***I.J. Engineering and Manufacturing*, 2012, 1, 69-77**
Published Online October 2012 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijem.2012.05.10

Available online at http://www.mecs-press.net/ijem

Research of Association Rule Mining Algorithm Based on Improved FP-Tree

Chen zhuo[a], Lu nannan[a], Li shiqi [a], Han tao [a]

[a] *Hubei University of Technology, Wuhan 430068,China*

Abstract

Mining algorithm of FP-Tree is one of the most effective mining algorithms in association rule mining. It must produce large amounts of the candidate set and scan database repeatedly, but it generates conditional FP-Tree recursively in the process of mining frequent pattern and wastes the storage space greatly using the common tree's memory structure. It proposed an algorithm for mining frequent patterns by constructing reverse FP-tree with a binary tree storage structure. In mining process, it mines left son tree recursively, but gets frequent pattern not by generating conditional FP-Tree, so it greatly reduces the storage space and running time. The experiments show that the improved algorithm can realize effective mining on the time and space.

Index Terms: data mining; frequent pattern; FP-tree; binary tree

1. Introduction

The improvement technology of data mining promoted the rapid development of data mining. Association rule mining is the most commonly used and important research direction in data mining. Association rule mining is mainly used for discovering the interesting connection or the relation among the mass data. Choosing the good association rule algorithm is the critical key of realizing the good mining.

The Apriori algorithm is the most classical algorithm of association rule algorithm. It uses iterative method of cascade searching, and explores the k+1- frequent item set through the k- frequent item set. But it possibly needs to produce large amounts of the candidate set and scans database repeatedly, which can affect the performance of algorithm inevitably. The FP- tree algorithm opened the new way to mining frequent pattern effectively. It does not need to produce large amounts of the candidate set and not need to scan database repeatedly, so it raised the algorithm efficiency greatly. [1] But its efficiency both in storage and running time

* Corresponding author.
E-mail address: chenzhuo_hust@163.com

still needed to be improved. The reference [2] used a one-way tree structure and proposed a frequent pattern-mining algorithm based on constrained sub-tree, which saved one third of memory. The reference [3] proposed an algorithm for mining frequent patterns by finding the frequent extensions and merging sub-trees in a conversely constructed FP-tree. The reference [4], [5], [6] also proposed the improved algorithm of mining frequent patterns, which raised its space and time efficiency to a certain extent. In this article, it mainly used the thought of constructing reverse tree which in reference [3], and took binary tree as memory structure to construct binary reverse FP-Tree. In the process of mining, it mined left son tree recursively, but got frequent pattern not by generating conditional FP-Tree, which realized effective mining to FP-Tree.

2. RELATED CONCEPT

Definition 1: $I = \{i_1, i_2, ..., i_m\}$ is a set. D, which is the duty related data is the database business's set, and each business T is a set, $T \subseteq I$. Each business has an identifier which is called TID. Supposed A is an item set, business T contains A and when only works as $A \subseteq T$ .The association rule is the shape like implication type $A \Rightarrow B$, and $A \subset I$ ， $B \subset I$, $A \cap B = \varnothing$ .The rule $A \Rightarrow B$ is tenable in business D and has the support s which is the percentage of $A \cup B$ in business D. The rule $A \Rightarrow B$ is tenable in business D and has the confidence c which is the percentage of containing business A simultaneously also containing business B in business D.

Definition 2: The support and confidence of rule are the interest measure standards of two rules and reflect the usefulness and determinism of discovery rule separately. The rule which simultaneously satisfies the smallest support valve (min_sup) and the smallest confidence valve (min_conf) is called as the strong rule.

Definition 3: A set is called the item set. The item set containing k item is called the k- item set. The appearance frequency of item is the business number of containing the item set, which called item set frequency or the support counting. If the appearance frequency of item is bigger than or equal to min_sup and the business number in D products, the item set satisfies the smallest support. The item set which satisfies the smallest support is called the frequent item set.

