GAIT- A POTENTIAL TOOL FOR PROLIFIC MINING OF ASSOCIATION RULES

S. Sabeen
Assistant Professor,
SRM Institute of Science and Technology,
Department of Computer Science,
Chennai, India.
sabeens@rediffmail.com

Dr. Arunadevi R
Department of Computer Science
Vidhya Sagar Women's College
Chengalpettu, Tamilnadu India

Abstract

This paper introduces the concept of data mining. Basic idea of the GAIT(Graph of Associations in TD) is based on graph SGMAT (Symmetric Graph Matrix). It is a symmetric matrix of graph representing transactions in a transaction database for mining association rules. Association rules are to find relationship in the database. It can even be said, when referring to data mining, people first think of mining association rules. Many algorithms proposed recently have been motivated by FP-Growth (Frequent Pattern Growth) process and uses an FP-Tree (Frequent Pattern Tree) to mine frequent patterns. This paper introduces an algorithm called GAIT (which constructs a graph instead of tree to arrange the items for mining frequent itemsets. The benefit of using graph structure comes in the form of space complexity because graph uses an item as node exactly once rather than two or more times as was done in tree. This algorithm consists of several steps. 1. Scanning the entire transactions from the database only once. 2. Generating a new compressed graph, GAIT, that contains all the transactions in the database with its relationships. 3. Constructing a special type matrix called 2D-SGMAT, Two Dimensional Symmetric Graph Matrix, for the compressed graph, GAIT. 4. Mining all frequent itemsets with its support count. 5. Finally derive all strong association rules satisfying both minimum support threshold and minimum confidence. Here introduce new algorithms for the above five steps.

Index Terms: symmetry graph matrix, data mining, association rule, path traversal, transaction database

1. Introduction

Data mining is the process of tracing knowledge from large amount of data. Data mining involves an integration of techniques from multiple disciplines such as database and data warehouse technology, statistics, machine learning, association rule mining, high-performance computing, pattern recognition, neural networks, data visualization, information retrieval, image and signal processing and spatial or temporal data analysis. Association rule learning is a method for discovering interesting relations between datasets in large databases. Based on the concept of strong rules, introduced association rules for discovering regularities between products in large scale transaction data recorded in supermarkets. Association rule mining comprises of two steps for the complete process. First step finds the frequent itemsets from given transaction database and second step generates rule using these frequent itemsets.

Frequent pattern mining is an important part in the data mining tasks. R.Srikant proposed the classical algorithm called Apriori[5]. That used the generate-and-test approach to find all frequent patterns. According to the property of Apriori, many like-Apriori algorithms had been proposed, but all algorithms had the bottle-neck that generating many candidates. In order to solve this problem, Han jia-wei proposed the FP growth algorithm based on FP_tree that used the compressed FP_tree structure to store the frequent patterns and did not generate candidates. But constructing the FP_tree need to scan the database twice, and the time for constructing FP_tree needs much time. Jun Gao [6] proposed a new association rule mining algorithm called MFP (modified FP Growth). The MFP algorithm can convert a transaction database into an MFP_tree through scanning the database only once, and then do the mining of the tree. All algorithms based on FP Growth process uses tree for arranging the items before mining, where more than one node can contain single item. This causes repetition of same item and needs more space to store many copies of same item.

In this paper an algorithm is desinged, which uses a compressed graph for arranging items before mining. The benefit of using graph is that there is only one node for an item. But in FP_Tree or in any tree structure many nodes are created for representing the same product i.e., more memory is required in Tree based approach for frequent pattern mining.

The proposed graph based approach requires less memory. Section I of this paper gives introduction to association rule mining and related algorithms. Section II explains the principle of the GAIT. Section 3 shows the algorithm to implement the GAIT Graph, Section 4 shows the construction of 2D-SGMAT. Section 5 shows the algorithm for mining all frequent itemset with its support count from the GAIT. Section 6 shows the algorithm to derive all strong association rules and Section 7 shows the conclusion.

2. GRAPH OF ASSICIATIONS IN TRANSACTION DATABASE, GAIT

Definition 1: Transaction Data Base: Transaction database: Transaction database stores transaction records. Every transaction in a transaction database has a unique identifier and all items included in the transaction are listed in order. Table1 is a simple transaction database named TDB with two attributes namely Tid and List of Products. T00l is the identifier of the first transaction record in TDB.

