Horizontal Fragmentation for Most Frequency Frequent Pattern Growth Algorithm

Media NoamanSolagh
Computer Science Dept. Education Col., Al-Mustansiriyah Uni. Baghdad, Iraq medonoman@yahoo.com

Enas Mohammed Hussien
Computer Science Dept. Education Col., Al-Mustansiriyah Uni. Baghdad, Iraq enas.seead38@yahoo.com

Abstract: Data mining is become very important at the present time, especially with the increase in the area of information it's became huge, so it was necessary to use data mining to contain them and using them, one of the data mining techniques are association rules here using the Pattern Growth method kind enhancer for the apriori. The pattern growth method depends on fp-tree structure, this paper presents modify of fp-tree algorithm called HFMFFP-Growth by divided dataset and for each part take most frequent item in fp-tree so final nodes for conditional tree less than the original fp-tree. And less memory space and time.

Keywords: frequent pattern mining, association rules, algorithm, performance modification

1. Introduction

Data Mining is also called as Knowledge discovery in database (KDD).It helps to explore, analyze and then finally retrieve the data.

Data Mining has many application areas. One of the Most important areas are association rules.[1].

The key of mining association rules is to find frequent itemsets, There are various serial algorithms for mining association rules, such as Apriori. However, the database for mining association rules is generally large, traditional serial algorithms cost much time.[2]. In order to improve efficiency mining association rules based on fp-tree by using FP-growth algorithm.[2].

The goal of Association rule Mining is to find all rules having-
Support $\geq$ minsup threshold
Confidence $\geq$ minconf threshold
Rules that satisfy both minsup and minconf are called strong rules.[1].

Frequent pattern mining techniques can also be extended to solve many problems, Thus the effective and efficient frequent pattern mining is an important and interesting research problem.[3].

One of the currently fastest and most popular algorithms forfrequent item set mining is the FP-growth algorithm [4]. It is based on a prefix tree representation of the given database of transactions (called an FP-tree), which can save considerableamounts of memory for storing the transactions. The basic idea of the FP-growth algorithm can be described as a recursive elimination scheme: in a preprocessing step delete all items from the transactions that are not frequent individually, do not appear in a user-specified minimum number of transactions. Then select all transactions that contain the least frequent item (least frequent among those that are frequent) and delete this item from them.[4].

Similar to several other algorithms for frequent item set mining, like, for example, Apriori, FP-growth preprocesses the transaction database as follows: in an initial scan the frequencies of the items (support of single element item sets) are determined. All infrequent items—that is, all items that appear in fewer transactions than a user-specified minimum number—are discarded from the transactions, since, obviously, they can never be part of a frequent item set. [3][4].

In addition, the items in each transaction are sorted, so that they are in descending order w.r.t. their frequency in the database. Although the algorithm does not depend on this specific order, experiments showed that it leads to much shorter execution times than a random order. An ascending order leads to a particularly slow operation in my experiments, performing even worse than a random order.[5][6].

The FP-tree is a prefix-tree structure that stores information about each frequent 1-itemset, in which the items are arranged in order of decreasing support value. Then, the mining process is transformed to mine the FP-tree.[7].

2. Research work.

In 2010, SLP-Growth Algorithm, this paper proposes a scalable trie-based algorithm named SLP-Growth. This algorithm generates the significant patterns using interval support and determines its correlation. Experiments with the real datasets show that the SLP-algorithm can discover highly positive correlated and significant of least association. Indeed, it also outperforms the fast FP-Growth algorithm up to two times, thus verifying its efficiency , By definition, least item is an itemset whose rarely found in the database but still can produce interesting and potentially valuable ARs. These rules are very important in discovering rarely occurring but significantly important, such as air pollution detection, critical fault detections, network intrusions, etc, Highly correlated least ARs are referred to the itemsets that its frequency does not satisfy a minimum support but are very highly correlated. ARs are classified as highly correlated if it is positive correlation and in the same time fulfils a minimum degree of predefined correlation , A new algorithm named Significant Least Pattern Growth (SLP-Growth) to extract these ARs is proposed. The proposed algorithm imposes interval support to capture all least itemsets family first before continuing to construct a significant least pattern tree (SLP-Tree). The correlation technique for finding relationship between itemset is also embedded to this algorithm [8].

