in Fig. 8. This is an example result of incremental mining technique.

C. *Experiment 3*

Initially, both algorithms will take almost same amount of time with 50% threshold. But when user will change threshold to 40% WAP-mine algorithm will rebuild the whole WAP-tree for same set of database. On the other hand, iWAP algorithm does not need to rebuild the tree as it has recorded full database in iWAP-tree earlier. Now only time for mining will be consumed by iWAP for further changing in threshold whereas WAP-mine algorithm will consume time for both tree construction and mining procedure, shown in Fig. 9.

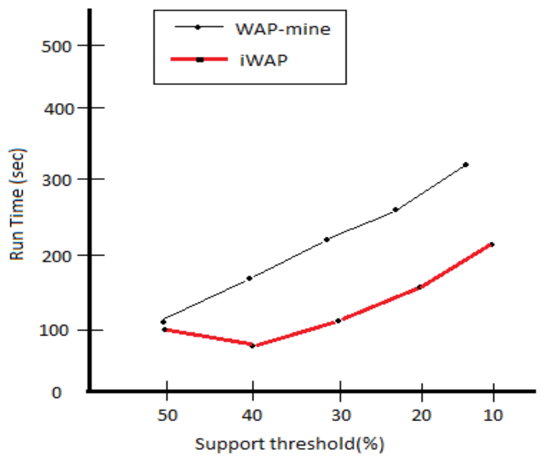

Fig. 9 Experimental result for interactive mining.

IV. Conclusion

Now a days web access sequential pattern mining is very important topic for research. It has lots of application and also attracts lots of researchers to research for a better solution. In this research we try to find another better solution based on the WAP-mine algorithm for web access sequential pattern mining. We do the improvement in case of building the WAP-tree which helps us to speed up the total CPU time. We have proposed an efficient tree structure iWAP-tree and corresponding mining algorithm for web access sequential pattern mining which support the interactive and incremental mining facility. The performance of the iWAP algorithm has been evaluated in comparison with the WAP-mine algorithm. Experimental results have shown that the iWAP algorithm performs much more efficient than the WAP-mine algorithm, especially when the support threshold becomes small. Here in the proposed algorithm the memory issue is not considered. As for future work, the mining process and solving the memory problem of iWAP is considered.

References

[1] Grcar., M.: USER PROFILING: WEB USAGE MINING. Department of Knowledge Technologies Jozef Stefan Institute Jamova 39, 1000 Ljubljana, Slovenia, pp. 45-49.

[2] Han, J., Pei, J., Yien, Y., Mao,R.: Mining Frequent patterns without candidate generation; a frequent pattern tree approach. Data mining and knowledge discovery,pp. 54-87(2004)

[3] Agrawal, R., Srikant, R.: Mining Sequential Patterns. In: Proceedings of the 11th International Conference on Data Engineering, Taipei, Taiwan(1995)

[4] Pei, J., Han, J., Mortazavi-Asl, B., Zhu, H. : Mining access patterns efficiently from web logs. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD00). Kyoto, Japan, pp. 396-399, 400-402 (2000)

[5] EZEIFE, C., LU, Y.: Mining Web Log Sequential Patterns with Position Coded Pre-Order Linked WAP-Tree. In: International Journal of Data Mining and Knowledge Discovery(DMKD) Kluwer Publishers, pp. 6,10, 12-15, 27-34 (2005).

[6] Zhou, B., Cheung Hui, S., Fong, A. : CS-Mine: An Efficient WAP-Tree Mining for Web Access Patterns. In: School of Computer Engineering, Nanyang Technological University, Singapore, pp. 523- 527, 530-532.(2004)

[7] Maged, E., Elke, A.R., Carolina, R. : FS-Miner: An Efficinet and Incremental System to Mine Contiguous Frequent Sequences. In: Computer Science Technical Report Series,Worcester Polytechnic Institute, pp. 128-130 (2003).

[8] Parmar, J., Garg, S.: Modified Web Access Pattern (mWAP) Approach for sequential Pattern Mining. In: International Conference on Data Mining, pp. 30-35 (2007)

[9] Cooley, R., Mobasher, B., Srivastava, J.: Data Preparation for Mining World Wide Web Browsing Patterns. In: Journal of Knowledge and Information Systems, Vol. 1, No. 1, pp. 67-71 (1999)

[10] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: 20th International Conference on Very Large Databases. Santiago, Chile, pp. 487-499 (1994)

[11] Kosala, R., Blockeel, H.: Web Mining Research: A Survey. ACM SIGKDD Explorations,Vol. 2., pp. 1-15 (2000)

[12] Ahmed, C. F., Tanbeer, S. K., Jeong B. S. , "An Efficient Method for Incremental Mining of Share-Frequent Patterns". In: 12th International Asia-Pacific Web Conference (APWeb), Busan, South Korea, pp. 147-153 (2010)

Tarannum Shaila Zaman received her B.S degree in 2011 and now studying M.S in Computer Science & Engineering from the University of Dhaka, Bangladesh. From September, 2011 to December, 2011 she worked as a Junior Quality Assurance Engineer in Structured Data Systems Limited. Since January, 2011 she has been working as a Junior Software Engineer in Together Initiative. Her research interest is in the area of data mining.
E-mail: imagine_prottasha@yahoo.com

Nafisah Islam received her B.S degree in 2011 and now studying M.S in Computer Science & Engineering from the University of Dhaka, Bangladesh. Since September, 2011 she has been working as a Junior Quality Assurance Engineer in Structured Data Systems Limited. Her research interest is in the area of data mining.
E-mail:nafisah001@yahoo.com

Chowdhury Farhan Ahmed received his B.Sc. and M.Sc. degrees in Computer Science from the University of Dhaka, Bangladesh in 2000 and 2002 respectively, and Ph.D. degree in Computer Engineering from Kyung Hee University, South Korea in 2010. From 2003-2004 he worked as a faculty member at the Institute of Information Technology, University of Dhaka, Bangladesh. Since 2004, he has been working as a faculty member in the Department of Computer Science and Engineering, University of Dhaka, Bangladesh. His research interests are in the areas of data mining and knowledge discovery.
E-mail: farhan@cse.univdhaka.edu

Byeong-Soo Jeong received his B.S. degree in Computer Engineering from Seoul National University, Korea in 1983, his M.S. degree in Computer Science from the Korea Advanced Institute of Science and Technology, Korea in 1985, and his Ph.D. in Computer Science from the Georgia Institute of Technology, Atlanta, USA in 1995. In 1996, he joined the faculty at Kyung Hee University, Korea where he is now a professor at the College of Electronics & Information. From 1985 to 1989, he was on the research staff at Data Communications Corp., Korea. From 2003 to 2004, he was a visiting scholar at the Georgia Institute of Technology, Atlanta. His research interests include database systems, data mining, and mobile computing.
E-mail: jeong@khu.ac.kr

.