A New MFI Mining Algorithm with effective Pruning Mechanisms

K.Sumathi
Department of Computer Applications,
K.L.N.C.I.T, Madurai

S.Kannan
Department of Computer Science, DDE, MKU,
Madurai

K.Nagarajan
Chief Architect of Business Intelligence, Tata Consultancy Services, Chennai.

ABSTRACT
Mining of frequent patterns is a basic problem in data mining applications. Frequent Itemset Mining is considered to be an important research oriented task in data mining, due to its large applicability in real world applications. In this paper, a new Maximal Frequent Itemset mining algorithm with effective pruning mechanism is proposed. The proposed algorithm takes vertical tidset representation of the database and removes all the non-maximal frequent item-sets to get exact set of MFI directly. Pruning is done for both search space reduction and minimizing the number of frequency computations. It works efficiently when the number of item-sets and tid-sets are more. The proposed approach has been compared with Mafia algorithm for mushroom dataset and the results shows that the proposed algorithm performs effectively and generates frequent patterns faster. In order to understand the algorithm easily, an example is provided in detail.

Keywords
Data Mining, Frequent Itemset Mining, Maximal Frequent Itemset Mining.

1. INTRODUCTION
The 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 is mining frequent itemsets. Frequent pattern mining has become an important data mining task and a focused theme in data mining research. Frequent pattern mining was first proposed by Agrawal et al. (1993)[13] for market basket analysis in the form of association rule mining. The Technical issue that was attempted to resolve is to find all relevant frequent itemsets that are present in a very large database with provision of limiting factor that comes as a user input (i.e. minimum support). For a transaction database, a frequent itemset is one that occurs in at least a user-specific percentage of the database. That percentage is called support. An itemset is closed if none of its supersets has the same support as the itemset. An itemset is maximal frequent if none of its supersets is frequent.

Let $I = (i_1, i_2, \ldots i_m)$ be a set of *m* distinct items. Let D be a set of database transactions where each transaction T is an on empty itemset such that $T \subseteq I$. A transaction T is defined as any subset of items in I. Each Transaction is associated with an identifier, called a TID. A set of items is called an Itemset. A transaction T is said to contain A if $A \subseteq T$. A set of items is said to be an itemset. An itemset that contains K items is a K-itemset. Support Count is the total frequency of appearance of a particular pattern in a database. Frequent Itemset is an itemset whose support count is greater than or equal to the minimum support threshold specified by the user. Data mining to determine the frequent pattern will be based on various elemental analyses such as Pattern completeness, Rule Set Abstraction Levels, different kinds of pattern rules, values and different dimensions involved as part of the larger data sets.

Once the frequent itemsets from transactions in a database D have been found, it is straightforward to generate strong association rules from them (where strong association rules satisfy both minimum support and minimum confidence). Most of the existing frequent itemset mining algorithms use a breadth-first approach, In breath-first search all K- itemsets are generated before finding (k+1)itemsets. In depth first search k+1 itemsets can be generated before finding all FIs. A lot of algorithm is available for generating all frequent itemsets efficiently [6, 7, 8, 9, 10, 11, and 12].

The major hitch of mining frequent itemsets for a huge database is the number of frequent itemsets may be large when the user specified minimum support is low and the memory requirement for storing these itemsets is also very high. This leads to the introductions of closed frequent itemsets and maximal frequent itemsets. An itemset P is closed frequent itemset if none of its supersets has the same support as the itemset P. An itemset S is maximal frequent itemset if S is frequent and none of its supersets is frequent. Maximal frequent itemset mining is efficient in terms of time and space when compared to frequent itemsets and closed frequent itemsets because frequent ietmsets and closed frequent itemsets are subsets of maximal frequent itemset. Some of the existing algorithms for mining maximal frequent itemsets are MaxMiner [1], DepthProject [2], MAFIA [5], Pincer Search [14], FPMax [4] and GenMax [3].

The database representation is also an important factor in the efficiency of generating Maximal patterns. In horizontal data format, the data is represented as tid-itemset format, where tid is the transaction identifier and itemset is the set of items included in the transaction. In vertical data format, the data is represented as item-tidset format, where item is the name of the item and tidset is the set of transaction identifiers containing the item. The vertical representation allows simple and efficient support counting.