Definition 2: Graph: A diagram that exhibits a relationship, often functional, between two sets of numbers as a set of points having coordinates determined by the relationship. It is a set of items connected by edges. Each item is called a vertex or node. Formally, a graph is a *set* of vertices and a binary relation between vertices, adjacency. A graph G can be defined as a pair (V,E), where V is a set of vertices, and E is a set of edges between the vertices $E \subseteq \{(u,v) \mid u, v \in V\}$. If the graph is undirected, the adjacency relation defined by the edges is symmetric, or $E \subseteq \{\{u,v\} \mid u, v \in V\}$ (sets of vertices rather than ordered pairs). If the graph does not allow self-loops, adjacency is irreflexive

Definition 3: Frequent Itemset: It is a data structure used to store Frequent Itemsets found while mining graph. The set of items occur frequently in the transactions with greater than or equal to minimum support threshold

3 THE PRINCIPLE OF THE PROPOSED ALGORITHM GAIT

This algorithm can be divided into three parts: In first part scan all transaction for given transactional database TDB. In second part all the nodes of GIAT are constructing for given TDB. The number of node is equal to the number of distinct products in the TDB. Each node is representing two values as node weight. They are frequency and name of the product[13]. Each directed edge in the GIAT contains two values marked on it. First value represents the frequency of edge and the second represents array of names of the nodes with which the concerned transaction has been started and traversed to reach the destination of this edge. All these two values are written on each edge as shown in figure2.

3.1 Scanning the database TDB

In this part scan all transactions from the given database TDB. Table 1 is a simple transaction database named TDB with 11 transactions and 5 distinct products.

3.2. CONSTRUCTION OF GAIT

This is the second part of algorithm which explains the formation of GIAT. As shown in figure 2. The various inputs to this algorithm is the number of node, n is 5 which is same as number of distinct products in the fig.1.and number of transactions, m is 11. First create all the 5 nodes, with its frequency and name field, of the GIAT. Start with the first transaction of TDB. Each product in the transactions is ordered in lexicographical order.

Table1. Transaction Database (TDB)

Tid	List of Products
T001	P1,P3
T002	P1,P2,P3
T003	P2,P4,P5
T004	P2,P3,P5
T005	P1,P4,P5
T006	P3,P4,P5
T007	P1,P2,P3,P4
T008	P4,P5
T009	P1,P4,P5
T010	P3,P4
T011	P2,P3,P4

Take first product P1 of the first transaction T001 and identify the identical node from the GAIT and increase its frequency by one[12]. Since the product P1 has no predecessor in T001, read the second item P3 and increase its occurrence frequency by one. Then draw a directed edge from its predecessor to second node. Set the edge weight for this edge as edge count is equal to 1 and vertex array as names of nodes connected the edge as |P1,P3|. Check any edge with same edge weight connecting to P3 then increase edge frequency count by one and merge it. Each time this algorithm creates a chain of edge and number of nodes increased by one. While making the graph for T001 it includes the nodes P1 and P3 and an directed edge connecting these two nodes with edge weight 1|P1,P3|. Next check the transaction T001 for any other item. There are no more items in T001.

