Hypergraph Model for Association Rule Mining

S .Sabeen[1]
Department of Computer Applications
Siddharth Institute of Engineering and Technology

Abstract

A transaction database (TDB) consists of a set I of items and a multiset D of nonempty subsets of I, whose elements are called transactions. Association rule mining seeks to discover associations among transactions encoded in a database. It can be used to improve decision making in a wide variety of applications such as: market basket analysis, medical diagnosis, bio-medical literature, protein sequences, census data, logistic regression, and fraud detection in web, CRM of credit card business etc. There are several algorithms for solving the popular and computationally expensive task of association rule mining from a TDB. In this paper we propose a novel hypergraph based model for TDB. We deal with hypergraphs as a tool to model and solve some classes of problems arising in Association rule mining in Data Mining. Algorithms to perform visits of hypergraphs and to find frequent itemset are studied in detail. Hypergraph is a generalization of a graph wherein edges can connect more than two vertices and are called hyper edges. We give efficient algorithms for generating the hypergraph and extracting frequent patterns for association rule mining. We also propose several hypergraph theoretic parameters which lead to a better understanding of the system. Our study shows that a new approach has high performance in various kinds of data, outperforms the previously developed algorithms in different settings, and is highly scalable in mining different databases.

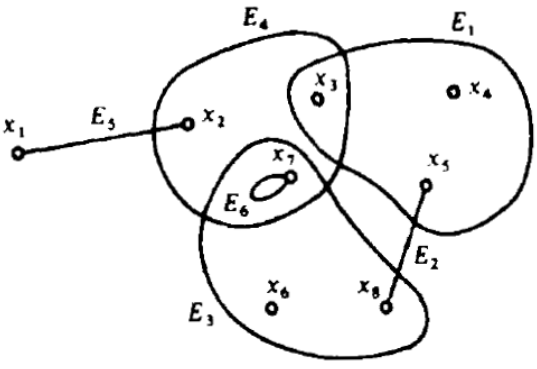
Keywords: Hyper graph, Association rule mining, frequent patterns, data mining.

1 Introduction

The task of association rule mining in a large database of transactions was proposed by Agarwal et al. [1]. Since then this problem has received a great deal of attention and association rule mining is one of the most popular pattern discovery methods in Knowledge Discovery from Database (KDD). A broad variety of efficient algorithms such as Apriori Algorithm [3], FPgrowth [4], SETM [6], DIC [5], Eclat [11] FI-tree(Frequent Itemset-tree)[13], H-mine[10] and SaM algorithm[12] have been developed during the past few years. In this paper we propose a data structure consisting of a hypergraph [8]. We give an algorithm which generates the Hypergraph D and which also simultaneously computes several other measures such as frequent items, non frequent items, total number of hyper edges, length of a largest transaction, frequency of occurrence of various nodes and the number of occurrences of each arc in D. The second algorithm generates all the patterns of the transaction database using the above data structure. This algorithm can be modified to extract frequent patterns. The third algorithm deals with association rule mining. In this process we scan the database exactly once and the hypergraph is constructed dynamically.

2 Hypergraph and Path Systems

A hypergraph[13] is a set V of vertices and a set of non-empty subsets of V, called hyper edges. Unlike graphs, hypergraphs can capture higher-order interactions in social and communication networks that go beyond a simple union of pair wise relationships. Just as graphs naturally represent many kinds of information in mathematical and computer science problems, hypergraphs also arise naturally in important practical problems, including circuit layout, numerical linear algebra, etc. A hypergraph is a natural extension of a graph obtained by removing the constraint on the cardinality of an edge: any non-empty subset of V can be an element (a hyper edge) of the edge set E. see figure 1.



$$A = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array} \begin{array}{c} \begin{array}{cccccc} E_1 & E_2 & E_3 & E_4 & E_5 & E_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Fig.1: Hypergraph and its incidence matrix

X = { $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$} and E = { $E_1, E_2, E_3, E_4, E_5, E_6$ } = {{ x_3, x_4, x_5}, {x_5, x_8},{ x_6, x_7, x_8},{ x_2, x_3, x_7 },{ x_1, x_2 },{ x_7}}

