Chapter 5
FOLMax-Mine – First Occurrence List Maximal Mine

A major challenge in mining frequent patterns from a large data set is the huge number of patterns generated during mining. The reason is, larger patterns contain an exponential number of smaller, frequent sub-patterns. Maximal Frequent Pattern Mining has evolved to overcome this situation. This chapter presents a new method FOLMax-Mine for the efficient mining of maximal web access patterns. It is a top-down method that uses the concept of first occurrence to reduce search space and thus to improve the efficacy of mining.

5.1 Introduction

Web access pattern mining is an application of sequence mining on web log data to generate interesting user access behavior in World Wide Web. In this chapter a new method, FOLMax-Mine, is presented for mining Maximal Web Access Patterns efficiently. It is a top-down method that uses the concept of first occurrence to reduce search space and thus improving the performance.

5.1.1 Maximal Access Pattern Mining

Mining of web access patterns generated by the users' interaction with the World Wide Web is a thrust area of research. Web access pattern mining is an application of sequence mining on web log data to generate interesting user access behavior on World Wide Web.

Huge volume of patterns generated during mining of large datasets is a major concern in any frequent pattern mining scenario. The situation becomes worse as the support threshold decreases. This is because if a pattern is frequent, each of its sub-patterns is frequent as well. In the case of lengthy patterns, the number of inclusive sub-pattern is exponential in number. The concept of Maximal Frequent Pattern Mining was proposed as a solution to this situation. A pattern α is a maximal frequent pattern in set D if α is

frequent, and there exists no super-pattern β such that $\alpha \subseteq \beta$ and β is frequent in D [Han et al., 2007].

As a web log entry is generated for each and every user access to the Web, volume of Weblog database is very high. Length of Web Access Pattern also is very large, especially at a low minimum support threshold. Because of these reasons, the number of access patterns generated in Web Access Pattern Mining is very large, making the space requirement a major concern.

Concept of Maximal Pattern Mining can be adapted to Web Log Mining to solve the issue of space requirement. This can be done by generating only Maximal Access Patterns by blocking the generation of all its sub access patterns. The set of max-patterns, though more compact, usually does not contain the complete support information regarding its sub-frequent patterns.

FOL-Mine (First Occurrence List Mine), a pattern growth based web access pattern mining algorithm, proposed in Chapter 4 was proved to be linearly scalable and more efficient than the prominent WAP-Tree based access pattern mining algorithms. The FOL-Mine algorithm uses a highly efficient linked structure, to hold the web access sequences, and employed the concept of first occurrence of symbols to grow patterns during mining. This chapter presents FOLMax-Mine algorithm, which is a modified version of FOL-Mine. FOLMax-Mine algorithm mines all and only maximal web access patterns from a file of preprocessed web access sequences.

5.1.2 Related Works

Mining max-patterns was first studied by Bayardo and MaxMiner, an Apriori-based, level-wise, breadth-first search method was proposed to find maximal patterns [Bayardo, 1998]. To reduce search space, MaxMiner performs a subset infrequency pruning and a superset frequency pruning. Though the super set pruning reduces search space and thus reduces time, the algorithm needs many passes to get all maximal patterns.

Another efficient depth first method MAFIA (Maximal Frequent Itemset Algorithm) was proposed in [Burdick, Calimlim and Gehrke, 2001] for mining maximal frequent itemsets from a transactional database. This algorithm uses vertical bitmaps to compress the transaction- id list and thus improves the previous work by a factor of three to five. But in [Gouda and Zaki, 2010] the authors conducted extensive experiments on a

variety of data sets and they showed that many times MAFIA yields a superset of maximal patterns.

M. J. Zaki and Gouda K introduced GenMax [Gouda and Zaki, 2010], a back track search based algorithm for enumerating all maximal patterns. GenMax uses a number of optimizations to quickly prune away a large portion of the subset search space. It also uses a progressive focusing technique to eliminate non-maximal itemset and employs diffset propagation for fast frequency checking. They compared their work with previous works in the field using a good set of real and synthetic datasets and claimed that performance could vary significantly depending on the dataset characteristics. But GenMax performs better than the previous algorithms on majority of datasets [Burdick, Calimlim and Gehrke, 2001].