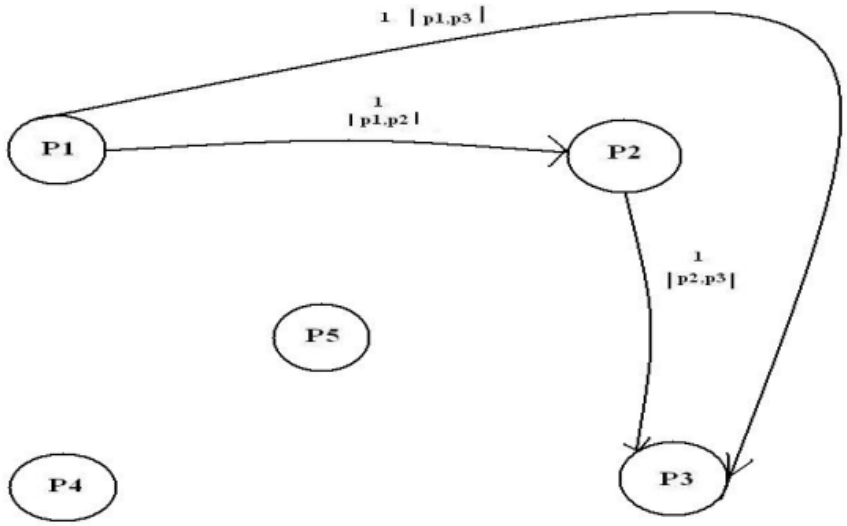

Figure 1 GAIT for T001 and T002

Now read the first item P1 from the second transaction T002 then start with its identical node P1 in the GAIT and increase its occurrence frequency by one. Since it has no predecessor in T002 take the second item P2 and increase its frequency by one then make an edge between these two nodes since this edge does not exist in all the edges generated so far. Set the edge weight as 1|P1,P2|. Read the next product P3 from T002 and identify the identical node P3 from the GAIT and increase its frequency count by one. Establish a new directed edge that connect P3 from its predecessor P2 with edge weight 1|P1,P2,P3|. Check any edge with same edge weight connecting to P3 then increase edge frequency count by one and merge it. This can be seen in the figure 1. Next scan the first item P2 of the T003 and increase its frequency count of its identical node in the GIAT. Since it is first product of T003 take the next item P4 and increase its node count by one in GIAT[10]. Draw an edge connecting P2 and P4 with edge weight 1{P2, P4}. Check any edge with same edge weight connecting to P4 then increase edge frequency count by one and merge it. Scan next item P5 of T003 and add 1 to its node frequency in the GAIT. Establish a new edge from its predecessor P3 to P5 with edge weight =[1(P2, P4, P5)]. Check any edge with same edge weight connecting to P5 then increase edge frequency count by one and merge it. This process continues till all transactions of database TDB is compressed in the GAIT. Figure 2 shows the GAIT generated for the given TDB in table1.

3.3 PROPOSED ALGORITHM GAIT

Algorithm : Construction of Graph GAIT

Input: Transaction Database TDB, N- number of distinct products in TDB, M-number of Transactions,
s- Minimum support threshold.
Output: GAIT, Graph of Associations in the TDB.

Method:
for item Create_Node(c P_i)
end for
Assign V and predecessor of each T_i as null
for each item $P_i \in T_j$
c is increased by one // c, node count
if (Predecessor($P_i \in T_j$) $\neq$ NULL) //$1 \leq j \leq N$
if (V U P_i) does not exist in all the generated edges to P_i so far
E = CreateEdge(Predecessor, P_i)
$E_{c=1}$
SetEdgeWt(E) = [E_c; (V U P_i)]
Set Predecessor = P_i;
endif
increase E_c by one
end if
Predecessor = P_i
V= V U P
end if
end if
Predecessor = P_i
V=V U P_j
end for
Return GAIT.

