ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

Improved Maximal Length Frequent Item Set Mining

P.C.S.Nagendra setty, D. Haritha, Vedula Venkateswara Rao

Abstract*— Association rule mining is one of the most important techniques in data mining. Which wide range of applications It aims it searching for intersecting relationships among items in large data sets and discovers association rules. The important of association rule mining is increasing with the demand of finding frequent patterns from large data sources. The exploitation of frequent item set has been restricted by the large number of generated frequent item set and high computational cost in real world applications. To avoid these problems we can use maximum length frequent item sets in generating association rules. The maximum length frequent item sets can be efficiently discovered on very large data sets. At present in research we have LFIMiner algorithm and MaxLFI algorithm to generate maximum length frequent item sets. Here we are proposing a new algorithm called FPMAX for generating maximum length frequent item sets that uses lattice graph data structure.

***Index Terms*— Data Mining, Association Rule Mining, Maximal Length Frequent Item Sets, Lfiminer, Maxlfi Algorithm, FPMAX Algorithm.**

I. INTRODUCTION

Mining frequent item sets is a fundamental and essential problem in many data mining applications such as the discovery of association rules, strong rules, correlations, multidimensional patterns, and many other important discovery tasks. The problem is formulated as follows: Given a large data base of set of items transactions, find all frequent itemsets, where a frequent item set is one that occurs in at least a user-specified percentage of the data base. Many of the proposed item set mining algorithms are a variant of Apriori [2], which employs a bottom-up, breadth first search that enumerates every single frequent item set. In many applications (especially in dense data) with long frequent patterns enumerating all possible $2^m - 2$ subsets of a m length pattern (m can easily be 30 or 40 or longer) is computationally unfeasible. Thus, there has been recent interest in mining maximal frequent patterns in these"hard" dense databases. Another recent promising direction is to mine only closed sets [9, 11]; a set is closed if it has no superset with the same frequency. Nevertheless, for some of the dense datasets we consider in this paper, even the set of all closed patterns would grow to be too large. The only recourse is to mine the maximal patterns in such domains. MAFIA uses a vertical bitmap representation for support counting and effective pruning mechanisms for searching the item set lattice [6]. The algorithm is designed to mine maximal frequent itemsets (MFI), but by changing some of the pruning tools, MAFIA can also generate all frequent itemsets (FI) and closed frequent itemsets (FCI). MAFIA assumes that the entire database (and all data structures used for the algorithm) completely fit into main memory. Since all algorithms for finding association rules, including algorithms that work with disk-resident databases, are CPU-bound, we believe that our study sheds light on some important performance bottlenecks. In a thorough experimental evaluation, we first quantify the effect of each individual pruning component on the performance of MAFIA. Because of our strong pruning mechanisms, MAFIA performs best on dense datasets where large sub trees can be removed from the search space. On shallow datasets, MAFIA is competitive though not always the fastest algorithm. On dense datasets, our results indicate that MAFIA outperforms other algorithms by a factor of three to thirty. At present, LFIMiner ALL is the fastest algorithm for mining maximum length frequent itemsets. Exploiting the optimization techniques in LFIMiner ALL algorithm, we develop the FPMax algorithm to discover maximum length frequent itemsets by adding Lattice Graph maximum length frequent itemsets to prune the search space. Experimental results on real-world datasets show that our proposed algorithm is faster than LFIMiner algorithm for mining maximum length frequent itemsets. The rest of the paper is organized as follows. Sect. II provides Literature Survey of frequent Item Sets. The System Design is presented in Sect. III. The Proposed Algorithm is presented in Sect. IV, whereas in Sect. V we discuss experimental results. Several final remarks and a brief discussion on future work conclude in Section VI.