2. RELATED WORKS

Data mining or knowledge discovery in databases is a collection of exploration techniques based on advanced analytical methods and tools for handling a large amount of information. The problem of mining frequent itemsets has been a topic of Intensive research. Efficient algorithms for mining frequent items are crucial for mining association rules. Most existing work focuses on mining all frequent item sets (FI). However, since any subset of a frequent item set also is frequent, it is sufficient to mine only the set of maximal frequent item sets (MFI). In this paper we study the performance of existing algorithm Mafia and also present a new algorithm to find MFIs quickly by obtaining the Effective possible frequent items (EPMFI). Some of the existing MFI mining algorithm is described below.

Max Miner [1] is an algorithm introduced by Roberto Bayardo for finding the maximal frequent patterns. It uses efficient pruning techniques such as item reordering to quickly narrow the search. It introduces Support lower bound computation method for frequency computations. Max Miner employs a breadth first traversal of set enumeration tree of itemset. It reduces database scanning by employing a look ahead pruning strategy.

Depth Project [2] finds long itemsets using a depth first search of a lexicographic tree of itemsets, Depth project uses a bitstring representation of database and counting method based on transaction projections along its branches. Bucketing technique is used to improve the counting times. It returns superset of the MFI and requires post-pruning to eliminate non-maximal itemsets.

Mafia [5] is one of the recent method for mining the maximal frequent pattents. In Mafia the Search strategy combines a vertical bitmap representation of the database with an efficient relative bitmap compression schema. Mafia uses three pruning strategies to remove non-maximal sets. The first one is the look-ahead pruning introduced in MaxMiner. The second technique checks if $t(X) \subseteq t(Y)$. If so X is considered together with Y for extension. The last method is to check if any existing maximal set includes the new set. Mafia requires a post-pruning step to eliminate non-maximal patterns.

Pincer Search Algorithm was proposed by, Dao-I Lin & Zvi M. Kedem [14]. This algorithm uses both, the top-down search to maintain a candidate set of maximal patterns and bottom-up search like apriori to construct the candidates. In this the main search direction is bottom-up except that it conducts simultaneously a restricted top-down search, which basically is used to maintain another data structure called Maximum Frequent Candidate Set. This can help in reducing the number of database scans, by eliminating non-maximal sets quickly. The maximal candidate set is a superset of the maximal patterns, and in general, there is a high overhead while maintenance. It uses the two properties Downward Closure Property, Upward Closure Property for pruning candidate sets.

FpMAX is also an MFI mining algorithm introduced by Gosta Grahne and Jianfei Zhu [4], and it is an extension of the FP-growth method, for mining MFI. Here an FP-tree is used to store the frequency detail of the entire dataset. A structure called Maximal Frequent Item set tree is utilized to keep track of all maximal frequent item sets. Using this stucture , Fp-MAX effectively reduces the search time and the number of subset testing operations.

GenMax is a backtrack search based algorithm introduced by K. Gouda and M.J.Zaki [3] for mining maximal frequent itemsets. GenMax uses a vertical database format, where data is represented in item- tidset format. GenMax uses a number of optimizations to prune the search space. It introduces new techniques such as progressive focusing to perform fast superset checking, reordering for search space pruning and diffset propagation to perform fast frequency computation.

3. PROPOSED APPROACH

The proposed approach focuses on Mining Maximal Frequent Itemset Generation. This paper introduces a new approach with Effective Pruning Mechanism to generate Maximal frequent itemsets directly.

There are two main focus elements to develop an efficient MFI algorithm. The first is the set of techniques used to reduce the size of search space, and the second is the representation used to perform fast frequency computations. This paper describes how proposed algorithm achieves the same.

In general the structure of the transactional database may be in two different ways - Horizontal data format and Vertical data format. Here, we are using vertical data format for storing the transactions in the database. The data is represented as item-tidset format, where item is the name of the item and tidset is the set of transaction identifiers containing the item.

Consider our example database which includes six different items, I = {A, B, C, D, E, F} and six transactions T= {1, 2, 3, 4, 5, 6}. The vertical data format of the database d is given below.

