A NEW ASSOCIATION RULE MINING BASED ON FREQUENT ITEM SET

Ms. Sanober Shaikh[1] Ms. Madhuri Rao[2] and Dr. S. S. Mantha[3]

[1]Department of Information Technology, TSEC, Bandra (w), Mumbai
s.sanober1@gmail.com
[2]Department of Information Technology, TSEC, Bandra (w), Mumbai
my_rao@yahoo.com
[3]AICTE, Chairman, Professor, CAD/CAM Robotics, VJTI, Mumbai
ssmantha@vjti.org.in

ABSTRACT

In this paper a new mining algorithm is defined based on frequent item set. Apriori Algorithm scans the database every time when it finds the frequent item set so it is very time consuming and at each step it generates candidate item set. So for large databases it takes lots of space to store candidate item set. The defined algorithm scans the database at the start only once and then makes the undirected item set graph. From this graph by considering minimum support it finds the frequent item set and by considering the minimum confidence it generates the association rule. If database and minimum support is changed, the new algorithm finds the new frequent items by scanning undirected item set graph. That is why it's executing efficiency is improved distinctly compared to traditional algorithm.

KEYWORDS

Undirected Item set Graph, Trade List

1. INTRODUCTION

Mining Association rule is very important field of research in data mining. The problem of mining Association rule is put forward by R.S Agarwal first in 1993. Now the Association rules are widely applied in E-commerce, bank credit, shopping cart analysis, market analysis, fraud detection, and customer retention, to production control and science exploration. etc. [1]

Now a days we will find many mining methods for finding the frequent item set such as Apriori algorithm, Frequent Pattern-Tree algorithm etc. Apriori algorithm's disadvantage is it generates lot of candidate itemsets and scans database every time. If database contains huge number of transactions then scanning the database for finding the frequent itemset will be too costly and it generates a lot of candidates. Next FP-Tree algorithm's advantage is it does not produce any candidate items but it scans database two times in the memory allowed. But when the memory does not meet the need, this algorithm becomes more complex. It scans the database more than two times and the I/O expenses will increase [2]. That is why there is need to design an efficient algorithm which updates, protects and manages the association rule in large transactional database. So far many researchers made analysis and research for how to efficiently update the association rules and put forward corresponding algorithm. There are two instances in the problem of Association Rule update. The first instance is when the database is changed then how to find frequent item sets. FUFIA Algorithm is the representational updating method for this

David Bracewell, et al. (Eds): AIAA 2011,CS & IT 03, pp. 81–95 , 2011.
 DOI : 10.5121/csit.2011.1308

problem. The second instance is when the minimum support is changed then how to find frequent items sets. IUA algorithm is the representational updating method for this problem. These updating algorithms have both advantages and disadvantages. This paper proposes a dynamic algorithm of frequent mining based on undirected item set graph which scans the database only once and then saves the information of original database in undirected item set graph and finds the frequent item sets directly from the graph. It does not generate any candidate items. When database and minimum support is changed, the algorithm rescans the undirected item set graph to get the new frequent item sets.[3]

2. BASIC CONCEPT OF ASSOCIATION RULE

Association rule finds interesting associations and/or correlation relationships among large set of data items. Association rule shows attribute value conditions that occur frequently together in a given dataset. A typical and widely-used example of association rule mining is Market Basket Analysis.

For example, data are collected using bar-code scanners in supermarket. Such 'market basket' databases consist of a large number of transaction records. Each record lists all items bought by a customer on a single purchase transaction. Managers would be interested to know if certain groups of items are consistently purchased together. They could use this data for adjusting store layouts (placing items optimally with respect to each other), for cross-selling, for promotions, for catalog design and to identify customer segments based on buying patterns.

Association rules do not represent any sort of causality or correlation between the two item sets The problem of mining association rules can be described as below: if $I = \{I_1, I_2 \ldots I_n\}$ is the set of items. Suppose D is database transaction set and each transaction T contains set of items, such that $T \subseteq I$. Each transaction has identifier called as TID i.e. transaction id. Suppose A is a set of items and transaction T is said to contain A only if $A \subseteq T$.

Association rule is an implication like as $A \Rightarrow B$ in which A, $B \subset I$ and $A \cap B = \varnothing$. [6]

Definition of support: The support is the percentage of transactions that demonstrate the rule. An item set is called frequent if its support is equal or greater than an agreed upon minimal value – the support threshold. [8]

Definition of Confidence: Every association rule has a support and a confidence.

An association rule is of the form: X => Y.

X => Y: if someone buys X, he also buys Y.

The confidence is the conditional probability that, given X present in a transition, Y will also be present. Confidence measure, by definition:

Confidence(X=>Y) = support(X, Y)/ support(X)

The aim of association rule is to find all association problems having support and confidence not less than given threshold value. For the given support i.e. minsupp, if the item set of D's support is not less than minsupp, then it can say that D is the frequent item set.

3. OPTIMIZING ALGORITHM OF MINING FREQUENT ITEM SETS

3.1. Definition And Algorithm Of Undirected Item Set Graph:

3.1.1. Undirected Item Set Graph:

Definition: Undirected item set graph is set of nodes V (V_1, V_2, …..V_n) in the database .Each node contains: the node name, the pointer to other nodes, and the number of nodes to which it

points. The side set E<i, j> of undirected item set graph has two attributes: the side name and the number of side appear. <V_i, V_j> express two frequent item sets; <V_1, V_2…V_n > express n frequent item set.