II. LITERATURE SURVEY

A. *Preliminaries*

The problem of mining maximal frequent patterns can be formally stated as follows: Let I= $\{I_1, I2, I3\ldots I_{n\}}$ be a set of m distinct items. Let D denote a database of transactions, where each transaction has a unique identifier (tid)and contains a set of items. The set of all tids is denoted T=$\{T_1$, T2, T3,……., $T_{n\}}$. A set $X \subseteq I$ is also called an item set. An item set with k items is called a k-item set. The set t(X) $\subseteq T$, consisting of all the transaction tids which contain X as a subset, is called the tidset of X . For convenience we write an item set {A, C, W} as ACW and its tidset is {1, 3, 4, 5}. The support of an item set X, denoted σ (X)=t(X). Support is the number of transactions in which that item set occurs as a subset. An item set is frequent if its support is more than or equal to some threshold minimum support (min sup) value, i.e., if σ (X)>=min sup. We denote by the set of frequent F_K the set of frequent itemsets, and by **FI** the set of all frequent itemsets. A frequent item set is called maximal if

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

it is not a subset of any other frequent item set. The set of all maximal frequent itemsets is denoted as **MFI**. Given a user specified *min sup* value our goal is to efficiently enumerate all patterns in **MFI**. The Figure1 shows an example for frequent item sets.

TID	Items	Frequent Itemsets Min_Sup = 3 trans	Frequent Itemsets Min_Sup = 2 trans	Itemset Size	Maximal Itemsets Min_Sup=3 trans	Maximal Itemsets Min_Sup = 2 trans
1	ACTW	A, C, D, T, W	A, C, D, T, W	1		
2	CDW	AC, AT, AW, CD, CT, CW, DW, TW	AC, AD, AT, AW, CD, CT, CW, DT, DW, TW	2		
3	ACTW					
4	ACDW	ACT, ACW, ATW,CTW, CDW,	ACD, ACT, ACW, ADW, ATW, CDT, CDW, CTW	3	CDW	CDT
5	ACDTW					
6	CDT					
		ACTW	ACDW, ACTW	4	ACTW	ACDW, ACTW

Fig 1 Mining Frequent Item sets

B. *GenMax for Maximal Itemsets*

There are two main ingredients to develop an efficient MFI algorithm. The first is the set of techniques used to remove entire branches of the search space, and the second is the representation used to perform fast frequency computations. We will describe below how GenMax extends the basic backtracking routine for FI, and then the progressive focusing and diffset propagation techniques it uses for fast maximality and frequency checking. The basic MFI enumeration code used in GenMax is a straightforward extension of FI-backtrack. The main addition is the superset checking to eliminate non-maximal itemsets, as shown in Figure 2.

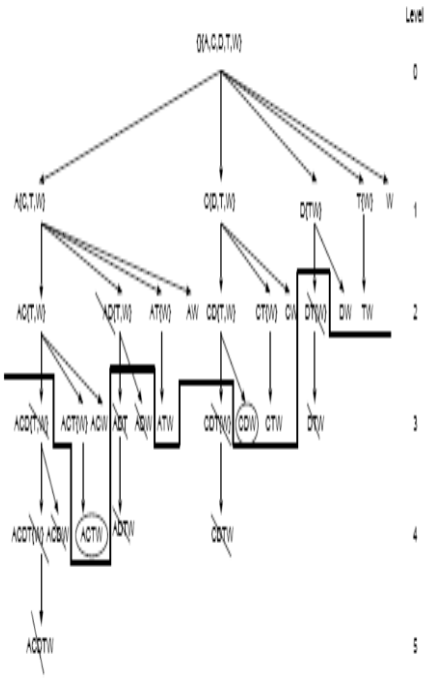

Fig 2 GenMax Algorithm.

C. *Search Space Pruning*

MAFIA uses the lexicographic subset tree originally presented by Rymon [9] and adopted by both Agarwal [3] and Bayardo [4]. The item set identifying each node will be referred to as the node's head, while possible extensions of the node are called the tail. In a pure depth-first traversal of the tree, the tail contains all items lexicographically larger than any element of the head. With a dynamic reordering scheme, the tail contains only the frequent extensions of the current node. Notice that all items that can appear in a sub tree are contained in the sub tree root's head union tail (H ∪T), a set formed by combining all elements of the head and tail. In the simplest item set traversal, we traverse the lexicographic tree in pure depth-first order. At each node n, each element in the node's tail is generated and counted as a 1-extension. If the support of {n's head}∪ {1-extension} is less than min_sup, then we can stop by the Apriori principle, since any item set from that possible 1-extension would have an infrequent subset. For each candidate item set, we need to check if a superset of the candidate item set is already in the MFI. If no superset exists, then we add the candidate item set to the MFI. It is important to note that with the depth-first traversal, itemsets already inserted into the MFI will be lexicographically ordered earlier.

