Algorithm of Mining Association Rules Based on Compressed Association Graph

Wei Wan
School of Computer Science & Engineering
Southeast University
Nanjing, P.R.China
wanwei120@126.com

Hao Jiang
School of Computer Science & Engineering
Southeast University
Nanjing, P.R.China
hjiang@seu.edu.cn

Xiaojie Li
School of Computer Science & Engineering
Southeast University
Nanjing, P.R.China
xiaojielee@yahoo.cn

***Abstract*—Graph-based algorithm DLG in mining association rules is introduced first. Then another improved algorithm CAG is proposed. The features of node degree and frequent itemset are used to compress the association graph. Experiment is done to give comparison on efficiency between DLG and CAG algorithm. Result shows CAG algorithm performs better than DLG algorithm under the conditions of different support and transactions.**

Key Words：Apriori algorithm; Transaction matrix; Association graph

I. INTRODUCTION

Mining association rules is one of the most active methods in data mining field. It is also one of the most important and widely used methods in this area currently. Since Agrawal proposed Apriori[1] algorithm, Apriori has become a classic algorithm in mining association rules. A large number of scholars have made much improvement on the basis of this algorithm. AprioriTid [2] algorithm replaces the original transaction database with shrinking Tid table to improve the search efficiency. Partition [3] algorithm introduces the idea of parallel mining. Sampling algorithm makes compromise for mining accuracy and efficiency. DHP [4] algorithm uses hashing to improve the efficiency of candidate itemsets (especially the 2-candidate itemset) formating process. However, these algorithms are all based on Apriori algorithm which uses continuous iterative procedure to generate frequent itemsets. Although these algorithms have taken some measures in pruning candidate itemsets, however, a large number of useless candidate itemsets will still be produced when database is too huge. In addition that Apriori algorithm needs to scan the database repeatedly. These features have become the bottleneck restricting efficiency of the algorithm. FP-Tree algorithm [5] is a kind of association rule algorithm that does not generate candidate itemsets. This algorithm has greatly improved the efficiency of generating frequent itemsets, but disadvantage is that the algorithm takes too much memory. Especially, the cost is great or even it is impossible to establish FP-tree in the memory when the number of transactions is too large. Yen first proposed the graph-based association rule algorithm Direct Larger Itemset Generation (DLG) [6] .DLG algorithm translates frequent itemsets mining into association graph searching. This algorithm scans database only once, omitting the connection steps in apriori algorithm, reducing the size of candidate itemsets, so the efficiency is improved. In this paper, an improved algorithm CAG based on DLG, compresses candidate itemsets with the combination of node degree [7] in association graph and the relevant characteristics of frequent itemsets. And experiment is done to prove the effectiveness of the improved algorithm.

978-1-61284-365-0/11/$26.00 ©2011 IEEE

II. DLG ALGORITHM

Each transaction in database can be abstracted as a binary vector. If an item appears in the transaction, we can represent it as 1, else 0. Table 1 shows some transactions in database, and they can be expressed as binary vectors in the right column of the table.

TABLE I. Binary vector of transactions

Transactions	Vectors
I_2, I_3, I_4, I_5	01111
I_2, I_3, I_5	01101
I_1, I_2, I_3	11100
I_1, I_2	11000
I_3, I_4, I_5	00111
I_1, I_4, I_5	10011

When a transaction is represented as binary vector, all the binary vectors of the affairs can form a transaction matrix [8]. Each line in matrix represents a transaction, and each column represents an item. The number 1 in each column vector represents that the item shows in the corresponding transaction. When we need to compute the support of a k-itemset, we just need to take the operation "and" to the corresponding column vectors. The number of 1 in the result vector is the support of the k-itemset. The transaction database D which we can describe as: $BV= [BV_1, BV_2, BV_3 \ldots BV_N]$. Among them, N is the number of items in database; BV_i represents the column vector of item I_i. Thus $BV_1=[001101]^T$, $BV_2=[111100]^T$, $BV_3=[111010]^T$, $BV_4=[100011]^T$, $BV_5=[110011]^T$. In BV_i, the number 1 in position j represents that the item I_i appears in the transaction D_j. We can compute whether an itemset is a frequent itemset through this expression: $C_k=BV_1 \wedge BV_2 \wedge \ldots \wedge BV_k$. If the number of 1 appearing in C_k is larger than the minimum support, we can draw a conclusion that C_k is a frequent itemset.