The association rule mining divides into two steps:

(1) Finding out all frequent item sets: According to the definition, the appearance frequency of item is the same as the pre-definition smallest support at least.

(2) Generating strong association rule by frequent item set: According to the definition, the appearance frequency of item is the same as the pre-definition smallest confidence at least.

The overall performance of association rule mining is depended on the first step. When finding all biggest frequent item sets, the corresponding connection rule is very easy to produce. Therefore, seeking for the frequent item set is the key of current research.

3. The algorithm and realize of improved FP- tree

The process of improved algorithm proposed in this article is similar with the FP- growth algorithm, similarly needs to two important processes: Constructing the FP- tree and mining the FP- tree. But in this article, it used the binary tree as storage structure and constructed FP- tree according to the rise order of support counting in order to omit the process of producing conditional FP-Tree.

3.1. Constructing the FP-Tree

In the process of constructing FP-Tree, the improved algorithm was basic same as the original FP- growth algorithm, lied in the following two differences: First, the improved algorithm constructs according to the rise order of support counting but not the descend order. Second, the storage structure is used binary tree (left is child and right is brother).According to the two differences, it mainly manifested two aspects: the support registers and the function insert_Tree([p|P], T).There are two main steps in insert_Tree([p|P],T):

First, seek for whether exists node N in T, and N.item_name = p. item_name.

Second, create a new node N. The improved algorithm is used binary tree, so the algorithm both seeking for node and creating new nodes are different from the original algorithm.

The example of entire construct process is as follows. There are business database, the smallest support is 2, as shown in Table Ⅰ.

Table.1 BUSINESS DATABASE

TID	**项 ID 的列表**
T100	I1，I2，I5
T200	I2，I4
T300	I2，I3
T400	I1，I2，I4
T500	I1，I3
T600	I2，I3
T700	I1，I3
T800	I1，I2，I3，I5
T900	I1，I2，I3

First, scan business database and obtaining the support increasing arranged table L= [I5:2, I4:2, I3:6, I1:6; I2:7]. Then, start to create the FP- tree. Create the root node of tree first, which marks with “null”. Scanning database for the second time and sorting each business item according to the order in L and creating branching for each business. For example, according to the order in L, the first business is arranged as {I5, I1, I2}. Current tree is null, so I5 is connected to root node directly and becomes the child node of root. Then, I1 becomes the child node of I5, and I2 becomes the child node of I1. All their frequency counting is 1. Similarly, carry on the second business. The current business sorting is {I4,I2}.Search the tree first, seek for whether exists I4 in the child of root. Here, first judge the left sub-tree of root node, if it has not found, carry on the recursion traversal to the brother of left sub-tree. Here, I4 is not found, so it must create a branch as the brother of I5, simultaneously set its frequency counting as 1, then connect I2 to the left sub-tree of I4 and set its frequency counting as 1. According to this method, carrying on the nine business of database in turn and obtaining the FP- tree shown as Figure 1.

3.2. Mining FP-Tree

Mining the improved FP-Tree is the core of this algorithm. The original algorithm mined by generating conditional FP-Tree, while the improved algorithm mined left son tree recursively and rebuild FP-Tree in order to obtaining frequent pattern not by producing conditional FP-Tree. The process of mining is as follows.

First, mine the left sub-tree of constructed FP-Tree root node. Obtaining all node frequency of left sub-tree, remove some nodes which not satisfy the support. Take the above constructed FP- tree as an example. First mining I5, there are I1 and I2 and I3 under I5 and their frequency are I1:2, I2:2, I3:1. The current support is 2, so remove I3 node and only leave I1, I2.