D. *Parallel Partitioning Pruning*

One method of pruning involves comparing the transaction sets of each parent/child pair. Let x be a node n's head and y be an element in n's tail. If t(x) ⊆ t(y), then any transaction containing x, also contains y. Since we only want the maximal frequent itemsets, it is not necessary to count itemsets containing x and not y. Therefore, we can move item y from the node's tail to the node's head.

III. System Design

In this section we describe the System Design to implement the system. The overall system design of Item Set Mining for Generating Maximal Frequent Item Sets is described in Figure 1. Item Set Mining for generating Maximal Item sets is composed of 5 layers.

1) The User Interface Layer: It is responsible for interaction with the user and various calls to various graphical and visualization utilities. This Module provides an Interface for each user to invoke with the system and to execute Queries for generating Maximal Frequent Item Sets (Frequent Item Patterns). It provides the following services

1.1) To design interface for LFI Miner algorithm to generate Maximal frequent patterns.

The Interface receives parameters Minimum support and Minimum confidence called Interesting measures(interesting constraints) and produces frequent patterns as output.

1.2) to design interface for FPMax algorithm to generate frequent patterns. The Interface receives parameters Minimum support and Minimum confidence called Interesting measures (interesting constraints) and produces frequent patterns as output.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

1.3) An Interface to displays the performance results of LFI Miner and FPMax algorithms.

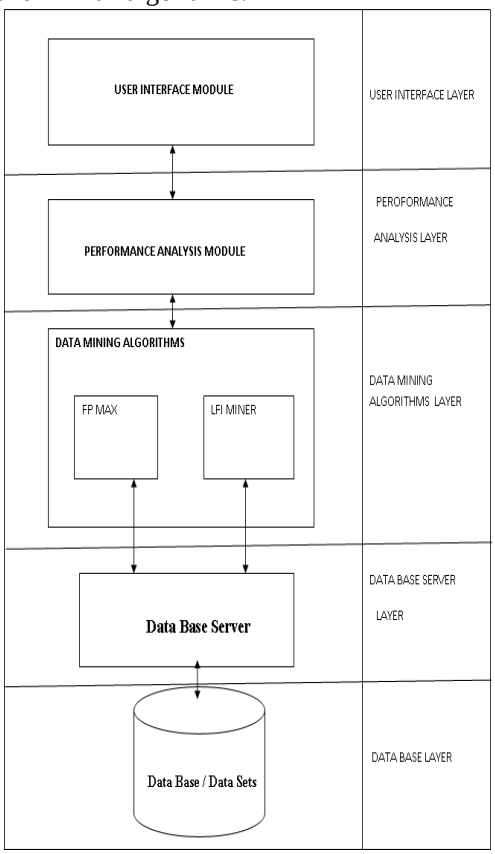

Fig 3. System Design

2). Data Mining Algorithms Layer: This is the main module connecting all system components. In this module there are set of Data Mining Algorithms to execute Query and generate frequent Item Sets. Here there are two algorithms to generate frequent item sets for finding Association rules among them.

1) LFI Miner Algorithm 2) FPMax Algorithm

2.1) LFI Miner **Algorithm:**

It receives set of transactional data items from relational data model and two interesting measures Min Support,

Min Confidence as input, and then it constructs an FPTree , performs pruning search space for generates frequent item sets.

2.2) FPMax algorithm:

This is Frequent Maximal Item Set Algorithm based on FP Tree. It receives set of transactional data items from relational data model, two interesting measures Min Support, Min Confidence and then generates Frequent Item Sets with the help of FPTree. During the process of generating Frequent Item Sets, it uses array based structure than tree structure.

3) Performance Analysis Layer:

This Module mainly focuses on analyzing the performance of LFIMiner and FPMax algorithms by comparing the time and space values for different support values.

4) Data Base Server Layer: MSACCESS relational data base. Contains data base of transactional data items. Receives Queries from the user interface design and transfers results from transactional data base.

5) Data Base Layer: Currently available in the proposed system are Transactional data about market basket analysis.