In DLG algorithm, we need to find all the 2-frequent itemsets. For instance, In Fig.1, the 2-frequent itemsets are listed as follows: $\{I_1,I_2\},\{I_2,I_3\},\{I_2,I_5\},\{I_3,I_4\},\{I_3,I_5\},\{I_4,I_5\}$. Then we create the association graph using 2-frequent itemsets, and extension is made to each itemset. As shown in Fig.1，edge $\{I_1,I_2\}$ can be extended to $\{I_1,I_2,I_3\}$ and $\{I_1,I_2,I_5\}$. Then we compute the support of all extended itemsets. For a k-frequent itemset, the rule of extension: if there is an edge from I_k to I_j, we extend the k-frequent itemset to (k+1)-candidate itemsets $\{I_1, I_2 \ldots I_k, I_j\}$, cycling until no edge to be extended.

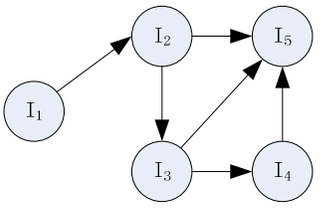

Figure 1. Original association graph

III. IMPROVED ALGORITHM CAG

To describe conveniently, we name the improved algorithm as CAG. Before introducing the specific algorithm, we dig out some useful theorems in mining association graph, detailed as follows:

Theorem 1: In association graph, if node degree is smaller than k-1 when we generate k-itemsets, this node can not be extended to k -frequent itemsets. The node associated with its edge can be removed from the association graph.

Proof: Suppose that the node can be extended to k-frequent itemsets, considering all the 2-subsets of frequent itemsets which contain the node, according to the law that any subset of

frequent itemsets is frequent itemsets, all the k-1 2-subsets are frequent itemsets. Then there are k-1 edges between this node and the other k-1 nodes, contradiction arising because the degree of this node is smaller than k-1. So the former part of the theorem is established. Then we prove why we can delete the node and its connected edges from the association graph. We just need to prove that the node showing in higher frequent itemsets is impossible. Suppose the node can be extended to higher frequent itemsets, denoted by (k + m)(m>0), thus there are at least k + m-1 edges connected with the node. We have already known the number of edges connected to this node is not more than k-1. Therefore, it is impossible that the node appears in the (k+m)-frequent itemsets(m>0). So we can remove the node from the association graph, with no impacts to the future generation of frequent itemsets.

Theorem 2: In association graph, when we make extensions to known k-frequent itemsets, if we find out at least one edge does not exist from one item in k frequent itemsets to the node to be extended, we should not make this extension [9]. That is because the extended (k+1)-itemset will not be a frequent itemset.

Proof: Suppose the extended (k+1)-itemset would be a frequent itemset, then the (k+1)-itemset would have k 2-subsets containing the extended node, and all the 2-subsets would be frequent itemsets. Then, it is inevitable that there will be edges between the k nodes and the extended node, as is shown in Fig.2.

