IMPROVING PERFORMANCE OF FREQUENT ITEMSET ALGORITHM

Kuldeep Singh Malik*

Neeraj Raheja**

ABSTRACT

Frequent itemset mining leads to the discovery of associations among items in large transactional database. The apriori algorithm adopts candidate generation and testing which is easy to implement but candidate generation and support counting is very expensive in this. In the fp-growth, there is no candidate generation and requires only 2 passes over the database but in this fp-tree is very expansive to built and support is counted only when entire dataset is added to fp-tree. In this paper, I introduce enhancedfp which does its work without prefix tree or any other complicated data structure and there is no re-representation of transaction is necessary. Finally I compare the performance of enhancedfp with fp-growth and apriori.

*M.Tech (CSE), M.M.E.C,Mullana (Haryana, India)

**Asst. Prof., M.M.E.C,Mullana (Haryana, India)

I. INTRODUCTION

Association rule mining searches [2] for relationships between items in a dataset. It finds association among set of items in transactional database. Each transaction is a list of items. Association rules is in form A⇒B which means customer buys A also tends to buy B. To mine association rule, basic concepts of support and confidence are needed. **Support** s is the probability that a transaction contain (X, Y).**Confidence** C is the measure of the strength of the association rule, suppose the confidence of the association rule x⇒y is 90%, it means that 90% of the transactions that contain X also contain Y together. Also minimum support and minimum confidence is needed to eliminate the unimportant association rules. Such that the association rules is hold when it is greater than the minimum support and minimum confidence.

T_id	items
100	a,b,c
200	a,c
300	a,d
400	b,e,f

Equation for support and confidence:

Support (A⇒ B) =Probability (A∩B).

Confidence (A⇒B) =Probability (B/A).

Let the min_support and min_confidence are 50%.for association rule a⇒c, support (a, c) =2/4*100%=50%.Confidence=Support (a, c)
/Support (a) =50%/75%=66.6%, means that customer buys a also have 66.6% chance to buy c. In this paper on the basis of study of the existing data mining algorithm apriori and fp-growth and according to disadvantage of them in transactional database, an enhancedfp is presented the enhancedfp avoids the problem of complex data structure and the experiments results shows that the performance of the enhancedfp is better than fp-growth and apriori.

II. APRIORI ALGORITHM

The apriori algorithm is firstly purposed by R.aggrwal and R.srikant [6] in 1994 for mining frequent itemset. . Apriori has monotonicity property which states-all non-empty subsets of a frequent itemset must also be frequent Apriori algorithm contains a number of passes over the database .In this firstly the algorithm scan the database to find the set of frequent 1-itemset and do count for each item and collecting those items that satisfy the minimum _support. The resulting set is denoted by L1.Next L1 is used to find L2(frequent 2-itemset) which is used to find L3 and so on, until no more frequent k-itemsets can be found. The finding of each k requires on full scan of a database.

ADVANTAGES:

1. Use large itemset.
2. Easy to implement.
3. Easily parallelized.

DISDVANTAGE:

1. It may need to generate a huge no of candidate sets. So its generation is expensive.
2. Assumes transactional database is memory resident.
3. Support count is expensive because require many database scan.

III. FP-GROWTH

FP-GROWTH approach is based on divide and conquers strategy for producing the frequent itemsets. It reduces the multiple scan over database Fp-growth is mainly used for mining frequent itemsets without candidate generation.

Major steps in Fp-growth is-

Step1- It firstly compresses the database showing frequent itemset in to Fp-tree .Fp-tree is built using 2 passes over the dataset.

First pass: frequent 1-itemset is same as in apriori frequent1-itemset.

Second pass: in this pass, the fp-tree is constructed. Nodes show the items and each node a counter. Each transaction is processed in descending order &creates a branch for each transaction. Fp reads one transaction at a time. When the transactions shares the same items (have same prefix).The path can overlap & counters are incremented. Pointers are maintained between nodes contains the same item, creating singly linked list {dotted lines).more the overlap more the compression

Step2: It divides the Fp-tree in to a set of conditional database and mines each database separately, thus extract frequent itemsets from Fp-tree directly.

ADVANTAGES:

1. It compresses the database.
2. Require only 2 pass over database.
3. There is no candidate generation.

4. Faster than apriori.

5. Reduces search cost

DISADVANTAGE:

1. It may not fit in main memory.

2. FP tree is expensive to build.

I) takes time to build but once built frequent itemset can be obtained easily.

II) Support can only be calculated once the entire dataset is added to fp-tree.

IV. ENHANCEDFP

As the major disadvantage of fp-growth is construction of fp-tree as it takes a lot of time to build and also expensive to built. Enhanced fp does its work without any complicated data structure, processing the transactions directly. Its main strength is simplicity of its structure. In this all the work is done in one simple recursive function. Enhancedfp is based on a step by step elimination of items from the transaction database together with a recursive processing of transaction subsets.