IV. METHODOLOGY

The FPMax, a variation of the FP-growth method, for mining maximal frequent item sets. Since FPMax is a depth-first algorithm, a frequent item set can be a subset only of an already discovered MFI. In FPMax we introduced a global data structure, the *Maximal Frequent Item set tree* (MFI-tree), to keep the track of MFI's. A newly discovered frequent item set is inserted into the MFI-tree, unless it is a subset of an item set already in the tree. However, for large datasets, the MFI-tree will be quite large, and sometimes one item set needs thousands of comparisons for subset testing. Inspired by the way subset checking is done in [4], in FPMax*, we still use the MFItree structure, but for each conditional FP-tree TX, a small MFI-treeMX is created. The treeMX will contain all maximal item sets in the conditional pattern base of X. To see if a local MFI Y generated from a conditional FP-tree TX is maximal, we only need to compare Y with the item sets in MX. This achieves a significant speedup of FPMax. Each MFI-tree is associated with a particular FP-tree. Children of the root of the MFI-tree are item prefix sub trees. In an MFI-tree, each node in the sub tree has three fields: item-name, level and node-link. The level-field will be useful for subset testing. All nodes with same item-name are linked together, as in an FP-tree. The MFI-tree also has a header table. However, unlike the header table in an MFI-tree is constructed based on the item order in the table of the FP-tree it is associated with. Each entry in the header table consists of two fields, item-name and head of a linked list. The head points to the first node with the same item-name in the MFI-tree. FP-tree, which is constructed from traversing the previous FP-tree or using the associated array. FPMax is based on lattice graph structure. A formal context is a triple K=(U,D,R) , where *U* is a finite set of elements called objects, *D* is a finite set of elements called attributes and *R* is a binary relation between *U* and *D*. For arbitrary $U \in x \bullet$, and $D \in y$, (X,Y) $\in R$, ((also denoted by *xRy*) if the object *x* has the attribute *y* .

Now we define two mappings f: $P(U) \rightarrow P(D)$ and g : $P(D) \rightarrow P(U)$ as follows :

$f(X) = \{y \in D / \forall\ x \in X, xRy\}$, *for* $X \in P(U)$

$g(Y) = \{ x \in D / \forall\ y \in Y, xRy\}$, *for* $Y \in P(D)$

Where P(*U*) and P(*D*) are the power sets of *U* and *D* respectively. Then, the maps *f* and *g* form a Galois connection

Between power sets P(*U*) and P (*D*) .The Extraction of Maximal Item Sets consists of following steps.

1) Generating item sets.
2) Generating frequent item sets.
3) Construct cyclic graph(lattice graph) of frequent and Item sets.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

4) Compare the support of every frequent Item set with its super set using closure property and ppc.

The following shows FPMax algorithm.

Procedure *FPMax* (T, M)

1. Input: T, an FP-tree
2. M, the MFI-tree for T.base
3. Output: Updated M
4. Method:
5. **if** T only contains a single path P
6. insert P into M
7. **else for each** i in T. header
8. set Y = T.base U {i};
9. **if** T.array is not NULL
10. *Tail*={frequent items for i in T:array}
11. **else**
12. *Tail*={frequent items in i's conditional pattern base}
13. sort *Tail* in decreasing order of the items' counts
14. **if not** *subset_checking*(Y U Tail, M)
15. construct Y^1 's conditional FP-tree TY and its array AY ;
16. initialize Y^1's conditional MFI-tree MY ;
17. call *FPmax*(TY ,MY);
18. merge MY with M

The following shows an example for constructing Lattice graph and generating association rules. Table 1 is a given transaction database, and Table 2 is corresponding formal context. Fig. 4 is closed label lattice based on formal context in Table 2.Generation set of implication rules extracted from the lattice is as follows:

{ f→*e* , *h* →*d* , *d*→ *h* , *i*→ *a* , *g*→ *b* , *f*→ *b* , *cd*→e , *ed*→c , *ch*→e , *eh*→c , *ca*→g , *ga*→ci , *gc*→a, *gi*→a , *ei*→f , *fi*→ec , *ei*→ *c* , *di*→a , *hi*→a } , there are 19 rules. For example, the rules generated from concept (2 {gc,ci,ga,ca}, acgi) are { *ca*→g , *ga*→ci , *gc*→a , *gi*→a }, then we can produce the following rules: { *gca*→ i , *gia*→c *gci*→a , *gi*→ac , *gc*→ai , *ca*→gi }, so the rules that we extracted are more simpler. From above we can see that generation set is only a smaller set of rules. Other rules could be generated from generation set if needed.

