Scholars Journal of Engineering and Technology (SJET) **ISSN 2321-435X**
Sch. J. Eng. Tech., 2013; 1(2):63-67

(An International Publisher for Academic and Scientific Resources)
www.saspublisher.com

Research Article

Efficient Mining Tree in Association Rule Mining for Reducing the Time complexity

Richa Sharma, Pream Narayan Araya
Samrat Ashok Technological Institute, Vidisha-464001 (M.P.) ,INDIA

***Corresponding author**
Richa Sharma
Email: richasharmamtech@gmail.com

Abstract: Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. However, no method has been shown to be able to handle data streams, as no method is scalable enough to manage the high rate which stream data arrive at. More recently, they have received attention from the data mining community and methods have been defined to automatically extract and maintain gradual rules from numerical databases. In this paper, we thus propose an original approach to mine data streams for Association rule mining. In this paper we present a novel ABFP tree technique that greatly reduces the need to traverse FP-trees and array based FP tree, thus obtaining significantly improved performance for FP-tree based algorithms. The technique works especially well for sparse datasets. We then present a new algorithm which use the FP-tree data structure in combination with the FP- Experimental results show that the new algorithm outperform other algorithm in not only the speed of algorithms, but also their CPU consumption and their scalability.
Keywords: FP-Tree, WSFP –Tree, Frequent Patterns, Array Technique.

1. INTRODUCTION

Frequent-pattern mining plays an essential role in mining associations [1] if any length k pattern is not frequent in the database, its length (k + 1) super-pattern can never be frequent. The essential idea is to iteratively generate the set of candidate patterns of length (k+1) from the set of frequent-patterns of length k (for $k \geq 1$), and check their corresponding occurrence frequencies in the database.

The Apriori heuristic achieves good performance gained by (possibly significantly) reducing the size of candidate sets. However, in situations with a large number of frequent patterns, long patterns, or quite low minimum support thresholds, an Apriori-like algorithm may suffer from the following two nontrivial costs: – It is costly to handle a huge number of candidate sets. For example, if there are 104 frequent 1-itemsets, the Apriori algorithm will need to generate more than 107 length-2 candidates and accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as {a1. . . a100}, it must generate $2100 - 2 \approx 1030$ candidates in total.

This is the inherent cost of candidate generation, no matter what implementation technique is applied. It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for mining long patterns. Can one develop a method that may avoid candidate generation-and-test and utilize some novel data structures to reduce the cost in frequent-pattern mining? This is the motivation of this study [5].

Let I= {i1, i2, in} be a set of items, we call x and I an item set, and we call X a k-item set if the cardinality of item set X is k. Let database T be a multi set of subsets of I, and let support(X) be the percentage of item set Y in T such that X U Y .Informally, the support of an item set procedures how often X occurs in the database. If support(X) + minus , we say that X is a frequent item set , and we denote the set of all frequent item sets by FI.A closed frequent item set is a frequent item set X such that there exists no superset of X with the same support count as X. If X is frequent and no superset of X is frequent, we

Say that X is a maximal frequent item set, and we denote the set of all maximal frequent item sets by MFI. [7]

2.RELATED WORK

Association Analysis is the discovery of association rules attribute-value conditions that occur frequently together in a given data set. Association analysis is widely used for market basket or transaction data analysis. Association Rule mining techniques can be used to discover unknown or hidden correlation between items found in the database of transactions. An association rule [1, 3, 4, and 7] is a rule, which implies certain association relationships among a set of objects (such as 'occurs together' or 'one implies to other') in a database. Discovery of association rules can help in business decision making, planning marketing strategies etc.Apriori was proposed by Agrawal and Srikant in 1994. It is also called the level-wise algorithm. It is the