In 2013, Performance Measure of Similis and FP-Growth Algorithm,in this study show the mining association rules is one of the main application areas of Data Mining, here use market basket dataset,
at present there are various Association Rules Algorithms are in market. This paper define the survey done on various algorithms of Association Rules of Data Mining and also compare two main algorithms-Similis Algorithm and FP-Growth Algorithm depending upon the different criteria that included in paper.[1].

In 2013, FP-Growth Approach to Mining Association Rules, in this paper of data mining researchers implements lots of algorithms for improving theperformance of mining. This work is also related to that strategy. This work, introduce an idea in this field.Here use Sampling Technique to convert text document in to the appropriate format. This format containsdata in the form of word and topic of word. This format take as a input in FP-Growth algorithm for givensupport value and get association rules of that transaction data, and after getting association rules applyclustering process and then get clusters for that association rules.[9].

In 2014, MS-FP-Growth, In this study suggested a new way to find association rules and through the application of a set of tests based on monitoring the difference between increasing and decreasing the minsup value , The main objective of these algorithms is to produce interesting knowledge from low level to high level. In this context, they propose to introduce the multi-support approach in the FP-Growth algorithm, This variant is applied in three cases, In the first case, we increase the minsup value from one level to another, In the second case, we decrease the minsup value between levels, In the last and third case, we vary the minsup value randomly increasing and decreasing The aim of this section is to study the impact of multi-supports on number of Association Rules (AR) generated [10].

3.Frame Work

Apriori-like algorithms expensively handle a great number of candidates. Additionally, it repeated scanning of the database is tedious. Therefore, Han et al. developed an efficient FP-tree based method, FP-growth, for mining frequent itemsets without generating candidates; this approach scans the database only twice [7, 8].

3.1 FP-growth

first discovers all frequent 1-itemsets and then establishes a compact treestructure, called an FP-tree (frequent-pattern tree). The FP-tree is a prefix-tree structurethat stores information about each frequent 1-itemset, in which the items are arrangedin order of decreasing support value. Then, the mining process is transformed to mine the fp-tree.[7].

Example1:the transaction database in Table 1 with a minimum support threshold 30%,FP-growth scans the database to discover all frequent1-itemsets and sorts these 1-itemsets in order of descending frequency of occurrence.

Table 1: example of database

TID	Transactions	Frequent itemset (orderd)
001	A1 B1 C1 D1 F1 I1 J1 K1	A1,D1,F1,C1,I1,J1,B1,K
002	A1 B2 C1 D1 F1 G H I1 J2 K2	A1,D1,F1,B2,C1,G,I1,K2,H,J
003	A1 B2 C2 D1 F1 G I1 J1 K2 L	A1,D1,F1,B2,G,I1,K2,J1,L,C
004	A1 B2 C1 D1 F1 G H I2 J2 K2 L	A1,D1,F1,B2,C1,G,K2,H,J2,L,I

The root of the tree is created first, and labeled with null,Next, the database is scanned again. The first transaction is employed to The establish thebranches of tree. In figure1.Construction of fp-tree.