Preprocessing of transactional database

Supports of the items are determined in initial scan. All the items whose support is less than the predefined minimum support are discarded from the transaction because they all are infrequent items. All the items in each transaction are sorted in ascending order w.r.t to their support in the database. Suppose the transactional database is:

a d

a c d e

b d

b c d g

b c f

a b d

b d e

b c d e

b c

a b d f

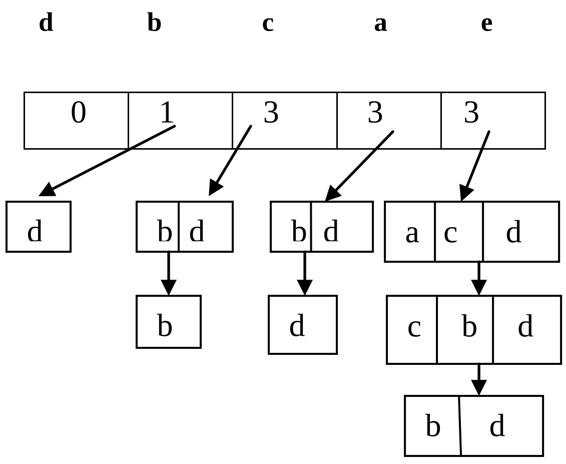
Suppose the predefined minimum support is 3.

g	1
f	2
e	3
a	4
c	5
b	8
d	8

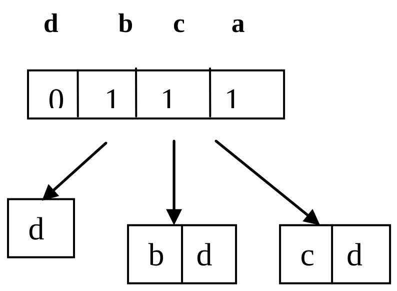
The items g and f is discarded because their support is less than minimum support. Transaction sorted lexicographically in descending order.

a d		e a c d
e a c d		e c b d
b d		e b d
c b d	$\longrightarrow$	a b d
c b		a b d
a b d		a d
e b d		c b d
e c b d		c b
c b		c b
a b d		b d

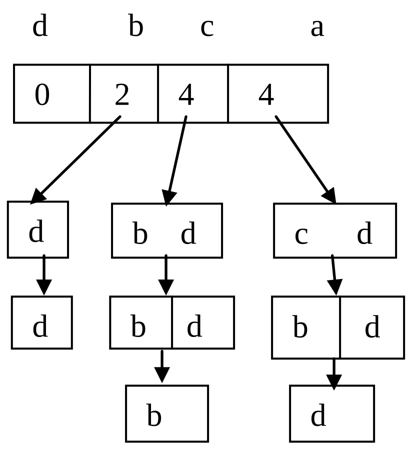
Each transaction is represented as a simple array of item identifiers. The transaction database is changed into a set of transaction lists, with one list for each item. Each element of which contains a support counter and a pointer to the head of the list. The list elements themselves consist only of a successor pointer and a pointer to the transaction. The transactions are inserted one by one into this structure by simply using their leading item as an index. However, the leading item is removed from the transaction, that is, the pointer in the transaction list element points to the second item.



initial database



Prefix e



e eliminated

Before a transaction list is processed, however, its support counter is checked, and if it exceeds the user-specified minimum support, a frequent item set is reported consisting of the item associated with the list and a possible prefix associated with the whole list array.

Single transaction list is processed as follows: for each list element the leading item of its transaction is retrieved and used as an index into the list array; then the element is added at the head of the corresponding list. In such a reassignment, the leading item is also removed from the transaction, which can be implemented as a simple pointer increment. In addition, a copy of the list element (with the leading item of the transaction already removed by the pointer increment) is inserted in the same way into an initially empty second array of transaction lists.

Since the elements of a transaction list all share an item, this second array collects the subset of transactions that contain a specific item and represents them as a set of transaction lists. This set of transaction lists is then processed recursively. After the recursion the next transaction list is reassigned, copied, and processed in a recursive call and so on. a list element representing a transaction that contains only one item is neither reassigned nor copied, because the transaction would be empty after the removal of the leading item. Instead only the counter in the lists array element is incremented as an indicator of such list elements.

ADVANTAGES

1. Simple data structure and processing scheme.
2. Fastest than apriori and fp-growth.

V. EXPERIMENTAL RESULTS

T40I10D100k is synthetic data resemble market basket data with short frequent patterns. Pumsb is a real dataset contains census data for population and housing. Mushroom is real data which are dense in long frequent patterns. All the implementations is done in c and compiled in Visual C++. All experiments are performed on 2.60GHZ Pentium [R] Dual-Core CPU with 0.99 GB of memory, running Microsoft Windows XP Professional Version 2002.All times shown include time for outputting all the frequent itemset.

Results of apriori,fp-growth and enhancedfp on mushroom.dat.