Figure 2 Algorithm for Constructing graph, GAIT of a transaction database

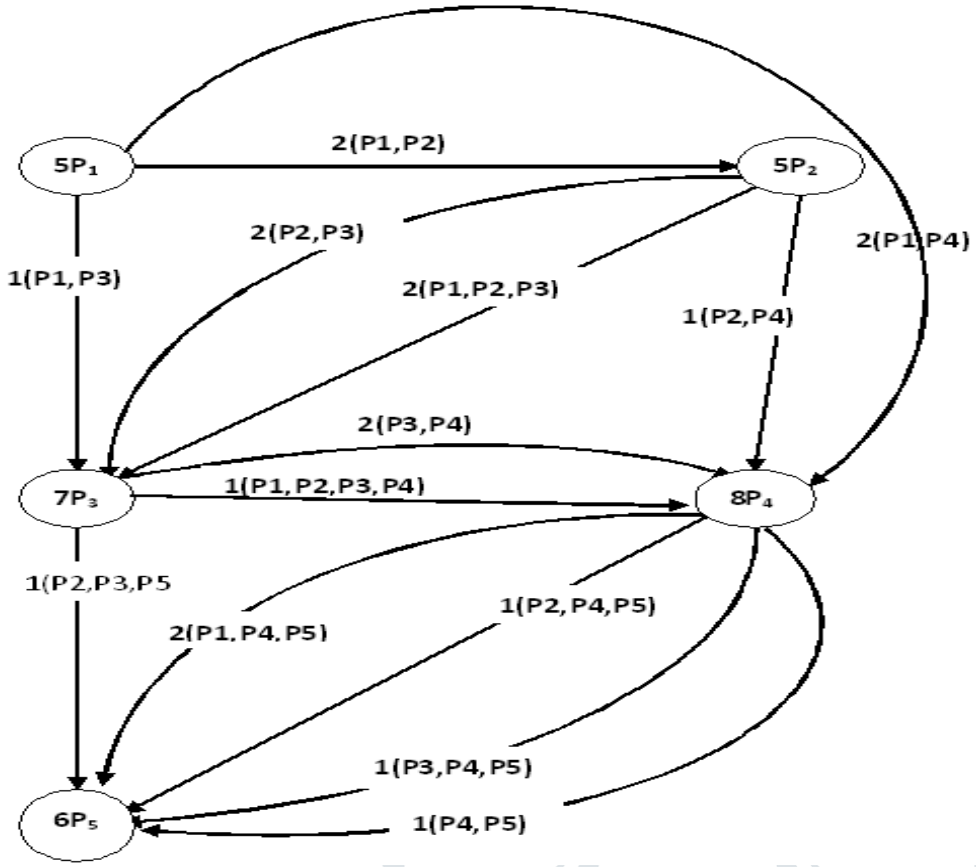

Figure 3 GAIT for the given TDB

4 PROPOSED ALGORITHM 2D-SGMAT

2D-SGM is a two dimensional symmetric graph matrix. It is a square matrix whose size is equal to the number of nodes in the GAIT. Diagonal elements of this matrix represent the frequency count of each 1-frequent item. This is very useful for finding all the 1-frequent itemset(L1) and 2-frequent itemset(L2). This algorithm is used for mining all the binary associations among the transactions in the TDB[11].

Algorithm 2D-SGMAT

Input: N – number of distinct items, s-min-support threshold,
GAIT- Graph of Associations in TDB
L_1= { }; //Assuming
L_2= { };

Output: 2D-SGMAT
L1 – set of all frequent 1- itemset.
L2 -- set of all frequent 2-itemset.

Method:

```
for ( i = 1; i < = N-1;  i++ )
   for (j =i ; j < =  N;  j++ )
      if (i = j)
      G2MAT[i][j] = Frequency(Pi);
       end if
       if (Frequency(Pi) >= s)
       Add {Pi : Frequency(Pi) } to L1
       end if
       Assign f=o
      if((Frequency(Pi)&&Frequency(Pj))>= s)
     for each edge E to Pj contains Pi & Pj
        if (Pi , Pj ε Edgewt(E)
            f = f + Frequency(E);
         end if
           if (f  >= s)
              Add { Pi , Pj :f } to L2
          end if
              G2MAT[i][j] = f;
          end for
         End if
      End for
```

```
End for
return(L1);
return(L2);
```

Figure 4 2D-SGMAT Algorithm

A sample execution of the 2D-SGMAT algorithm is given in this section. The transactions in the TDB given in the table1 are considered for this execution. Assume that the minimum support, s = 2. First this algorithm generates a symmetric matrix of size n equal to the number of nodes in the GAIT. Figure.3 it visit the first node of GAIT and its frequency is written in the 2D-SGMAT [1][1]. This algorithm defines a path traverse from first node to remaining nodes through various edges connecting the vertices. All the values of first row will be filled first by traversing the edge starting from the first node P1. Each edge to Pj it finds the total frequency of nodes Pi and Pj includes in the edges connecting from Pi to Pj. This total frequency corresponding to Pi and Pj will be written in the cell P[i][j] of 2D-SGMAT. If this frequency is greater than or equal to the s, minimum support then Pi,Pj is included in the L2 with its frequency count. All elements in the diagonal represent the frequency of each node in the GAIT. Frequency of each node Pi is greater than or equal to the s, minimum support then Pi, is included in the L1 with its frequency count. This process will be repeated for all the edges connecting two nodes.

Two properties of 2D-SGMAT are i).This is symmetric matrix, that is the values above the diagonal elements are same as that of the below diagonal elements. Hence no need to repeat the steps for the elements below the diagonal elements. ii) It shows the all the binary relationships in the TDB. 2DSGMAT generated for the given TDB is given below. It displays all the L1 and L2.

	P1	P2	P3	P4	P5
P1	5	2	3	2	2
P2		4	4	3	2
P3			7	4	2
P4				8	5
P5					6

Figure 5. 2D-GMAT of GAIT in fig.2.