Table1 Transaction Database

TID	T
0	b, e, f, g
1	c, d, e, h
2	$a, c, g\, i$
3	c, e, f, i
4	a, d, h, i

Table 2 Formal Context

	a	b	c	d	e	f	g	h	i
0	0	1	0	0	1	1	1	0	0
1	0	0	1	1	1	0	0	1	0
2	1	0	1	0	0	0	1	0	1
3	0	0	1	0	1	1	0	0	1
4	1	0	0	1	0	0	0	1	1

The following figure shows lattice grapgh.

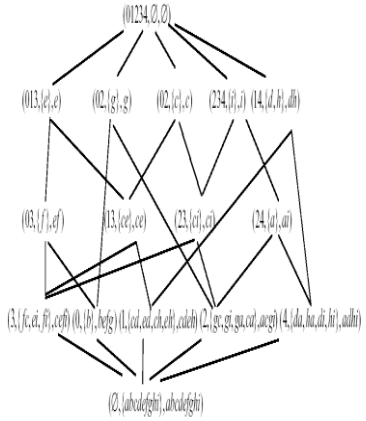

Fig 4. Lattice Graph

The time complexity of incremental building lattice in is $O(2^k \|\|L\|\|)$, in which $\|\|U\|\|$ is the number of objects and *k* is the maximum value of the number of attributes $\|f(\{x\})\|$ ($x \in U$), . $\|L\| = 2^k \|\|U\|\|$ is the number of nodes in lattice. In the paper, because the computing closed label of every new added node or of generate node need scan all its father nodes. The maximum number of father nodes is $2^k - 1$, so the time complexity of the algorithm is $O(2 (\|\|L\|\| 2^k + 2^k - 1) = O(2^k \|\|L\|\|)$. The number of repeat times that the algorithm of mining implication rules is the number of father nodes, so the time complication of mining generation set is $O(\|\|L(\|\|2^k - 1))= O(2 \|\| L\|\| 2^k)$. In practice, the number of father nodes is far less that the worst, and we recorded father nodes of every node, so the efficiency of mining rules will has some improvement. As for mining association rules, the scan number is even less because of pruning based on support.

V. Experimental Results

The proposed approach is validated by means of a large set of experiments addressing the following issues:

1. Performance of the Btree index creation, in terms of both creation time and index size.
2. Performance of frequent item set extraction, in terms of execution time, memory usage and I/O access time.
3. Effect of the index layered organization.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

4. Effect of the DBMS buffer cache size on hit rate
5. Scalability of the approach.

A. Frequent Item Set Extraction Performance We analyzed the runtime with various supports. Since our objective is to compare the performance of the approaches on the extraction phase. For large data sets, than FPMax based algorithm produces better performance. Since Lattice graph paths compactly represent the transactions, reading the needed data requires a lower number of I/O operations with respect to accessing the flat file representation of the data set. This benefit increases when the data set is larger and more correlated.

B. Memory Consumption. FPMax algorithms store in memory all data needed during the extraction process, these algorithms show the same peak main memory behavior. For both LFIMiner algorithm and FPMax based algorithm average and peak memory are close. Thus, memory consumption is rather constant in time. The difference in both average and total memory is more significant between the LFIMiner based and the FPMax algorithms. Since the buffer cache size is kept constant, FPMax-based memory requirements are rather stable with decreasing support thresholds.

C. Scalability The scalability of the LFIMiner based and FPMax based algorithms has been analyzed by varying 1) the number of transactions and 2) the pattern length. Figures 5 and 6 describe generating Maximal Frequent Item sets Using LFIMiner and FPMax.

