A NOVEL APPROACH FOR WEB PAGE SET MINING

R.B.Geeta [1], Omkar Mamillapalli [2]
Shasikumar G.Totad [3] and Prof Prasad Reddy P.V.G.D [4]

[1]Associate Professor, Department of CSIT, GMRIT, Rajam, (A.P), India.
geetatotad@yahoo.co.in
[2]Department of CSE, GMR Information Technology, Rajam, (A.P), India.
omkar256@gmail.com
[3]Professor & HOD, Department of CSE, GMRIT, Rajam, (A.P), India.
skumartotad@yahoo.com
[4]Professor, Department of CS & SE, AndhraUniversity, Visakhapatnam,(A.P),India
Prasadreddy.vizag@gmail.com

ABSTRACT

The one of the most time consuming steps for association rule mining is the computation of the frequency of the occurrences of itemsets in the database. The hash table index approach converts a transaction database to an hash index tree by scanning the transaction database only once. Whenever user requests for any Uniform Resource Locator (URL), the request entry is stored in the Log File of the server. This paper presents the hash index table structure, a general and dense structure which provides web page set extraction from Log File of server. This hash table provides information about the original database. Web Page set mining (WPs-Mine) provides a complete representation of the original database. This approach works well for both sparse and dense data distributions. Web page set mining supported by hash table index shows the performance always comparable with and often better than algorithms accessing data on flat files. Incremental update is feasible without reaccessing the original transactional database.

KEYWORDS

Web mining, URL, Web Pages set extraction, HTTP transaction, Log File.

1. Introduction

Many researchers and practitioners have been investigated Association rule mining has been for many years [1], [3], [4], [5], [6]. Agrawal et al. introduced the problem of mining frequent itemsets for the first time[5], who proposed algorithm Apriori. The Apriori algorithm must scan the transcation database several times and FP_growth algorithm needs to scan the database only twice. If the the database is larger, the efficiency of FP growth algorithm is higher. To reduce scanning of database twice, the rapid association rule mining algorithm came into existence. The rapid association Rule Mining algorithm (QFP) requires to scan the transaction database once compared to FP growth algorithm, so it can increase the time efficiency of mining association

DOI : 10.5121/ijwest.2011.2412

rules [2].The correlations among web pages in a transactional database D can be identified using association rules. Association refers to how the web pages in the web site can be grouped. These are used to assist retail store management, marketing, grocery store problems, and inventory control. Each transaction in D is a set of web pages. Association rules are usually represented in the form A -> B, where A and B are web page sets, i.e., set of web pages. Web page sets are characterized by their frequency of occurrence in D, which is called support. Research activity usually focuses on defining efficient algorithms for web page set extraction, which represents the most computationally intensive knowledge extraction task in association rule mining [7]. In this paper, we propose a similar approach to support data mining queries. The WebPages-Mine (WPs-Mine) index is a novel data structure that provides a compact and complete representation of transactional data supporting efficient item set extraction from a relational DBMS. The following Web Pages data set shows 13 hypertext transfer protocol (http) transaction requests in given session threshold.

Table 1. Example Web Pages data set

TID	WebPagesID	TID	WebpagesID	TID	WebPagesID
1	b,h,e,p,v,d,g	6	a, r, n, u ,i, b,s	11	r,e,h,b,a
2	m,h,n,d,b,e	7	b, g, h, d, e,p	12	z,i,a,n,r,b
3	I,f,o,h,e,c,p	8	a, i, b	13	b,d,h,p,e
4	a,w,e,k,h,j	9	f, e ,i, c ,h, p		
5	d,b,e,h,n	10	h, a, e, b, r ,t		

2. The WPs-Mine Index

Whenever user requests for any Uniform Resource Locator, the details of request is entered into the Log File of the server. The log file entry contains various fields like IP address, time at which request is made, status code, number of bytes transferred and which page is requested. The web pages information collected in the log file is stored in the form of database. This data is stored in the form of relational model, as a relation R. Assuming some session threshold the frequency of each webpage is counted and stored in dataset as shown in table 1.

2.1 WPs-Hash Indexed tree Structure

The structure of the WPs-Mine index is characterized by two components: the Web Page Set-Tree and the Web Pages-Hash table tree. The two components provide two levels of indexing. The Web Pages set-Tree (WPs-Tree) is a prefix-tree which represents relation R by means of a brief and compact structure. The hash table of 26 buckets [A-Z] is created. Each bucket stores the information about the support of each web page in a assumed threshold. Each bucket holds the physical location of each web page in the website. Linked list with various nodes is attached for each bucket which holds the addresses of different web pages. The WebPages-Hash index (WPs-H index) table structure allows reading selected WPs-Tree portions during the extraction task. For each item, it stores the physical locations of all item occurrences in the WPs-Tree.

2.1.1 WPs-Tree:

The Web Pages set-Tree (WPs-Tree) is a prefix-tree which represents relation R by means of a short and compact structure. Implementation of the WPs-Tree is based on the FP-tree data structure, which is very effective in providing a compact and lossless representation of relation R as shown in Figure .1.

2.1.2 WPs-Hash-indexed tree:

The WPs-Hash-tree is a Hash table with tree structure which allows access of selected WPs-Tree portions during the extraction task. For each web page in the given website, it stores the physical locations of all web page occurrences in the Web Pages set Tree.

Figure 2 shows the WPs-Mine Hash indexed tree allows selectively accessing the WPs-Tree blocks during the extraction process. It is based on a Hash indexed Tree structure. For each item i in relation R, there is one entry in the WPs-Mine Hash indexed tree.

