Journal of Computer Science 4 (8): 638-645, 2008
ISSN 1549-3636

An Efficient Algorithm for Mining Maximal Frequent Item Sets

A.M.J. Md. Zubair Rahman and P. Balasubramanie
Kongu Engineering College, Perundurai, Tamilnadu, India

Abstract: Problem Statement: In today's life, the mining of frequent patterns is a basic problem in data mining applications. The algorithms which are used to generate these frequent patterns must perform efficiently. The objective was to propose an effective algorithm which generates frequent patterns in less time. **Approach:** We proposed an algorithm which was based on hashing technique and combines a vertical tidset representation of the database with effective pruning mechanisms. It removes all the non-maximal frequent item-sets to get exact set of MFI directly. It worked efficiently when the number of item-sets and tid-sets is more. **Results:** The performance of our algorithm had been compared with recently developed MAFIA algorithm and the results show how our algorithm gives better performance. **Conclusions:** Hence, the proposed algorithm performs effectively and generates frequent patterns faster.

Key words: Mining-frequent item sets-hashing-MAFIA

INTRODUCTION

Frequent pattern mining plays a major role in many data mining applications like mining association rules, correlations. Frequent patterns are the patterns that occur frequently in the given data set. If a set of items appear frequently together in a transaction data set, it is referred as frequent item set. If a set of items are occurring frequently in a sequential manner, it is referred as frequent sequential pattern. If a substructure like subtrees, subgraphs occurs frequently it is called as frequent structured pattern. These frequent items can be represented in the form of association rules. The association rule problem is a very important problem in the data-mining field with numerous practical applications including consumer market-basket analysis, inferring patterns from web page access logs and network intrusion detection. The association rule model was introduced by Agrawal[1]. Support and confidence are the two measures of rule interestingness. They reflect the usefulness and certainty of discovered rules respectively. If there is an association rule like X => Y (support = a%, confidence = b%) then it means that a% of all transactions show that X and Y are together and b% of customers who purchased X also bought Y, where X and Y are items. Association rules must satisfy both minimum support threshold and a minimum confidence threshold. Such rules are called strong[2].

Support (X => Y) = P(X U Y)
Confidence (X => Y) = P(Y|X)
= Support (X U Y) / Support (X)

Let I be a set of items. If there is an item A $\subseteq$ I, A is a k-itemset if the cardinality of itemset A is k. The support of an itemset measures how often A occurs in the database. If Support (A) $\geq$ minsup, A is referred as frequent itemset. Informally, if the support of an itemset A satisfies a predefined minimum support threshold, then A is a Frequent Itemset (FI). An itemset is closed if there exists no proper super itemset B such that B has the same support as A. An itemset A is Frequent Closed Itemset (FCI) if A is both closed and frequent. An itemset A is a Maximal Frequent Itemset (MFI) if A is frequent and there exists no super itemset B such that A$\subset$ B and B is frequent. It is straight forward to observe that the following relation ship holds: MFI $\subseteq$ FCI $\subseteq$ FI[29]. Frequent pattern mining can be classified based on the completeness of patterns to be mined, the levels of abstraction involved in the rule set, the number of data dimensions involved in the rule, the types of values handled in the rule, the kinds of rules to be mined, the kinds of patterns to be mined. Algorithms for frequent itemset mining can be classified into three categories as Apriori-like algorithms, frequent pattern growth based algorithms such as FP-growth and algorithms that use the vertical data format.

The process of finding association rules has two separate phases[13]. In the first phase, we find set of Frequent Itemsets(FI) in the database. In the second phase, we set the set FI to generate "interesting patterns. In practice, the first phase is time consuming.

Wherever there are very long patterns (patterns containing many items) are present in the data, it is

Corresponding Author: A.M.J. Md. Zubair Rahman, Kongu Engineering College, Perundurai, Tamilnadu, India

often impractical to generate the entire set of frequent itemsets or closed itemsets[27]. There is much research on methods for generating all frequent itemsets efficiently[3,7,8,9,15,18,31] or just the set of maximal frequent itemsets[6,10,12,14,27]. Most of these algorithms use a breadth-first approach, i.e., finding all k-itemsets before considering (k+1) itemsets. However, with dense datasets such as telecommunications and census data, where there are many, long frequent patterns, the performance of these algorithms degrades incredibly.

