# A Graph-based Algorithm For Mining Maximal Frequent Itemsets

Bo Liu, Jiuhui Pan
*Department of Computer Science, Jinan University, Guangzhou, China, 510632*
*lbxldd@sohu.com, jhpan@jnu.edu.cn*

## Abstract

*Association rule mining is an important research branch of data mining, and computing frequent itemsets is the main problem. The paper is designed to find maximal frequent itemsets only. It presents an algorithm based on a frequent pattern graph, which can find maximal frequent itemsets quickly. A breadth-first-search and a depth-first-search techniques are used to produce all maximal frequent itemsets of a database. The paper also analyzes the complexity of the algorithm, and explains the computation procedure by examples. It has high time efficiency and less space complexity for computing maximal frequent itemsets.*

## 1. Introduction

Association rule mining was first proposed by Agrawal et al [1], and discovering all frequent itemsets is the key problem of it. Apriori [1][2] is a typical frequent itemset mining algorithm, and there are some improved algorithms[3][4] based on it. In addition, a number of methods based on FP-tree[5] and hypergraph [6][7] appear.

A database D consists of several records. Let L={$i_1$,$i_2$,……$i_n$} denote a record of data items. A itemset T is defined to be a subset of L. A k-itemset is a set with k items. The support or frequency of T is the percentage of records in D that contain the itemset. T is a frequent itemset if its support exceeds a given threshold $\sigma$. All subsets of a frequent itemset must also be frequent， and any superset of a maximal frequent itemset is not frequent.

Given n items, there are potentially $2^n$ itemsets, however, only a small fraction of them are usually frequent. The frequent itemsets discovery algorithms aim to find them quickly. In the paper, we focus on the methods for discovering maximal frequent itemsets only.

The paper presents a frequent pattern graph (FP-graph) to study the problem of maximal frequent itemset mining. The motivation behind the application of a FP-graph in mining maximal frequent itemsets is to improve time efficiency and reduce space complexity in comparison to some existing algorithms.

The work was supported by science and technology research plan grant form Guangdong Province (2003c101011), China.

## 2. Related work

In the literature[7], it proves that the problem of deciding if there is a maximal $\sigma$-frequent itemset with at least t attributes for a given 0-1 relation r, and a threshold $\sigma \in [0,1]$, is NP-complete. Clearly exact exponential time algorithms cannot work in lots of domains. A number of algorithms[7][8][9][11][12] have been proposed to improve efficiency of the problem.

Aprior algorithm is a famous algorithm for mining all frequent itemsets. It employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets. For computing the maximal elements, it performs well when the size of the maximal elements is small. If there is a maximal itemset whose size is large, then it would require too much time.

Pincer-Search[8] combines a bottom-up and a top-down techniques to find the maximal frequent itemsets. The bottom up process finds frequent itemsets, and non-frequent itemsets. The non-frequent itemsets are used by a top down process to refine a set of potential maximal frequent itemsets.

Max-Miner[9] uses Rymon's set enumeration[10] to enumerate all the itemsets. It reduces the search space by pruning the tree to eliminate both supersets of infrequent itemsets and subsets of frequent itemsets. While the algorithm computes the support of an itemset, it also computes the support of the largest itemset that appears in the subtree rooted at this itemset. If this superset is frequent, then all other itemsets in this subtree must be frequent too. Thus the algorithm avoids having to compute the support of all the frequent itemsets.

Similar to Max-Miner[9], DepthProject[11], MAFIA[12], and MaxGen[13] use Rymon's set enumeration technique.

To our best knowledge, Dualize and Advance[7] has the lowest complexity bounds for computing all maximal itemsets. It applies a greedy search to locate some maximal elements, and uses the simple fact that if some maximal frequent itemsets are known, then every unknown one must contain a minimal transversal of the complements of the known itemsets. The algorithm alternates between finding maximal frequent itemsets and finding minimal traversals of the complements of the already discovered maximal frequent itemsets, until no new maximal frequent itemsets can be found.

Among the above algorithms, only Dualize and Advance has provably worst case complexity that is sub-exponential to the size of the output(that is, to the number of maximal frequent itemsets)[7].

## 3. Description of a FP-graph

Assume that the schema of a database D is A={A1, A2, …… $A_m$}, Dom($A_i$) is the domain of the attribute $A_i$, and each transaction including in D is denoted as ($t_1$, $t_2$, …... $t_m$), where $t_i$ ( i.e. $A_i$=a, a∈ Dom($A_i$)) is an item of the transaction.