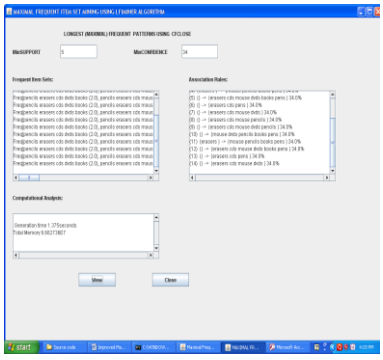

Fig 5 Generating Maximal Frequent Item Sets by LFIMiner

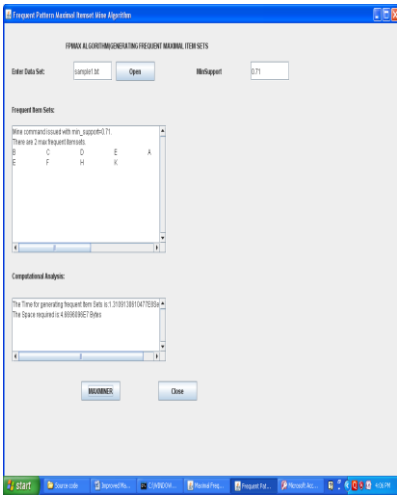

Fig 6 Generating Maximal Frequent Item Sets by FPMax

The Following Figures shows Comparison of time complexity for LFIMiner and FPMax algorithms.

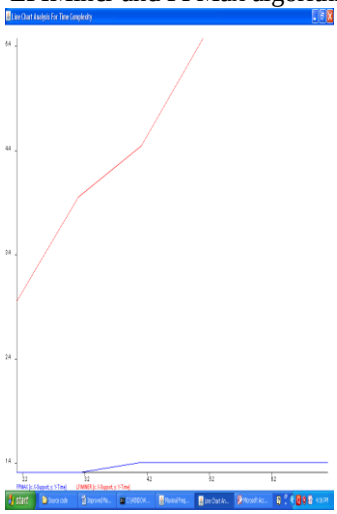

Fig 7 Time Complexity Analysis of LFIMiner and FPMax

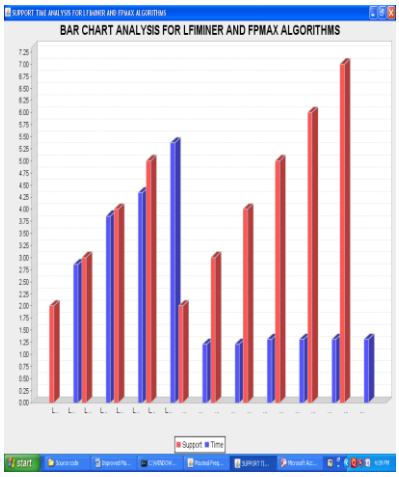

Fig 8 Time Complexity Analysis of LFIMiner and FPMax

The Following Figures shows Comparison of Space complexity for LFIMiner and FPMax algorithms.

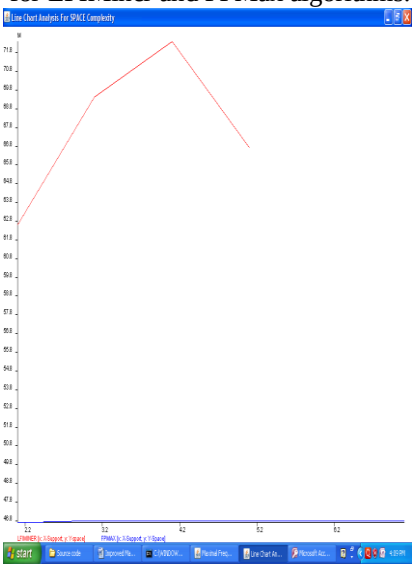

Fig 9 Space Complexity Analysis of LFIMiner and FPMax.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 2, August 2012

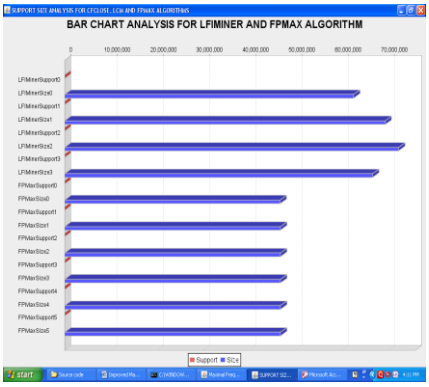

Fig 10 Space Complexity Analysis of LFIMiner and FPMax

VI. CONCLUSION

We have introduced a novel Lattice Graph based technique that generates Maximal item sets more efficiently. Our technique greatly reduces the time spent traversing Graph than traversing FP-trees, and works especially well for sparse datasets. For mining maximal frequent itemsets, we extended our earlier algorithm LFIMiner to FPMax that not only uses the graph technique, but also a new subset-testing algorithm. For the subset testing, a variation of the FP-tree, an MFItree, is used for storing all already discovered MFI's. For all of our algorithms we have presented several optimizations that further reduce their running time. Our experimental results showed that FPMax always outperforms existing algorithms.