Cheng et al. [Cheng, Wei and Zhang, 2006] proposed a method based on MFP (Maximum Frequent Pattern) for discovering maximal access pattern of web users. They introduced a WUAP-Tree (Web User Access Pattern Tree) as basic data structure for the mining process. WUAP-Tree is a variation of suffix tree with user count and shared prefix sub sequence count. WUAP-Mine is the mining algorithm proposed by the authors.

5.2 FOLMax-Mine – First Occurrence List Maximal Mine

FOL-Mine used a linked list with header node to store the first occurrences of each event in a very compact and efficient way. The count of first occurrences, which gives the total support, is held in the header node. Unnecessary generations of intermediate projected databases and tedious support counting are eliminated in FOL-Mine algorithm.

FOL-Mine uses three main procedures i) the *Main* algorithm ii) *Construct-FOL,* the algorithm to construct FOL structure and iii) the mining algorithm, FOL-Mine.

In FOLMax-Mine to mine maximal access patterns, the first two algorithms remain the same [Sections 4.3.2.1 and 4.3.2.3]. The access pattern mining algorithm FOL-Mine is modified to mine each and every maximal pattern efficiently. FOLMax-Mine, the proposed algorithm to mine maximal access pattern is given in Figure 5.1. A linked list, FOL (First Occurrence List), holds the first occurrences of a frequent event. Each node of FOL-list is a pointer to the first occurrence of an item in a web access sequence.

Head node stores the total number of occurrences and address of the first-occurrence position of the current frequent event in the first access sequence in which it occurs.

Structure of Header node and FOL-list are as described below:

1. struct htype { int supp;

   symbol item;

   folnode *next; }

2. Nodes of FOL-list is of the format

   struct folnode { wasdlist *occr;

   folnode *next };

Algorithm for Mining Maximal Pattern

FOLMax-Mine (pattern q, FOL L, η)

// q: Current Pattern

L: Current FOL list

η: Absolute Minimum Support //

1. For each ***a*** Є ∑ do

   i. La← Construct-FOL (L, ***a***)

   ii set max-flag to 1

   iii. if support of ***a*** > η

   FOLMax-Mine (*q* .***a*** , La, η);

   If max-flag=1;

   F←*F* U q;

   Set max-flag to 0;

   endif

   iv. delete La

   endfor

2. return

Figure 5.1: Algorithm FOLMax-Mine

Step 1(i) of the algorithm uses the algorithm *Construct-FOL*(L, ***a***) [Section 4.3.2.3] to build the First Occurrence List of the symbol ***a***, that stores the first occurrences of symbol ***a*** in each sequence**.** Step1 (iii) calls FOLMax-Mine recursively to mine all maximal patterns.

5.3 Performance Evaluation

5.3.1 Test Environment and Datasets

FOLMax-Mine is implemented in Microsoft visual C++ 6.0 Professional version 2002. Data sets used for the experiments are *msnbc* and T10I4D100k. Descriptions of datasets are given in Section 4.4.1. All the tests were performed on an Intel Dual Core machine with 1GB RAM and running Microsoft Windows XP.

5.3.2 Experimental Results

Experiments are conducted to evaluate the performance of the proposed method using the two popular databases. Performance of the method is evaluated by considering the execution time and number of patterns generated.

5.3.2.1 Evaluation of Execution Time

The execution time requirement of the algorithm is tested. The total execution time required by the method is measured by introducing codes in the original implementation.

Table 5.1: Execution Time at Various Supports for the Database *msnbc*

Support	Execution Time (secs)	Number of Patterns
0.01	75	127
0.009	79	152
0.008	87	194
0.007	95	238
0.006	105	292
0.005	140	421

Figure 5.2 shows the execution time variation of FOLMax-Mine with varying support on *msnbc* dataset. Table 5.1 gives the output information in tabular form. Experiment was repeated on T10I4D100 synthetic dataset and the result is shown in Table 5.2 and

Figure 5.3. The evaluative experiments reveal that the execution time increases as the minimum support threshold decreases. This is because at a lower minimum support the number of frequent element is more and thus the number of patterns also is more.