A hypergraph is also called a set system or a family of sets drawn from the universal set X. The difference between a set system and a hypergraph (which is not well defined) is in the questions being asked. Hypergraph theory tends to concern questions similar to those of graph theory, such as connectivity and colorability, while the theory of set systems tends to ask non-graph-theoretical questions

Hypergraph and Hyper path

Let V be a finite set and E a family of subsets of V. If for all $E_i \in E$, the following conditions are satisfied:

$$E_i \neq \phi, \bigcup_{E_i \in E} E_i = V$$

Then the couple D = (V,E) is called a hypergraph. Each element v ∈ V is called a vertex and each element $E_i \in E$ a hyperedge.

Definition 1:

A hyperpath between two vertices u and v is a sequence of hyperedges $\{E_1, E_2, ...E_m\}$ such that u ∈ E_1, v ∈ E_m, and $E_i \cap E_{i+1} \neq \phi$ for i = 1, ...,m− 1. A hyperpath is simple if non-adjacent hyperedges in the path are non overlapping, i.e., $E_i \cap E_j = \emptyset$, ∀j ≠ i, i ±1.

3.2 Hypergraph Model for Transaction Database

I = {1, 2, . . . , n} set of items, ordered according to natural order of *N*. $T \subset I, T \neq \phi$ is a transaction. A multiset(collection) D of transactions is a transaction database $\forall i \in I, \exists T \in D$ such that $i \in T$, that is $\bigcup_{T \in D} T = D$

Let $T \in D$ and $X \subset I$. Then T supports X if $X \subset T$, the support of X is $f(X)$ and is defined as $\frac{|\{T \in D | X \subset T\}|}{|D|}$. For a minimum threshold $s \in [0,1], X \subset I$, is a frequent pattern if $f(X) \geq$ s. An association rule is an implication of the form

$X \Rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \phi$.The support of the rule $X \Rightarrow Y$ is $f(X \cup Y) = \frac{|\{T \in D | X \cup Y \cup T\}|}{|D|}$.

The confidence of the rule $X \Rightarrow Y$ is $Conf(X \Rightarrow Y) = \frac{f(X \cup Y)}{f(X)}$.

We can build the hypergraph D = *(I,TDB)*. Each transaction corresponds to an edge, the number of distinct items is the order of *D*, and the number of m (distinct) of transactions is size of *D*. Since every item belongs to at least one transaction, D is a hypergraph in the classical sense (that is without isolated vertices).

The number of transaction to which an item *i* belongs (frequency of i) is $d_D(i)$, the degree of I in D. The maximum length of a transaction in TDB is the rank of D. It is defined as γ(D)=max(|T| such that $T \in D$). Although D has no order by itself, the order given to the items allows regarding each transaction (edge) as an ordered set.

Given D, we can recover the TDB, that is, there is no loss of information. There is a bijective correspondence between hypergraph of order n and TDB's on a set of n items. To check the support of a given set $X \subset I$, we need m|X|=|D|.|X| steps, that is for every item in X, we must check if it belongs to each edge (transaction) in D. To check the confidence of an association rule X=>Y, the maximum number of steps required is $m|X|.|y|\{T \in D | X \subset T\}|$.After checking the support of X, we only have to check of Y is obtained in the edges containing X.

For basic terminology in TDB and association rule mining we refer to the book by Han and Kamber [7]. Normally we generate association rules for frequent patterns. In this paper we propose a data structure which consists of a hypergraph D and a system of hyper edges in D for representing a transaction database. The vertex set of the hypergraph is the item set I. Any transaction T of the form {x} is represented as a loop at x.

This paper proposes an algorithm to construct the hypergraph representing a TDB. The algorithm scans the data exactly once, dynamically constructs the hypergraph D and simultaneously computes several parameters such as frequency of occurrence of each node, number of loops at each node, number of occurrence of each arc uv, total number of arcs in D, the maximum length of a transaction.