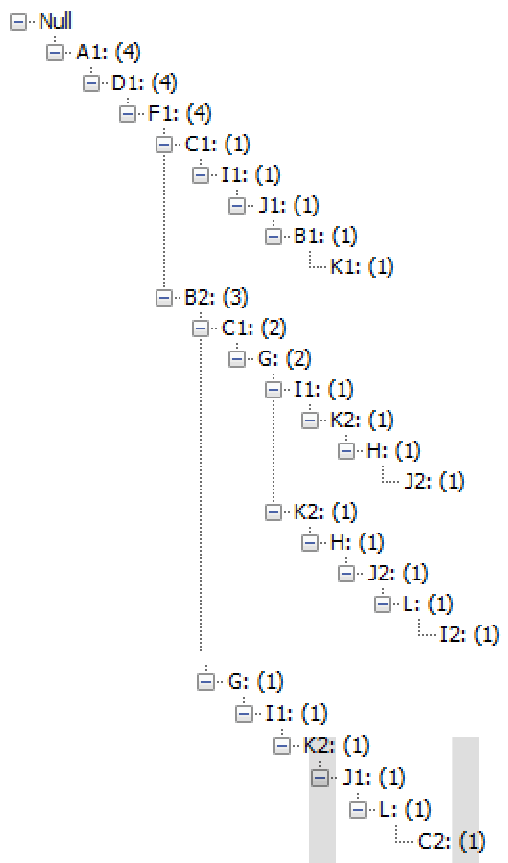


Figure1.constrection of FP-tree

Consider the example in Table 1 and minSup = 30% ,The mining process begins from the bottom of theheader table, and moves toward the top the process of generating the conditional FP-treeterminates will No more frequent itemsets can be generated in this conditionalfp-tree.In table 2 consider mining process.

Table2. Mining rules

Items	**conditional pattern base**	**conditional fp-tree**

A1	{(:4)}	{()} | A1
D1	{(A1,:4)}	{(A1:4,)} | D1
F1	{(A1,D1,:4)}	{(A1:4,D1:4,)} | F1
C1	{(A1,D1,F1,:1),(A1,D1,F1,B2,:2)}	{(A1:3,D1:3,F1:3,)} | C1
I1	{(A1,D1,F1,C1,:1),(A1,D1,F1,B2,C1,G,:1),(A1,D1,F1,B2,G,:1)}	{(A1:3,D1:3,F1:3,C1:3,B2:3,G:3,)} | I1
J1	{(A1,D1,F1,C1,I1,:1),(A1,D1,F1,B2,G,I1,K2,:1)}	{(A1:2,D1:2,F1:2,C1:2,I1:2,)} | J1
B1	{(A1,D1,F1,C1,I1,J1,:1)}	{(A1:1,D1:1,F1:1,C1:1,I1:1,J1:1,)} | B1
K1	{(A1,D1,F1,C1,I1,J1,B1,:1)}	{(A1:1,D1:1,F1:1,C1:1,I1:1,J1:1,B1:1,)} | K1
B2	{(A1,D1,F1,:3)}	{(A1:3,D1:3,F1:3,)} | B2
G	{(A1,D1,F1,B2,C1,:2),(A1,D1,F1,B2,:1)}	{(A1:3,D1:3,F1:3,B2:3,C1:3,)} | G
K2	{(A1,D1,F1,B2,C1,G,I1,:1),(A1,D1,F1,B2,G,I1,:1),(A1,D1,F1,B2,C1,G,:1)}	{(A1:3,D1:3,F1:3,B2:3,C1:3,G:3,I1:3,)} | K2
H	{(A1,D1,F1,B2,C1,G,I1,K2,:1),(A1,D1,F1,B2,C1,G,K2,:1)}	{(A1:2,D1:2,F1:2,B2:2,C1:2,G:2,I1:2,K2:2,)} | H
J2	{(A1,D1,F1,B2,C1,G,I1,K2,H,:1),(A1,D1,F1,B2,C1,G,K2,H,:1)}	{(A1:2,D1:2,F1:2,B2:2,C1:2,G:2,I1:2,K2:2,H:2,)} | J2
L	{(A1,D1,F1,B2,G,I1,K2,J1,:1),(A1,D1,F1,B2,C1,G,K2,H,J2,:1)}	{(A1:2,D1:2,F1:2,B2:2,G:2,I1:2,K2:2,J1:2,)} | L
C2	{(A1,D1,F1,B2,G,I1,K2,J1,L,:1)}	{(A1:1,D1:1,F1:1,B2:1,G:1,I1:1,K2:1,J1:1,L:1,)} | C2
I2	{(A1,D1,F1,B2,C1,G,K2,H,J2,L,:1)}	{(A1:1,D1:1,F1:1,B2:1,C1:1,G:1,K2:1,H:1,J2:1,L:1,)} | I2

3.2 Horizontal Fragmentation of most frequent FP-growth algorithm

This method was proposed based on the problems that were apply in the MFFP-growth proposed algorithm. in the first proposal, it was observed when apply MFFP-growth algorithm on huge dataset along time-consuming in execution approximation to FP-growth time.

