Global Journals LaTeX JournalKaleidoscope™
Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

Improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree

Dr. Sujatha Dandu[1], B.L.Deekshatulu[2] and Priti Chandra[3]

[1] JNTU

Received: 6 December 2012 Accepted: 5 January 2013 Published: 15 January 2013

Abstract

Frequent itemset mining plays an important role in association rule mining. The Apriori FP-growth algorithms are the most famous algorithms which have their own shortcomings such as space complexity of the former and time complexity of the latter. Many existing algorithms are almost improved based on the two algorithms and one such is APFT [11], which combines the Apriori algorithm [1] and FP-tree structure of FP-growth algorithm [7]. The advantage of APFT is that it doesn?t generate conditional sub conditional patterns of the tree recursively and the results of the experiment show that it works fasts than Apriori and almost as fast as FP-growth. We have proposed to go one step further modify the APFT to include correlated items trim the non correlated itemsets. This additional feature optimizes the FP-tree removes loosely associated items from the frequent itemsets. We choose to call this method as APFTC method which is APFT with correlation.

Index terms— data mining, correlation, correlation coefficient.

1 Introduction

Association rules are created by analyzing data for frequent if/then patterns and using the criteria of support and confidence to identify the most important relationships. Support is an indication of how frequently the items appear in the database. Confidence indicates the number of times the if/then statements have been found to be true. Mining association rules purely on the basis of minimum support may not always give interesting relationships between the item sets.

Consider a case where in a sample set of 100 transactions, item A with support SA = 50 & item B with support SB = 50 have a combined support of SAB = 5. If the minimum support threshold is 5, it would appear as if A and B are frequent item sets because they satisfy the minimum support criteria. The drawback of this method is that only 10% of all A and 10% of all B are Author ? : Aurora's Technological and Research Institute, Hyderabad, India. E-mail : sujatha.dandu@gmail.com Author ? : Distinguished Fellow, IDRBT, Hyderabad, India. E-mail : deekshatulu@hotmail.com Author ? : Scientist, Advanced System Laboratory, Hyderabad, India. E-mail : priti_murali@yahoo.com involved in association rules together. So the relationship between A and B cannot be of much use even though they occur together more than the support value. A new method is required wherein we measure not only the support but also the confidence of B occurring when A occurs & vice versa. This way we can make sure that the interestingness of the rules is preserved. The concept of correlation is introduced in order to filter the result from the association rules that not only satisfy the minimum support criteria but also have a linear relationship amongst them. This approach combines the concepts of FP-growths tree generation technique with Apriori's candidate generation step along with a correlation condition so as to improve the interestingness of rules as well as to optimize the space and time consumption.

2 II.

3 EXISTING SYSTEM

The concept of frequent itemset was first introduced by Agarwal et al in 1993.Two basic frequent itemset mining methodologies: Apriori & FP-growth, and their extensions, are introduced. Agarwal and Srikanth [2] observed an interesting downward closure property which states that: A k-itemset is frequent only if all of its sub-itemsets are frequent. It generates candidate itemset of length k from itemset of length k-1. Since the Apriori algorithm was proposed, there have been extensive studies on the improvements of Apriori, eg. partitioning technique [3], sampling approach [4], dynamic itemset counting **??**6], incremental mining [5] & so on.

Apriori, while historically significant, suffers from(1) generating a huge number of candidate sets, and (2) repeatedly scanning the database. Han et al [7] derived an FP-growth method, based on FP-tree. The first scan of the database derives a list of frequent items in which items in the frequency descending order are compressed into a frequent-pattern tree or FP-tree. The FP-tree is mined to generate itemsets. There are many alternatives and extensions to the FP-growth approach, including depth first generation of frequent itemset [8]; H-mine, by **??**9] which explores a hyper structure mining of frequent patterns; and an array-based implementation of prefix tree structure for efficient pattern growth mining **??**10]. To overcome the limitation of the two approaches a new method named APFT [11] was proposed. The APFT algorithm has two steps: first it constructs an FP-tree & then second mines the frequent items using Apriori

4 C

ssociation rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database. An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

5 A

algorithm. [The results of the experiment show that it works faster than Apriori and almost as fast as FPgrowth]. Extending this approach, we have introduced APFTC, which includes the concept of correlation to filter (reduce) the association rules that not only satisfy the minimum support but also have liner relationships among them. The computational results verify the good performance of APFTC algorithm.