The algorithm first creates all nodes of D, one node for each item, with support count 0. Then each transaction is scanned and hyper edge in D representing the transaction is constructed. If (i_1, i_2,….,i_k) is a transaction, the arc (i_j , i_{j+1}) is represented as a linked list. The header of this list has two fields. One field is used to store the list of vertices (i_1, i_2,….,i_{j+1}) which is called the label of the arc (i, j) and the other field is used to store the frequency of occurrence of the arc (i, j). Dynamic memory allocation method is used for storing these values. The pseudo code for the construction of hypergraph is given in Figure 2.

Algorithm: Construction of Hypergraph, D for TDB

Input :
TDB, Transaction Database
I, set of items in TDB

Output:
D, Hypergraph of given TDB
n, Order of D
$\gamma(D)$, Rank of D
$\delta(D)$, Antirank of D
$f(E_i)$, Frequency of each E_j
starD(*i*), Partial hypergraph formed by the edges containing *i*, $i \in I$
V(E_i), Set of vertices in the edge E_i
$d_D(i)$, support count or degree of vertex , $i \in I$

$\underset{\max}{\Delta}(D)$, Maximum degree of D

$\underset{\min}{\nabla}(D)$, Minimum degree of D

Method:

$E = \phi, m = 0;$

for each $i \in I$,

CreateNode(i);

end for;

for each $T_j \in TDB$

if($T_j = E_i \in E$) // *E is the set of edges in D created so far.*

++ $f(E_i)$, // *increment the edge count of* E_i *by 1.*

`else

CreateEdge(E_j)

$f(E_j) = 1;$

$V(E_j)$ = { T_j};

$E = \mathrm{Y}\{E_j : f(E_j)\}$

m++; // *number of distinct edges*

end if

end for

n = | I |;

$\gamma(D) = \underset{j}{Max}|E_j|$;

$\delta(D) = \underset{j}{Min}|E_j|;$

if ($\gamma(D) = \delta(D)$,

return the given *TDB* is uniform;

end if

for each $i \in I$,

starD(i) = $\{E_j | (E_j \in E) \,\&\, \&(i \in E_j)\}$

end for

for each $i \in I$,

$d_D(i) = \sum_{\forall E_j \in starD(i)} f(E_j)$

end for

$\underset{\max}{\Delta}(D) = \underset{i \in I}{Max}(d_D(i))$

$\underset{\min}{\nabla}(D) = \underset{i \in I}{Min}(d_D(i))$

Return D, Hypergraph of TDB

Fig.2: A greedy algorithm for constructing the hypergraph

We illustrate the algorithm in fig.2 with a dataset of diseases where a person is suffering from cold, fever and other related symptoms. The real time data set of seasonal fever is collected from the local doctors of Ramachandra medical college,Chennai which consists of six attributes as {cold, headache, fever, bodypain, allergy, cough}.

PATIENT ID	SYMPTOMS
T001	Cold, Fever and Allergy
T002	Cold, Headache and Cough
T003	Cold, Headache, Body pain and Fever
T004	Fever, Body pain and Cough
T005	Cold, Fever, Headache and Cough
T006	Cold, Body pain and Cough
T007	Cold, Allergy and Cough
T008	Cold ,Cough and Body pain
T009	Cold and Cough
T010	Cold, Headache and Fever

Table 1. Disease dataset

Different patients may have the different combination of symptoms. We applied our algorithm to find the association among the attributes with discritised dataset in the table 2 of the above table1.

PATIENT ID	ITEMS
T001	3, 5, 1
T002	3,6,4
T003	3,6,2,5
T004	5,2,4
T005	3,5,6,4
T006	3,2,4
T007	3,1,4
T008	3,4,2
T009	3,4
T010	3,6,5

SYMPTOMS	DESCRETISED VALUE
Allergy	1
Body pain	2
Cold	3
Cough	4
Fever	5
Head ache	6

Table 2: Discritised data set from the table 1

Hypergraph constructed from the descritised data set in table 2 is given in the figure3.