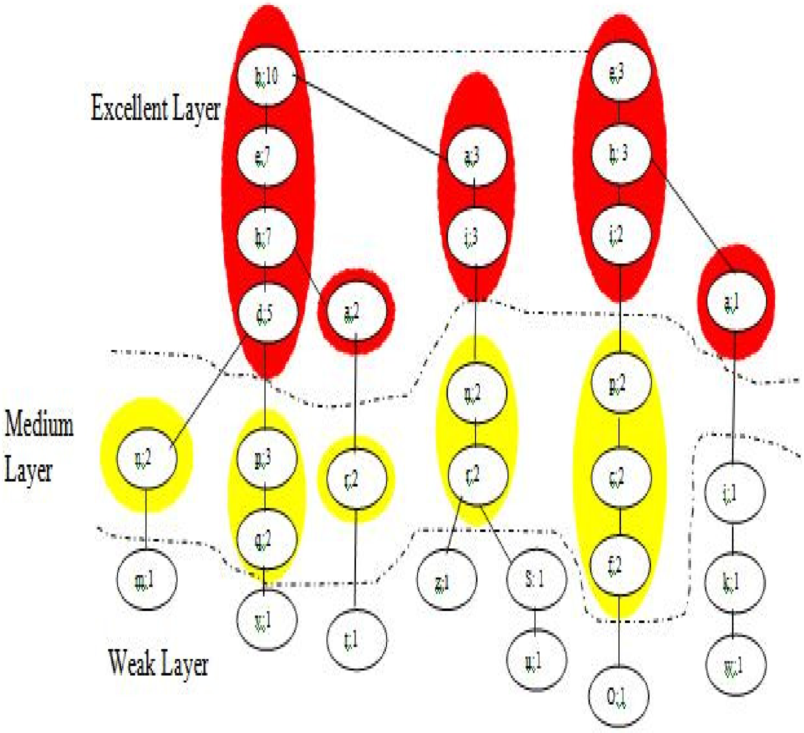


Figure 1. WPs-Mine index for the example dataset WPs-Tree

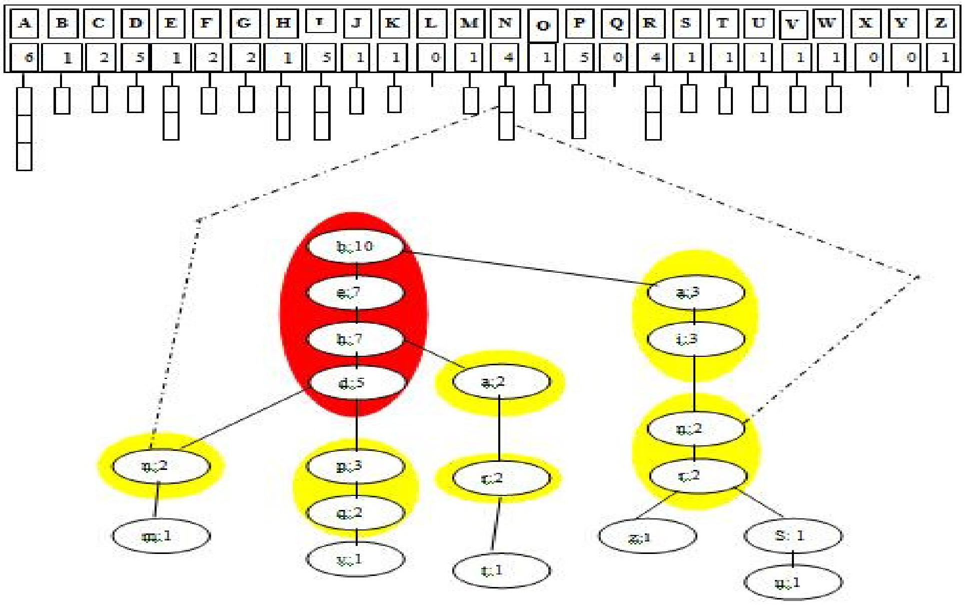


Figure 2 WPs-Mine Hash indexed tree for the example dataset WPs tree

2.2 WPs-Mine Data Access Methods

Three data access methods are devised to load from the WPs-Mine index the following projections of the original database: 1) Frequent WebPages-Tree to support projection-based algorithms (e.g., FP-growth [8]). 2) Support-based projection, to support level based (e.g., APRIORI [7]), and array-based (e.g., LCM v.2 [10]) algorithms. 3) web pages-based projection, to load all transactions where a specific webpage occurs, enabling constraint enforcement during the extraction process. The three access methods are described in the following sections.

2.2.1 Construction of Frequent WebPages-Tree:

From the relation R, the frequency of each web page is counted for a given session threshold time. The web pages are sorted in order based on its frequency but preceding in lexicographical order. In the WPs-Tree web pages are sorted by descending support lexicographical order as represented by WPs-Tree. This is represented as a prefix tree.

In the example data set, item p appears in two nodes, i.e., [p:3] and [p:2]. The access method reads two prefix paths for p, i.e., [p : 3 ->d :5 -> h : 7 !->e : 7 -> b : 10]and [p : 2 !->i : 2 !->h : 3 ->e:3] Each sub path is normalized to p node support. For example, the first prefix path, once normalized to [p:3], is [p : 3 ->d :3 -> h : 3 -> e : 3 !->b : 3]

2.2.2 Support-Based division of WPs-Tree:

The support-based projection of relation R contains all transactions in R intersected with the web pages which are frequent with respect to a given support threshold (Min Sup). The WPs-Tree paths completely represent the transactions. Web pages are sorted by decreasing support along the paths. Starting from a root node, the WPs-Tree is visited depth-first by following the node child pointer. The visit ends when a node with an Un-frequent item or a node with no children is reached.

The WPs-Tree is partitioned into three layers based on given minimum support threshold. The web pages whose support is greater than or equal to given minimum threshold is considered to be belonging to Excellent layer. The web pages whose support is greater than 1 and less than given minimum support belong to weak layer as shown in Fig. 1.