6 III.

7 Proposed System a) Correlation Concept

The concept of correlation can be extended to transaction databases with the following modifications. An item 'a' is said to be correlated with item 'b' if it satisfies the following conditions: P (ab) > P(a)P(b) Here P(ab) = probability of items 'a' and 'b' occurring together in the transaction database i.e. the number of transactions in which both 'a' and 'b' occur together/total number of transactions. P(a) =The number of transactions in which 'a' occurs/total transactions. P(b) = The number of transactions in which 'b' occurs/total transactions Therefore the formula essentially represents

8 Observed probability > Expected Probability

This condition is said to be positive correlation between items 'a' and 'b'.

9 b) APFTC

The idea of correlation is introduced at step 5 of the APFT algorithm. We are basically deriving the frequent itemsets of size 2 at this step, so it is only appropriate to introduce the idea of correlation here. There is another change to the algorithm where we can calculate the support of each branch at the time of construction of calculation N Table itself instead of traversing the tree again later. This is a more economical way of calculating support than the one suggested in the original paper where repeated traversal of the tree is necessary for support calculation.

Algorithm APFT [ ] Input: FP-tree, minimum support threshold ? Output: all frequent itemset L L = L1; for each item Ii in header table, in top down order LIi = Apriori-mining (Ii); Return L = {LULi1ULi2U?Lin}; Pseudo code Apriori-mining (I i) 1. Find item p in the header table which has the same name with Ii (2. q = p.tablelink; 3. while q is not null Introducing correlation coefficient at line4, we continue for each node qi != root on the prefix path of q All Paths of Tree **??**i].add (qi.item-name);//AllPathsOfTree is an array of all paths from q to root. if NTable has a entry N such that N.Item-name= q i.item-name.

N.Item-support = N.Item-support + q.count; else Check For Correlation Between qi and q PA=Map Support (q.itemName); PB=Map Support (qi.item-name); P(AB)= q.count; If(P(AB)>P(A)P(B)/transaction Count) . add an entry N to the NTable; . N.Item-name = q i. item-name; . N.Item-support = q.count; All Paths of Tree [i].support=q.count;//here we have the individual path //and its support stored to be used //later q = q.tablelink;

10 c) Example

We follow an example of a simple database with 6 transactions as shown below.

Transaction Database Once the tree has been constructed we proceed with the APFT algorithm with construction of an N Table for each of the nodes. We start of with the node 4 which is at the bottom of the header table for the given fp tree. Let us take the minimum support value as 2 for this example. As shown in the figure we use the Ntable along with Apriori's candidate generation step to successively generate supersets of the the smaller itemsets and then perform the pruning step by calculating support by scanning the tree paths instead of scanning the entire database as is the case with apriori.

The candidate support calculation procedure is as shown in the diagram below each path from the node to the root is stored with that path's support. The algorithm APFTC is reportedly working efficiently and in many cases, it's much faster than FP-Growth. The results are found to be more interesting than association rules mined by FP-Growth although they are the subsets of itemsets mined by FP-Growth. The above graph shows the number of itemsets generated with respect to varying minimum support. Supporting our idea, APFTC has generated equal to the number of itemsets which are highly correlated when compared to the other algorithms. The above graphs gives the itemsets generated with respect to varying size. The itemsets generated by APFTC are equal to or less than those generated by the other two algorithms in some cases.

It can be concluded from the above results that APFTC performs as expected proving to be efficient in time consumed and also in retrieving the most correlated itemsets.