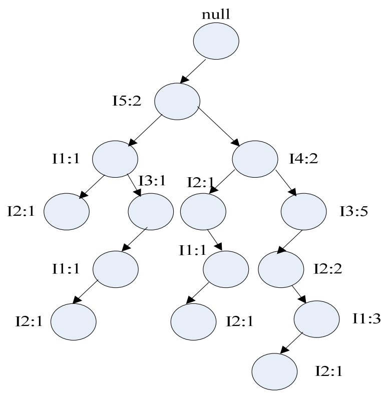

Fig.1. The constructed FP-Tree

Then, rebuild a new sub-tree with the retention node and the left sub-tree of root. If the new sub-tree only has a branch, then output the frequent pattern, otherwise, mine this sun-tree recursively. In this example, there is only one branch under the rebuild sub-tree of I5, so obtain frequent pattern I1 I5:2 I2 I5:2 I1 I2 I5:2.

Finally, connect the sub-tree of I5 to its brother node and delete I5 node, then connect its brother node I4 to the left sub-tree of root node and obtain the new tree, shown as Figure 2.

Carry on the above mining to the left sub-tree of root node repeatedly until left sub-tree has not any brother node.

3.3. The realize of improved algorithm

1) Data Structure

The improved algorithm does not need an item table. The data structure of node is as follows.

```
class FPTree
{    CString  item_name;  //the name of node
          int count;  //support counting
          FPTree* child;  //child node
          FPTree* brother;  /brother node
          FPTree* father;  //father node    }
```

2) The algorithm of constructing FP-Tree

The improved algorithm in constructing FP-Tree is mainly manifest in the function insert_Tree([p|P],T).It has two steps: Search(p|P],G) (G is the current seek node)and Create(p,G). There are the two algorithm descriptions as follows:

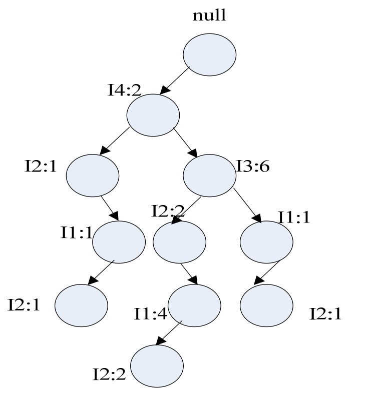

Fig.2. The new tree after the first mining

```
a)  Algorithm descriptions: Search(p|P],G) (G is the current seek node)
if (G is not null)
{ if(L,the left sub-tree of G is not null)
      { if( L.item_name = p. item_name)
                The count of L add 1;
         else
                Search(p|P],L.brother);
      }else
             Create(p,G);
 }else
        Create(p,G);
b)  Algorithm descriptions: Create(p,G)
if (G is not null)
 {  if(R,the right sub-tree of G is not null)
            Create(p,R);
     else
          Connect node p to R node, the right sub-tree of G
 }else
      Connect node p to the left sub-tree of G, return
```

3) The algorithm of mining FP-Tree

There are two steps in the improved algorithm of mining FP-Tree: MineTree_L(L) and MergeTree(L,T). There are the algorithm descriptions as follows.

```
a)  Algorithm descriptions: Mine(G) (G is the root node of current mining tree)
  if(G is only have left sub-tree)
        Outputs the frequent item set;
 else
 {    MineTree_L(L);
      MergeTree(L,T);   }
b)  Algorithm descriptions: MineTree_L(L)
  Count the frequent counting of L;
  Obtain the node information of the left node according to the support;
  newTree = Rebulid(t,K);
  if(newTree is only have one branch)
        Outputs the frequent item set;
  else
        Mine(newTree);
c)    Algorithm descriptions: MergeTree(L,T)
  if (L.child is not null)
 {     insert(L.child,T);
       C= L.child;
       B = C.brother;
       while(C.child is not null)
       {      insert(C.child,C);
              C=C.child;    }
       while(B is not null)
      {       insert(B,T);
              C1= B.child;
              B1= B.brother;
              while(C1is not null)
             {     insert(C1,B);
                   C1=C1.child;    }
             while(B1 is not null)
             {     insert(B1,T);
                   B1=B1.btorher;    }    }
      B = B.brother;    }
   Delete L;
Transfer the right sub-tree of root node to the left sub-tree
```

4. Experimental result analysis

4.1. Timely analysis

In order to indicate the execution efficiency of the improved algorithm, carrying on the simulation experiment. Under the similar experiment environment, comparing this improved algorithm with the FP-growth algorithm.