2.2.3 WPs-Hash-Table Tree

Log File of server contains information about how many visitors visited various web pages of web site. Given a session threshold, the frequency of each web page is counted and each page's count is stored in hash table. The hash table of 26 buckets [A-Z] is created. Each bucket stores the information about the frequency of each web page. Each bucket holds the physical location of each web page occurrences in the WPs-tree. Linked list with various nodes is attached for each bucket which holds the addresses of the occurrences of web pages in the WPs-Tree. The WebPages-Hash index (WPs-H index) table structure allows reading selected WPs-Tree portions during the extraction task. For each web page, it stores the physical locations of all page occurrences in the WPs-Tree.

2.3 WPs-Mine storage procedure

The organization of the WPs-Mine index is designed to minimize the cost of reading the data needed for the current extraction process. However, fetching a given record requires loading the entire disk block where the record is stored The WPs-Tree physical organization is based on the following correlation types:

i) Intra transaction correlation:. Web Pages appearing in a same transaction are thus intrinsically correlated. To minimize the number of read blocks, each WPs-Tree path should be partitioned into a block.

ii) Extra transaction correlation: In some transactions, set of web pages accessed may be same and some other pages accessed may be different, so block can be formed for common web pages accesses and separate block can be made for remaining web pages access.

2.3.1 WPS-Tree Layers

TheWPS-Tree is partitioned in three layers based on the node access frequency during the extraction processes. 1) the node level in the WPs-Tree, i.e., its distance from the root,2) the number of paths including it, represented by the node support, and 3) the support of its item.. The three layers are shown in Fig. 2a for the example WPs-Tree**.**

Excellent layer: This layer includes web pages that are very frequently accessed during the mining process. These nodes are located in the upper levels of the WPs-Tree. These web pages are most important pages as these are frequently accessed.

Medium Layer: This layer includes nodes that are quite frequently accessed during the mining process. This layer contains web pages which are frequently accessed during web site visits.
Weak layer: This layer includes the nodes corresponding to rather low support items, which are rarely accessed during the mining process. The web pages in this layer must be paid more attention to modify the content as these web pages are rarely accessed by web users.

3. Web Page set Mining

Web Page set mining has two sequential steps: 1) the needed index data is stored and 2) web page set extraction takes place on stored data.

Frequent Web Pages set Extraction

This section describes how frequent web pages set extraction takes place on the WPs-Mine index. We present two approaches, denoted as FP-based and LCM-based algorithms, which are an adaptation of the FP-Growth algorithm [3][9] and LCM v.2 [10]algorithm.

FP-based algorithm:

The FP-growth algorithm stores the data in a prefix-tree structure called FP-tree. First, it computes web page support. Then, for each transaction, it stores in the FP-tree its subset including frequent web pages. Web pages are considered one by one. For each web page, extraction takes place on the frequent-web page database, which is generated from the original FP-tree and represented in a FP-tree based structure.

4. EXPERIMENTAL RESULTS

The validation is done on both dense and sparse data distributions. We report the experiments on these parameters. The parameters include transaction and item cardinality, average transaction size (AvgTrSz),and data set size) as shown in Table 1. Connect is dense and medium-size data sets. Kosarak [10] is a Large and sparse data set.

4.1 Index Creation and Structure:

Table 2 reports both WPs-Tree and WPs-Hash index table tree size for the two data sets. The overall WPs-Mine index size is obtained by summing both contributions. The WPs-Mine indices have been created with the default value Kavg ¼ 1:2.

Furthermore, the Connect and Kosarak, and data sets have been created with KSup ¼ 0, while large synthetic data sets with KSup ¼ 0:05. In sparse data sets (e.g., Kosarak), where data are weakly correlated, data compression is low and storing the WPs-Tree requires more disk blocks.

Table 1 also shows the index creation time, which is mainly due to path correlation analysis and storage of the index paths on disk. The first factor depends on the number of WPs-Tree paths.

Table 2. Data Set Characteristics and Corresponding Indices

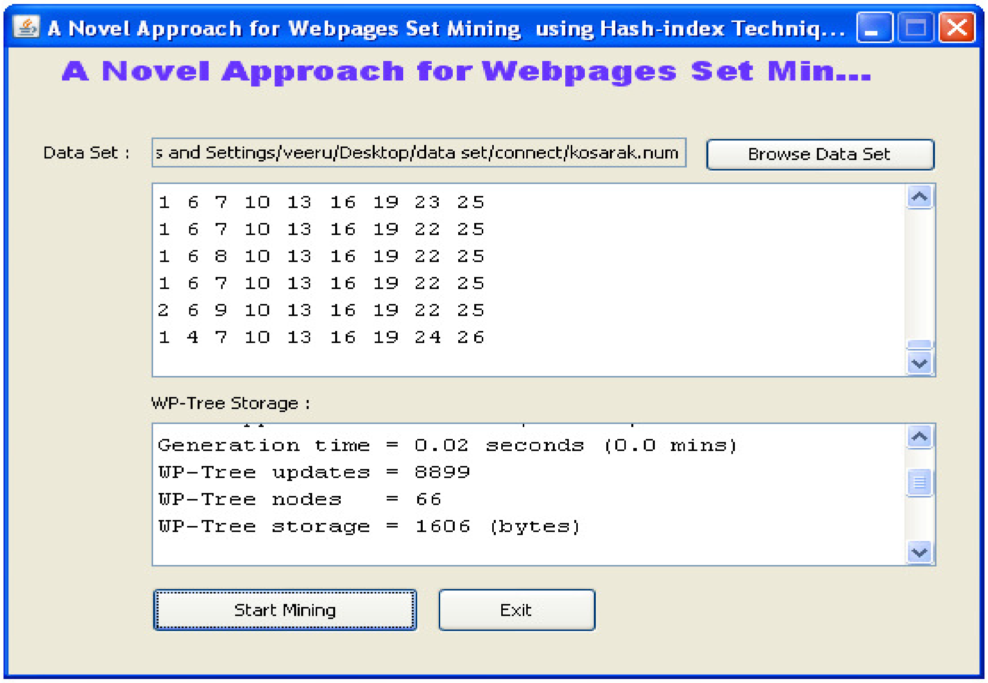
Dataset	Transactions	Dataset Items	AvtrSz	Size(KB)	WPs-Tree (Records)	WPs-Hash -indexed tree (Records)	Time (sec)
CONNECT	1200	38	7	8563	269	722	0.59
KOSARAK	1600	20	7.9	9564	291	729	0.61