L1 = { { P1:5}, {P2:4}, {P3:7}, {P4:8}, {P5:6} }

L2 = {{P_1, P_2:2}, {P_1, P_3:3}, {P_1, P_4:2}, {P_1, P_5:2}

{ {P_2,P_3:4}, {P_2,P_4:3}, {P_2,P_5:2}

{P_3, P_4:4}, {P_3, P_5:2}

{P_4 P_5:5}}

5 PROPOSED ALGORITHM FOR MINING HIGHER ORDER FREQUENT ITEMSETS, KDGMAT

This algorithm used for mining all higher order frequent itemset with its support count from the GAIT. This is a generalization of 2D-SGMAT. The logic behind this is multi-dimensional array concept. This is an efficient method for mining all the higher order frequent itemset for mining the association rules. No candidate sets are generated. Here directly mining the frequent items set from the traversal through the GAIT. The generalized algorithm for mining frequent itemset is givenin the figure.

Name of Algorithm: **Generalized KD-GMAT**

Input: N – number of distinct items, s-min-support threshold,
GAIT- Graph of Associations in TDB
Lk= { };

Output: Lk – set of all frequent k- itemset.

Method:

```
for ( i1= 1; i1 < = N-k+1;  i1++ )
   for ( i2= i 1 +1; i2 < = N-k+2;  i 2++ )
     ……..
       ……
         for ( i k= ik +1; ik< = N-k+1;  ik ++)
         Assign f=o
        if((Frequency(Pi1)&&Frequency(Pij)&&…...&&  Frequency(Pik)))>= s)
         For each Edge E to P i k
            If (Pi 1, Pi 2, Pi 3 ε Edgewt(E)
                 f = f +Frequency(E)
             end if
          end for
```

```
        If (f >= s)
           Add { Pi1, Pi2, .....Pik :f } To Lk
           KDGMAT[i1][i2 ]….[ik ]=f;
        End if
        End for
       ……..
      …….
    End for
End for
Return LK
```

Figure 6 KD-GMAT algorithm

Mining of Frequent 3 Item sets

Modified version of the above algorithm for mining L3 is given below.

Name of Algorithm: 3D-SGMAT

Input: N – number of distinct items, s-min-support threshold,
GAIT- Graph of Associations in TDB
L_3= { };

Output: L_3 – set of all frequent 3 itemset.

Method:

```
for ( i = 1; i < = N-2  i ++ )
   for ( j = i +1; j <= N-1  j++)
       for ( k = j+1;k < = N;  k++)
            Assign f=0
            if((Frequency(Pi)&&Frequency(Pj)
           && Frequency(Pk))>= s)
                for each Edge E to Pk
                  if {Pi , Pj , Pk } Є Edgwt(E))
                    f = f +Frequency(E)
                end if
              end for
                 If (f >= s)
                   Add {Pi, Pj, Pk : f } to L3
                   3DGMAT[i][j][k]=f
                end if
            end if
      end for
   end for
end for
Return L3
```

The above algorithm works as follows.

When i_1=1, i_2=2 and i_3= 3. If frequency of each P_{i1},P_{i2},P_{i3} satisfies minimum support, s, Traverse Each edge to P_3, contains P_{i1},P_{i2}, P_{i3} . Only one edge contain P1,P2,P3. The frequency of this edge weight,

F (P_1, P_2, P_3) = 2 >= s; therefore add this to L3. i.e., L_3= {{P_1, P_2, P_3 : 2}}.

When i_1=1, i_2=2 and i_3= 4.Since the frequency (P_1, P_2, P_4) satisfies s. Find how many edges to the node P_4 contain P_1, P_2, P_4. Only one edge contains these nodes as its edge weight. But its frequency of this edge is less than the minimum support threshold. 3DGMAT[P_1],[P_2],[P_4] = 1. This set is not added to the L_3. This process repeat till the graph traverses the edge connecting the nodes P_3, P_4, P_5. Frequency of this node is 1. So this set is not added to the L_3. The set of frequent 3-itemset mined from the given TDB is given below.

L_3 = {{ P_1, P_2, P_3: 2}, { P_2, P_3, P_4: 2}{ P_1, P_4, P_5: 2}}.

6 MINING ALL STRONG ASSOCIATION RULES

Once the frequent itemsets from transactions in a database D have been found, next generate strong association rules from them. Strong association rules satisfy both minimum support and minimum confidence. The main idea for defining the confidence of the rule X => Y is given below.