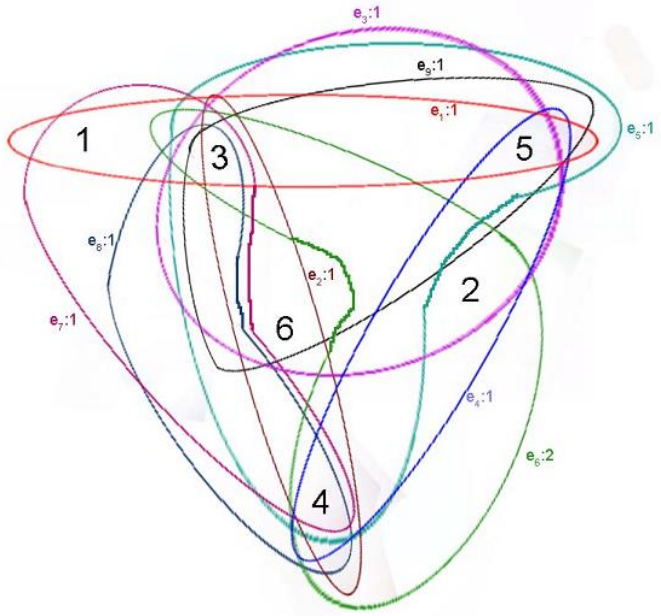


Figure.3 hypergraph for the data set in table 2

4 Algorithm for extracting frequent patterns from hypergraph D of a TDB

In this section we present an algorithm for extracting the set **L** of all frequent patterns from weighted hypergraph constructed in Section 3.2. This algorithm used to traverse all hyper edges and extracts all the frequent patterns. The pseudo code for extracting frequent patterns is given the figure 4.

Algorithm for extracting frequent patterns from hypergraph D

Input:

D, hypergraph of TDB

s, minimum support threshold

Output:

L, set of all frequent patterns

Method:

For each $E_j \in E$ in hypergraph *D*

S=Generate non empty subset of $V(E_i)$ // $V(E_i)$ is the set of vertices in E_i

for each $S_j \in S$

$f(S_j) = f(E_i)$

$if(S_j \notin L)$

$if(S_j \notin C)$

$if(f(S_j) \geq s)$

$L = L \cup \{S_j : f(S_j)\}$

else

$C = C \cup \{S_j : f(S_j)\}$

end if

else

Add $f(S_j)$ to the count of identical set in C

$if(f(S_j) \geq s)$

$L = L \cup \{S_j : f(S_j)\}$

$C = C - \{S_j : f(S_j)\}$

end if

end if

Add $f(S_j)$ to the count of identical set in L

end if

end for

end for

Return L

Fig.4: Greedy algorithm for extracting frequent patterns from hypergraph.

The set of all frequent patterns generated from the above hypergraph is given in the table 4. Here we assume minimum support s = 20%.

SL. NO.	FREQUENT PATTERNS	FREQ-UENCY	SL. NO.	FREQUENT PATTERNS	FREQ-UENCY
1	{3}	9	12	{6,4}	2
2	{5}	4	13	{3,6,4}	2
3	{1}	2	14	{2}	3
4	{3,5}	4	15	{3,2}	4
5	{3,1}	2	16	{6,5}	3
6	{5,1}	2	17	{2,5}	2
7	{3,5,1}	3	18	{3,6,5}	3
8	{6}	4	19	{5,4}	2
9	{4}	6	20	{2,4}	2
10	{3,6}	4	21	{3,2,4}	2
11	{3,4}	6			

Table4: Frequent patterns generated from the hypergraph with s= 20%

Strong association rules can be generated from the set of frequent patterns mined from the given TDB. An association rule which satisfies both minimum support threshold and minimum confidence threshold is called a strong association rule. For each frequent pattern X and for each nonempty proper subset Y of X the algorithm computes the support and confidence of the association rule $Y \Rightarrow X - Y$.

Example 4.1. From Table 4 we have X = {3, 6, 5} is a frequent pattern with frequency 3. The set of all association rules generated from this pattern with the confidence and support for each rule is given in Table 5. We have taken the minimum confidence threshold c and the minimum support threshold s as 75% and 20% respectively.

Sl.No.	Association Rules	Confidence of the Rule	Support of the Rule	R / R'
1	{3} ➔ {6,5}	33 %	33 %	R'
2	{6} ➔ {3,5}	75 %	33 %	R
3	{5} ➔ {3,6}	75 %	33 %	R
4	{3,6} ➔ {5}	75 %	33 %	R
5	{3,5} ➔ {6}	75 %	33 %	R
6	{5,6} ➔ {3}	100	33 %	R

Table 5. Association Rules mined from the frequent pattern {3,6,5}.