4.2 Frequent Web Pages Set Extraction Performance:

The WPs-Mine structure is independent of the extraction algorithm. To validate its generality, we compared the FP-based and LCM-based algorithms with three very effective state-of-the-art algorithms accessing data on flat file.

Figure 3 generates statistics about Konark dataset containing how many WP-Tree nodes are created, how much storage required and how many WP-Tree updates. Figure 4 produces Minimum support, Number of records, Number of columns and WP-Tree updates. Figure 5 shows what is the storage required by WP-Hash-index Tree along with generation time for the tree. Similarly Figure 6 generates statistics about Connect Dataset.

Figure 8 compares the FP-based algorithm with the FP-growth algorithms [3] on flat file, all characterized by a similar extraction approach. For real data sets (Connect, and Kosarak), differences in CPU time between the FP-based and the Prefix-Tree algorithms are not visible for high supports, while for low supports the FP-based approach always performs better than Prefix-Tree. Comparison is made between connect and Kosarak dataset. In connect data set WPs-Hash index Tree 3 times faster than WPs-tree.so WPs-Hash index Tree reduced the search time and I/O cost. In Kosarak data set WPs-Hash index Tree 3 times faster than WPs-tree.so WPs-Hash index Tree reduced the search time and I/O cost. This effect is particularly relevant for low supports, because representing in memory a large portion of the data set may significantly reduce the space for the extraction task, hence causing more memory swaps. Figure 9 displays run Time Comparison Between (a)Connect and (b) Kosarak dataset


A Novel Approach for Webpages Set Mining using Hash-index Techniq...
A Novel Approach for Webpages Set Min...
Browse Data Set
Start Mining
Exit

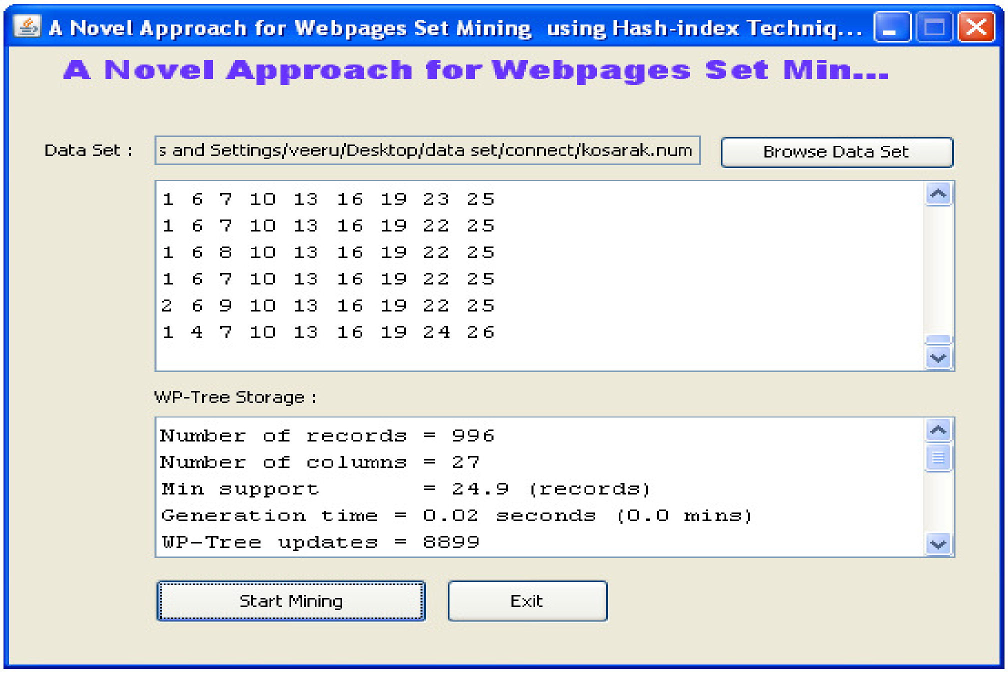
Data Set : s and Settings/veeru/Desktop/data set/connect/kosarak.num

```
1  6  7  10  13  16  19  23  25
1  6  7  10  13  16  19  22  25
1  6  8  10  13  16  19  22  25
1  6  7  10  13  16  19  22  25
2  6  9  10  13  16  19  22  25
1  4  7  10  13  16  19  24  26
```

WP-Tree Storage :

```
Generation time = 0.02 seconds (0.0 mins)
WP-Tree updates = 8899
WP-Tree nodes   = 66
WP-Tree storage = 1606 (bytes)
```

Figure 3. Kosark statistics


A Novel Approach for Webpages Set Mining using Hash-index Techniq...
A Novel Approach for Webpages Set Min...
Browse Data Set
Start Mining
Exit