3.1.2. Construction of Undirected Item Set Graph:

First step is to scan the database. It makes each item as a node and at the same time it makes the supporting trade list for each node. Supporting trade list is a binary group T= {Tid, Itemset} (where Tid is transaction id and Itemset is trade item set). So the side between nodes can be accomplished by corresponding trade list operation. The algorithm does the intersection of two nodes with supporting trade list. When trade list is not empty, that means there is a side between two nodes. The appearance number of each side is the resultant number which algorithm finds by the side's intersection. Given database that includes five items and nine transactions (shown in table one). Suppose that minimum support minsupp is two. Table two contains the information of support trade list of table one.

Table 1: A Store Business Data

TID	The List Of Item ID
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

Table 2: Trade List of Commodity Item

Commodity Item	Support Trade List
I1	T400,T500,T700,T800, T900
I2	T100,T200,T300,T400, T600
I3	T300,T500,T600,T700, T800
I4	T200,T400
I5	T100,T800

Algorithm one: Construction of undirected item sets graph
Input: Database D
Output: Undirected item set graph
Begin
1. Add the items into the vertex set V;
2. For i=1 to n-1
   2.1. Select V_i from V;
   2.2. For each V_j ($j \neq i$)
      2.2.1. If ($I_i \cap I_j$) $\neq \emptyset$ then
      2.2.2. Add link between Vi and Vj //Vi and Vj become adjacent nodes.

2.2.3. End if.
2.3. Next
3. Next
4. End;

3.1.3. Optimize Mining Algorithm Based On Undirected Item Sets Graph

The algorithm in this paper uses the search strategy of Depth first- Search to set universal undirected item graph. The specific steps are shown as follows:

Select a node V_i from node set V. If the number of times Vi appears in the database is not less than the minimum support minsupp, then {Vi} will belong to the item in frequent 1-item set. If count of node Vi adjacent to node Vj's side is not less than support S, then { Vi, Vj } will belong to the item in frequent 2-iterm set. When there are three nodes in undirected item set graph and count of each side of the node is not less than minimum support minsupp, these three nodes < Vk,Vm,Vn> will belong to frequent 3-item set. When there more than three nodes in undirected item sets graph then count of each side of the node should not be less than minimum support minsupp and all the subset of these n nodes should be frequent.

Algorithm two: To find frequent item set based on undirected item sets graph.

Input: Undirected item set graph, minimum support minsupp, minconf
Output: frequent item set L, Association rule
Begin
1. The node set V is empty or not. If it is empty then stop;
2. Find count of each item (e.g. V_i) and check count of each item is greater than or equal to minimum support minsupp. If greater then the items are stored in frequent-1 item set;
3. (frequent item set) = L;

4. Select any unvisited node (e.g. V_j) from adjacent list of V_i;
5. If count ((V_i, V_j) >= minsuppp) then
   5.1. L U V_j;
   5.2. L.adjacentlist = (L.adjacentlist) intersection (V_j .adjacent list);
   5.3. Call DFS (V_j) Procedure;
6. End if;
7. Confidence of each item is compared with minconf and strong association rules are generated.
8. End;

Procedure DFS (V_j):
Begin
1. If V_j.adjacentlist ≠ Φ then
   1.1. Select any other node, suppose V_k from V_j.adjacentlist;
   1.2. Call isloop (L, V_k) Procedure;
   1.3. If count (L, V_k) is greater than or equal to minimum support then combine L U (V_k).
      1.3.1. Call DFS (V_k);
      1.3.2. Output is frequent

item set;
1.3.3. Delete V_k from V_j.adjacentlist;
1.3.4. Call DFS (V_j);
1.4. Else Return to its parent vertex V_i;
1.5. Call DFS (V_i);
2. End;

Procedure isloop (L, V_k):
Begin
1. If V_k € L.adjacentlist then return V_k;
2. Else delete V_k from V_j.adjacentlist;
3. Call DFS (V_j);
4. End;

According to algorithm two, it contains frequent 1-item is L_1= {I_1, I_2, I_3, I_4, I_5}; frequent 2-item is L_2 ={{ I_1, I_2 }, { I_1, I_3}, { I_1, I_5}, { I_2, I_3 }, { I_2, I_4}, {I_2, I_5 }}; frequent 3-item is L_3 = {{ I_1, I_2, I_3}, {I_1, I_2, I_5}}.

3.2. Updating Undirected Item Set Graph

When database and minimum support i.e. minsupp is changed the undirected graph should be changed accordingly. If we want to add some new items to the database, then undirected item set graph is updated accordingly. At this time, the new frequent item sets can be found only by running algorithm two again. When the minimum support is changed, new frequent item set can be found only by adjusting the parameter of algorithm two again.

3.2.1. Database Affair Changed

For example, when a new item T910 is added to table one; the result is as shown as in table three.

Table 3: The New Data in a Store

TID	The list of items
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
T910	I1,I4

A new item T910 have added at this time. So the arisen number of side <$I1, I4$> is two. As shown in fig.1, frequent 1-item set is L1= {I_1, I_2, I_3, I_4, I_5};
frequent 2-item set is L2={{ I1 , I2},{ I1 , I3 },{ I1 , I5},{ I2 , I3 },{ I2 , I4 }, {I2, I5}, {I1, I4 }}; frequent 3-item set is L3={{ I_1, I_2, I_3 },{ I_1, I_2, I_5},{ I_1, I_2, I_4}}according to algorithm two.