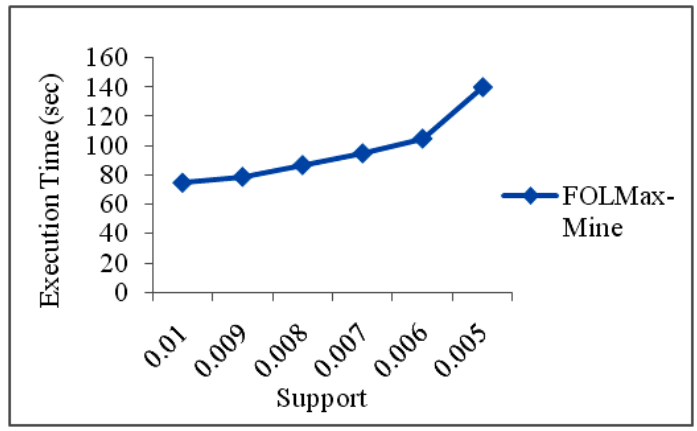

Figure 5.2: Execution Time at Various Supports for the Database *msnbc*

Table 5.2: Execution Time at Various Supports for the Database T10I4D100K

Support	Execution Time (secs)	Number of Patterns
0.04	4	10
0.035	8	40
0.03	9	50
0.025	15	107
0.02	22	155
0.015	34	236
0.01	57	377

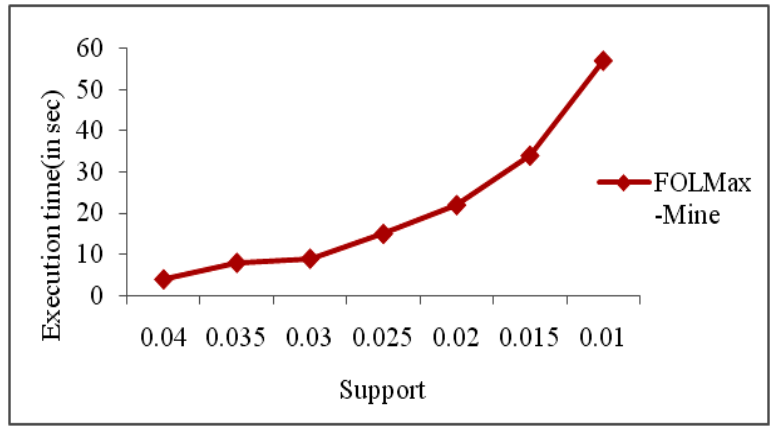

Figure 5.3: Execution Time at Various Supports for the Database T10I4D100K

5.3.2.2 Evaluation of Number of Maximal Patterns Generated

Various experiments are conducted to study the relation between execution time and number of maximal access patterns generated. Experiments are conducted using the datasets *msnbc and* T10I4D100K. Figure 5.4 gives a comparison of execution time with number of maximal patterns generated on *msnbc* dataset. Table 5.1 provides the corresponding data in tabular form. Experiment was repeated on T10I4D100 synthetic dataset and the result is shown in Table 5.2 and Figure 5.5.

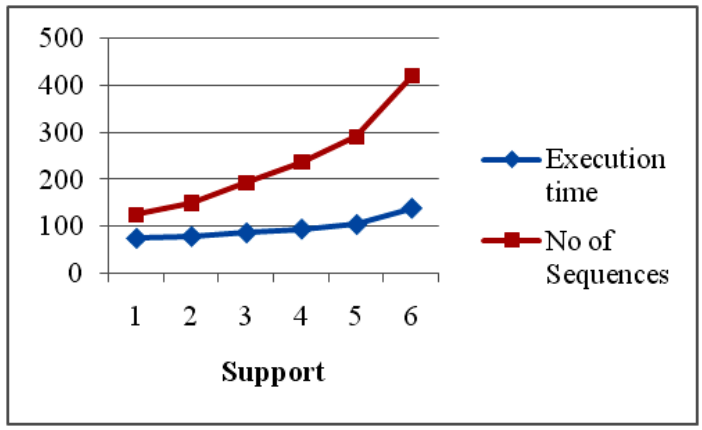

Figure 5.4: Comparison of Execution Time and Number of Maximal Access Patterns at Various Supports on *msnbc*