For the disease dataset given in Table 1 we have generated 21 frequent patterns. The number of frequent patterns generated with various support counts is given in Table 4. Various association rule generated from the frequent items set {3,6,5}.is given in table 5. One of the association rule is **{5,6} ➔ {3}** [cofidence=100, support=30 %], the infromation that the patient who is suffering from the disease Fever and Head ache also tend to have the disease Cold. A support of 33 % for association rule means that 33% of all the patients under analysis suffering from the diseases Fever, Head ache and Cold together. A confidence of 100% means that 100% of patients suffering from Fever and Head ache also suffering from Cold. The number of association rule generated from the above disease data set is 51.

6 Completeness and Compactness of Hypergraph

Several important properties of Hypergraph can be observed from the Hypergraph construction process.

Order of Hypergraph, D of the TDB

Order of the D, denoted by n(D), is the number of vertices and the number of edges will be denoted by the m(*D*). For the hypergraph D, in figure 1, n(D) = 8, m(D) = 6

Rank of D

The rank of a hypergraph for a TDB defined as

$\gamma(D) = \underset{j}{Max}|E_j|$

The anti-rank is defined as $\delta(D) = \underset{j}{Min}|E_j|$.

For hypergraph, D generated from TDB, $\gamma(D)$ indicates the number of the items in the largest transaction, having maximum number of items present. s(D) indicates the minimum number of items present in the transaction. All the transactions having a same number of items then the hypergraph D generated from such TDB will be a uniform. Uniform hypergraph of rank γ will also be called as γ-uniform and no repeated transactions will be there in the TDB.

For D, given in figure 1, $\gamma(D) = \underset{j}{Max}|E_j| = 3$ and $\delta(D) = \underset{j}{Min}|E_j| = 1$ this means that largest transactions in TDB consists maximum 3 items and smallest transaction consists of only one item. The D is not a Uniform hypergraph since $\gamma(D) \neq \delta(D)$

Knowledge Extracted during the construction of hypergraph D for the given TDB

1. O (D), order of hypergraph is defined as the no of vertices in the D.
   O (D) = | I | = number of distinct items.
2. γ (D) , rank of D, $\gamma(D) = \underset{j}{Max}|E_j|$

   The transaction having highest number of items will be the rank of TDB for D, the edge having the maximum number of vertices called rank of D, γ (D)

   Anti rank of D, $\delta(D) = \underset{j}{Min}|E_j|$

   The edge Ej having the minimum number of vertices in D
3. **Uniform TDB**

   For D, Hypergraph of TDB, if(γ (D) = δ (D)) and for all $E_i \in E$, $f(E_i)=1$ then D is uniform hypergraph. That is, all the edges in D, hypergraph of TDB have the same rank with no repeated edges, D is a uniform hypergraph. Its TDB is called uniform transaction database. $\forall$ $T_j \in$ TDB, having unique number of items in it that is a uniform TDB.
4. StarD(i) with centre *i*, $i \in I$, is the partial hypergraph formed by the edges containing the vertex *i*.

   $\underset{starD(i),\forall i \in I}{Y}$ gives the dual of the hypergraph.
5. The degree $d_D^{(i)}$ of i to be the number of edes of starD(i): No of edges containing vertex *i* is the $d_D^{(i)} = m(D(i))$
   The number of transactions containing the item 'i' is the degree or support count or frequency of the item i.
6. The degree of the hypergraph D, is defined by,

   $$\Delta(D) = \max_{i \in I} d_D^{(i)}$$

   This gives the information regarding support countt of the most frequently occur item or pattern in the TDB.
7. Regular hypergraph/TDB
   All the vertices of the hypergraph having the same degree then D is a regular hypergraph.

   A TDB, is regular one $\forall i, j \in I$ & i≠j such that $d_D^{(i)} = d_D^{(j)}$, where i≠j. That is the occurrence frequency of all the items in TDB is unique in a regular TDB.