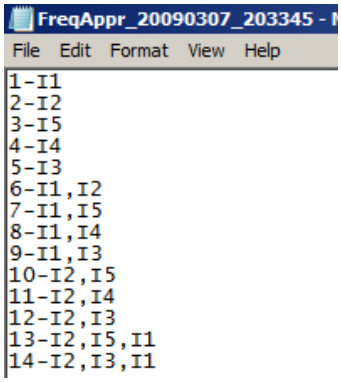
3.2.2 Minimum support changed

For example, when the minimum support minsupp is three, frequent 1-item set={I_1, I_2 , I_3 }; frequent 2- item is L2={{ I_1, I_2},{ I_1 , I_3},{ I_2 , I_3 }} according to algorithm two.

4. RESULTS

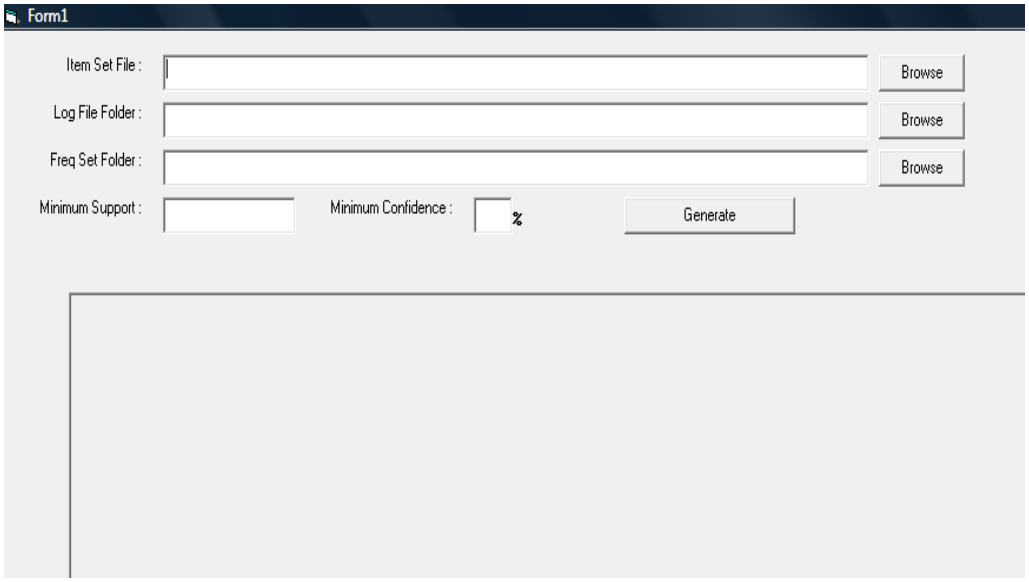
4.1 Results of Apriori Algorithm

Fig1: Frequent Item Set with Apriori Algorithm with database shown in Table 1



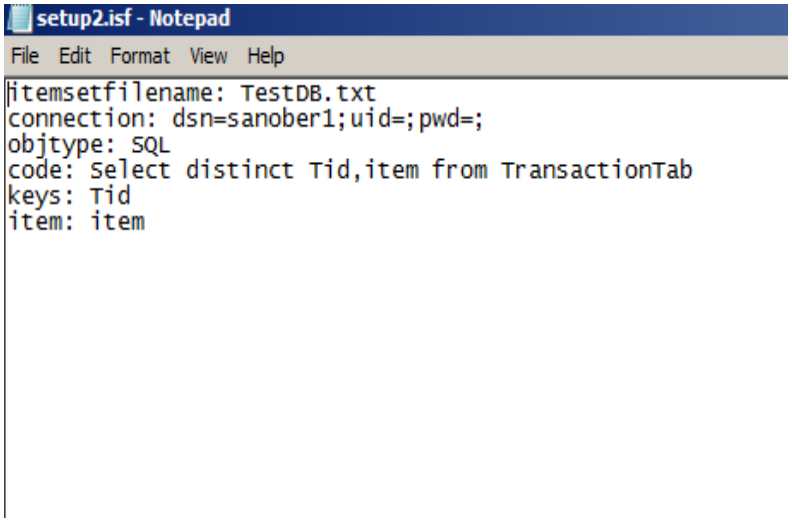
4.2 Results of Undirected Item Set Graph