This degradation is due to the following reasons: these algorithms perform as many passes over the database as the length of the longest frequent pattern. Secondly, a frequent pattern of length l, when l is large, the frequent itemset mining methods become CPU bound rather than I/O bound. In other words, it is practically unfeasible to mine the set of all frequent patterns for other than small l. There are two current solutions to the long pattern mining problem. The first one is to mine only the maximal frequent itemsets[19], which are typically orders of magnitude fewer than all frequent patterns. While mining maximal sets help understand the long patterns in dense domains, they lead to a loss of information; since subset frequency is not available maximal sets are not suitable for generating rules. The second is to mine only the frequent closed sets[19-21]. Closed sets are lossless in the sense that they uniquely determine the set of all frequent itemsets and their exact frequency. At the same time closed sets can themselves be orders of magnitude smaller than all frequent sets, especially on dense databases.

The large itemset problem is reasonably well solved at least for the case of very sparse sales transaction data, when the pattern lengths are short[6, 13]. An interesting analysis of the impact of different kinds of data on access costs has been provided in[18]. An Apriori-style algorithm with improved counting techniques using column wise data access for databases with a larger number of items has been also been discussed in the same study. We maintain that when the actual frequent patterns are wide, even the CPU-costs of any algorithm which is based on the Apriori-framework would be compromised by the investigation of all 2k subsets of frequent k-patterns. In such cases, the frequent itemset generation algorithms become CPU-bound. GenMax utilizes a backtracking search for efficiently enumerating all maximal patterns. GenMax uses a number of optimizations to quickly prune away a large portion of the subset search space. It uses a novel progressive focusing technique to eliminate non-maximal itemsets, and uses diffset propagation for fast frequency checking.

Mafia is good for mining a superset of all maximal patterns, GenMax is the method of choice for enumerating the exact set of maximal patterns. We further observe that there is a type of data, where MaxMiner delivers the best performance. We denote by Fk the set of frequent k-itemsets, and by FI the set of all frequent itemsets. A frequent itemset is called maximal if it is not a subset of any other frequent itemset. The set of all maximal frequent itemsets is denoted as MFI. Given a user specified miti-sup value our goal is to efficiently enumerate all patterns in MFI. Backtracking algorithms are useful for many combinatorial problems. 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 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.

Some of the algorithms in the literature such as MaxMiner avoid this by implementing look aheads[27], in which supersets of frequent patterns are used in order to prune off potential candidates in the search. Other innovative ideas for handling these problems are discussed in[23]. Recently, the merits of a depth-first approach have been recognized[6].

The database representation is also an important factor in the efficiency of generating and counting itemsets. Generating the itemset Z = (X U Y) refers to creating t(Z) = t(X) ∩ t(Y), and counting is the process of determining support(Z) in T. Most previous algorithms use a horizontal row layout, with the database organized as a set of rows and each row representing a transaction. The alternative vertical column layout associates with each item X a set of transaction identifiers (tids) for the set t(X). The vertical representation allows simple and efficient support counting

Basic Properties of Itemset-Tidset Pairs We use the concept of a closure operation[24,25] to check if a given itemset X is closed or not. We define a closure of an itemset X, denoted c(X), as the the smallest closed set that contains X. Recall that i(Y) is the set of items common to all the tids in the tid set Y , while t(X) are tids common to all the items in X. To find the closure of an itemset *X* we first compute the image of *X* in the transaction space to get t(X). We next map t(X) to its image in the itemset space using the mapping i to get i(t(X)). It is well know that the resulting itemset must be closed[25], i.e., c(X) = i ◦ t(X) = i(t(X)). It follows that an itemset X is closed if and only if X = c(X). For example the itemset ACW is closed since c(ACW) = i(t(ACW)) = i(1345) = ACW. The support of an itemset

X is also equal to the support of its closure, i.e., $\sigma(X) = \sigma(c(X))$[26, 27].