Theorem

1. A hypergraph is regular ifand only if its D is regular.
2. For a D of a TDB of order n=|I|, the degree $d_D^{(i)} = d_i$, in decreasing order form n tuple $d_1 \geq d_2 \geq \ldots \geq d_n$ whose properties can be characterized as below.

Proposition 1

Consider $d_1 \geq d_2 \geq \ldots \geq d_n$ is the degree sequence of uniform hypergraph of ranks γ and order n possibly with repeated edeges if and only if $\sum_{i=n}^{n} d_i$ is a multiple of γ(D) and $d_n \geq 1$

Partial Hypergraph

For a set J $\subset$ {1,2,3,4…….,m}, the partial hypegraph D', generated by the set J can be defined as $D^{!} = (E_j / j \in J)$.

The set of vertices of D' is a non-empty subset of the set of all items, I in TDB. For a set A $\subset I$, we call the family of transaction formed with set A,

$D_A = \{E_j \text{ I } A | 1 \leq j \leq m, E_j \text{ I } A \neq \phi\}$ the sub graph induced by the set A. It is similar to the definition of partial sub hypergraph. The subgraph induced by the set A can be defined as the set of all transactions formed using the induced set $A \subset I$.

Example: for a set A = {$x_2, x_3, x_4, x_5, x_7, x_8$} of the subset of the X ,the set of vertices of the hypergraph given in figure 1.

$D_A = \{E_j \cap A | 1 \leq j \leq m, E_j \text{ I } A \neq \phi\}$, the set edges in the sub hypergraph induced by a set A contains the following edges, E = { E1,E2,E4,E6} is given in figure 5

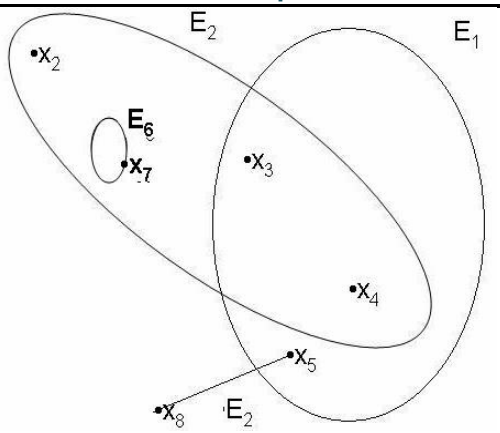


Figure 5: D_A , sub hypergraph induced by set A.

Frequent subhypergraph of D induced by a frequent pattern L

Frequent subhypergraph of D induced by a frequent pattern L can be defined as follows:

For a frequent pattern $l \in L$, we call the family

$D_l = \{E_j \text{ I } l \mid 1 \le j \le m, E_j \text{ I } l \ne \phi\}$, the frequent sub hypergraph induced by the frequent pattern $l \in L$

A frequent sub hypergraph induced by the frequent pattern $l \in L$ is similar to the definition of frequent partial hypergraph. Frequent partial hypergraph can be defined as for a set $j \subset \{1,2,3,...,m\}$ the frequent partial hypergraph *D'* generated by the set j is $D' = \{E_j \mid (j \in J) \,\&\, \&(V(E_j) \in L)\}$

The set of vertices of *D'* is a non empty subset of I

Extraction frequent sub hypergraph from the D of a TDB

Algorithm: Extraction frequent sub hypergraph from D

Input:

D, the hypergraph of D
L, set of all frequent itemset
S, minimum support
m, number of distinct edges is D

Output:

D_L, set of all frequent sub hypergraph induced by L

Method:

$D_L = \phi$
For each $l_i \in L$
$D_l = \{E_j \text{ I } l_i \mid 1 \le j \le m, E_j \text{ I } l_i \ne \phi\}$
$if (D_i \notin D_L)$
$D_L = \text{Y} D_l$
end if
end for
return D_L

Fig:6 algorithm extracting set of frequent sub hypergraphs, D_L from TDB

Lemma Given a transaction database TDB and a support threshold min sup, the support of every frequent itemset can be derived from TDB's hypergraph, D.

Proof. Based on the Hypergraph construction process, for each transaction in the TDB, its frequent item projection is mapped to a path through the edges in D.

Given a frequent itemset X = x1, x2,…, xn in which items are sorted in the support descending order. We can visit all the edges with label e: n in the Hypergraph.