So it was necessary to find a solution to this problem and overcome the time spent during the implementation.HFMFFP-growth is a hyper method between horizontal fragmentation for dataset with MFFP-growth algorithm and this step can be summarized as follow:

Algorithm 1: HFMFFP-growth algorithm

```
Input:
    D, a database of transaction;
    min sup, the minimum support count threshold.
Output:
HFMFFP-tree.
Procedure:
  1. Divide D to equal n size → d1….dn
  2. scan the dn once,Store all the frequent 1-itemsets F and their individual support
     values in each sub D
  3. Sort F in order of decreasing support value to generate a list of frequent 1-itemsets
     H.
  4. Select the most frequent 1-itemset (MF) to generate the root node of the MFFP-
     tree, for each dn
  5. For each transaction, T⊂dn, perform the following;
      1. Select the frequent 1-itemsets in T and sort them in order of H. Let the frequent
      1-itemsets list in T be {m1, m2, ..., mn}.
```

2. If *m*1 == *MF*, call function Insert_node(*m*2, root); else call function Insert_node(*m*1, root).

Figure 2: HFMFFP-tree construction

Example 2: Consider the transaction database in Table 1 with a minimum support threshold of 30%. Use HFMFFP-growth scans each sub database to discover all frequent1-itemsets and sorts these 1-itemsets in order of descending frequency of occurrence.