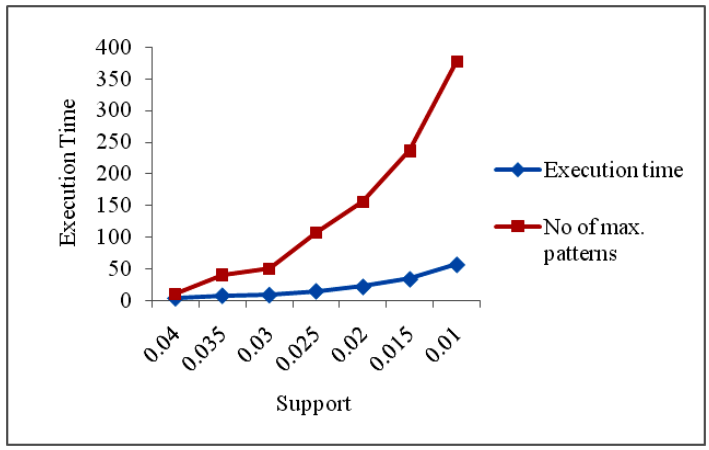

Figure 5.5: Comparison of Execution Time and Number of Maximal Access Patterns at Various Supports on T10I4D100K.

Experiments conducted to study the number of maximal access patterns and execution time requirement showed that the number of maximal patterns goes up as the support decreases. Execution time also increases, when support is decreased. But from the graphs in Figure 5.4 and Figure 5.5, it is evident that even though the number of

maximal access patterns shoots up as minimum support decreases, increase in the corresponding execution time is very minimal and under control.

5.3.2.3 Comparison between Conventional and Maximal Access Pattern Mining

Main objective of Maximal Access Pattern Mining is to reduce the space requirement of the generated set of patterns. To study the advantage of the proposed FOLMax-Mine method in this regard, experiments are conducted by running both FOL-Mine and FOLMax-Mine on the same datasets. FOL-Mine is already proved to be an efficient method for Web Access Pattern Mining. Datasets used for the experiments are T25I10D10K and T10I4D100K. The results of the tests on T25I10D10K are provided in Table 5.3 and the results of the tests conducted using T10I4D100K is in Table 5.4. Graphical representation of the resuts is given in Figure 5.6 and Figure 5.7.

Table 5.3: Number of Patterns Generated by FOL-Mine and FOLMax-Mine for T25I10D10K

Support	Number of Patterns		Difference
	FOL-Mine	FOLMax-Mine	
0.01	3300	2561	739
0.0095	6893	4521	2372
0.009	11164	7245	3919
0.0085	13370	8216	5154
0.008	15198	9220	5978

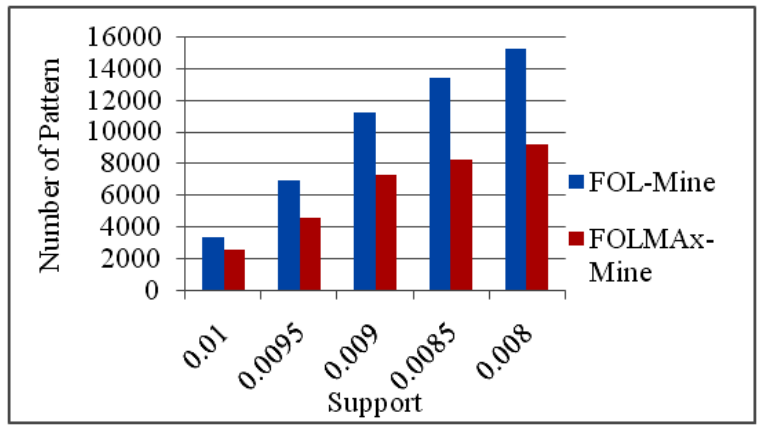

Figure 5.6: Number of Patterns Generated by FOL-Mine and FOLMax-Mine for T25I10D10K