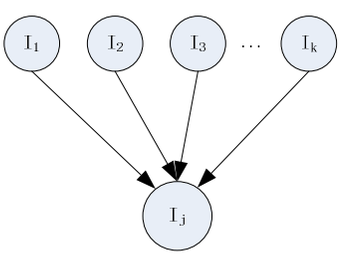

Figure 2. Condition of theorem 2

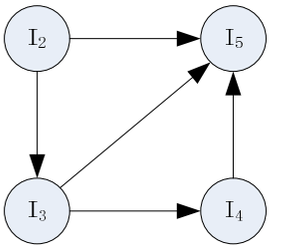

Figure 3. Association graph for sample

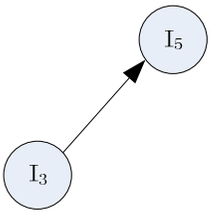

Figure 4. Association graph for sample

Here we have a specific example to demonstrate the procedure of the algorithm:

When we compute the support of the 3-frequent itemsets in Fig.1, the degree of I_1 is smaller than 2. Thus we delete I_1 and its connected edge, and we get figure 3. In Fig.3, edge $\{I_2, I_3\}$ can be extended to $\{I_2, I_3, I_5\}$ and $\{I_2, I_3, I_4\}$, but the latter extension $\{I_2, I_3, I_4\}$ is not necessary. That is because there is no edge between I_2 and I_4. Thus itemset $\{I_2, I_3, I_4\}$ is not a frequent itemset. Edge $\{I_3, I_4\}$ can be extended to $\{I_3, I_4, I_5\}$ which is proved to be another 3-frequent itemset. Finally we find out two frequent itemsets $\{I_2, I_3, I_5\}$ and $\{I_3, I_4, I_5\}$. Now we know we need not extend this edge to 4 frequent itemsets, because the degree of I_2 and I_4 is smaller than 3. Then we can delete them from association graph, after that we get Fig.4 which contains only two nodes, and mining procedure is over.

Description of CAG algorithm:

1. Scan the database and create transaction matrix.

2. Find out all the 2-frequent itemsets.

3. Create association graph.

4. Find out all the k-frequent itemsets(k>2), and compute the degree of the nodes. If the degree is smaller than k-1, delete the node and its connected edge, applying theorem 2 when extending the (k-1)-itemsets.

5. Execute k++. If the number of nodes in association graph is smaller than k (theorem 1), repeat step 5, else the algorithm is over.

Pseudo-code of CAG algorithm (core step 4 and 5)

Input: all the transactions

Output: frequent itemsets

Method:

```
While Node.size>=k do
  If Node[i].degree<k /*Theorem 1*/
    Delete Node[i]
    For k=1 to Node.size
      If ExistEdge[i, k] or ExistEdge [k,i]
        Delete Edge[i, k] or Edge[k,i];
      End If
    End For
  End If
  For Each Freq_ItemSet{I1,I2,…,Ik-1,Ik}
  In all frequent itemsets
  While ExistEdge[k,j] /*Theorem 2*/
    Num=0
    For p=1 to k-1
      If ExistEdge[p,j]
        Num++
      End If
    End For
    If Num==K-1
      Extend to
      CandidateItem{ I1,I2,…,Ik-1,Ik ,Ij}
      Sup(Ck+1)=BV1^BV2^ … ^BVk-1^
      BVk^ BVj /*Compute support*/
      IF Sup(Ck)>Min_Sup;
        Extend to Freq_Itemset Lk+1
      End If
    End If
  End While
 End For
End while
```

This algorithm makes some improvement in reducing redundant candidate itemsets through two measures. On the one hand, we compress the association graph; on the other hand, we restrict the extension of the frequent itemsets.

IV. EXPERIMENT

We make the comparison between DLG and CAG algorithm through experiments. In Fig.5, we set the number of transactions for 1000, and we test two algorithms with different support. In Fig.5 we set the support for 2%, we also test two algorithms with different number of transactions. As the Fig.5 and Fig.6 show, under the conditions of different number of transactions and support, CAG has a better performance.