Experiment environment: Intel Core 2 2.10GHz CPU, memory 1GB, Windows 2003 server.

Experiment data set: Data set D1 contains 1000 business, 800 different items. The average length of business is 35. Data set D2 contains 100000 business, 8000 different items, the average length of business is 50. The experiment result is shown as Figure 3.

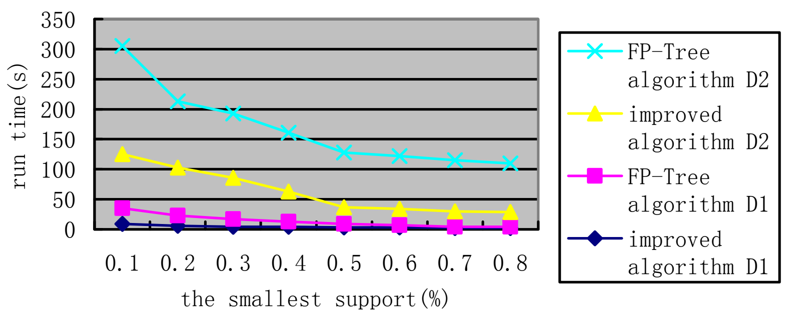

Fig.3. the comparison between improved algorithm and original algorithm

From the Fig.3, compared with the original algorithm, the improved algorithm has the obvious superiority in the time efficiency. Especially when the support is quite small, the execution time of improved algorithm is only one third of the originals'. When the business number in the data set is quite small, the growth of improved algorithm is quite steady.

In order to test the elasticity of this algorithm, select 0.01 as the support, carry on the test to the D2 data set. Run the original algorithm and the improved algorithm in 10000, 200000,…,100000 business data sets separately. The experiment result is shown as Figure 4.

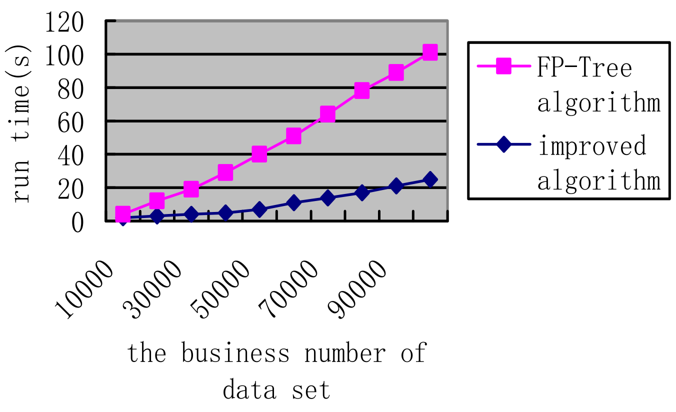

Fig.4. the elasticity of algorithms

From the Fig.4, with the business number increased, the improved algorithm presents the rather steady, and displays the good elasticity.

4.2. Spatiality analysis

The original FP- Tree algorithm takes the common tree as its storage structure. The storage structure of each node is shown in Figure 5.

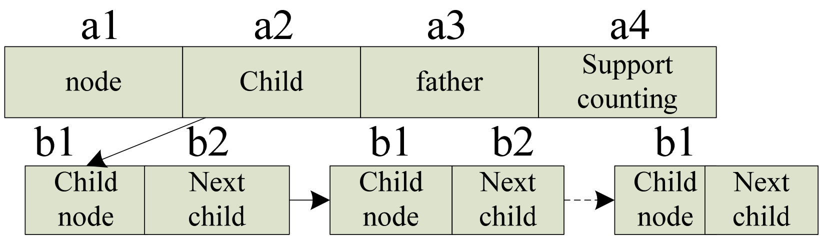

Fig.5. the storage structure of original algorithm

Supposed the storage space of node is a_1, the child's point is a_2, the father's point is a_3, the support counting's is a_4, the storage space of child is b_1, the storage space of next child is b_2.Then the storage space of each node is

$$K_i = a_1 + a_2 + a_3 + m_i(b_1 + b_2) \qquad (1)$$

And m_i is the child numbers of this node. If there are n nodes, the storage space is

$$K = K_1 + K_2 + K_3 + ... + K_n = n(a_1 + a_2 + a_3 + a_4)$$
$$+(m_1 + m_2 + ... + m_n)(b_1 + b_2) \qquad (2)$$