Fig 2: Main Form of Undirected Item Set Graph



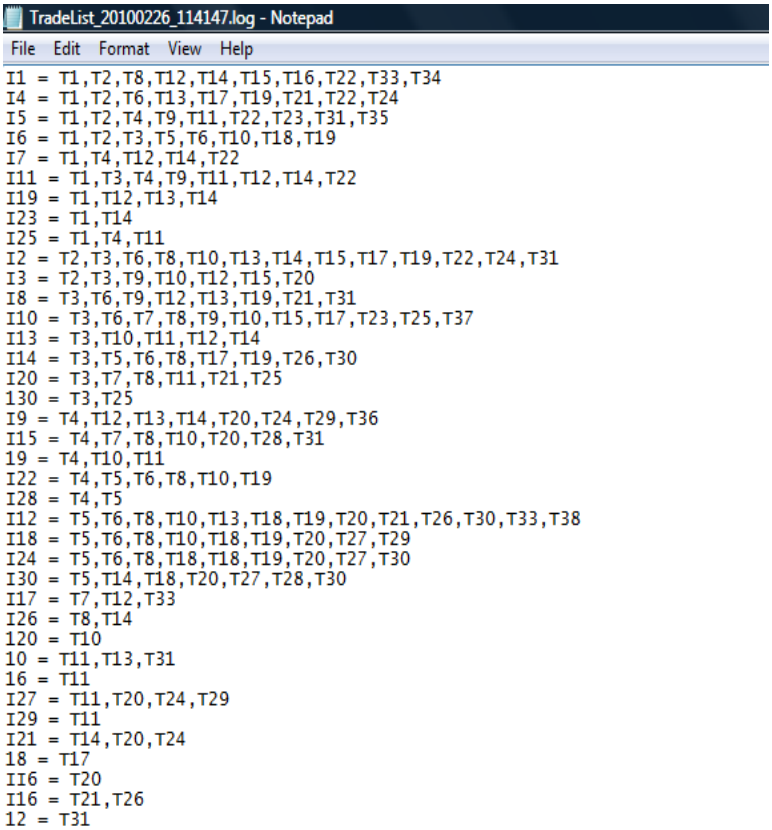
In Fig 2form the first i.e. Item Set File asks for the database from which you want to retrieve the frequent items. Here for input of Item set file one .isf file is made as shown in Fig 3. In that file the code for connectivity with database is made. Through the code the database is converted to a text file. In the first line write name of .isf file that will be converted to a format which the code will accept.

Fig 3: Item Set File



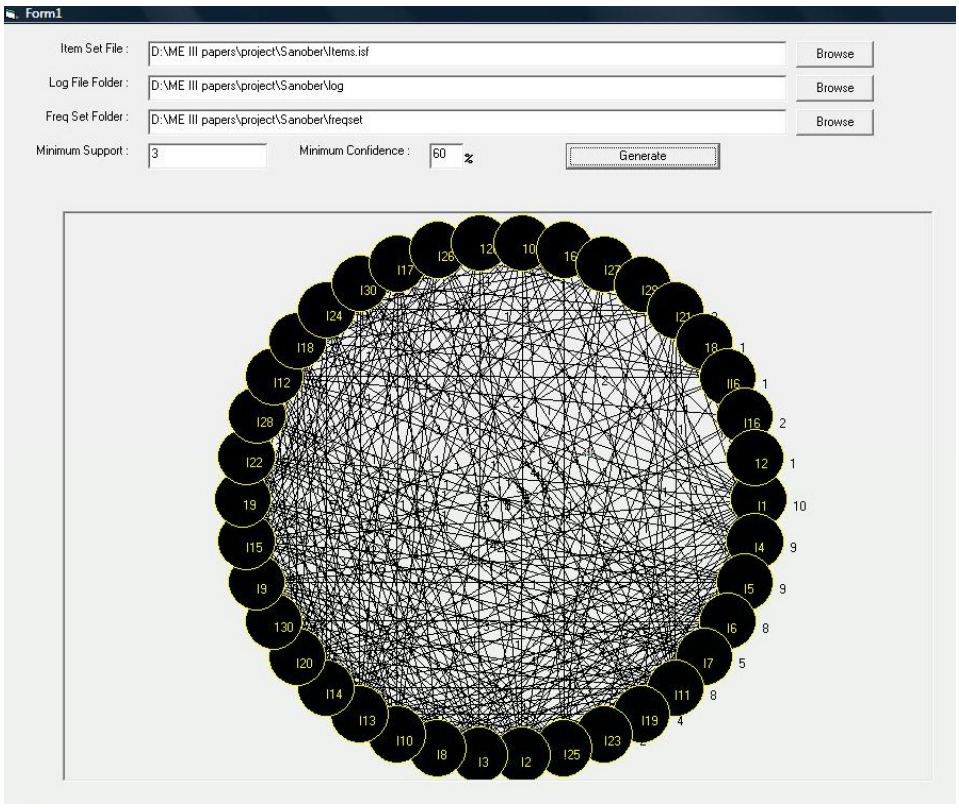
When we will click on Generate button in Fig 2, Trade list is made from which we can come to know how many number of items are present in input database as shown in Fig 4.

Fig 4: Trade List



Then with the help of this Trade list Undirected Item Set graph is made as shown in Fig 5. All the items will become nodes in Undirected item set graph. From this graph frequent item are found and kept in Freq set folder.

Fig 5: Undirected Item Set Graph



Here minimum support is 3. Now the count of each item is compared with minimum support. If count is greater than minimum support those items will be frequent item sets as shown in fig 6.