Confidence of the rule X => Y is equal to the frequency (X U Y) / frequency of (X).Where frequency (X U Y) is the number of transactions containing the itemsets (X U Y) and frequency(X) is the number of transactions containing the itemset X. Based on this method association rule can be generated as follows

6.1 Algorithm for Generating Association Rules from Frequent Sets

Input: L_1 – set of all frequent-1-itemset.
L_2 – set of all frequent-2-itemset.
L_h– union of all higher order frequen itemset.

c- Minimum confidence threshold.
s- Support of the rule

Output: R set of all strong association rules.

Method:
R = { };
Set of all frequent itemset, L = L_2 U L_h
Find m = | L| // number of frequent itemset.
For all frequent itemset x € L
Generate_All_Nonempty_Subsets(x);
For each nonempty susbset y € x
Generate_Rule("y => (x-y)")
Confidence of the rule, conf= f(x) / f(y)
//f(x) is the frequency of itemset x. // f(y) is the frequency of itemset y.
if (conf > = c)
R U = {y => (x-y): support=s, confidence = conf}
End if;
End for;
End for;
Return(R);// list of all Strong association rules.

Figure 7 Algorithm for Generating Association Rules from Frequent Sets

Non empty subset of x = {{ P_1 },{ P_2 },{ P_3 },{ P_1, P_2},{ P_1, P_3},{ P_2, P_3}}
The resulting association rules are shown below. Each listed with its confidence.

P_1 => P_2 ^ P_3, Confidence = 2/5 = 40%
P_2 => P_1 ^ P_3, Confidence = 2/4 = 50%
P_3 => P_1 ^ P_2, Confidence = 2/7 = 28.57%
P_1 ^ P_2 => P_3, Confidence =2/2 = 100%
P_1 ^ P_3 => P_2, Confidence = 2/3= 66.67%
P_2 ^ P_3 => P_1,, Confidence = 2/4= 50%

This process will be repeated for all the frequent itemset in L. This is a simple algorithm and can be used to mine all the strong association rules from the set of frequent itemset, L.

6.2 KNOWLEDGE MININED FROM GAIT

Theorem 1: A pattern P is not a frequent pattern. Any super pattern which contains this pattern P cannot be a frequent pattern.
Proof: If the support count of a pattern P is less than the minimum support threshold then P is not a frequent pattern. Hence support of a super pattern that contains the given pattern P must be also less than the minimum support. So the super pattern contain the pattern cannot be frequent.
Theorem 2: For a frequent pattern {i_1,i_2,……i_k}, if there is no directed edge from i_k to any other product p then the pattern (i_1,i_2,…….i_k,p) cannot be a super pattern.
Proof: We can prove this by mathematical induction. When i=1, i.e., the number of elements in the pattern is 1. If the support of the i_1 is less than the minimum support then p_1 is not frequent pattern. When i=2, i.e., number of elements in the pattern is 2. Suppose p_1 is frequent pattern. If there is no directed edge from p_1 to any other element p_2 then (p_1,p_2) is not a pattern. If an edge is exists from p_1 to any other element p_2 then it is a pattern. Frequentcy of this directed edge is greater than or equal to the minimum support s then the pattern (p_1,p_2) is frequent pattern. When i=k, i.e., number of elements in the pattern is equal to k. Suppose (p_1,p_2,…….p_k) is a frequent pattern. If there exist no directed edge from p_k to any other product p_{k+1} then the pattern cannot be extended to a super pattern with (k+1) elements. If there exists a directed edge from p_k to any other node p_{k+1} then (p_1,p_2,…….p_k,p_{k+1}) is a pattern and the frequency (p_1,p_2,…….p_k,p_{k+1}) is greater than or equal to s then this will become a super frequent pattern according to the theorem 1.

When i=k+1, number of elements =k+1. Suppose (p_1,p_2,…….p_k,p_{k+1}) is frequent pattern. If there exist a directed edge from p_{k+1} to any other item p_{k+2} then (p_1,p_2,…….p_k,p_{k+1},p_{k+2}) is a large pattern and its frequency is greater than or equal to s then it becomes the super frequent pattern by theorem 1. Hence this is true for i=1, i=2 and i=k an arbitrary value and i=k+1. Hence the theorem is true.