**Definition 1:** A database transaction graph G=(V, E) is a directed graph, which has a vertices' set V and an edges' set E: V is a set of all transaction items, and is divided into m number of subsets, i.e. $\bigcup_{i=1}^{m} V_i = V$, where $V_i$ is a set of all items with attribute name $A_i$ ; E is a set of two-item directed edges, i.e. E={<$v_n$, $v_k$>|$v_n$∈ $V_i$, $v_k$∈ $V_j$, j=i+1, $v_n$ and $v_k$ appear in a same transaction .}.

**Definition2:** A database frequent pattern graph (or FP-graph) G=(V', E') is a subgraph of its transaction graph, i.e., V'∈V, E' ∈E, where V' is a set of frequent 1-itemsets, and E' is a set of directed edges that connect two frequent 1-itemsets.

For example, a relational database includes three transactions, shown in table 1. Suppose that V={ $A_1$=1, $A_1$=2, $A_2$=2, $A_2$=3, $A_3$=2, $A_3$=3}={$v_1$, $v_2$, $v_3$, $v_4$, $v_5$, $v_6$ }, the minimal support threshold is 40%, then the database transaction graph is shown in Figure 1, and the database frequent graph is shown in Figure 2.

**Table 1 A relational database example**

| A1 | A2 | A3 |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 3 | 3 |
| 2 | 2 | 2 |

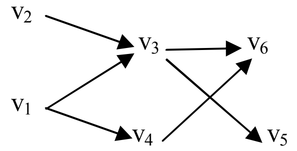

**Figure 1 A transaction graph**

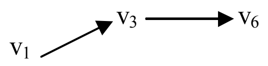

**Figure 2 A frequent pattern graph**

Therefore, the problem of computing a database's frequent itemsets can be changed to the problem of creating, transforming and traversing a FP-graph. Firstly, scan the database once, result in all frequent 1-itemsets and frequency of each item; then scan the database again and create its frequent pattern graph according to definition 2. Lastly, use the algorithm described in section 4 to produce all maximal frequent itemsets.

## 4. The MFSET algorithm

### 4.1. Basic idea

According to the following properties, we can transform a FP-graph, and then compute all maximal frequent itemsets efficiently.

Property 1: If {$v_i$,$v_j$} is not a frequent itemset, and {$v_j$,$v_k$} is a frequent itemset or not, then {$v_i$,$v_j$,$v_k$} is definitely not a frequent itemset.

According to the above property , if {$v_i$, $v_j$} is not frequent, the edge <$v_i$,$v_j$> in a FP-graph can be deleted, and the edge < $v_j$,$v_k$ > can also be deleted if {$v_j$,$v_k$} is not a frequent itemset; Meanwhile the edge <$v_i$,$v_k$> should be added if it doesn't exist currently and {$v_i$,$v_k$} is frequent.

Property 2: If {$v_i$,$v_j$…$v_k$} is a frequent itemset, then there is a path $v_i \rightarrow v_j \rightarrow \ldots \rightarrow v_k$ in a FP-graph.

Given a FP-graph, all maximal σ-frequent itemsets can be worked out according to the following steps. Assuming S be a set of vertices in the FP-graph, in which a vertex's in-degree is zero and out-degree is not zero.

(1) Choose a vertex from S randomly, and remove it from S, i.e. S＝S\$v_i$, and then traverse other vertices by a breadth-first-search technique along the directed edges.

(2) At the vertex $v_i$, choose an adjacent edge <$v_i$,$v_j$>, and compute frequency($v_iv_j$). If frequency($v_iv_j$ ) ≥σ, then {$v_i$,$v_j$} is a frequent 2-itemset; Otherwise delete the edge <$v_i$,$v_j$>, and add $v_j$ into S if $v_j$'s in-degree becomes zero, meanwhile test the directed edges <$v_i$,$v_{j1}$>……<$v_i$,$v_{jk}$> if they don't exist, where $v_{j1}$……$v_{jk}$ are adjacent vertices of $v_j$, and add the edge <$v_i$,$v_{jp}$> (1≤p≤k) if frequency($v_iv_{jp}$) ≥σ. Iterate this step until all adjacent edges of $v_i$ are processed.