For any two nodes in the IT-tree, Xi × t(Xi) and Xj × t(Xj), if Xi ⊆ Xj then it is the case that t(Xj) ⊆ t(Xi). For example, for ACW ⊆ ACTW, t(ACW) = 1345 ⊆ 135 = t(ACTW). Let us define f : P(I) → N to be a one-to-one mapping from itemsets to integers. For any two itemsets Xi and Xj , we say Xi ≤ f Xj iff f(Xi) ≤ f(Xj). The function f defines a total order over the set of all itemsets. For example, if f denotes the lexicographic ordering, then itemset AC ≤ AD, but if f sorts itemsets in increasing order of their support, then AD ≤ AC if σ(AD) ≤ σ(AC). There are four basic properties of IT-pairs that CHARM leverages for fast exploration of closed sets. Assume that we are currently processing a node P × t(P) where [P] = {l1, l2, · · · , ln} is the prefix class. Let Xi denote the itemset Pli, then each member of [P] is an IT-pair Xi × t(Xi).

Theorem 1: Let Xi ×t(Xi) and Xj ×t(Xj) be any two members of a class [P],with Xi ≤ f Xj,wher e f is a total order (e.g., lexic ographic or support-based). The following four properties hold:

- If t(Xi) = t(Xj),then c(Xi) = c(Xj) = c(Xi U Xj)
- If t(Xi) ⊂ t(Xj),then c(Xi) ≠ c(Xj),but c(Xi) = c(Xi U Xj)
- If t(Xi) ⊃ t(Xj),then c(Xi) ≠ c(Xj),but c(Xj) = c(Xi U Xj)
- If t(Xi) ≠ t(Xj),then c(Xi) ≠ c(Xj) ≠ c(Xi U Xj)

In a thorough experimental evaluation, we first quantify the effect of each individual component on the performance of the algorithm. We then compare the performance of MAFIA against depth project, the most efficient previously known algorithm for finding maximal frequent itemsets[6]. Our results using some of the standard machine learning benchmark datasets indicate that MAFIA outperforms depth project by a factor of three to five on average. The main aim of developing this algorithm is to achieve CPU efficiency.

MATERIALS AND METHODS

The problem of mining frequent itemsets has been a topic of Intensive research[14, 26]. Since the number of such sets is huge, it is common and more efficient to restrict the search to closed item-sets[26], where a set is closed if all its supersets have strictly lower frequency in the database. The collection of frequent closed sets contains the same information as the overall collection of frequent item-sets, but is much smaller. There is also a growing interest in mining structured data, such as graphs, and more generally multi-relational databases, and the notion of closed sets has also been imported to this richer setup. Another variation exist between mining in a single interpretation (graph), or across multiple interpretations. Finally, some authors restrict the implication relation used in defining closures to range-restricted clauses only. In addition to these differences, the notion of a closed set can be coupled with a closure operator that takes a set and calculates its closure and there is more than one way to define such closures. The literature gives the impression that these different choices are unimportant and that algorithmic issues can be studied independently of the semantics. Our investigation shows that this impression is false and that semantics do matter.

Methods for finding the maximal elements include All-MFS[28], which works by iteratively attempting to extend a working pattern until failure. A randomized version of the algorithm that uses vertical bit-vectors was studied, but it does not guarantee every maximal pattern will be returned.

Max Miner[27] is another algorithm for finding the maximal elements. It uses efficient pruning techniques to quickly narrow the search. Max Miner employs a breadth first traversal of the search space; it reduces database scanning by employing a look ahead pruning strategy Depth Project[30] finds long itemsets using a depth first search of a lexicographic tree of itemsets, and uses a counting method based on transaction projections along its branches.

It returns a superset of the MFI and would require post-pruning to eliminate non-maximal patterns. FP growth[31] uses the novel Frequent Pattern tree (FP-tree) structure, which is a compressed representation of all the transactions in the database.

Mafia[29] is the most recent method for mining the MFI. Mafia uses three pruning strategies to remove non-maximal sets. The first is the look-ahead pruning first used in Max Miner. The second is to check if a new set is subsumed by an existing maximal set.

The most important category of approaches in multi-relational classification is ILP. Besides ILP, probabilistic approaches are also popular for multi-relational classification and modeling. The most important one is the probabilistic relational models (PRM's)[22,17] which is an extension of Bayesian networks for handling relational data. PRM's can integrate the advantages of both logical and probabilistic approaches for knowledge representation and reasoning. In[16] an approach is proposed to integrate ILP and statistical modeling for document classification and retrieval.