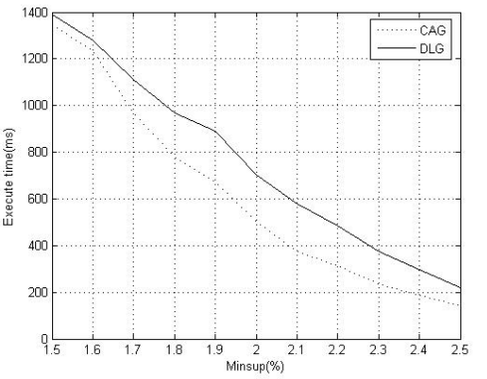

Figure 5. Comparison under different support

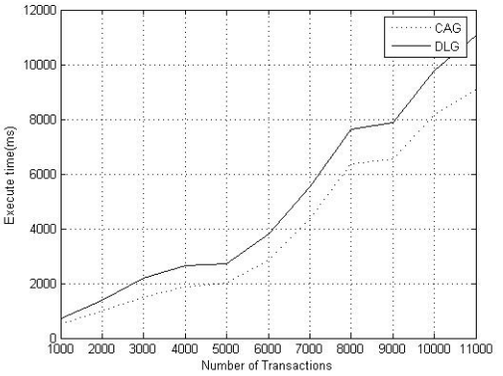

Figure 6. Comparison under different transactions

V. CONCLUSION

In this paper, the association rule mining algorithm based on DLG is put forward, some possible improvement is proposed. On the one hand, we propose a method to compress the association graph by digging out the potential relationship between node degree and frequent itemsets; on the other hand, according to the law that any subset of frequent itemset is a frequent itemset, we restrict the extension of the candidate itemsets. However, CAG algorithm also has its disadvantages. When most transactions contain nearly all the items or the support is fairly low, CAG can not improve the efficiency obviously. That is because under this condition, the association graph is close to be a complete graph and the condition to compress the association graph is hard to satisfy.

REFERENCES

[1]Rakesh Agrawal, Tomasz Imielinski, Arun Swami. Mining Association Rules Between Sets of Items in Large Databases. ACM SIGMOD, 1993. 207~216

[2] Rakesh Agrawal, Ramakrishnan Srikant. Fast Algorithm for Mining Association Rules. Proceedings of 20th Int. Conf. Very Large Data Bases (VLDB), Jorge B. Bocca, Matthias Jarke and Carlo Zaniolo, eds. Morgan Kaufmann Press, 1994. 487~499

[3]A. Savasere, E. Omiecinski, S. Navathe. An Efficient Algorithm for mining association Rules in Large Databases. Proceeding of 21th Int'l Conf. Very Large Data Bases. San Francisco: Morgan Kaufmann, 1995. 432~444

[4]Jong Soo Park, Ming-Syan Chen, Philip S. Yu. An Effecive Hash Based Algortithm for Mining Association Rules. Proceedings of ACM SIGMOD International Conference on Management of Data. Michiael J. Carey and Donovan A. Schneider eds. San Jose, California, 1995, 175~186

[5]Jiawei Han, Jian Pei, Yiwen Yin. Mining Frequent Patterns without Candidate Generation. Proceedings of ACM SIGMOD Intl. Conference on Management of Data, Weidong Chen, Jeffery Naughton, Philip A. Bernstein eds. ACM Press, 2000. 1~12

[6] Show-Jane Yen, Arbee L.P. Chen. An Efficient Approach to Discovering Knowledge from Large Databases [C]. Proceedings of The IEEE/ACM International Conference on Parallel and Distributed Information Systems. Los Angeles: IEEE Computer Society Press, 1996:8~18

[7]LIU Du-yu, ANG Jin-hao, ZHONG Shou-ming. Research on high efficiency algorithm based on graph for mining association rules. Computer Engineering and Design. 2006,27(23):4475~4478

[8]L.Jaba Sheela, V. Shanthi, D.Jeba Singh, Image Mining using Association rules derived from Feature Matrix, In proceedings of the 2009 International Conference on Advances in Computing, Communication and Control (ICAC3'09), Mumbai, Maharashtra, India, 2009, pp.440~443.

[9] Huang Hongxin. Revised Algorithm of Mining Association Rules Based on Graph[J]. Computer & Digital EngineerinG, 2009, 37(12): 38~42