(3) From one adjacent vertex of $v_i$, traverse the FP-graph backwards as step (2). Iterate this step until the vertices whose out-degree are zero.

(4) Return to (1), iterate (2)(3) until S is null. Now if there are vertices whose in-degree and out-degree are zero, they must be maximal frequent itemsets.

(5)Reset S be a set of vertices in the transformed FP-graph, in which a vertex's in-degree is zero and out-degree is not zero.

(6)Remove a vertex V from S, and traverse the transformed FP-graph from V by a depth-first-search technique. For each path(except the path which is a subset of another path) which starts from V and ends in a vertex whose out-degree is zero, compute frequency of the set (denoted as U) that consists of all vertices on the path, and test whether U is a maximal frequent itemset. If frequency(U) ≥σ U is a maximal frequent itemset; otherwise test U\$v_i$($v_i$∈U), U\$v_i$ is a candidate for maximal frequent itemsets if U\$v_i$ is a frequent itemset and is not a subset of any current known maximal frequent itemset, if not, go on to test $U\backslash v_iv_j(v_i \in U, v_j \in U)$, ……,until the test set includes only two vertices, it is a candidate for maximal frequent itemsets if it is not a subset of any current known maximal frequent itemset. In the subsequent computation, if a candidate for maximal frequent itemsets is a subset of any new maximal frequent itemset or a subset of any new candidate for maximal frequent itemsets, it is a non-maximal frequent itemset.

(7)Repeat (6) until S is null.

On a whole, the above procedure is divided into two phases: step (1) till (4) belong to the first phase. After the phase, the vertex whose in-degree and out-degree are zero must be a maximal frequent itemset. Then go to the second phase, and compute other maximal frequent itemsets by a top-down fashion.

For example, assume that a relational database includes four attributes, which attributes domain are $A_1$, $A_2$, $A_3$, $A_4$. Figure 3 is its FP-graph, where $v_1 \in A_1$, $v_2 \in A_1$, $v_3 \in A_2$, $v_4 \in A_2$, $v_5 \in A_3$, $v_6 \in A_3$, $v_7 \in A_4$, $v_8 \in A_4$.

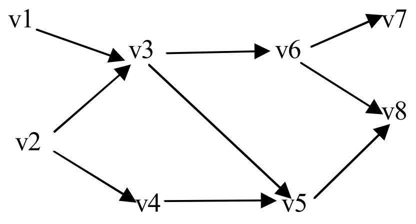

**Figure** 3 **An example of a FP-gaph**

In Figure 3, let $\{v_1,v_3\}$,$\{v_4,v_5\}$,$\{v_5,v_8\}$ be non-frequent, the edges $<v_1,v_3>$,$< v_4,v_5>$,$<v_5,v_8>$ are pruned, meanwhile test the edges $<v_1,v_6>$ and $<v_4,v_8>$, let $\{v_1,v_6\}$ be non-frequent and $\{v_4,v_8\}$ be frequent, then Figure 3 is transformed to Figure 4. Now $\{v_1\}$ must be a maximal frequent itemset.

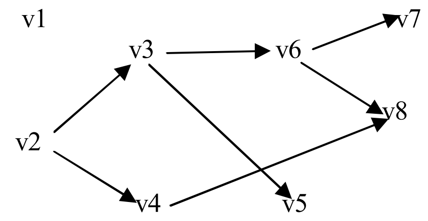

**Figure** 4 **A transformed FP-graph**

In Figure 4, start from $v_2$, the paths $v_2v_3v_6v_7$, $v_2v_3v_6v_8$ , $v_2v_3v_5$, $v_2v_4v_8$ can be traversed by a depth-first-search technique, then compute maximal frequent itemsets based on these paths by a top-down fashion in turn. Firstly, test a longest path $v_2v_3v_6v_7$, if it is frequent it must be a maximal frequent itemset, otherwise test $v_3v_6v_7(v_2v_3v_6v_7\backslash v_2)$, $v_2v_6v_7(v_2v_3v_6v_7\backslash v_3)$, $v_2v_3v_7(v_2v_3v_6v_7\backslash v_6)$ and $v_2v_3v_6(v_2v_3v_6v_7\backslash v_7)$ respectively to determine whether they are frequent. Let $v_3v_6v_7$, $v_2v_6v_7$ be frequent, they must be maximal frequent itemsets. Then the subsets of $v_2v_3v_7$ and $v_2v_3v_6$, i.e., $v_2v_3$, $v_3v_7$, $v_2v_7$ ,$v_2v_6$ and $v_3v_6$ are candidates for maximal frequent itemsets. But $v_3v_7$ ,$v_2v_7$ , $v_2v_6$ and $v_3v_6$ are all subsets of a known maximal frequent itemset, so only $v_2v_3$ maybe a maximal frequent itemset. The same computation method is applied to paths $v_2v_3v_6v_8$, $v_2v_3v_5$, $v_2v_4v_8$.