There are n nodes. There are (n-1) child nodes,

$$m_1 + m_2 + m_3 + ... + m_n = n - 1 \qquad (3)$$

Then the storage space is

$$K = n(a_1 + a_2 + a_3 + a_4) + (n-1)(b_1 + b_2) \qquad (4)$$

The improved algorithm takes binary tree as its storage structure. The storage structure is shown in Figure 6.

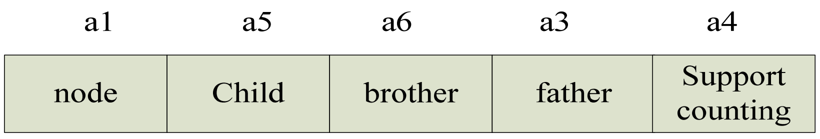

Fig.6. the storage structure of improved algorithm

Similarly are n nodes, its storage space is

$$C = n(a_1 + a_3 + a_4 + a_5 + a_6) \qquad (5)$$

Then the economical spatial is

$$K - C = n(a_2 - a_5 - a_6) + (n-1)(b_1 - b_2) \qquad (6)$$

Each field expresses the indicator type, so the storage space is equal, supposed $a_2 = a_5 = a_6 = b_1 = b_2 = a$ .So the finally economical spatial is

$$K - C = (n-2)a \qquad (7)$$

From this, the improved algorithm reduced the storage space greatly. Especially when the value increases, this kind of superiority displays obviously.

5. Conclusions

Finding out the frequent item set is the most essential step in the association rule mining, so the highly effective algorithm of mining frequent pattern is very important. This article improved the storage structure of original FP-Tree algorithm, omitted the process of producing conditional FP- tree by mining left son tree recursively. The improved algorithm surpasses the original algorithm in the space and time efficiency. But how effectively mine FP-Tree and obtain the frequent pattern is still a topic which is worth further studying.

References

[1] Van der Geer J, Hanraads JAJ, Lupton RA. The art of writing a scientific article. J Sci Commun 2000;163:51-9.
[2] Strunk Jr W, White EB. The elements of style. 3rd ed. New York: Macmillan; 1979.
[1] HAN J, KAMBER M, Data Mining Concepts and Techniques(in Chinese) [M], Beijing: China Machine Press, 2001.
[2] Fan Ming, Li Chuan, Mining Frequent Patterns in an FP-tree Without Conditional FP-tree Generation(in Chinese) [J], Journal of Computer Research And Development, 2003, 8: 1216 - 1222.
[3] Zhao Yanduo, Song Binheng, Algorithm for mining frequent patterns based on converse FP-tree(in Chinese) [J], Computer Applications, 2005,6： 1385- 1387.
[4] Meng Xiangping, Wang Huajin, Wang Xiangyong, Mining Maximal Frequent Patterns Based on Improved FP-Tree(in Chinese) [J], Computer Engineer and application, 2005,14:179- 181.
[5] Liang Bizhen, Lu Yueran,Qin Liangyi,An Improved FP-Tree -Based Algorithm for Maximal Target Frequent Itemsets Mining(in Chinese) [J], Computer Engineer And Science, 2007,10:70- 72.
[6] Ma Xuhui, Zhang Ahong, Association Rules Generated By the FP Tree Depth-First Algorithm(in Chinese) [J], Artificial intelligence and recognition technology, 2010,6: 3439-3440.