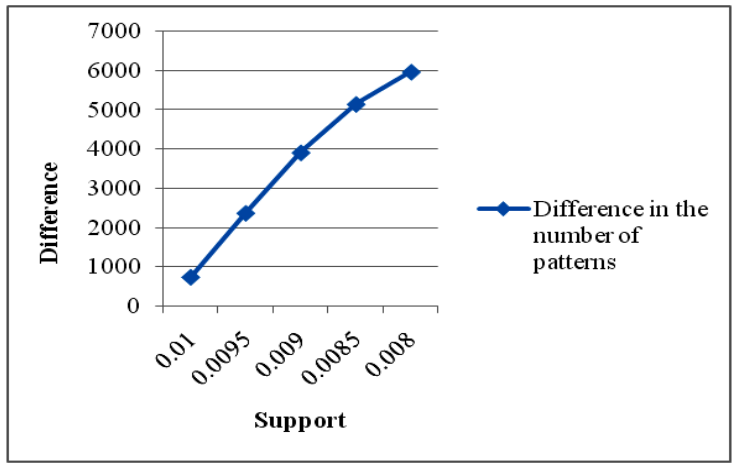

Figure 5.7: Difference in the Number of Patterns Generated by FOL-Mine and FOLMax-Mine for T25I10D10K

Table 5.4: Number of Patterns Generated by FOL-Mine and FOLMax-Mine for T10I4D100K

Support	Number of Patterns Generated		Difference in number
	FOL-Mine	FOL-MAx-Mine	
0.003	4552	2882	1670
0.0025	7703	4632	3071
0.002	13255	7653	5602
0.0015	19126	10975	8151
0.001	27532	16220	11312

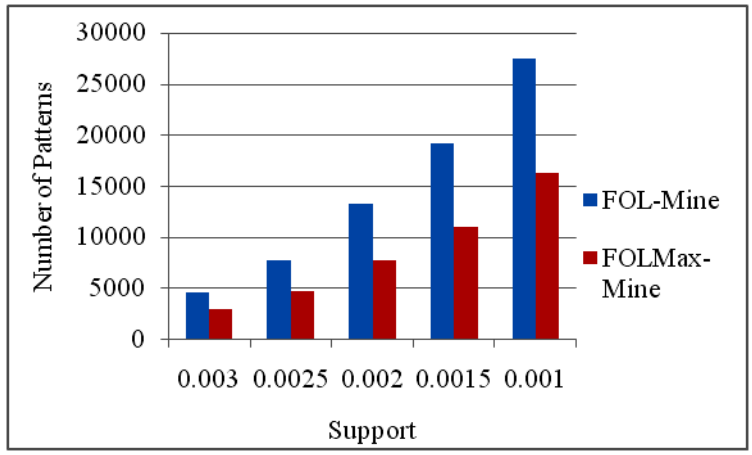

Figure 5.8: Number of Patterns Generated by FOL-Mine and FOLMax-Mine for T10I4D100K

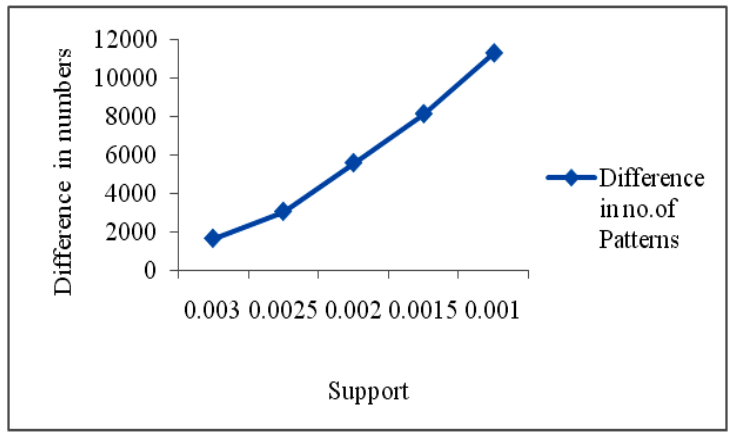

Figure 5.9: Difference in Number of Patterns Generated by FOL-Mine and FOLMax-Mine for T10I4D100K