Given this conceptual framework, we can describe the most recent approaches to the maximal frequent

itemset problem. As a baseline, Apriori traverses the lattice in a pure breadth-first manner, discovering all frequent nodes at level k before moving to level (k+1); Apriori finds support information by explicitly generating and counting each node[13]. Max Miner performs a breadth-first traversal of the search space as well, but also performs look aheads to prune out branches of the tree. The look aheads involve superset pruning, using apriori in reverse (all subsets of a frequent itemset are also frequent). In general, look aheads work better with a depth-first approach, but Max Miner uses a breadth-first approach to limit the number of passes over the database. Depth Project performs a mixed depth-first traversal of the tree, along with variations of superset pruning[6]. Instead of a pure depth-first traversal, Depth Project uses dynamic reordering of children nodes. With dynamic reordering, the size of the search space can be greatly reduced by trimming infrequent items out of each node's tail. Also proposed in Depth Project is an improved counting method and a projection mechanism to reduce the size of the database. The other notable maximal pattern methods are based on graph-theoretic approaches. MaxClique and MaxEclat[10] both attempt to divide the subset lattice into smaller pieces ("cliques") and proceed to mine these in a bottom-up Apriori-fashion with a vertical data representation. The VIPER algorithm has shown a method based on a vertical layout can sometimes outperform even the optimal method using a horizontal layout[11]. Other vertical mining methods for finding FI are presented by Holsheimer[4] and Savasere *et al.*[5]. The benefits of using the vertical tid-list were also explored by Ganti *et al.*[3].

Proposed Work: 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. 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. We use hash data structure to represent this data format. Initially one hash is maintained for itemset and another hash for tidset. Different pointers are maintained as links between itemset and tidset as shown in the Fig. 1.

Let us consider the support to be 3. In such case, from the above structure the items I1, I2, I3 are frequent items (Table 1). So, only these items will be considered to next level.

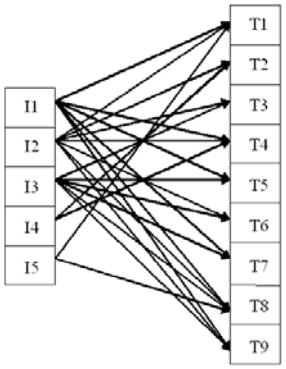

Fig. 1: Hash structure for itemset and tidset.

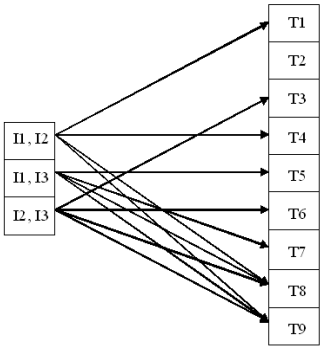

Fig. 2: Hash structure in second level

Table 1: The vertical data format of the transactional database D in first level

Itemset	Tidset
I1	T1, T4, T5, T7, T8, T9
I2	T1, T2, T3, T4, T6, T8, T9
I3	T3,T5,T6,T7,T8,T9
I4	T2,T4
I5	T1,T8

Now in the second level, intersection of tidset of all permutations of the frequent items is taken.

In the second level, the itemsets {I1, I2}, {I1, I3}, {I2, I3} are frequent itemsets. Now, total frequent itemsets are {I1, I2, I3, {I1,I2}, {I1,I3}, {I2, I3}}. From these, maximally frequent itemsets are {I1, I2}, {I1, I3}, {I2, I3}. This is shown in Fig. 2 and Table 2.

If we observe, the third level itemset {I1, I2, I3} has only 2 transactions {T8, T9} in tidset. So, this is not even frequent itemset. So, the frequent itemsets and maximally frequent itemsets obtained from the second level are the final result. It can observed that the number of levels increase if the support is less. If we increase the support, then number of levels decreases and so as the time to find MFI decreases.

Table 2: The vertical data format of the transactional database D in second level

Itemset	Tidset
I1, I2	T1, T4, T8, T9
I1, I3	T5, T7, T8, T9
I2, I3	T3, T6, T8, T9

The process will be continued till intersection can be taken. In this procedure we need not calculate the support of the itemset separately. It can be taken by the number of transactions in the tidset. Also the pruning can be done while finding the MFI itself, but not after finding FI completely. The proposed algorithm is given below.