Corollary 2.1: Suppose (p_1,p_2,…….p_k) is a frequent pattern. If there is no directed edge from p_k to any other node p_{k+1} then (p_1,p_2,…….p_k,p_{k+1}) is not a pattern.
For example, from the above example, p_1 is a pattern. There is a directed edge from p_1 to p_2. So (p_1,p_2) is a pattern. Frequency of this pattern is 2, satisfying minimum support, hence (p_1,p_2) is a frequent pattern.

There exists another directed edge from p_2 to another node p_3. So (p_1,p_2,p_3) is a super pattern. Frequency of this pattern is satisfying minimum support. So it is a large frequent pattern.

7 PERFORMANCE EVALUATION

FAR[2] algorithm proposed by Jeb Sheela [6] transforms the TDB in Feature Matrix and stored it in bits. This algorithm used Boolean Vector 'Relational Calculas' for discovering frequent items set. It generates candidate itemsets by combining the coloumn of feature matrix. If the number of column in the feature matrix is very large, FAR algorithm will cost much time to useless calculus. GMA[2](Association Graph and Matrix Pruning Algorithm proposed by Haiwei Pan, Xiades Tan. It reduces the number of candidate set but this is not optimum. It uses 'and calculus' to generate frequent itemset. This algorithm generates several candidate sets for higher order frequent itemsets. If the minimum support threshold is changing then GMA is not suit for mining corresponding frequent itemsests satisfying new minimum support threshold. In DLG[1] proposed by Yen and Chen there will be large number of redundant computing and time taken for generating frequent itemset is more.

CONCLUSION

To get optimum solution for mining frequent itemset is to mine the frequent itemset directly with minimum candidate set and number of times scanning the data base is also minimum. In this proposed approach there is only once read the TDB and constructing a GAIT graph. It contains all the transaction with its item. This is a potential tool can be used to prolific analysis of the transaction data using the SGMAT. This method generates all frequent itemset in level wise with minimum comparison. This can be achieving the traversal through the nodes of GAIT satisfying minimum support. This algorithm suggests a simple data structure to keep all the frequent itemset and association rules. This will work even the number of items in TDB large.

REFERENCES

[1] SHANG-ping Dai, DUAN Xin 'Research on a Graph base Algorithm 'IEEE International symposium on computational Intelligence and Design.2008

[2]Haiwei Pan, Xiaolei, Qilong Han, Guisheng Yin ' En effective Algorithm based on Association Graph and Matrix for mining Association rules' IEEE, 2010.

[4] Borgelt C, "Efficient Implementations of Apriori and Eclat," Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations, CEUR Workshop Proc., voL 80, Nov. 2003.

[5] Agrawal R, and RSrikant, "Fast algorithms for mining association rules", In VLDB'94, pp. 487-499,1994.

[6] Gao Jun: "A New Algorithm of Association Rule Mining" 2008 International Conference on Computational Intelligence and Security.

[7] Goethals 8., "Survey on Frequent Pattern Mining," manuscript, 2003.[8] Grahne Gosta, and Jianfei Zhu, " Fast Algorithms for Frequent Itemset Mining Using FP-Trees " IEEE Transactions On Knowledge
And Data Engineering, VoL 17, No. 10, October 2005 . ppI3471362.

[9] Grahne G. and L Zhu, "Efficiently Using Prefix-Trees in Mining Frequent Itemsets," Proc. ICDM 2003 Workshop Frequent Itemset Mining Implementations, Dec. 2003.

[10] Sabeen. S., "Classification of Message in Dynamic Notice Board using Android", European Journal of Scientific Research, Vol. 92, No. 4, December 2012.

[11] Arumugam, S., Sabeen, S. "Association Rule Mining using Path Systems in Directed Graphs", INT J COMPUT COMMUN, ISSN 1841-9836, 8(6):791-799, December, 2013

[12] Dr. Sabeen, S ,"Frequent Pattern Mining Using Hyper Path In Hypergraphs" International Journal of Innovative Research in Computer and Communication Engineering,(IJIRCCE), Vol. 4, Issue 6, June 2016

[13] Dr. Sabeen, S ,"Hypergraph Model for Association Rule Mining" International Journal of Creative Research Thoughts (IJCRT), Vol. 6, Issue 2,Page No pp.830-838 April 2018.