Figure 5.6 and Figure 5.8, emphasize that as the support decreases, number of patterns generated by the proposed FOLMax-Mine also decreases. This is mainly due to the fact that at a lower support, length of generated patterns will be more and so is the number of frequent subsequences. In Maximal Pattern Mining, all frequent sub-patterns are blocked from generation. Thus it is clear that the new FOLMax-Mine generates only maximal patterns and gives a reduction of about 50% in the number of generated patterns. Moreover, rate of reduction increases as support threshold decreases.

5.3.2.4 Scalability Test

To test the scalability of the algorithm, size of the database is changed keeping the support threshold constant. Experiment is done using the dataset T10I4D100k. Data size is changed from 20k to 100k. Table 5.5 provides the execution time taken by the method for the various database sizes. Figure 5.10 shows the result of the scale-up experiments graphically. The experimental result shows that the algorithm is linearly scalable.

Table 5.5: Execution Time for Various Database Sizes at Support Threshold for T10I4D100K

Database Size (in K)	Execution Time
20	18
40	44
69	72
80	103
100	128

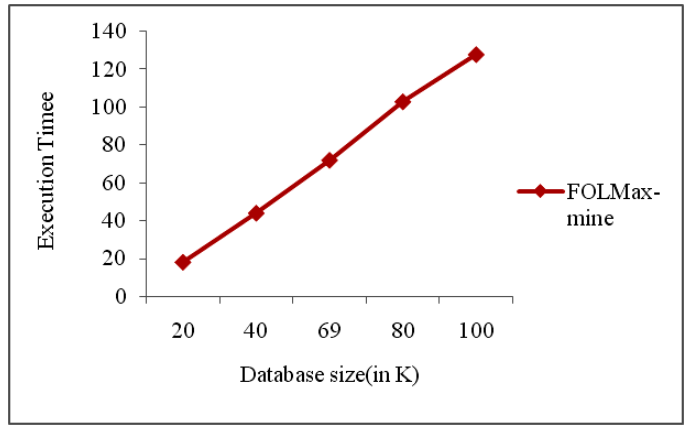

Figure 5.10: Scalabilty of FOLMax-Mine on Dataset T10I4D100K

5.4 Summary

Finding out interesting patterns is a fundamental task in many data mining algorithms. Many methods have been suggested by the researchers towards this task. Execution time and space for holding input sequences and output patterns are two major concerns in pattern mining. A major challenge in mining frequent patterns from a large data set is the fact that such mining often generates a huge number of patterns satisfying the minimum support threshold, especially when minimum support is set low. This is because if a pattern is frequent, each of its sub-patterns is frequent as well. A large pattern will contain an exponential number of smaller, frequent sub-patterns. Maximal frequent pattern mining was proposed to overcome this situation.

In this chapter, a new efficient method for generating Maximal Access Patterns is proposed. Thorough analysis of the method is performed to prove its correctness and efficiency. Execution time and the Number of Patterns generated by the proposed method at various support thresholds are studied for the datasets *msnbc* and T10I4D100K. The study reveals that even though the number of maximal access patterns increases rapidly at lower support thresholds, the execution time increases only at a low rate and it is under control.

Proposed FOLMax-Mine is compared with a conventional Access Pattern Mining method FOL-Mine, which is introduced in Chapter 4, to study the advantage of the new method. The experimental study shows that the proposed method gives about 50% reduction in the number of patterns generated. Results of the experimental analysis also are provided. Scaling up experiments proved the new method to be linearly scalable.