Figure 1: Improved

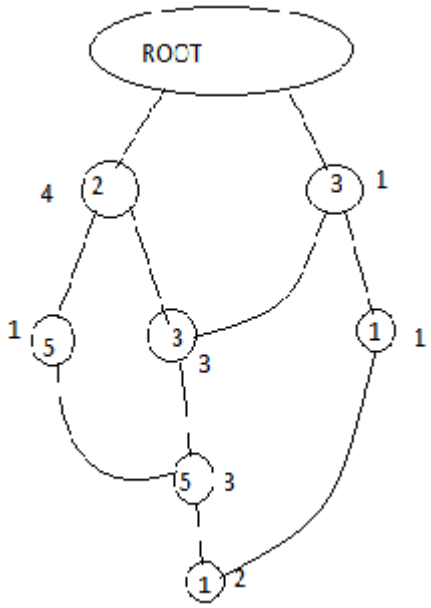
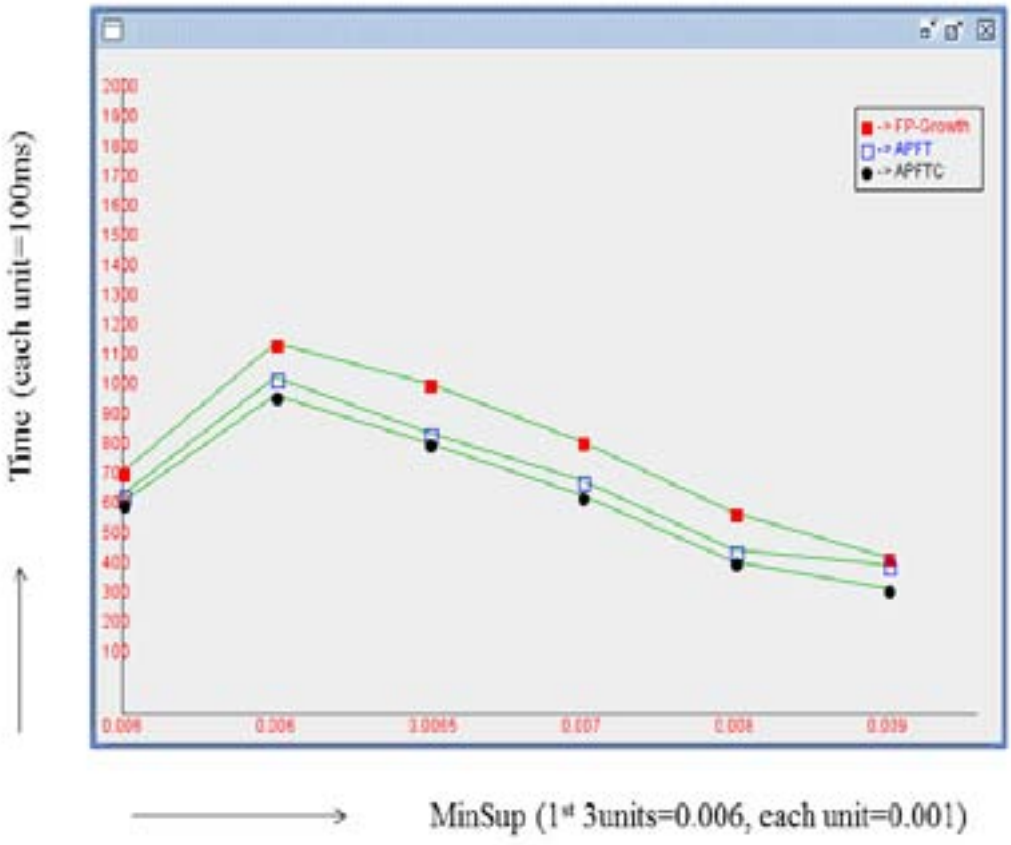
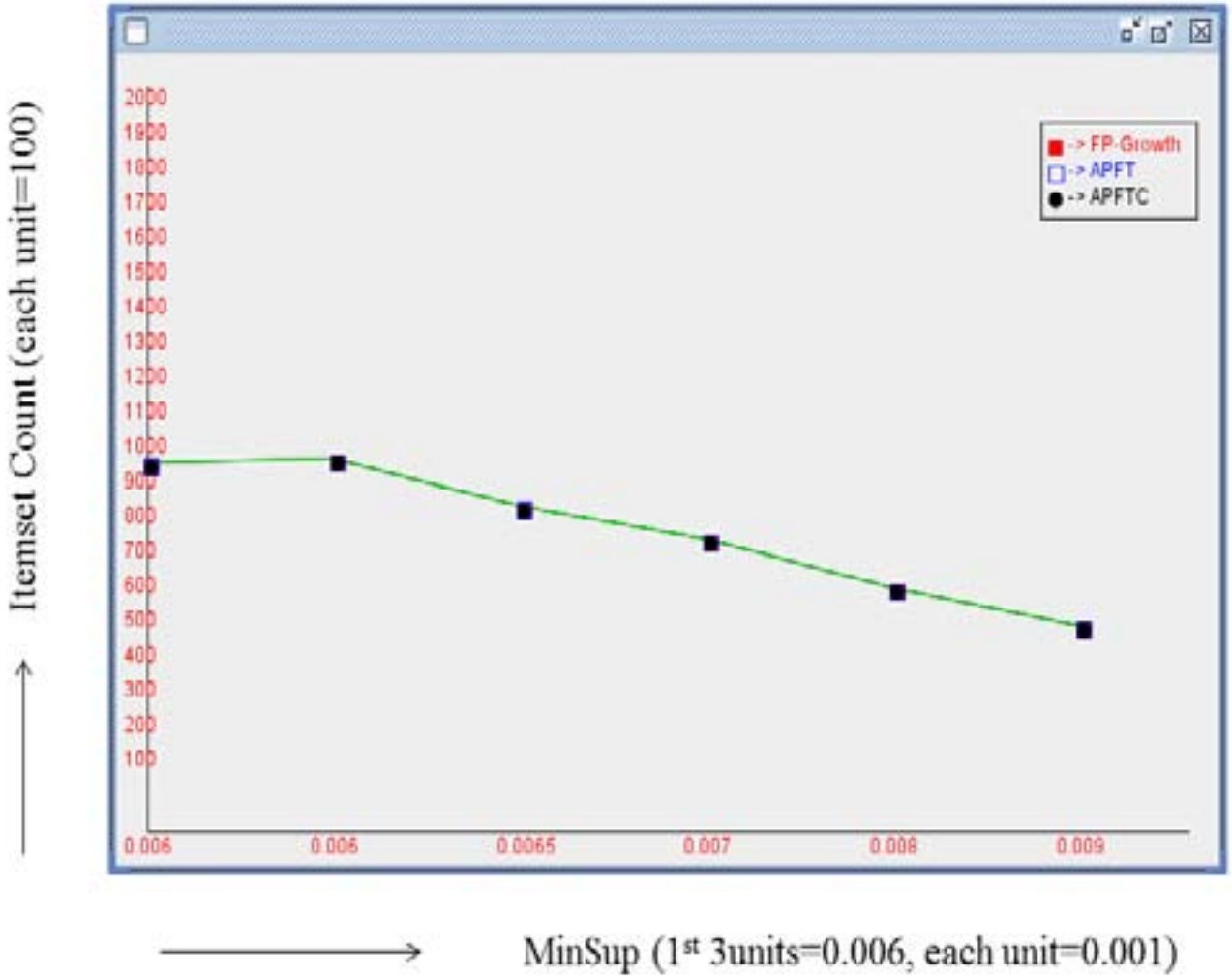