Item	**Tidset**
A	T1, T2, T3, T4, T5
B	T2, T5, T6
C	T1, T2, T4, T5, T6
D	T1, T3, T4, T5
E	T2, T3, T4, T5, T6
F	T2, T4, T6

Table 1 : Vertical Data format of the transactional database D.

All Frequent items are extracted first. The support is directly given by the number of transactions in the tidset of each item. Let us consider the minimum support to be 3. From the above structure, all items are frequent. The items A, B, C, D, E and F are frequent items and will be considered to next level.

In the next level possible frequent extension and infrequent extension for each frequent item are obtained. PFE and IFE for each item is given below.

Frequent Item	Possible Frequent Extension (PFE)	Infrequent Extension (IFE)
A	C, D, E	B, F
B	C, E	D, F
C	D, E, F	-
D	E	F
E	F	-
F	-	-

Table 2: PFE and IFE of every frequent item.

An item j is added to possible frequent extension of item i if number of transaction occurred in the intersection of tidset of frequent item i and j satisfies the user specified minimum support. Otherwise the item j is added to infrequent extension of item i.

Definition 1:

An item j can be added to PFE(i) if number of transaction in (T(i) ∩ T(j)) achieves the user specified minimum support and itemset (i,j) can be a frequent itemset. Otherwise the item j is added to IFE (i) and itemset (i j) can't be a frequent itemset. Preliminary Possible Maximal Frequent Itemsets(PPMFIs) for each frequent item are generated from Possible Frequent Extensions of that item. For example PPMFI of item A is ACDE. Infrequent itemsets are also generated from infrequent extension of that item. IFitemset of A is AB, AF, and ABF. AB, AF and BF are infrequent. So ABF is also an infrequent itemset. The Preliminary PMFIs and IFitemsets of example database are given below.

Frequent Item	EPMFIs
A	ACDE
B	BCE
C	CDE,CEF
D	DE
E	EF
F	F

Table 3: Preliminary PMFIs and IFitemset of every frequent item.

Effective PMFIs(EPMFIs) are generated from preliminary PMFI by eliminating infrequent itemsets.

Definition 2:

A PPMFI can't be a MFI, if it includes any IFitemsets.

For example the third PPMFI CDEF has an IFitemset DF. So this PPMFI can't be a MFI.

Definition 3:

When a PPMFI includes any IFitemset, it can be split into N new PPMFIs with respect to the IFitemset, it contains, where N is the size of IFitemset.

So that the PPMFI, CDEF can be split into two PPMFIs with respect to DF.(CDE and CEF. The size of IFitemset is 2).

Definition 4:

Any PPMFI can be an Effective PMFI, if it has no IFitemsets.

The first and second PPMFIs are ACDE, BCE and these itemsets have no infrequent itemset and added to effective PMFI. The third PPMFI is CDEF and it has one infrequent itemset (DF). So this itemset is split into two itemsets(CDE, CEF) with respect to DF and the new PPMFIs have no infrequent itemsets and added to effective PMFIs. Next 3 preliminary PMFIs(DE, EF, F) are added to (EPMFIs).The effective PMFIs of each frequent item is given below.

Frequent Item	Preliminary PMFIs	IFitemsets(IF itemsets)
A	ACDE	AB, AF, ABF
B	BCE	BD, BF, BDF
C	CDEF	-
D	DE	DF
E	EF	-
F	F	-

Table 4: Effective PMFIs of every frequent item

All MFIs are generated from EPMFI using the proposed algorithm. The First EPMFI is ACDE has no superset in MFI and it is not frequent. So (n-1) combinations of ACDE starts with A (first item)is obtained (ACD,ACE, ADE)and the same algorithm is called for these combinations and all of them have no superset in MFI, are frequent and are added to MFI. The Second itemset BCE has no superset in MFI and it is a frequent item set. So it is added to MFI. The Third itemset CDE has no superset in MFI and it is a infrequent item set. So all (n-1)-itemsets (includes C) of CDE are generated. Every 2-itemsets of CDE (CD and CE) has superset in MFI and ignored. The fourth itemset CEF has no superset in MFI and it is a frequent item set and added to MFI. The subsequent EPMFIs (DE, EF, F) have supersets in MFI and Ignored.

From above example, MFIs with support count 3 are ACD, ACE, ADE, BCE and CEF.