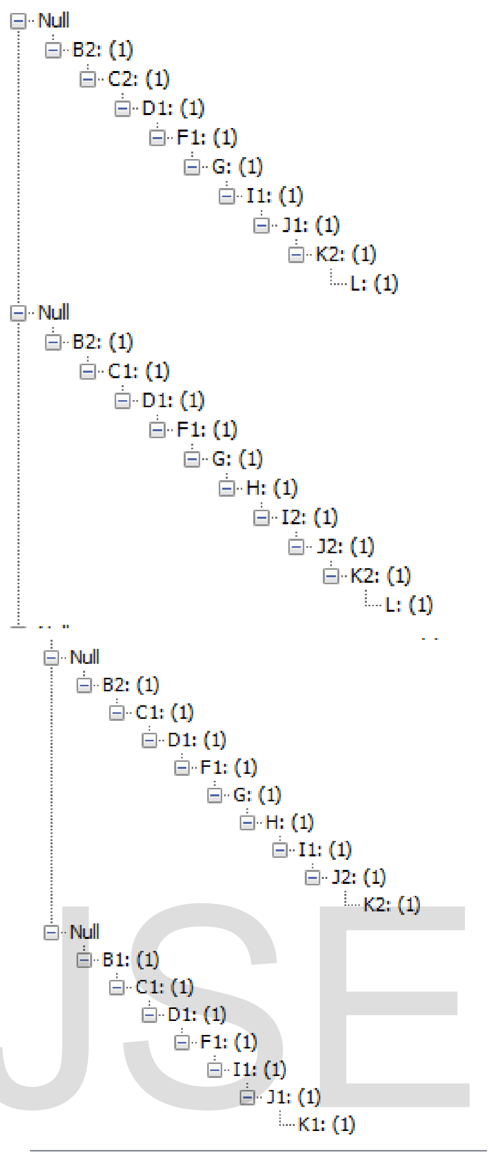


Figure 3 HFMFFP-tree construction

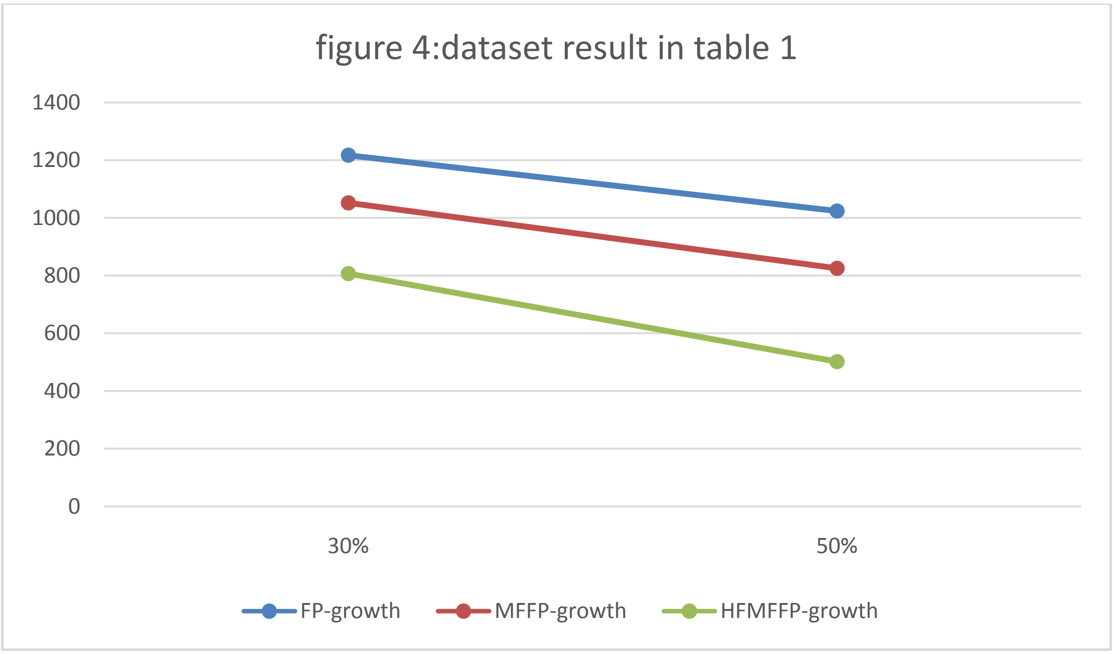
Table 3 generating association rules

Items	conditional pattern base	conditional fp-tree
Tree #1		
B2	{(:1)}	{()} | B2
C2	{(B2,:1)}	{(B2:1,)} | C2
D1	{(B2,C2,:1)}	{(B2:1,C2:1,)} | D1
F1	{(B2,C2,D1,:1)}	{(B2:1,C2:1,D1:1,)} | F1
G	{(B2,C2,D1,F1,:1)}	{(B2:1,C2:1,D1:1,F1:1,)} | G
I1	{(B2,C2,D1,F1,G,:1)}	{(B2:1,C2:1,D1:1,F1:1,G:1,)} | I1
J1	{(B2,C2,D1,F1,G,I1,:1)}	{(B2:1,C2:1,D1:1,F1:1,G:1,I1:1,)} | J1
K2	{(B2,C2,D1,F1,G,I1,J1,:1)}	{(B2:1,C2:1,D1:1,F1:1,G:1,I1:1,J1:1,)} | K2
L	{(B2,C2,D1,F1,G,I1,J1,K2,:1)}	{(B2:1,C2:1,D1:1,F1:1,G:1,I1:1,J1:1,K2:1,)} | L
Tree #2		
B2	{(:1)}	{()} | B2
C1	{(B2,:1)}	{(B2:1,)} | C1
D1	{(B2,C1,:1)}	{(B2:1,C1:1,)} | D1
F1	{(B2,C1,D1,:1)}	{(B2:1,C1:1,D1:1,)} | F1
G	{(B2,C1,D1,F1,:1)}	{(B2:1,C1:1,D1:1,F1:1,)} | G
H	{(B2,C1,D1,F1,G,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,)} | H
I2	{(B2,C1,D1,F1,G,H,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,H:1,)} | I2
J2	{(B2,C1,D1,F1,G,H,I2,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,H:1,I2:1,)} | J2
K2	{(B2,C1,D1,F1,G,H,I2,J2,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,H:1,I2:1,J2:1,)} | K2
L	{(B2,C1,D1,F1,G,H,I2,J2,K2,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,H:1,I2:1,J2:1,K2:1,)} | L
Tree #3		
B2	{(:1)}	{()} | B2
C1	{(B2,:1)}	{(B2:1,)} | C1
D1	{(B2,C1,:1)}	{(B2:1,C1:1,)} | D1
F1	{(B2,C1,D1,:1)}	{(B2:1,C1:1,D1:1,)} | F1
G	{(B2,C1,D1,F1,:1)}	{(B2:1,C1:1,D1:1,F1:1,)} | G
H	{(B2,C1,D1,F1,G,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,)} | H
I1	{(B2,C1,D1,F1,G,H,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,H:1,)} | I1
J2	{(B2,C1,D1,F1,G,H,I1,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,H:1,I1:1,)} | J2
K2	{(B2,C1,D1,F1,G,H,I1,J2,:1)}	{(B2:1,C1:1,D1:1,F1:1,G:1,H:1,I1:1,J2:1,)} | K2
Tree #4		