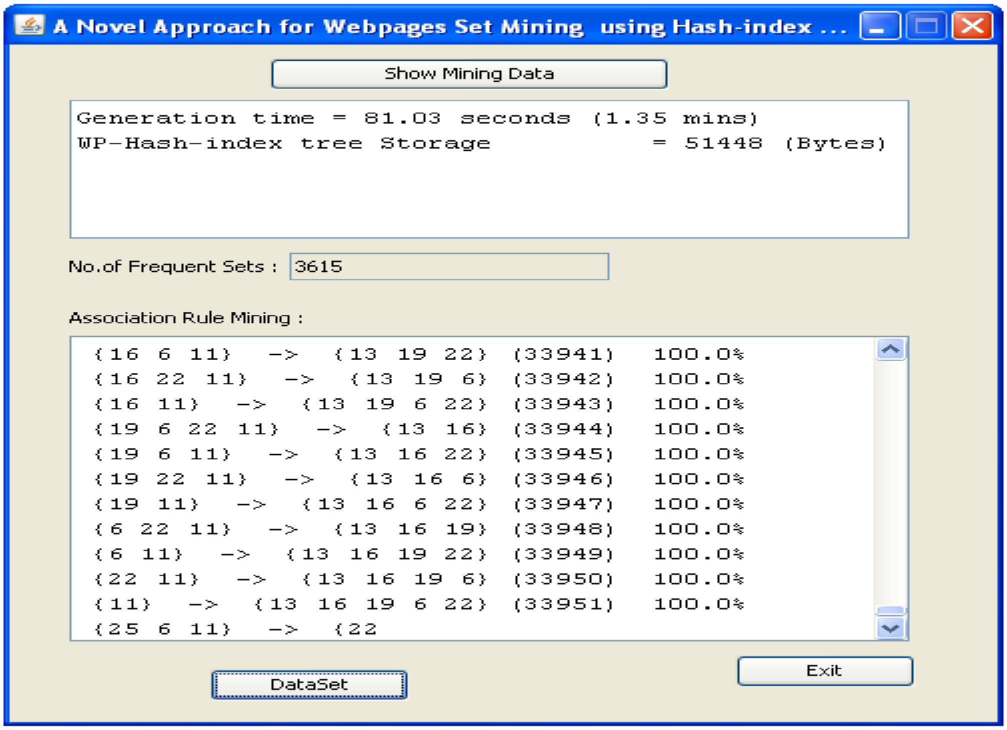
Data Set : s and Settings/veeru/Desktop/data set/connect/kosarak.num

```
1  6  7  10  13  16  19  23  25
1  6  7  10  13  16  19  22  25
1  6  8  10  13  16  19  22  25
1  6  7  10  13  16  19  22  25
2  6  9  10  13  16  19  22  25
1  4  7  10  13  16  19  24  26
```

WP-Tree Storage :

```
Number of records = 996
Number of columns = 27
Min support       = 24.9 (records)
Generation time = 0.02 seconds (0.0 mins)
WP-Tree updates = 8899
```

Figure 4. The Dataset Kosark statistics

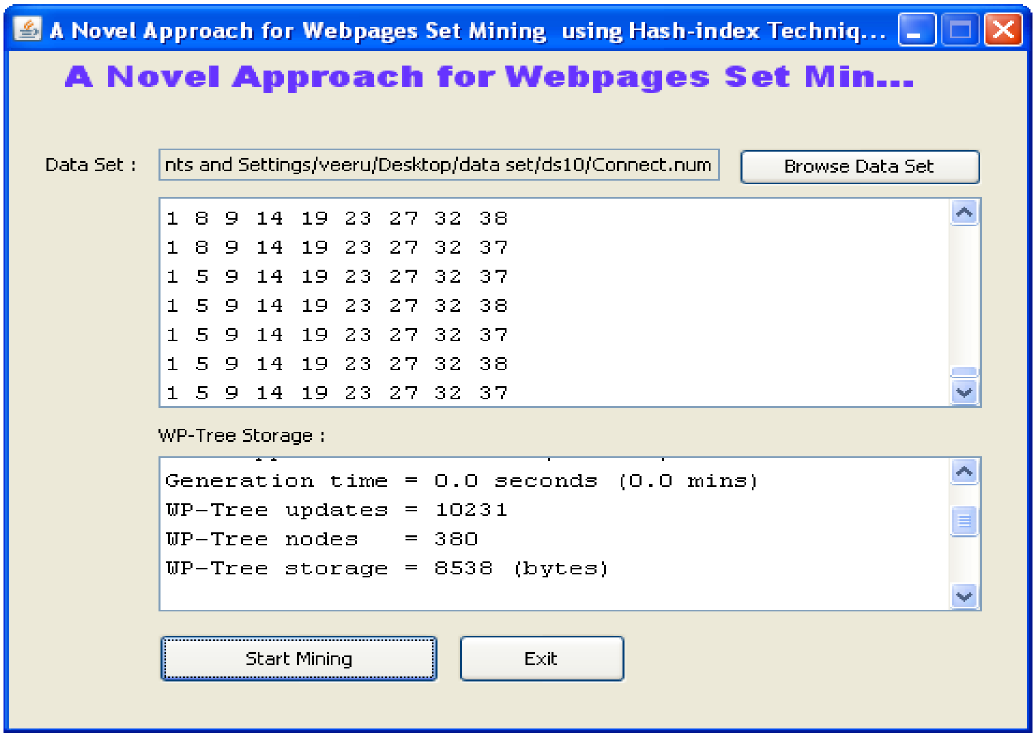

A Novel Approach for Webpages Set Mining using Hash-index ...
Show Mining Data
Association Rule Mining :
DataSet
Exit

```
Generation time = 81.03 seconds (1.35 mins)
WP-Hash-index tree Storage          = 51448 (Bytes)
```