most popular and influent algorithm to find all the frequent sets. The mining of multilevel association is involving items at different level of abstraction. For many applications, it is difficult to find strong association among data items at low or primitive level of abstraction due to the sparsity of data in multilevel dimension. Strong associations discovered at higher levels may represent common sense knowledge. For example, instead of discovering 70% customers of a supermarket that buy milk may also buy bread. It is also interesting to know that 60% customer of a super market buys white bread if they buy skimmed milk. The association relationship in the second statement is expressed at lower level but it conveys more specific and concrete information than that in the first one. To describe multilevel association rule mining, there is a requirement to find frequent items at multiple level of abstraction and find efficient method for generating association rules. The first requirement can be fulfilled by providing concept taxonomies from the primitive level concepts to higher level. There are possible to way to explore efficient discovery of multiple level association rules. One way is to apply the existing single level association rule mining method to mine Q based association rules. If we apply same minimum support and minimum confidence thresholds (as single level) to the Q levels, it may lead to some undesirable results. For example, if we apply Apriori algorithm [1] to find data items at different level of abstraction under the same minimum support and minimum confidence thresholds. It may lead to generation of some uninteresting associations at higher or intermediate levels.

Large support is more likely to exist at high concept level such as bread and butter rather than at low concept levels, such as a particular.The FP-growth method relies on the following principle: if X and Y are two item sets, the support of item set X UY in the database is exactly that of Y in the restriction of the database to those transactions containing X. This restriction of the database is called the conditional pattern base of X. Given an item in the header table, the growth method constructs a new FP-tree corresponding to the frequency information in the sub-dataset of only those transactions that contain the given item. Figure 2(a) shows the conditional pattern base and the FP-tree for item {p}, this step is applied recursively, and it stops when the resulting smaller FP tree contains only one single path. The complete set of frequent item sets is generated from all single path FP-trees [5]. When adding an item i to the existing item set head, we denote the item set head i by Z, the path from the parent node of this node (node's item-name is i) to the root node in the head's FP-tree is called Z's prefix path. Figure 2(b) shows the prefix paths for item {p}.

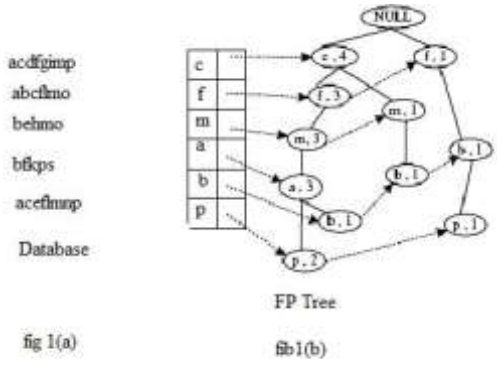

Figure 1: FP-Tree

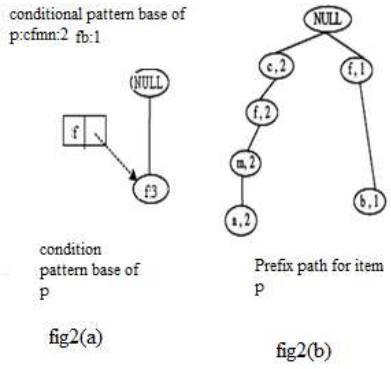

Figure 2: Prefix paths for Item

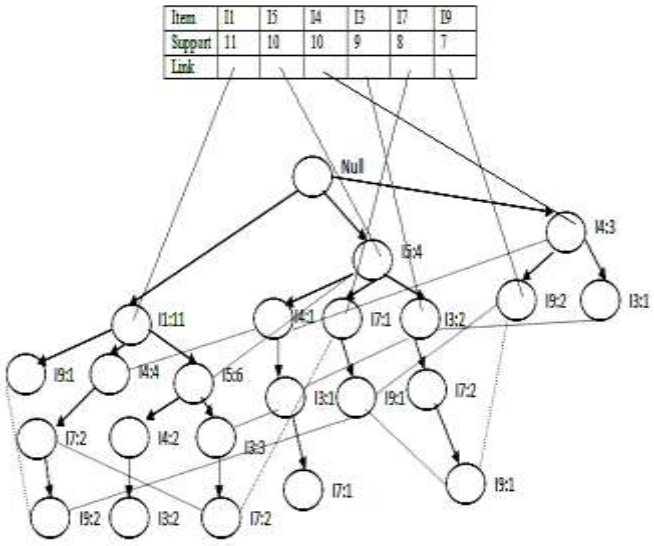

Figure 3: Frequency of Sample Database

3. PROPOSED WORK

Algorithm of WS with Array based technique: Improved FP-tree (IFP-tree) is similar with FP-tree and each node in IFP-tree consists of four fields: item, count, ahead and next. Where item registers which item this node represents, count registers the number of transactions represented by the portion of the path