HBMFI

Input:

D, a database of transactions Min_sup, the minimum threshold support

Output: M, Maximal frequent itemsets in D

Method:

- Count the support of each item in a given set of transactions
- if support(itemset) > = min_sup then
- FI[i] = itemset;
- find all permutations of the FI
- find the intersection of the transactions of each items in each permutation
- count the support of each FI
- MFI = FI - subsets of FI
- go to 2
- return MFI

The proposed algorithm performs better because MFI is being calculated directly before computing FI completely. At each level, after computation of FI, we are computing MFI also. So, the time taken to compute MFI is negligible. And also it shows that no separate pruning is required. Hash data structure can be maintained to store database. This makes easy in performing several tasks. As we are following vertical data format, support also need not be calculated separately. In this case, support is directly given by the number of transactions in the tidlist of each FI.

RESULTS

Figures 3-5 illustrate the results of comparing HBMFI to our implementation of MAFIA method, the state of the art method for finding maximal frequent items. Support is taken as X-axis and the time taken to find the MFI is taken as Y-axis.

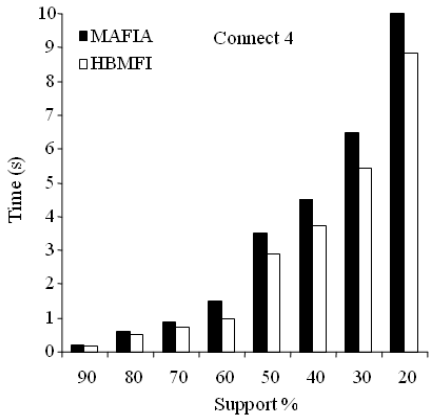

Fig. 3: Time comparison of MAFIA and HBMFI on connect - 4 dataset

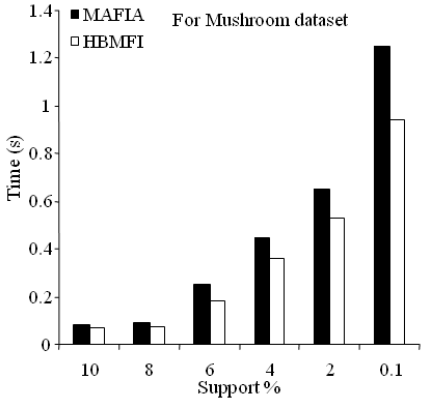

Fig. 4: Time comparison of MAFIA and HBMFI on mushroom dataset

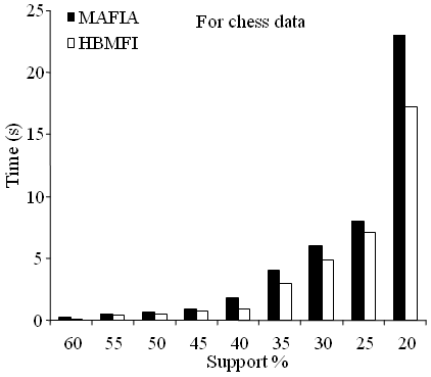

Fig. 5: Time comparison of MAFIA and HBMFI on chess dataset

The comparison of algorithms MAFIA and HBMFI is shown using different datasets in Fig. 3, Fig. 4 and Fig. 5.

The percentage in the improvement of the performance of the proposed algorithm, HBMFI is less in Chess dataset when compared to other datasets. The extremely low number of transactions and small number of frequent items at low supports muted the factors that HBMFI relies on to improve over MAFIA.

Both the algorithms scale linearly with the database size, but HBMFI is about 2 to 3 times faster than MAFIA. Thus we see that HBMFI performs better with large number of transactions and long itemsets.

DISCUSSION

The testing of this algorithm has been carried out on the real datasets containing large number of transactions and long itemsets like chess, mushroom, connect4. At the lowest supports tested, the longest databases have over 20 items, making any algorithm that examines all possible subsets of these patterns infeasible. This makes the task of finding the MFI computationally intensive despite the small size of the databases.

For Connect-4, the increased efficiency of itemset generation and support counting in MAFIA and HBMFI explains the improvement. Connect-4 contains an order of magnitude more transactions than the other two datasets, amplifying the advantage in generation and counting.

For Mushroom, the improvement is best explained by how the MFI is computed at each level and found directly without waiting for FI completely. This leads to a much greater reduction in the overall search space than for the other datasets, since the reductions is so great at highest levels.

CONCLUSION

We presented HBMFI, an algorithm for finding maximal frequent itemsets. Our experimental results demonstrate that HBMFI consistently outperforms MAFIA by a factor of 2-3 on average. The vertical data format representation of the database, the easy manipulations on hash data structure and directly computing MFI are the added advantages of this algorithm.