No.of Frequent Sets : 3615

```
{16 6 11}  -> {13 19 22} (33941)  100.0%
{16 22 11}  -> {13 19 6} (33942)  100.0%
{16 11}  -> {13 19 6 22} (33943)  100.0%
{19 6 22 11}  -> {13 16} (33944)  100.0%
{19 6 11}  -> {13 16 22} (33945)  100.0%
{19 22 11}  -> {13 16 6} (33946)  100.0%
{19 11}  -> {13 16 6 22} (33947)  100.0%
{6 22 11}  -> {13 16 19} (33948)  100.0%
{6 11}  -> {13 16 19 22} (33949)  100.0%
{22 11}  -> {13 16 19 6} (33950)  100.0%
{11}  -> {13 16 19 6 22} (33951)  100.0%
{25 6 11}  -> {22
```

Figure 5. The Database Kosark statistics

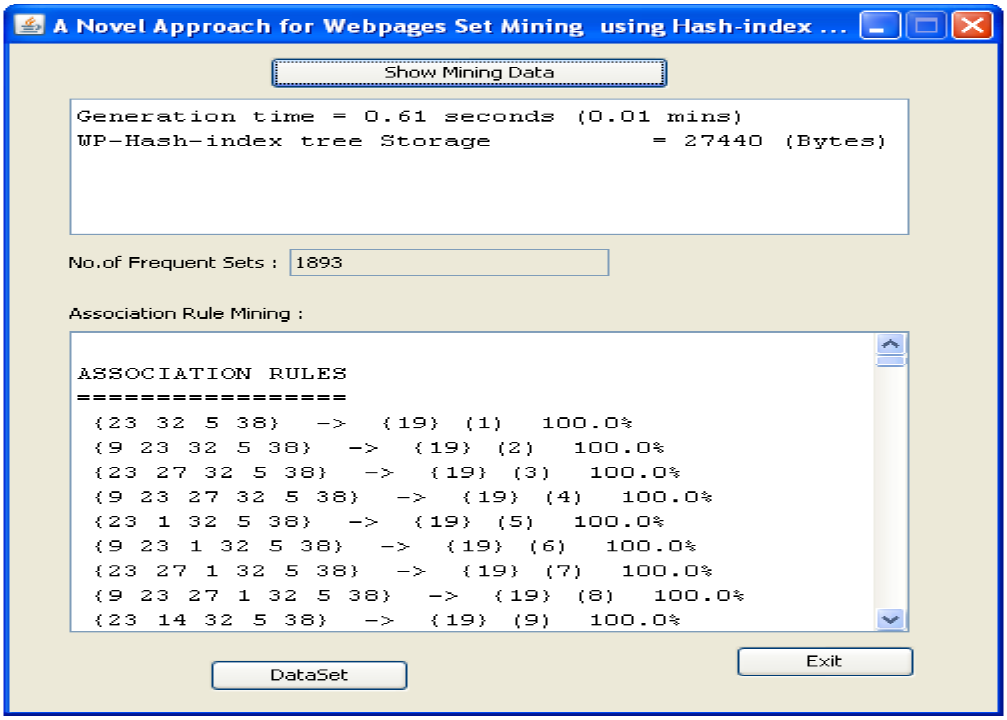

A Novel Approach for Webpages Set Mining using Hash-index Techniq...
A Novel Approach for Webpages Set Min...
Browse Data Set
WP-Tree Storage :
Start Mining
Exit

Data Set : nts and Settings/veeru/Desktop/data set/ds10/Connect.num

```
1 8 9 14 19 23 27 32 38
1 8 9 14 19 23 27 32 37
1 5 9 14 19 23 27 32 37
1 5 9 14 19 23 27 32 38
1 5 9 14 19 23 27 32 37
1 5 9 14 19 23 27 32 38
1 5 9 14 19 23 27 32 37
```

```
Generation time = 0.0 seconds (0.0 mins)
WP-Tree updates = 10231
WP-Tree nodes   = 380
WP-Tree storage = 8538 (bytes)
```

Figure 6. The Dataset Connect statistics


A Novel Approach for Webpages Set Mining using Hash-index ...
Show Mining Data
DataSet
Exit

```
Generation time = 0.61 seconds (0.01 mins)
WP-Hash-index tree Storage          = 27440 (Bytes)
```

No.of Frequent Sets : 1893

Association Rule Mining :

```
ASSOCIATION RULES
=================
 {23 32 5 38}  ->  {19} (1)  100.0%
 {9 23 32 5 38}  ->  {19} (2)  100.0%
 {23 27 32 5 38}  ->  {19} (3)  100.0%
 {9 23 27 32 5 38}  ->  {19} (4)  100.0%
 {23 1 32 5 38}  ->  {19} (5)  100.0%
 {9 23 1 32 5 38}  ->  {19} (6)  100.0%
 {23 27 1 32 5 38}  ->  {19} (7)  100.0%
 {9 23 27 1 32 5 38}  ->  {19} (8)  100.0%
 {23 14 32 5 38}  ->  {19} (9)  100.0%
```