B1	{(:1)}	{()} | B1
C1	{(B1,:1)}	{(B1:1,)} | C1
D1	{(B1,C1,:1)}	{(B1:1,C1:1,)} | D1
F1	{(B1,C1,D1,:1)}	{(B1:1,C1:1,D1:1,)} | F1
I1	{(B1,C1,D1,F1,:1)}	{(B1:1,C1:1,D1:1,F1:1,)} | I1
J1	{(B1,C1,D1,F1,I1,:1)}	{(B1:1,C1:1,D1:1,F1:1,I1:1,)} | J1
K1	{(B1,C1,D1,F1,I1,J1,:1)}	{(B1:1,C1:1,D1:1,F1:1,I1:1,J1:1,)} | K1

4. experimental results

The performance of HFMFFP-growth is compared with that of MFFP-growth is more efficient approach for miming frequent itemsets ,based on use horizontal fragmentation technique with most frequent itemset. memory space for HFMFF-tree in byte less than memory space for FP-growth and MFFP-growth . Test this algorithm on 30% and 50% minsup ,using for coded visual .net , and the dataset is real dataset from UCI machine learning, use bank marketing dataset after preprocessing dataset. Full dataset 4500 records in figure .show computational performance of mining the bank marketing dataset between FP-growth , MFFP-growth and HFMFFP-growth.



5.Conclusion

Mining frequent itemsets in a transaction database is critical for mining association rules. This research suggested a new mothod to performance FP-tree efficiency in two dimension effect:

- reducing the memory space of FP-tree
- reducing the time of execution tree
- Fragmentation the dataset to equal size
- applies a smaller tree to discover frequent itemsets faster.

References

1.A. Singh , J. Agarwa, A. Rana," Performance Measure of Similis and FP-Growth Algorithm", International Journal of Computer Applications, January 2013 .

2. Fei Tu and Bo He," A Parallel Algorithm for Mining Association Rules Based on FP-tree", Springer-Verlag Berlin Heidelberg 2011.

3. M. H. Margahny and A. Shakour, " FAST ALGORITHM FOR MININGASSOCIATION RULES", Journal of Engineering Sciences, Assiut University, Vol. 34, No. 1, pp. 79-87, January 2006.

4. C. Borgelt," An Implementation of the FPgrowth Algorithm".

5. OdedMaimon · LiorRokach," Data Mining and Knowledge Discovery Handbook", Springer Science+Business Media, LLC 2005, 2010.

6. Florian Verhein," Frequent Pattern Growth (FP-Growth) Algorithm",2008 Florian Verhein.

7. Yu-Chiang Li and Chin-Chen Chang," A New FP-Tree Algorithm for Mining Frequent Itemsets", Springer-Verlag Berlin Heidelberg 2004.

8.Zailani Abdullah1, Tutut Herawan2, and Mustafa Mat Deris3, "Mining Significant Least Association Rules Using FastSLP-Growth Algorithm",AST/UCMA/ISA/ACN 2010.

9. Rakesh Kumar Soni1, Prof. Neetesh Gupta2, Prof. Amit Sinhal3,"An FP-Growth Approach to MiningAssociation Rules ", IJCSMC, Vol. 2, Issue. 2, February 2013, pg.1 – 5.

10. Wiem Taktak1 and Yahya Slimani2,"MS-FP-Growth: A multi-support Vrsion of FP-Growth Agorithm", International Journal of Hybrid Information Technology,2014.