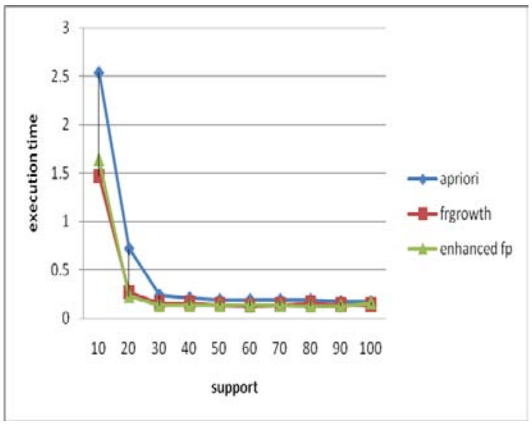


Figure 1

Results of apriori, fp-growth and enhancedfp on pumsb.tab.

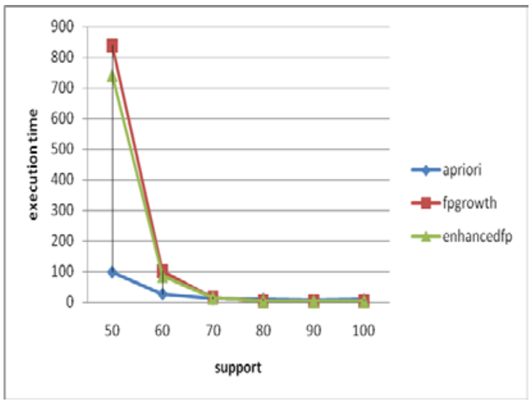


Figure 2

Results of apriori,fp-growth and enhancedfp on T40I10D100K.dat.

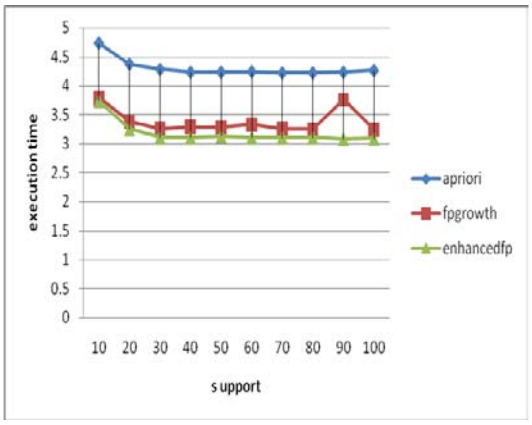


Figure 3

I run implementations of apriori,fp-growth and enhancedfp on three datasets mushroom,pumsb andT40I10D100K. Fig.1 to Fig3 shows the execution time of implementations over the various support.The blue line refers to apriori algorithm.The red line refers to fp-growth algorithm.The green line refers to enhancedfp.

Figure1. Shows the running time of the compared algorithms on mushroom data with different minimum supports represented by percentage of the total transactions. Under large minimum supports, enhancedfp run faster than fp-growth and aprirori as well as under small minimum supports. Thus on the dataset mushroom performance of enhancedfp is better than apriori,fp-growth.

Figure2. Shows the running time of the compared algorithms on pumsb data with different minimum supports represented by percentage of the total transactions. Under large minimum supports, enhancedfp run faster than fp-growth and aprirori, while under small minimum supports performance of apriori is better. Thus on the dataset pumsb performance of enhancedfp is better than apriori,fp-growth

Figure3. Shows the running time of the compared algorithms on T40I10D100K data with different minimum supports represented by percentage of the total transactions. Under large minimum supports, enhancedfp run faster than fp-growth and aprirori as well as
under small minimum supports. Thus on the dataset T40I10D100K performance of enhancedfp is better than apriori,fp-growth

VI. CONCLUSION

In this paper I introduced enhancedfp,which does its work without any complex data structure.By comparing it to frequent itemset mining algorithms apriori and fp-growth the strength of enhancedfp is analyzed. As the
Experimental results show,this implementa-
ion clearly outperforms apriori and fp-growth by some extent.

VII. REFERENCES

[1] A. Swami, T. Imielienski, R. Agrawal Mining Association Rules between Sets of Items in Large Databases. Proc. Conf. on Management of Data, 207–216. ACM Press, New York, NY, USA 1993.

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

[3] A. Savasere, E. Omiecinski, S. Navathe, An efficient algorithm for mining association rules in large databases, Proceedings of 21th VLDB Conference, Zurich, Switzerland, 2006 pp. 432–444.

[4] D. Yang , H. Lu, J. Pei, J. Han, S. Nishio, S. Tang, and D. Yang. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. IEEE Conf. on Data Mining (ICDM’01, San Jose, CA), 441–448. IEEE Press, Piscataway, NJ, USA 2001.

[5] H. Pei, J. Han, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In: Proc. Conf. on the Management of Data (SIGMOD’00, Dallas, TX). ACM Press, New York, NY, USA 2000.

[6] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", 2nd Edition, Morgan Kaufmann Publishers, August 2006

[7] M. Zaki, M. Ogihara, S. Parthasarathy, and W. Li. New Algorithms for Fast Discovery of Association Rules. Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD’97), 283–296. AAAI Press,
Menlo Park, CA, USA 1997.

[8] Grahne, G., & Zhu, J.”Fast Algorithm for frequent Itemset Mining Using FP-Trees”, IEEE Transactions on Knowledge and Data Engineer, Vol.17, NO.10, 2005.