For each edge e, vertex v with label e : n, the support count supv in vertex v is the number of transactions represented by e. If x1, . . . , xn all appear in e, then the supv transactions represented by e contain X. Thus, we accumulate such support counts. The sum is the support of X.

Based on this lemma, after a hypergraph for TDB is constructed, it contains the complete information for mining frequent patterns from the transaction database. Thereafter, only the D, hypergraph is needed in the remaining of the mining process, regardless of the number and length of the frequent patterns. The size of Hypergraph is bounded by the size of its corresponding TDB because each transaction will contribute at most one edge to the Hypergraph, with the length equal to the number of frequent items in that transaction. Since transactions often share frequent items, the size of the hypergraph is usually much smaller than its original database. A hypergraph never breaks a transaction into pieces. Thus, unlike the Apriori-like method which may generate an exponential number of candidates in the worst case, under no circumstances, may a Hypergraph with an

exponential number of edges be generated. The Hypergraph is a highly compact structure which stores the information for frequent pattern mining.

Results and Discussion

Dataset: For the experiment we have used datasets of different applications. These datasets was obtained from the UCI repository of machine learning databases[15] (http:\\www.ics.uci.edu/mlearn/MLRepository.html-1998). The characteristics of the datasets selected for the experiment is given in table 6.

Files	**No. of Records**	**No. of Columns**
adult.D14.N48842.C2.num	48842	14
Hepatitis.D19.N155.C2.num	155	19
heart.D75.N303.C5.num	303	75
Census	48842	14
letRecog.D106.N20000.C26.num	20000	17
MushroomD.90.N81424.C2.num	8124	23

Table 6 Data sets used in the analysis

To study the strategies we have conducted several experiments on a variety of data sizes comparing our approach with the well-known SaM algorithms, and FI- tree algorithm written by its original authors. The performance metrics in the experiments is the total execution time taken and the support count for adult, hepatitis and heart datasets. For this comparison also same dataset were selected as for the above experiment with 30% to 70% of minimum support threshold. The experiments were conducted on 2.6 GHz CPU machine with 3 Gbytes of memory using Windows XP operating system. Time needed to mine frequent itemset for different algorithms using the data set given in the table6 is discussed below.

Support in %	**Time in seconds**		
	FI-Tree	**SaM**	**HG model**
30	8.12	9.85	4.03
40	5.69	6.72	2.08
50	3.56	4.51	1.5
60	1.99	2.69	1.1
70	1.01	1.7	0.8

Table 6 Time Scalability with respect to support on the Adult dataset

Time taken to mine frequent pattern with various support threshold on Adult data set is given in the table 6. The total execution time for our HG model is very much less than that of FI-Tree and SaM methods. The SaM algorithm and FI-Tree algorithms take more time see figure 6 as that compared to our approach.

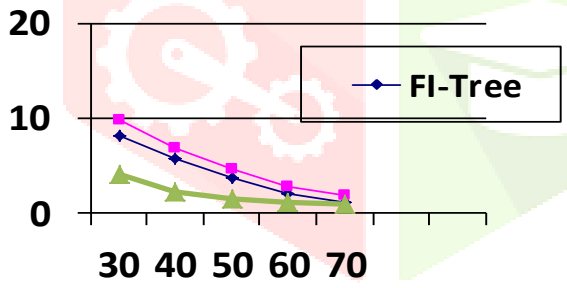


Figure 6: Time Scalability with respect to support on the Adult dataset

The total execution time for our new approach HG model and the other algorithms FI-Tree and SaM on Heart data set given in the table7 algorithms large reduces with the increase in support threshold from 30% to 70%. Our proposed approach takes less time as that compared to the other two algorithms Sam and FI-Tree,. The execution time of HG Model approach with SaM algorithms for hepatitis data set is given in table 8.