Figure 2:



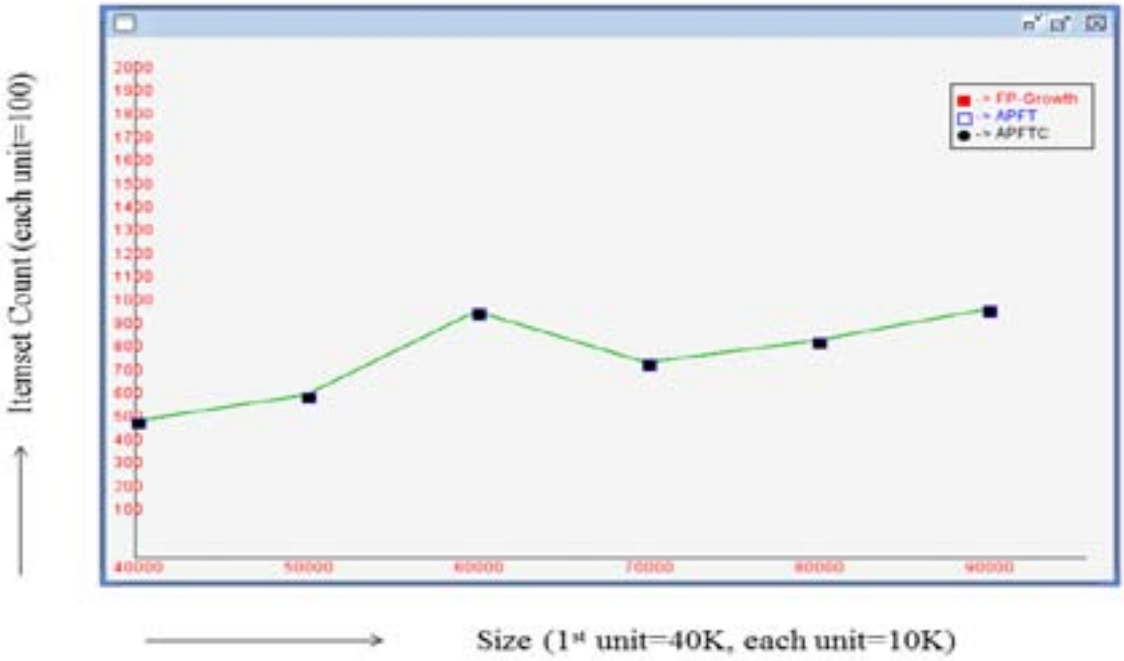
1

Figure 3: Figure 1 :C



2

Figure 4: Figure 2 :



3

Figure 5: Figure 3 :

[Item-Support = N] , N Item-Support = N . (Item-support + q.count; an entry N to the N Table)

[Ct and Subset] , = Ct , Subset . Ck, t.

[Fk (ed.)] , = Fk . c.support *,-./012} (ed.)

[Agarwalr ()] 'A tree projection algorithm for generation of frequent itemsets'. Aggarwalc Agarwalr , Prasadvvv . *Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining*, 2000.

[Park et al. ()] 'An effective hash-based algorithm for mining association rules'. J S Park , M S Chen , P S Yu . *Proceedings of ACM SIGMOD International Conference on Management of Data*, (ACM SIGMOD International Conference on Management of DataSan Jose, CA) 1995. p. .

[Park et al. ()] *An effective hash-based algorithm for mining association rules*, J S Park , M S Chen , P S Yu . InSIGMOD1995. p. .

[Savasere et al. ()] 'An efficient algorithm for mining association rules in large databases'. E Savasere , S Omiecinski , Navathe . *Proceedings of the 21st International Conference on Very large Database*, (the 21st International Conference on Very large Database) 1995.

[Contrasting Correlations by an Efficient Double-Clique Condition -Aixiang Li, Makoto Haraguchi, and Yoshiaki Okubo] *Contrasting Correlations by an Efficient Double-Clique Condition -Aixiang Li, Makoto Haraguchi, and Yoshiaki Okubo*,

[Techniques -Jia Wei Han and Micheline Kamber (ed.)] *Data Mining -Concepts and*, Techniques -Jia Wei Han and Micheline Kamber (ed.)

[Agrawal and Srikant] *Fast algorithms for mining association rules*, R Agrawal , R Srikant . 94 p. .

[Pei et al. ()] 'Hmine: Hyper-structure mining of frequent patterns in large databases'. J Pei , J Han , H Lu . *ICDM*, 2001. p. .

[Item-name = q i. item-name] *Item-name = q i. item-name*,

[N] *Item-support = q*, N . (count)

[Agrawal et al. (1993)] 'Mining association rules between sets of items in large databases'. R Agrawal , T Imielinski , A Swami . *InProc.1993 ACM-SIGMOD Int. Conf. Management of Data*, (Washington, D.C) May 1993. p. .

[Han et al. (2000)] 'Mining Frequent Patterns without Candidate Generation (PDF), (Slides)'. J Han , J Pei , Y Yin . *Proc. 2000ACM-SIGMOD Int*, (2000ACM-SIGMOD Int) May 2000.

[Fk] *NTable)j.Item-support*minsup} 14*, = Fk . (repeat 15. k = k + 1)

[SPMF -A Sequential Pattern Mining Framework -Philippe Fournier-Viger] *SPMF -A Sequential Pattern Mining Framework -Philippe Fournier-Viger*,