REFERENCES

1. Agrawal, R., T. Imielinski and A. Swami, 1998. Mining association rules between sets of items in very large databases. In the Proceedings of the ACM SIGMOD International Conference on Management of Data, May 25-28, Washington, D.C., US, pp: 207-216. http://doi.acm.org/10.1145/170035.170072.
2. Jiawei Han and Micheline Kamber, 2001. Data Mining: Concepts and Techniques. 1st Edn., Morgan Kaufmann pp: 500. ISBN-10: 1558604898.
3. Ganti, V., J. Gehrke and R. Ramakrishnan, 2000. DEMON: mining and monitoring evolving data. ICDE 2000, San Diego, CA., pp: 439-448. http://www-db.cs.wisc.edu/dbseminar/spring00/talks/demon_paper.pdf
4. Holsheimer M., M. Kersten, H. Mannila and H. Toivonen, 1995. A perspective on databases and data mining. Proceeding of the 1st International Conference on Knowledge Discivery and Data Mining, Aug. 1995, AAAI Press, Montreal, Canada, pp: 150-155, http://www.cs.helsinki.fi/research/fdk/datamining/pubs/kdd95.ps.gz
5. Savasere, A., E. Omiecinski and S. Navathe, 1995. An efficient algorithm for mining association rules in large databases. Proceedings of 21st International VLDB Conference on Very Large Data Bases, Sep. 11-15, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA ., pp: 432-444. http://portal.acm.org/citation.cfm?id=673300
6. Ramesh C. Agarwal, Charu C. Aggarwal and V.V.V. Prasad, 2001. A tree projection algorithm for generation of frequent itemsets. J. Parallel Distribut. Comput., 61: 350-371. DOI: 10.1006/jpdc.2000.1693
7. Agrawal, R. H. Mannila, R. Srikant, H. Toivonen and A.I. Verkamo, 1996. Fast Discovery of Association Rules. In: Advances in Knowledge Discovery and Data Mining, Usama Fayyad, M. G.P. Shapiro, P. Smyth, and R. Uthurusamy, (Eds.). AAAI/MIT Press, Menlo Park, CA, USA., pp: 307 -28 I. SBN:0-262-56097-6
8. Aggarwal, C.C. and P.S. Yu, 1998. Mining large itemsets for association rules. Bull. IEEE Comput. Soc. Technical Committee Data Eng.,: 23-31. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.306
9. Aggarwal, C.C. and P.S. Yu, 1998. Online generation of association rules. In Proceedings of the 14th International Conference on Data Engineering, Feb. 23-27,IEEE Xplore, Orlando, FL, USA., pp: 402-411. DOI: 10.1109/ICDE.1998.655803
10. Mohammed J. Zaki, 2000. Scalable algorithms for association mining. IEEE Trans. Knowl. Data Eng., 12: 372-390. DOI: 10.1109/69.846291.