Figure 7. The Dataset Connect statistics

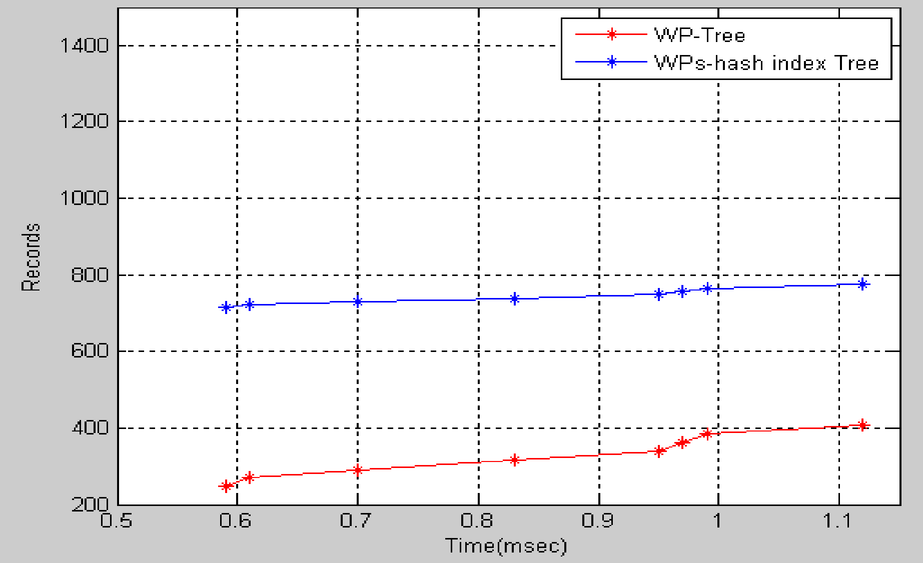


Figure 8. Frequent WPs set extraction time for the FP-based algorithm. (a) Connect

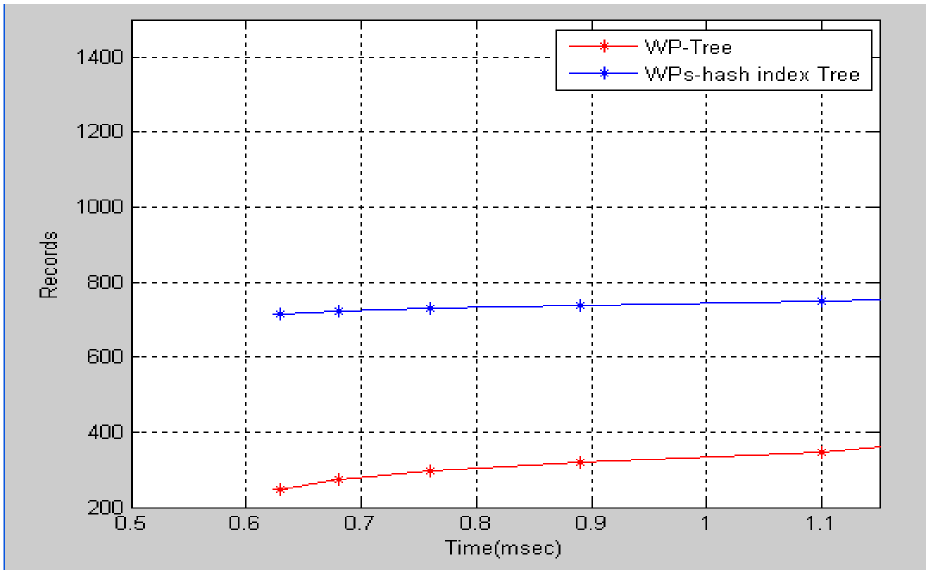


Figure .8 Frequent WPs set extraction time for the FP-based algorithm. (b) Kosarak.

4.3 Comparison

Run Time Comparison between WP-Tree and WPs-Hash index Tree for Dataset Connect and Kosarak dataset is shown in Figure 9a & Figure 9 b respectively. . In connect data set WPs-Hash index Tree 3 times faster than WPs Tree. So WPs-Hash index Tree reduced the search time and I/O cost. In Kosarak data set WPs-Hash index Tree 3 times faster than WPs-tree.so WPs-Hash index Tree reduced the search time and I/O cost.

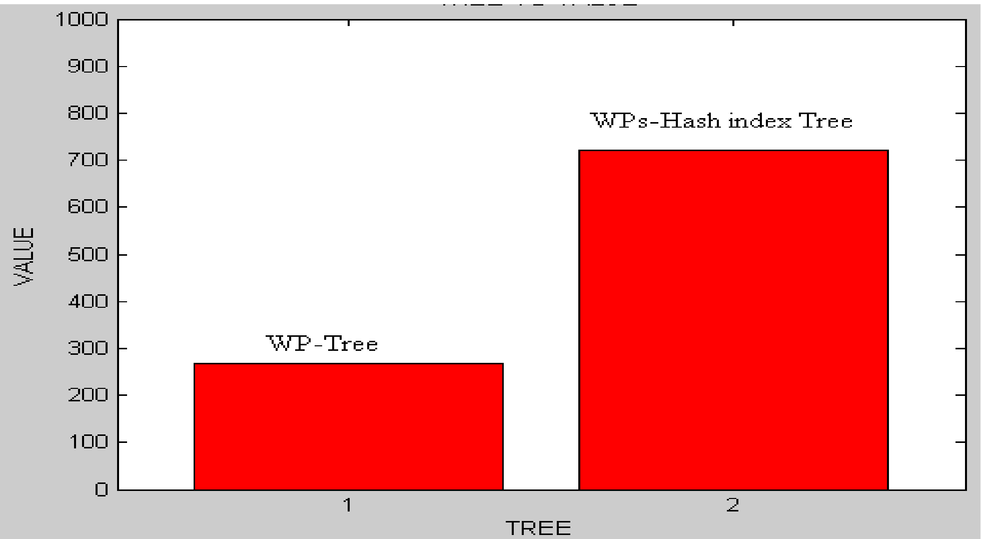


Figure 9 a. Run Time Comparison Between WPs-Tree and WPs-Hash index Tree Connect dataset