## 4.2. Description of MFSET algorithm

**Algorithm name:** MFSET

Input: G (a FP-graph of a database) with a vertex set V, an edge set E, and minimum support $\sigma$.

Output: Mset (maximal frequent itemsets).

Method:

```
1.  S ={v∈V| In-degree(v)=0∧out-degree(v)≠0};
2.  While (S≠Φ) Do
3.     S=S\v, v∈S;
4.     Repeat
5.        Breadth-first-search (G, AdjacentSet);
6.        Prune or add edges in G;
7.     Until (AdjacentSet =Φ)
8.  Enddo
9.  Mset＝{v∈FP-graph| In-degree(v)=0∧
            out-degree(v)=0}
10. S={v∈V| In-degree(v)=0∧out-degree(v)≠0} ;
11. While (S≠Φ) Do
12.    S=S\v, v∈S;
13.    Depth-first-search (G, v, Paths);
14.    For each path in Paths
15.       Y=MaxSetCompute(path);
16.       Mset=Mset∪Y-X;
17.    End for
18. End while
```

There are some specifications of the above algorithm:

(1) From line 1 till line 8( the first phase), the algorithm works iteratively, adjusting the initial FP-graph, deleting or adding edges, such that two vertices of any edge left in the FP-graph form a frequent 2-itemset.

(2) Breadth-first-search (G, AdjacentSet) in line 5 is a procedure, which searches G by a breadth-first technique. AdjacentSet is adjacent vertices' set of v;

(3) From line 11 on (the second phase), the algorithm works following procedure iteratively: search a path starting from a node whose in-degree is zero and ending in a node whose out-degree is zero, then compute maximal frequent itemsets based on the path.

(4) Depth-first-search (G, v, Paths) in line 13 is a procedure, which starts from v and searches G by a depth-first technique, where Paths is a set of all paths which begin from v and end in vertices whose out-degree are zero.

(5)MaxSetCompute(path) in line 15 is a function, which computes and returns new candidates for maximal frequent itemsets based on the path in a top-down fashion.

(6)X in line 6 is a set of candidates for maximal frequent itemsets computed before, which are subsets of one of itemsets in Y.

### 4.3 Complexity Analysis of MFSET algorithm and comparison with some algorithms

Given a database , all attributes are assumed to be categorical. For computing all frequent 1-itemsets of the database and constructing a FP-graph, it only needs twice scans of the database. Suppose m is the number of attributes, and $n = \mathrm{Max}\{n_i, 1 \le i \le m\}$, where $n_i$ is the number of possible values with attribute name $A_i$.

The number of vertices in the FP-graph is:

$n_1+n_2+n_3+\ldots\ldots+n_m \leqslant mn$

The number of edges in the FP-graph is:

$n_1*n_2*n_3*\ldots\ldots*n_m \leqslant n^m$

The complexity of the Breadth-first-search and the Depth-first-search is $O(m^2n^2)$ (Suppose the FP-graph represented by an $m \times n$ matrix). The complexity for computing maximal frequent sets based on the paths (from vertices whose in-degree are zero to vertices whose out-degree are zero) is $O(n^{\log mn})$. Therefore, the temporal complexity of MFSET algorithm is $O(n^{\log mn})$, i.e. the running time of our method is sub-exponential to total number of attribute values.

Compared with Aprior[1][2], MFSET produces fewer candidate itemsets in each traversal and reduces the scanning times of the database. Apriori produces frequent (k+1)-itemsets from frequent k-itemsets, and may produces a lot of candidate itemsets, for example, p number of frequent 1-itemsets will lead to $C_p^2$ candidate 2-itemsets, so that it need scan the database many times.