reaching this node, ahead links to the left child or the parent of the node, and next links to the right brother of the node or the next node in IFP-tree carrying the same item, or null if there is none. We also define two arrays: nodecnt and link, and link [item] registers a pointer which points to the first node in the IFP-tree carrying this item, nodecnt [item] registers the support count sum of those nodes in IFP-tree which carry the same item. In comparison with FP-tree, IFP-tree doesn't contain the path from root to leaf-node, contains fewer pointers than FP-tree in mining process, and so may greatly save cost in memory. The construction method of IFP-tree is similar with that of FP-tree, the difference from FP-tree exits in the process of Inserting frequent item sets in each transaction into IFP-tree. In this paper, we don't adopt the method of recursively performing the procedure, insert tree ([p|P], t), but employ a dynamic pointer to complete it.

3.1 The algorithm constructing IFP-tree as follows:

Procedure FP-tree constructs (T, min_sup)

1) Scan T and count the support of each item, derive a frequent item set (F) and a list (L) of frequent items, in which items are ordered in frequency-descending order;

2) The root of IFP-tree is created and labeled with "root";

3) For each transaction t UT do{Frequent item set It= t UF, in which items are listed to St according to the order of L, defines a dynamic pointer (p_current) which points to root.

Procedure WSFP-tree constructs (T, min_sup)

1) Scan T and count the support of each item, derive a frequent item set (F) and a list (L) of frequent items, in which items are in sequence of occurrence form;

2) The root of IFP-tree is created and labeled with "root";

3) For each transaction t UT do{ Frequent item set It= t UF, in which items are listed to St according to the order of occurrence L, defines a dynamic pointer (p_current) which points to root.

4}Traverse IFP-tree in a root-first order and transfer the pointers of ahead and next, count the sum of nodes' support carrying the same item and then list together. For example, let transaction database T be illustrated by TABLE I, and the minimum support (min_sup) be 4, then we can get the list (L) of frequent items.

A	G	D	C	B
B	D	E	A	M
C	E	F	A	N
A	B	N	O	I
A	C	Q	R	G
A	C	H	I	G
A	F	M	N	O
A	D	B	H	I
J	E	B	A	D
A	K	E	F	C
C	D	L	B	A

Table 1

Figure 4: constructed all frequent item set.

A	B	C	D
A	B	D	E
A	C	E	F
A	B	_	_
A	C	_	_
A	C	_	_
A	F	_	_
A	B	D	_
A	B	D	E
A	C	E	F
A	B	C	D

Table 2: Transaction database t with ascending order

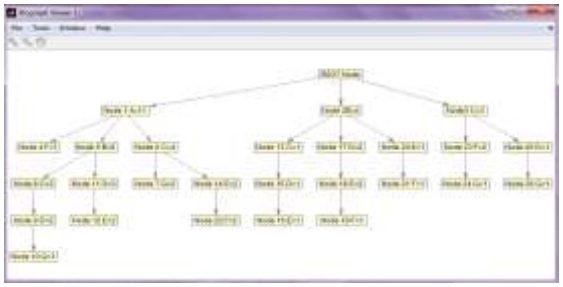

Figure: 5 FP Tree constructions

A	G	D	C	B
B	D	E	A	M
C	E	F	A	N
A	B	N	O	I
A	C	Q	R	G
A	C	H	I	G
A	F	M	N	O
A	D	B	H	I
J	E	B	A	D
A	K	E	F	C
C	D	L	B	A

Table 3: Transactional Dataset:

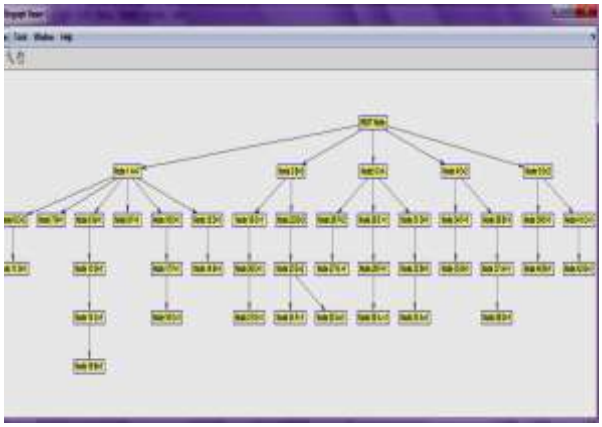

Figure: 6 WSFP Tree construction

5. ARRAY BASED APPROACH:

Firstly let an order (<) be the order of the list L, that is, the support descending order of frequent items. Let the letters (i,j,<,k) denote items in database, then i is called the minimum item and k is called the maximum item if i<j<....k. Let the letters (ik,<,i1[<1]) denote items, P be a path from root to the node N in IFP-tree. If there exists a child node N□of the node N and the items (ik,<,i1) appear the sub-path from N to N in order, that is, the item ik corresponds to the node N and i1 corresponds to N,then P is called the path with the array of the item sets{ik,<,i1}, the support count of the node N also is called the base count of A.

The process of building PT (a) is the following: firstly, each node in IFP-tree whose value of item is m is retained in PT (a), the support count of each inner node (except root) is initialized to be zero. Secondly, for each node, we summate the support count of its children.

The main work done in the mining process is traversing the postfix sub-tree to count the support of item sets and constructing new postfix sub-tree, Recall that for each item i of conditional PT(x), two traversals of PT(x) are needed for constructing the new sub-tree PT(k,i). The first traversal finds all frequent items and their counts of support, the second traversal constructs the new sub-tree PT (k, i). In this paper, we use the array technique presented by reference [2]. All cells in the array are initialized as 0.

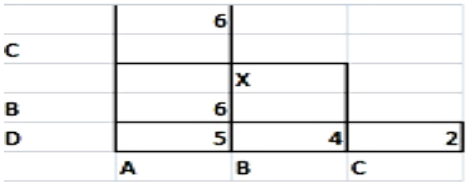

Figure7 (A): Array Examples

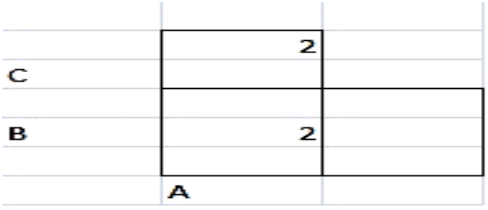

Figure7 (B) Array Examples

During the second scan each transaction, first all frequent items in the transaction are extracted. Suppose these items form item set I, to insert I into PT, the items in I are sorted according to the order in the header of PT. When we insert I into PT, at the same time AU[i,j] is incremented by 1 if {i,j} is contained in I. After the second scan, array A keeps the counts of all pairs of frequent items, as shown in table (a) of Fig 4.

6. EXPERIMENTAL EVALUATION

The experiments were conducted on 2.4 GHz Pentium with 512 MB of memory running Microsoft Windows XP. All codes were compiled using Matlab 7.10. We used Connect-4 downloaded form a website to test and compared FP tree with WSFP tree, which is a real and dense dataset. Fig 8 and Fig 9 shows the experimental results. Here we can see that ABWSFP outperforms WSFP for high levels of minimum support, but it is slow for very low levels.