Fig 6: Frequent Items

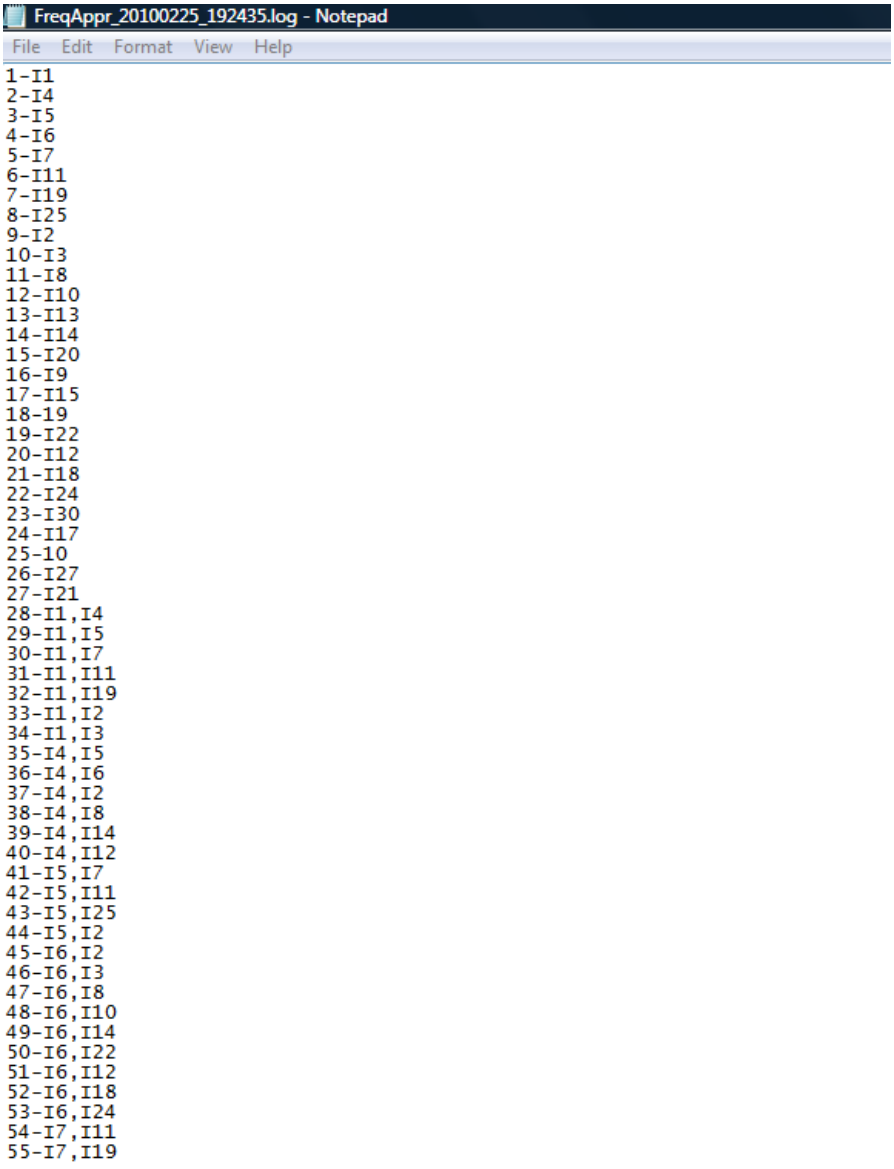


Fig 6 (cont): Frequent Items

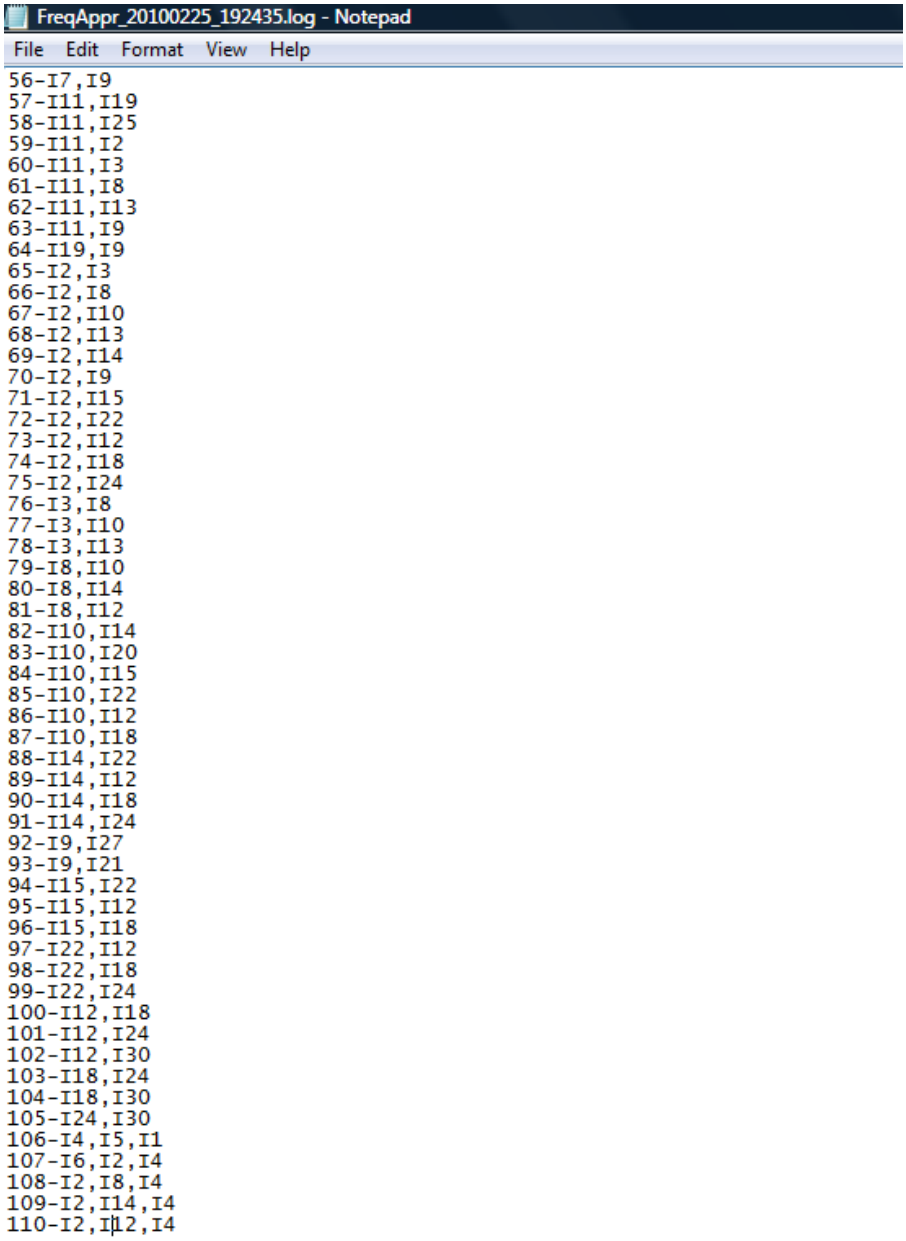