11. Shenoy, P., J. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa and D. Shah, 2000. Turbo-charging vertical mining of large databases. Proceeding of ACM SIGMOD International Conference on Management of Data, June 2000, Dallas, Texas, USA,pp:22-33. http://doi.acm.org/10.1145/335191.335376
12. Gunopulos, D., H. Mannila and S. Saluja, 1997. Discovering all most specific sentences by randomized algorithms. In Proceedings of the 6th International Conference on Database Theory, Jan. 08-10, Springer-Verlag London, UK., pp: 215-229. http://portal. acm.org/citation.cfm?id=656097
13. Agrawal, R. and R. Srikant, 1994. Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Databases, Sep. 12-15, Santiago de chile, Chile, pp: 487-499. DOI: 10.1.1.40.7506.
14. Lin, D.I. and Z.M. Kedem, 1998. Pincer search: A new algorithm for discovering the maximum frequent sets. In Proceedings of the 6th International Conference on Extending Database Technology, Mar. 23-27, Springer-Verlag London, UK.,pp:105-119. http://portal.acm.org/citation.cfm?id=645338.650396.
15. Park, J.S., M.S. Chen, P.S. Yu, 1995. An effective hash based algorithm for mining association rules. ACM SIGMOD Record, 24: 175-186. http://doi.acm.org/10.1145/568271.223813
16. Rin Popescul and Lyle H. Ungar and Steve Lawrence and David M. Pennock, 2002. Towards structural logistic regression: Combining relational and statistical learning. In Proceedings of KDD-2002 Workshop on Multi-Relational Data Mining 02, ACM, Alberta, Canada, pp: 130-141. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.6235
17. Taskar, B., E. Segal and D. Koller, 2001. Probabilistic classification and clustering in relational data. In Proceedings of the 17th International Joint Conference on Artificial Intelligence 01, Lawrence Erlbaum Associates Ltd, USA.,pp:870-876. http://direct.bl.uk/bld/PlaceOrder.do?UIN=107907671&ETOC=RN&from=searchengine
18. Dunkel, B. and N. Soparkar, 1999. Data organization and access for effcient data mining. In the Proceedings of the 15th International Conference on Data Engineering, Mar. 23-26, IEEE Xplore, Sydney, NSW, Australia, pp: 522-529. DOI: 10.1109/ICDE.1999.754968
19. Mohammed Zaki, J. and C.J. Hsiao, 2002. CHARM: An efficient algorithm for closed itemset mining. In Proceedings of SDM'02 Conference, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.520
20. Bastide, Y., R. Taouil, N. Pasquier, G. Stumme and L. Lakhal, 2000. Mining frequent patterns with counting inference. ACM SIGKDD Explorations Newsletter,2:66-75. http://doi.acm.org/10.1145/380995.381017
21. Pasquier, N., Y. Bastide, R. Taouil and L. Lakhal, 1999. Discovering frequent closed itemsets for association rules. In Proceedings of the 7th International Conference on Database Theory, Jan. 10-12, Springer-Verlag London, UK., pp: 398-416. http://portal. acm.org/citation.cfm?id=645503.656256
22. Getoor, L., N. Friedman, D. Koller and B. Taskar, 2001. Learning probabilistic models of relational structure. In Proceedings of International Conference on Machine Learning (ICML'01), Williamtown,MA,pp:170-177. http://direct.bl.uk/bld/PlaceOrder.do?UIN=100556121&ETOC=RN&from=searchengine
23. Zaki, M.J., S. Parthasarathy, M. Ogihara and W. Li, 1997. New algorithms for fast discovery of association rules. In Proceeding of the 3rd International Conference on Knowledge Discovery and Data Mining 97, AAAI Press, pp: 283-286. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.5143
24. Zaki, M.J., 2000. Generating non-redundant association rules. In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 20-23, Boston, Massachusetts, US., pp: 34-43. http://doi.acm.org/10.1145/347090.347101
25. Ganter B. and R. Wille, 1999. Formal Concept Analysis: Mathematical Foundations. 1st Edn., Springer-Verlag, USA., pp: 284. ISBN-10: 3540627715.
26. Gouda, K. and M.J. Zaki, 2001. Efficiently mining maximal frequent itemsets. In the Proceedings of International Conference on Data Mining, Nov. 29-Dec. 02 2001, IEEE Computer Society Washington, DC, USA., pp: 163-170. http://portal.acm.org/citation.cfm?id=645496.658047&coll=GUIDE&dl=GUIDE
27. Bayardo, R.J., 1998. Efficiently mining long patterns from databases. In the Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, June 001-04, Washington, United States, pp: 85-93. http://doi.acm.org/10.1145/276304.276313

28. Gunopulos, D., H. Mannila and S. Saluja, 1997. Discovering all the most specific sentences by randomized algorithms. In Intenational Conference on Database Theory, Jan. 08-10, Springer-Verlag London,UK.,pp:215-229. http://portal.acm.org/citation.cfm?id=645502.656097
29. Burdick, D., M. Calimlim and J. Gehrke, 2001. MAFIA: A maximal frequent itemset algorithm for transactional databases. In International Conference on Data Engineering, Apr. 02-06, IEEE Computer Society Washington, DC, USA pp:443-452. http://portal.acm.org/citation.cfm?id=645484.656386&coll=GUIDE&dl=GUIDE.
30. Agrawal, R., C. Aggarwal and V. Prasad, 2000. Depth first generation of long patterns. In the Proceedings of the 6th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, Aug. 20-23, Boston, Massachusetts, United States, pp: 108-118. http://doi.acm.org/10.1145/347090.347114
31. Han, J., J. Pei and Y. Yin, 2000. Mining frequent patterns without candidate generation. In the Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 15-18, Dallas, Texas, United States, pp: 1-12. http://doi.acm.org/10.1145/342009.335372