REFERENCES

[1] Tran Anh Tai, Ngo Tuhan Phong, Nguyen Kim Anh, An Efficient Algorithm for discovering maximal frequent item sets, Proceedings of IEEE 2011 third international journal of knowledge and systems engineering, pages 62-69.

[2] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. Depth first generationof long patterns. Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 108–118, 2000.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Vary Large Data Bases, pages 487–499, 1994.

[4] R. J. Bayardo. Efficiently mining long patterns from databases. Proceedings of ACM SIGMOD International Conference on Management of Data, pages 85–93, 1998.

[5] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transaction databases. Proceedings of the 17th International Conference on Data Engineering, pages 443–452, 2001.

[6] W. Cheung and O. R. Zaiane. Incremental mining of frequent patterns without candidate generation or support constraint. Proceedings of the 7th International Database Engineering and Applications Symposium, pages 111–116, 2003. [6] K. Gouda and M. J. Zaki. Efficiently mining maximal frequent itemsets. Proceedings of the 1st IEEE International Conference on Data Mining, pages 163–170, 2001.

[7] G. Grahne and J. Zhu. High performance mining of maximal frequent itemsets. Proceeding of the 6th SIAM International Workshop on High Performance Data Mining, pages 135– 143, 2003.

[8] J. Han, J. Pei, , and Y. Yin. Mining frequent patterns without candidate generation. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pages 1–12, 2000.

[9] T. Hu, S. Y. Sung, H. Xiong, and Q.Fu. Discovery of maximum length frequent itemsets. Information Sciences: an International Journal, 178(1):69–87, January 2008.

[10] R. Mehul, "Discrete Wavelet Transform Based Multiple Watermarking Scheme", in Proceedings of the 2003 IEEE TENCON, pp. 935-938, 2003.

[11] D. Lin and Z. M. Kedem. Pincer-search: A new algorithm for discovering themaximum frequent itemset. Proceedings of the 6th International Conference on Extending Database Technology, pages 105–119, 1998.

[12] M. Wojciechowski and M. Zakrzewicz. Dataset filtering techniques in constraint-based frequent pattern mining. Proceedings of ESF ExploratoryWorkshop on Pattern Detection and Discovery, pages 77–91, 2002.

[13] R. Agarwal, T. Imielinski, and A. Swami, "Mining Associationrules between Sets of Items in Large Databases", Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD'93), Washington, D.C.,USA, pp. 2007.

[14] G. Grahne, and J.F. Zhu, "Fast Algorithms for Frequent Item setMining Using FP-Trees," IEEE Transactions on Knowledge andData Engineering, pp. 1347-1362, Oct 2005.

[15] Lisheng Ma, Huiwen Deng, "Fast Algorithm for mining Maximal frequent Item sets", First International symposium on Data, Privacy and E-Commerce, pp. 86– 91, Nov 2007.

[16] Bo Liu, Jiuhui Pan "Graph-based Algorithm For mining Maximal Frequent Item sets", Forth International conference on Fuzzy Systems and Knowledge discovery (FSKD2007).

Author's Profile

Mr. P.C.S.Nagendar Setty
Studying final M.Tech(CSE) in University college of Engineering JNTU Kakinada, Kakinada.

D.Haritha is Assistant Professor in Computer science and Engineering Department at Jawaharlal Nehru Technological University Kakinada. She has 12+ years of experience. She guided 30 M.Tech students and 15 MCA students for their project. Her research interest is on image processing, Data structures and networking. She published 4 research papers in international journals. She published 3 research papers in international conferences.

Mr. Vedula Venkateswara Rao M.Tech. (CSE), M.Tech. (IT) Associate Professor, over 14 years of experience in teaching and software industry with Sri Vasavi Engineering College and Luscent Technologies. Handled courses for B.Tech, M.Tech and M.C.A and specialized in Data Mining, Data base Systems, Machine Learning, Language processors, Operating Systems. Venkateswara Rao has more than 20 publications in various international journals and conferences including 5 IEEE publications. He guided 8 M.Tech students and 15 MCA students for their project.