Similar to FP-graph, FP-Growth[5] method uses a FP-tree to represent all the transactions of the database, and then uses a recursive technique to find all frequent itemsets. Although the technique does not need to compute itemsets of candidate patterns, it still finds the support of all frequent itemsets and therefore can be less efficient than other techniques when used to find the maximal frequent itemsets only. On the other hand, a FP-graph may be pruned after the first phase, but a FP-tree is not changed during computing.

In addition, compared with Dualize and Advance [7] whose complexity is sub-exponential to the size of output, our method is designed for not only mining single dimensional Boolean association rules from transactional databases, but also mining multidimensional association rules from relational databases and data warehouses. However, Dualize and Advance only aims at mining single dimensional Boolean association rules from transactional databases.

As to other methods[8][9][11][12][13], their details of comparisons with Dualize and Advance have been made in literature[7], they are all based on Rymon's set enumeration technique, and are also only suitable for mining single dimensional Boolean association rules from transactional databases.

## 5. A further explanation using an example

In the section, we use an example of a transaction database, and explain the details how our method and Dualize and Advance compute all its maximal σ-frequent itemsets. The explanation instead of an experiment shows our method is more efficient than Dualize and Advance's.

The literature[7] has proven that the running time of Dualize and Advance is sub-exponential to the size of the output. For comparing our method with Dualize and Advance, we take a transaction database as an example( shown in Table 2), and explain the details how to compute all its maximal frequent itemsets for mining single dimensional association rules by the two different algorithms. Let σ=25%.

**Table 2 A transaction database**

| TID | List of items |
|---|---|
| 1 | A,B,C,D |
| 2 | A,B,C |
| 3 | A,C |
| 4 | A,D,E |
| 5 | A,D |
| 6 | B,C,E |
| 7 | B,D |
| 8 | C,E |

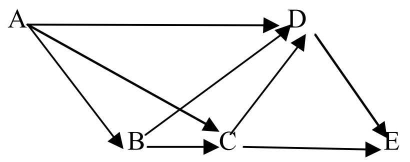
$$\begin{tikzcd}
A \arrow[rrr] \arrow[dr] \arrow[drr] & & & D \arrow[dr] & \\
& B \arrow[r] \arrow[urr] & C \arrow[ur] \arrow[rr] & & E
\end{tikzcd}$$

**Figure** 5 **The FP-gaph of the database in Table 2**

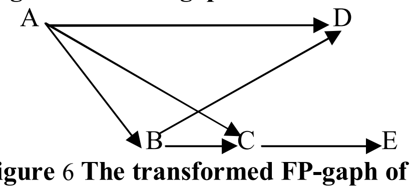
$$\begin{tikzcd}
A \arrow[rrr] \arrow[dr] \arrow[drr] & & & D \\
& B \arrow[r] \arrow[urr] & C \arrow[r] & E
\end{tikzcd}$$

**Figure** 6 **The transformed FP-gaph of Figure 5**

According to our method, firstly, create a FP-graph (Figure 5), then transform it to Figure 6. Secondly, start from vertex A, four paths may be traversed by a depth-first-search technique, i.e. ABCE，ACE，ABD and AD. Because ACE is a subset of ABCE, AD is a subset of ABD, we only need to test ABCE and ABD. First, based on the longer path ABCE, ABC and CE are worked out as maximal frequent itemsets. Second, based on ABD, AD is worked out as a maximal frequent itemset (AB is a subset of a known maximal frequent itemset) .

Using Dualize and Advance, the algorithm computes all maximal frequent itemsets based on a hypergraph shown in Figure 7, and traverses the graph by a random walk fashion. Let CE be the first maximal frequent itemset found. It's complement is ABD(i.e. ABCDE\CE), and the transversals of the complement are A, B and D. When a new maximal frequent itemset AD is found, the existing transversals(ABD) are extended, to intersect the complement of the new maximal frequent itemset(i.e. ABCDE\AD=BCE). Then the new transversals are ADC, ADE, and B. When ADC and ADE are found to be non-frequent, the branches of the trees are prune. Begin from B, BAC (i.e. ABC) is found to be a maximal frequent itemset. Each walk is random until the transversal is non-frequent.