The process will be continued till testing all effective possible maximal frequent itemsets. The ideal place to do pruning is to incorporate while finding the MFIs. Hence the efficiency can be achieved. The pseudo code for proposed algorithm is given below in figure 1.

Pseudo code

Function to find all MFIs from EPMFIs

Inputs

(i) EPMFIs (Effective Possible Maximal Frequent itemsets)

(ii) min_sup - Minimum support the has been defined for mining process Interfacing Functions

Output

(i)MFIs

Find All MFI(EPMFIs, min_sup)

For each x ε EPMFIs

if x has a superset in MFI

ignore x;

else if x is frequent

MFI.add(x);

else

// find new EPMFIs by obtaining (n-1)combinations of EPMFI that includes the first item of EPMFI.

NEPMFIs=Find (n-1)combinations(x,min_sup);

findAllMFI(NEPMFIs,min_sup);

For End;

Function to find all EPMFIs from PPMFIs

Inputs

(i) PPMFIs (Preliminary Possible Maximal Frequent itemsets)

(ii) min_sup - Minimum support that has been defined for mining process Interfacing Functions

Output

(i)EPMFIs

FindAllEPMFI(PPMFIs,min_sup)

For each x in PPMFIs

If x contains any IFitemsets

Divide x into n newPPMFIs with respect to IFitemset;

// where n is the size of IFitemset (ABCE is PPMFI , IFset is CE then PPMFI1=ABC PMFI2=ABE)

For each y in newPPMFIs

FindAllEPMFI (y, min_sup);

end for

else

EPMFI.add(PPMFI);

end if

end for

Figure 1. Pseudo code for MFI mining

The proposed algorithm performs better because all Maximal Frequent Itemsets are being calculated directly from EPMFI before computing Frequent Itemset completely. Once the association between every two items is obtained, the Preliminary Possible Maximal Frequent Itemsets are retrieved from possible Frequent Extensions and Effective PMFIs are obtained from PPMFIs by removing infrequent itemsets. The other itemsets are ignored automatically.

The Pruning mechanism works effectively and counting is not performed for the subset of MFI's. So, the time taken to compute MFI is negligible. As we are following vertical data format, there is no need to calculate the support separately. In this case, support is obtained by the counting the number of transactions in the tidlist of each item. The vertical representation of database has the following major advantages over the horizontal layout: When the vertical (item-tidset) format is used, the support of an item is obtained by computing the number of transaction in its tidset and finding support of an itemset involves only the intersection of tidsets. With the vertical layout, there is an automatic "reduction" of the database before each scan in that only those itemsets that are relevant to the following scan of the mining process are accessed from disk.

Pruning

Pruning at level 1:

Possible Frequent Extensions (PFEs) of every frequent item are created by finding association between every two items. PPMFIs are obtained from PFEs. So PPMFI of a frequent item includes only frequent extensions of that item. Infrequent items of every frequent item are also generated simultaneously to find EPMFIs.

Pruning at level 2:

The Effective Possible Maximal Frequent itemsets(Maximal Candidate Itemsets) are directly retrieved from PPMFIs by eliminating infrequent itemsets from PPMFIs. So maximum of infrequent itemsets are pruned automatically. All MFIs are generated from EPMFIs. The frequency check is not performed for the itemsets having supersets in MFI.

The proposed approach applies superset checking to eliminate the non maximal frequent itemsets. Once all EPMFIs are generated, each EPMFI is checked whether it is a subset of any maximal pattern. If so the itemset is eliminated entirely. Counting is not performed for this itemset and next EPMFI is taken for test.

4. RESULTS AND DISCUSSIONS

The testing of the proposed algorithm has been carried out on the real dataset (containing long itemsets) mushroom. We measured that, the number of candidate itemsets and frequency computation taken by the proposed algorithm to find MFIs and it is compared to Mafia algorithm for various values of minimum support. The support is varied from 10 to 1. The results show that the proposed algorithm generates MFIs very quickly than our implementation of Mafia method.

Figure 2 illustrates that, the proposed approach generates all MFIs very quickly than Mafia algorithm. Support is taken as x axis and the time taken to generate all MFI is taken as y axis.

For Mushroom, Computation of MFI at every stage has been explained and correlated to the overall search space reduction at upper levels. This brings significant performance improvement with respect to the time of execution and the data set operations in the downstream processing. This approach will be working very efficiently for any sparse and dense dataset.