Fig 6 (cont): Frequent Items

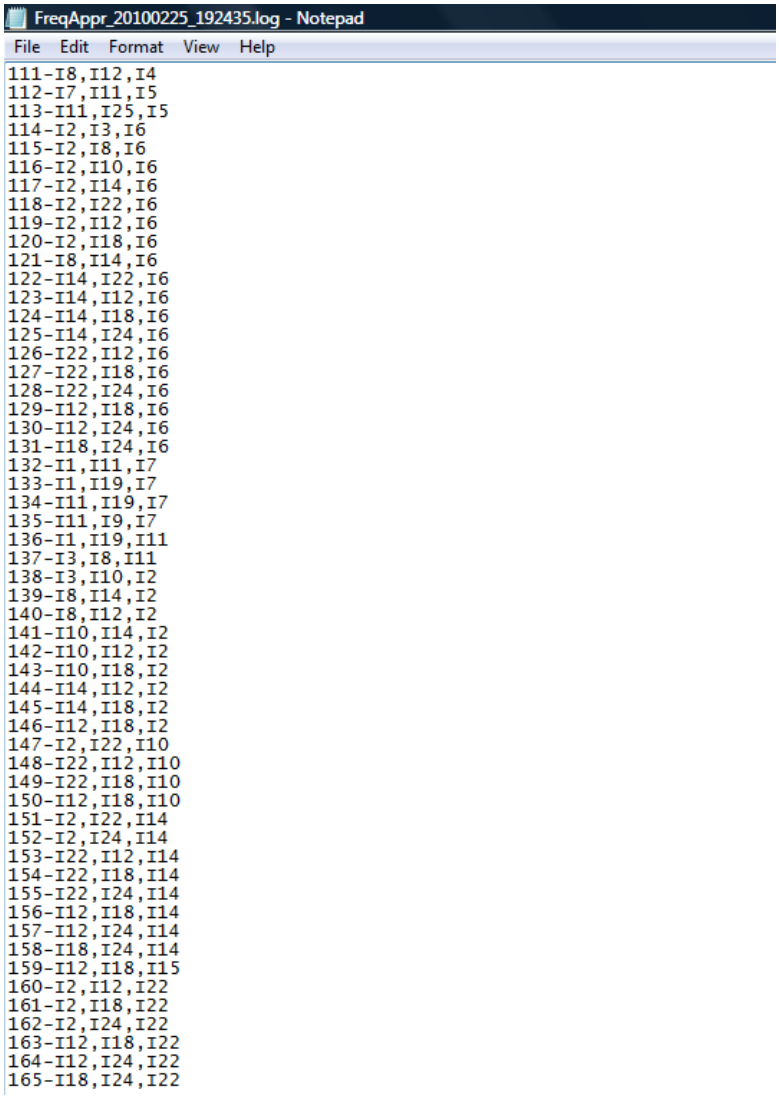
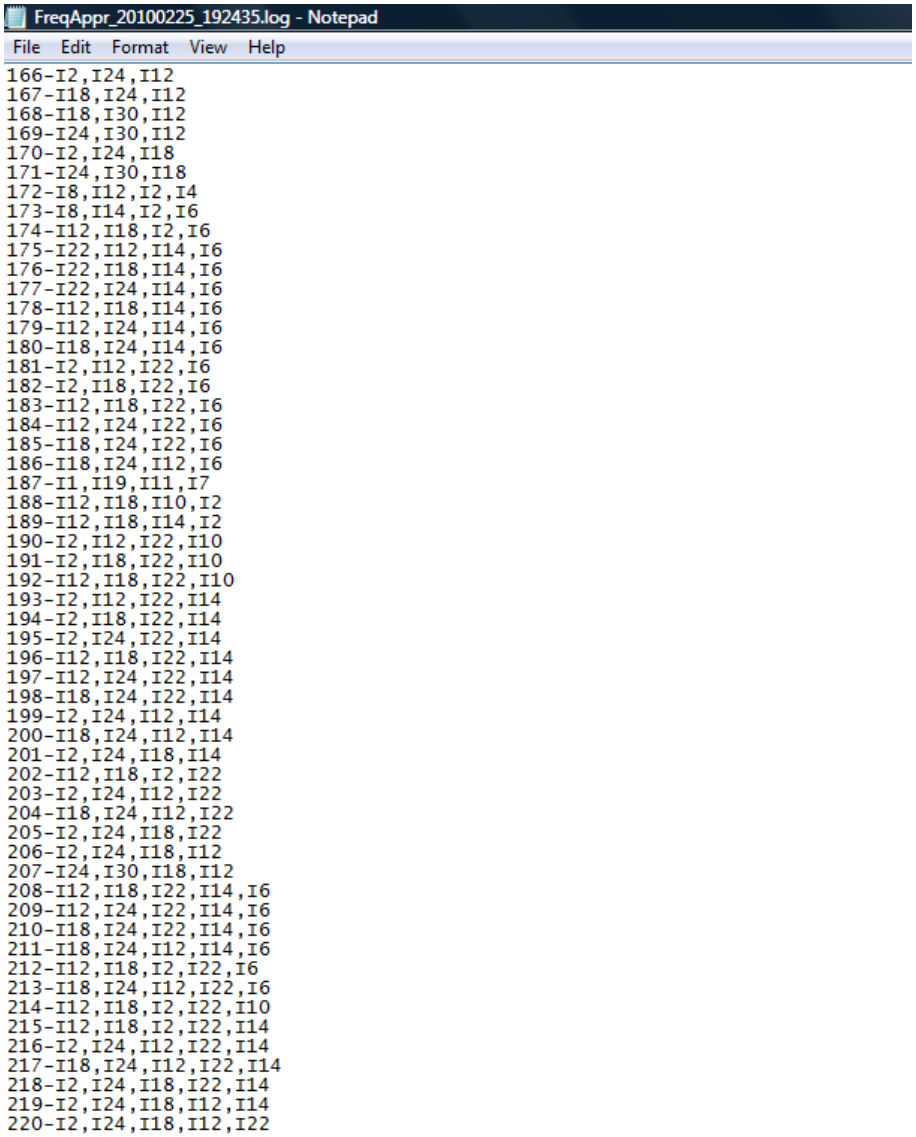