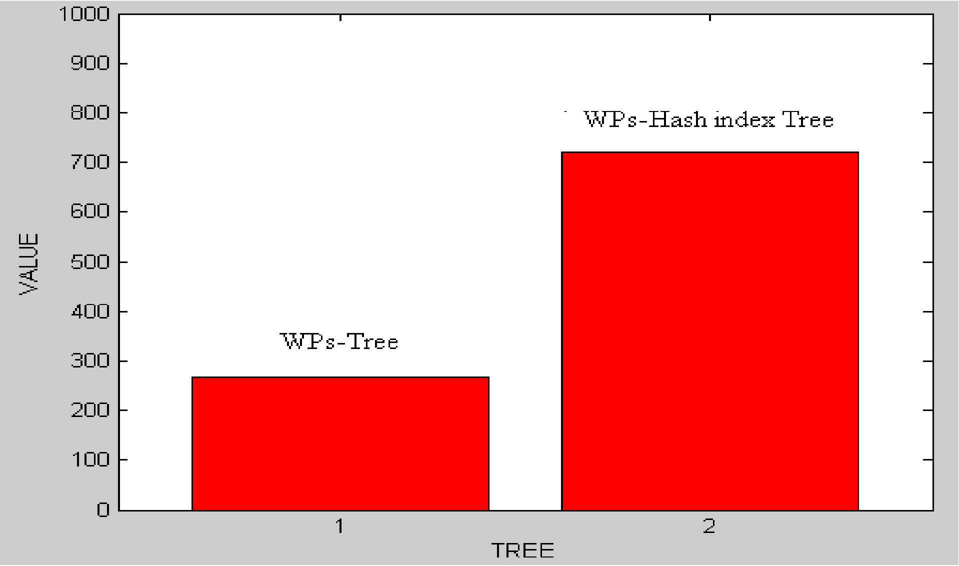


Figure 9 b. Run Time Comparison Between WPs-Tree and WPs-Hash index Tree Kosarak dataset

5. CONCLUSION

Efficient Data mining algorithms play an important role when transaction databases are very large. Since transaction databases are huge, it may not be stored in main memory. This Hash-indexing approach helps in completion of mining process through scanning the transaction database only once. This experiment showed that this algorithm is more efficient compared to other existing algorithms. The WPs-Hash index tree structure provides efficient access reducing I/O time. Further extension may be to have still efficient compact structure for different data distributions and incremental updating of index. Incremental update is feasible without re-accessing the original transactional database. Incremental update can be done considering the transaction where session was ended.

REFERENCES

[1] Agrwal, R, T. Imielinski, and A. Swami. (1993) " Mining association rules between sets of items in large databases " , in proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C. pp 207-216

[2] Li Juan, Ming De-ting (2010) "Research of an Association Rule Mining Algorithm Based on FP Tree" 978-1-4244-6585-9/10 IEEE

[3] Silverstein, C., S. Brin, and R. Motwani(1998) "Beyond market baskets: Generalizing association rules to dependence rules. Data Mining and Knowledge Discovery", pp 39–68,.

[4] Aggarawal, C. and P. Yu.(1998) "A new framework for itemset generation. In Proceedings of the PODS Conference, Seattle, WA, USA, June 1998" pp 18–24.

[5] Han, J., Pei, J., & Yin, Y, (2000), Mining frequent patterns Candidate generation. In Proc. 2000 ACM-SIGMOD Int. Management of Data (SIGMOD'00), Dallas, TX.

[6] Kotásek, P. & Zendulka J.,(2000) "Comparison of Three Mining Algorithms for Association Rules. Proc. of 34th Spring Int. Conf. on Modeling and Simulation of Systems", (MOSIS'2000), Workshop Proceedings Information Systems Modeling (ISM'2000), pp. 85-90.

[7] R. Agrawal and R. Srikant,(1994) "Fast Algorithm for Mining Association Rules," Proc. 20th Int'l Conf. Very Large Data Bases (VLDB '94). .

[8] J. Han, J. Pei, and Y. Yin, (200) "Mining Frequent Patterns without Candidate Generation," Proc. ACM SIGMOD.

[9] Elena Baralis, Tania cerquitelli, and Silvia Chiusano, (2009) "IMine: Index Support for Item Set Mining; ,IEEE Transactions on Knowledge and data engineering, Vol 21, No. 4, pp 493-506.

[10] T. Uno, M. Kiyomi, and H. Arimura, (2004) "LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets,"Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations (FIMI).

Authors

Mrs Geeta R.B, received her Master's degree from Walchand Engineering college, Kolhapur University, Sangli. She is pursuing her PhD degree from JNTU Hyderabad. She is a working as Associate Professor and head of the department of Information Technology GMR IT,RAJAM. She is member of Computer Society of India, ISTE and IEEE. Her research areas include data mining, distributed databases and computer network. She has published various papers on journal and conference proceedings

Mr. M. Omkar pursuing his M.Tech in Computer Science & Engg from GMR IT, RAJAM affiliated to JNTU Kakinada. He is a member of IEEE. His areas of interest include Data mining, Computer network and Databases. He is author of many papers

Prof Shashikumar G. Totad received his Master's degree from Walchand Engineering college, KolhapurUniversity, Sangli. He is pursuing her PhD degree from JNTU Hyderabad. He is a working as a Professor and head of the department of Computer Science & Engg GMR IT,RAJAM. He is member of Computer Society of India and ISTE. His research areas include data mining, mobile agents, data structures and design and analysis of algorithms. He has published various papers in journals and conference proceedings.

Dr. Prasad Reddy received the PhD degree from Andhra University, Vizag Andhra Pradesh, India. He is currently a Rector of Andhra University. He is specialized in Enterprise widecomputing, XML based object models and scalable web applications. His research areas include Soft computing,Knowledge discovery from databases, Image processing, Number theory and cryptosystems. He has published numerous papers National/International Journals andconference proceedings. He is a Fellow, Institution of Engineers, INDIA,member, International Association of Engineers, member,Indian Science Congress.