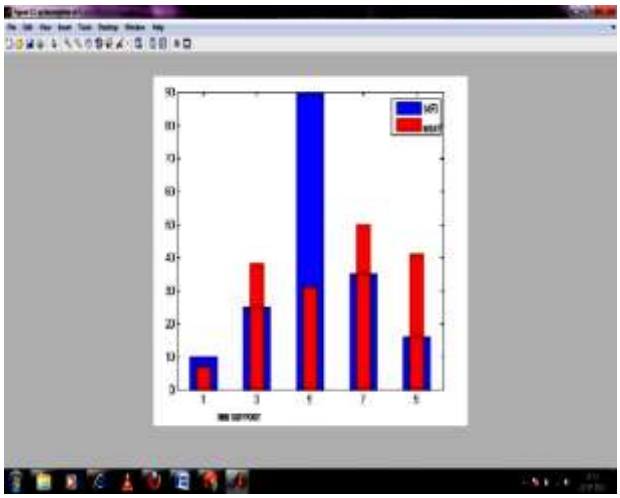

Figure8: Graphical Representation of Calculated Result

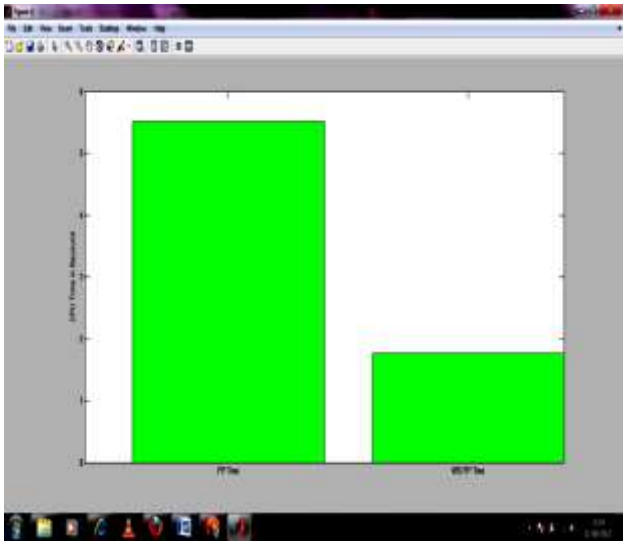

Figure9: CPU Utilization

7. CONCLUSIONS

In this paper, an efficient algorithm, called ABWSFP-max, for mining maximal frequent patterns based on improved FP-tree and array technique is proposed, the algorithm improves the conventional FP-tree and by introducing the concept of the array sub-tree, avoids generating the maximal frequent candidate patterns in mining process and therefore greatly reduces the memory consume, it also uses an array-based technique to reduce the traverse time to the improved FP-tree. Therefore it greatly improves the mining efficiency in time and space scalability. Experimental results show that it possesses high mining efficiency and scalability.

REFERENCES

1. Bharat Gupta, Dr.Deepak Garg, Karun Verma. A Novel Approach to Mine Frequent Item Sets Using Maximal Apriori and FP-Tree Method, International Journal of Advance Computing . 2011; 3(2)1-7.

2. Kuparala Chakrapani. Implementation of array based technique to improvise representation of fp-tree using iafp-max algorithm. Journal of Global Research in Computer Science. 2011; 9(2):216-222.

3. Mantha SS. Maximal Frequent Item set. International Journal of Computer Applications, 2010;10(3):12-15.

4. Sumathi K, An array based approach for mining maximal frequent itemsets "Computational Intelligence And Computing Research (ICCIC), 2010, 1-6.

5. Divya R, S.Vinod kumar. Survey on AIS, Apriori and FP-Tree algorithms International Journal of Computer Science and Management Research. 2012; 1(2):226-236.

6. Huanglin Zeng An Improved Algorithm of FP - tree Growth Based on Mapping Modeling, ICCASM. 2010; 4:463

7. Vaibhav Kant Singh and Vinay Kumar Singh. Minimizing Space Time Complexity by RSTDB a new method for Frequent Pattern Mining.First International Conference on Intelligent Human Computer Interaction ,Allahabad, 2009.

8. Christie I. Ezeife and Min Chen. Lecture Notes in Computer Science, Advances in Web-Age Information Management. 2004; 3129; 5th International Conference, WAIM 2004, Dalian, China.

9. Han J., Pei J, Yin Y and Mao R, Mining frequent patterns without candidate generations., Data mining and knowledge discovery, 2004; 8:53-87.

10. Han J., Pei J, Yin Y and Mao R, Mining frequent patterns without n improved frequent pattern growth method for mining association rule. ACM SIGMOD Record, 2000; 29(2):1-12.

11. Burdick Doug, Calimlim Manuel, and Gehrke Johannes, A Maximal Frequent Item set Algorithm for Transactional Database, Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2001; 443-452.

12. Han J, Pei J and Yin Y, Mining frequent patterns without candidate generation, Proceedings of Special Interest Group on Management of Data, Dallas, 2000; 1-12.

13. Agrawal R, Srikant S, Fast algorithms for mining association rules. In VLDB', 1994; 94: 487-499.