Fig 6 (cont): Frequent Items



Confidence of each item is compared with minimum confidence given by user and strong association rule is formed. The items having confidence greater than or equal to minimum confidence, are stored in file shown in Fig 7.

Fig 7: Association Rule

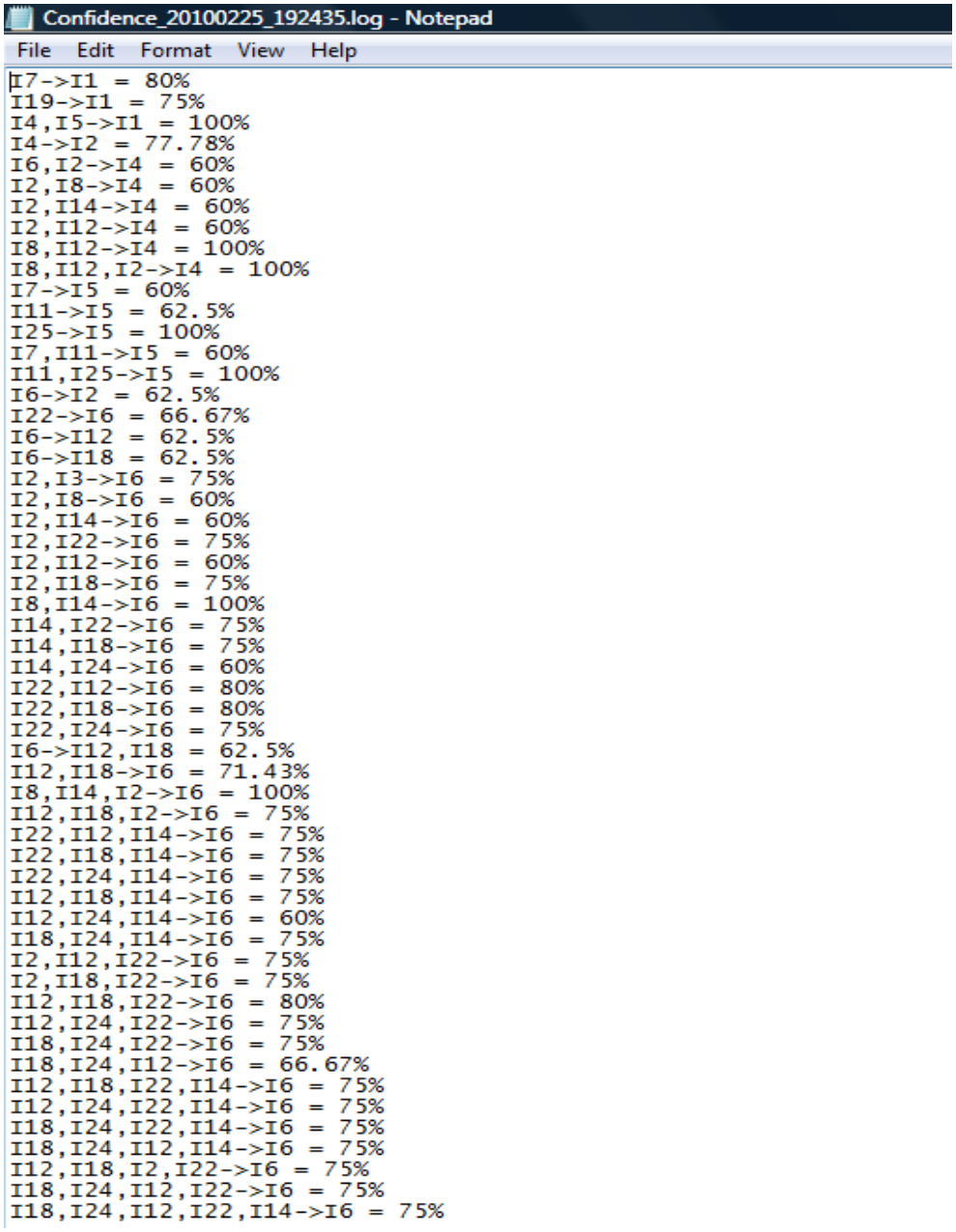


Fig 7(cont): Association Rule

Confidence_20100225_192435.log - Notepad

File Edit Format View Help

```
I2,I24,I22->I14 = 100%
I12,I18,I22->I14 = 80%
I12,I24,I22->I14 = 100%
I18,I24,I22->I14 = 100%
I2,I24,I12->I14 = 100%
I18,I24,I12->I14 = 66.67%
I2,I24,I18->I14 = 100%
I12,I18,I2,I22->I14 = 75%
I2,I24,I12,I22->I14 = 100%
I18,I24,I12,I22->I14 = 100%
I2,I24,I18,I22->I14 = 100%
I2,I24,I18,I12->I14 = 100%
I2,I24,I18,I12,I22->I14 = 100%
I27->I9 = 75%
I21->I9 = 100%
I22->I12 = 83.33%
I22->I18 = 83.33%
I22->I24 = 66.67%
I22->I2,I12 = 66.67%
I2,I12->I22 = 80%
I22->I2,I18 = 66.67%
I2,I18->I22 = 100%
I2,I24->I22 = 100%
I22->I12,I18 = 83.33%
I12,I18->I22 = 71.43%
I22->I12,I24 = 66.67%
I22->I18,I24 = 66.67%
I22->I12,I18,I2 = 66.67%
I12,I18,I2->I22 = 100%
I2,I24,I12->I22 = 100%
I22->I18,I24,I12 = 66.67%
I18,I24,I12->I22 = 66.67%
I2,I24,I18->I22 = 100%
I2,I24,I18,I12->I22 = 100%
I18->I12 = 77.78%
I24->I12 = 77.78%
I2,I24->I12 = 100%
I18,I24->I12 = 85.71%
I18,I30->I12 = 75%
I24,I30->I12 = 80%
I2,I24,I18->I12 = 100%
I24,I30,I18->I12 = 75%
I18->I24 = 77.78%
I24->I18 = 77.78%
I2,I24->I18 = 100%
I24,I30->I18 = 80%
I30->I24 = 71.43%
I2,I22->I12,I18 = 100%
I2,I22,I6->I12,I18 = 100%
I2,I22,I10->I12,I18 = 100%
I2,I22,I14->I12,I18 = 100%
```

5. CONCLUSION