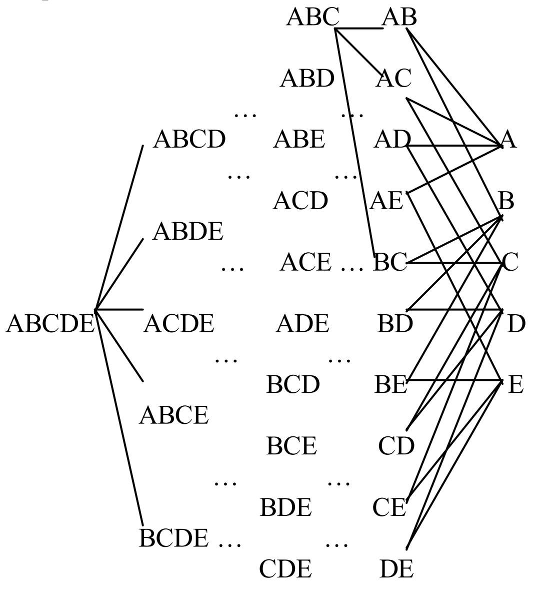

**Figure** 7 **A hypergraph used in Dualize and Advance**

Form above explanations of two methods, we can conclude the advantages of ours. The FP-graph is simpler than the hypergraph used in Dualize and Advance, so the possible transversals are fewer in our method. MFSET tries to prune the FP-graph , and then computes maximal frequent itemsets by testing long paths(the longest path has priority) in the transformed FP-graph. So it is very feasible to the scenario when the size of the maximal elements is large.It guarantees that a new maximal itemset or the subsets of maximal frequent itemsets are found in each iteration. Although Dualize and Advance starts from itemsets that have to be frequent and expands them to maximal frequent itemsets, and it also guarantees that new maximal itemsets are found in each iteration(if not, a itemset in the negative boundary is found), it performs the transversal computation by randomized permutations that may result different efficiency.

## 6. Conclusion

In a conclusion, MFSET can produce maximal frequent itemsets of a database or a data warehouse based on a FP-graph. The main contribution of the work is in two aspects: (1)it can be applied in mining single dimensional association rules and multidimensional association rules; (2)it uses a new data structure and presents a new algorithm whose the worst running time is sub-exponential to the number of all attributes' values.

## 10. References

[1]R.Agrawal, T.Imielinski, and A.Swami. Mining association rules between sets of items in large databases. In proc1993ACM-SIGMOD Int.Conf. Management of Data(SIGMOD'93), 207-216,1993.

[2]R.Agrawal and R.Srikant. Fast algorithms for mining association rules. In Proc.1994 Int.Conf.Very Large Data Bases(VLDB'94), 487-499,1994.

[3]J S Park, M S Chen, P S Yu. An effective hash-based algorithm for mining association rules. In Proc.1995 ACM-SIGMOD Int Conf on Management of Data, 175-186,1995.

[4]S.Brin, R.Motwani, J.D.Ullman, and S.Tsur. Dynamic itemset counting and implication rules for market basket analysis. In Proc.1997 ACM-SIGMOD Int.Conf. Management of Data(SIGMOD'97),pages 255-264.

[5]J.Han, J.Pei, and Y.Yin. Mining frenquent patterns without candidate generation. In Proc.2000 ACM-SIGMOD Int.Conf. Management of Data(SIGMOD'00),pages 1-12.

[6]D.Gunopulos, R.Khardon, H.Mannila, and H.Toivonen. Data mining, hypergraph transversals, and machine learning. In 16th ACM Symp. Principles of Database Systems,1997.

[7] D.Gunopulos, R.Khardon, H.Mannila, et al. Discovering all most specific sentences. ACM Transactions on Database Systems, Vol.V, No.N, 1-36, 2003.

[8] Lin,D.I., Kenden,Z.M. Pincer search: A new algorithm for discovering the maximum frequent set. In Extending Database Technology,105-119, 1998.

[9] Bayardo, R.J. Efficiently mining long patterns from databases. In Proc.of ACM SIGMOD Intl.Conf.on Management of Data, 1998.

[10] Rymon,R. Search through systematic set enumeration. In International Conference on Principles of Knowledge Representation and Reasoning,1992.

[11] Agarwal,R.C., Aggarwal, C.C. et al. Depth first generation of long patterns. In ACM Conf. on Knowledge Discovery and Data Mining(SIGKDD),108-118, 2000.

[12]Burdick,D., Calimlim,M. and Gehrke,J. Mafia: a maximal frequent itemset algorithm for transactional databases. In IEEE Intl. Conf. on Data Engineeringi(ICDE), 2001.

[13]Gouda,K., Zaki,M.J. Efficiently mining maximal frequent itemsets. In ICDM,163-170,2001.