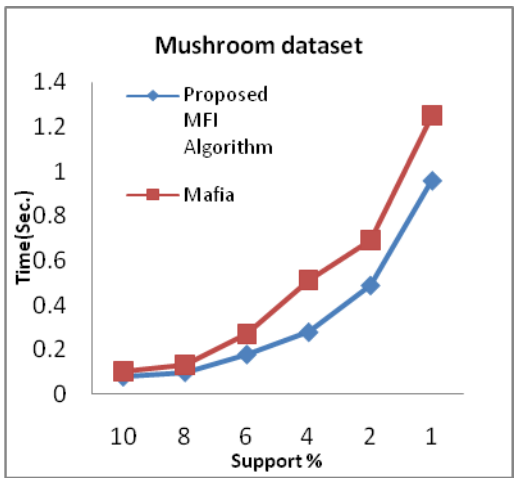

Figure 2. Execution time comparison of Proposed algorithm and Mafia on Mushroom dataset.

5. CONCLUSION

In this paper we have investigated a new approach and algorithm for mining MFIs. The algorithm is to approach the problem slightly different and leverage best techniques to reduce execution time – basic steps are finding frequent items, obtaining Possible Frequent Extensions of every frequent item, Generating PPMFIs from PFEs , obtaining EPMFIs from PPMFIs by eliminating Infrequent itemsets and extracting MFIs from EPMFIs. Our algorithm is compared with Mafia algorithm and obtained that the proposed algorithm generates all MFIs very quickly than Mafia. The vertical data format representation of the database, EPMFI generation and directly computing MFIs from EPMFIs are the added advantages of this algorithm.

6. REFERENCES

[1] Roberto Bayardo, "Efficiently mining long patterns from databases", in ACM SIGMOD Conference 1998.

[2] R. Agarwal, C. Aggarwal and V. Prasad, "A tree projection algorithm for generation of frequent itemsets", Journal of Parallel and Distributed Computing, 2001.

[3] K. Gouda and M.J.Zaki, "Efficiently Mining Maximal Frequent Itemsets", in Proc. of the IEEE Int. Conference on Data Mining, San Jose, 2001.

[4] Gosta Grahne and Jianfei Zhu, "Efficiently using prefix-trees in Mining Frequent Itemsets", in Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations Melbourne, Florida, USA, November 19, 2003.

[5] Burdick, D., M. Calimlim and J. Gehrke, "MAFIA: A maximal frequent itemset algorithm for transactional databases", In International Conference on Data Engineering, pp: 443 – 452, April 2001, doi = 10.1.1.100.6805

[6] J. Han, J. Pei, and Y. Yin. "Mining frequent patterns without candidate generation", In ACM SIGMOD Conf., May 2000.

[7] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules", Advances in Knowledge Discovery and Data Mining, pages 307-328, MIT Press, 1996.

[8] V. Ganti, J. E. Gehrke, and R. Ramakrishnan, "DEMON: Mining and Monitoring Evolving Data", ICDE 2000: 439-448

[9] Aggarwal, C.C. and P.S. Yu, "Mining largeitemsets for association rules", in Bulletin of the IEEE Computer Society Technical Committee onData Engineering, 1998, pp: 23-31. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.306

[10] Aggarwal C.C and P.S. Yu, "Online generation of association rules", in proceedings of the fourteenth International Conference on Data Engineering, 1998, pp:402-411.

[11] Park J.S, M.S. Chen, P.S. Yu, "An Effective Hash Based Algorithm for Mining Association Rules",ACM SIGMOD Record, Vol. 24, Issue 2, May 1995, pp: 175-186, ISSN: 0163-5808.

[12] Dunkel B. and N. Soparkar, "Data Organization and access for efficient data mining", in the proceedings of the 15th International Conference on Data Engineering, pp: 522-529, 1999, ISBN: 0-7695-0071-4

[13] R. Agrawal, T. Imielienski and A. Swami, "Mining association rules between sets of items in largedatabases. In P. Bunemann and S. Jajodia, editors, Proceedings of the 1993 ACM SIGMOD Conference on Management of Data, Pages 207-216, Newyork, 1993, ACM Press.