In this project candidate items are not generated. The information of items of original database is saved in undirected item set graph. Then the information of frequent item set is found by searching the undirected item set graph. The space complexity of algorithm depends on mainly storage of undirected item set graph and the size of undirected item set graph is decided by the length of database items. The time complexity of algorithm mainly depends on scanning the database and executive numbers of DFS when scanning undirected item set graph. The time of scanning database is decided by its length and its average estimate time is O (n). Executive numbers of DFS is decided by numbers of adjoin node and the length of frequent item. Its average estimate time is O (N*K*L) where N is the number of nodes; K is the number of adjust nodes and L is the length of frequent item set. Its time complexity is far less than Apriori algorithm.

6. ACKNOWLEDGEMENT

1. Ms Madhuri Rao(Guide)

2. Mr. Naushad Shaikh

7. REFERENCES

[1] S. Chai, J. Yang, Y. Cheng, "*The Research of Improved Apriori Algorithm for Mining Association Rules*", 2007 IEEE.

[2] S. Chai, H. Wang, J. Qiu, "*DFR: A New Improved Algorithm for Mining Frequent Item sets*", Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)

[3] R. Agrawal, T. Imielinski, A. Swami, "*Mining Association Rules between Sets of Items in Very Large Databases [C]*", Proceedings of the ACM SIGMOD Conference on Management of Data, Washington, USA, 1993-05: 207-216

[4] R. Agrawal, T. Srikant, "*Fast Algorithms for Mining Association Rules in Large Database [C]*", Proceedings of 20th VLDB Conference, Santiago, Chile, 1994: 487-499

[5] L Guan, S Cheng, and R Zhou, "*Mining Frequent Patterns without Candidate Generation [C]*", Proceedings of SIGMOD'00, Dallas, 2000:1-12.

[6] Dongme Sun, Shaohua Teng, Wei Zhang, "*An algorithm to improve the effectiveness of Apriori*", Proceedings of 6th IEEE International Conference on Cognitive Informatics (ICCI'07), IEEE2007.

[7] http://en.wikipedia.org/wiki/Apriori_algorithm.