Support in %	**Time in seconds**		
	FI-Tree	**SaM**	**HG model**
30	0.05	0.07	0.035
40	0.4	0.06	0.031
50	0.3	0.05	0.028
60	0.03	0.03	0.02
70	0.01	0.02	0.009

Table 7: Heart data set execution time

Support in %	Time in seconds		
	FI-Tree	SaM	HG model
30	0.64	0.91	0.538
40	0.09	0.28	0.08
50	0.04	0.06	0.03
60	0.03	0.04	0.028
70	0.00	0.0	0.0

Table 8: Hepatitis Data set

We have also conducted a detail analysis to assess the performance of the well known algorithm FP-Growth with respect to the other frequent itemset mining algorithms. The performance matrix in the experiments is the total execution time taken and the number of item sets generated for different data sets.

The following performance analysis graphs show the execution time for the algorithms FP-Growth, Eclat, Relim, SaM with our new approach HG Model.

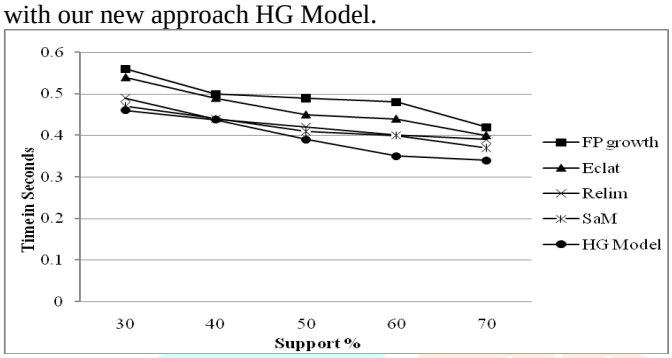


Figure 7: comparison of Execution time of the algorithms on Adult data set

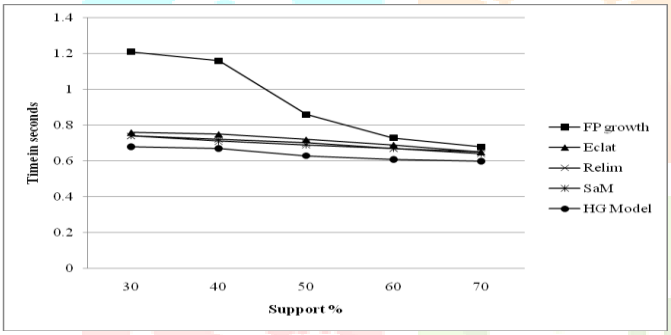


Figure8: Performance analysis on Census data set

7 Conclusion

In this paper we have proposed a new data structure consisting of a weighted hypergraph D and a path system in D for representing a TDB. We have presented algorithms for constructing D, for generating frequent patterns using D and for generating frequent subhpergraphs. During the entire process the data is scanned exactly once. We have conducted several types of experiments to test the effect of changing the support, transaction size, dimension, transaction length and use of other hypergraph theoretic parameters to extract new knowledge about the TDB and the comparison of the performance of this algorithm with other existing algorithms in the literature using real data set also studied and analyzed.

References

[1]R. Agrawal, T. Imielinski and A. Swami, Mining association rules between sets of items in large database, In Proc. of the ACM SIGMOD International Conference on Management of Data (ACM SIGMOD 93), Washington,USA, May 1993, 207-216.
[2]G. Chartrand and L. Lesniak, Graphs and Digraphs, Chapman and Hall, CRC, 4th edition, 2005.
[3] R. Agrawal and R. Srikant, Fast Algorithms for mining association rules, In Proc. of the 20th International Conference on Very Large Database (VLDB' 94), Santiago, Chile, June 1994.
[4] J. Han, J. Pei and Y. Yin, Mining Frequent Patterns without Candidate Generation, In Proc. of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, May 2000.
[5] S. Brin, R. Motwani, J.D. Ullman and S. Tsur, Dynamic itemset counting and implications rules for masket basket data, In Proc. of the ACM SIGMOD International Conference on Management Data, 1997.
[6] M. Hontsma and A. Swami, Set oriented mining for association rules in Centre, San Jose, California, October 1993.
[7] J. Han and M. Kamber, Data minining, Concepts and Applications,
[8] C. Berge. Graphs and Hypergraphs. North-Holland, 1973.
[9]M. Ozdal and C. Aykanat. Hypergraph models and algorithms for data- pattern- based clustering. Data Mining and Knowledge Discovery, 9(1):29{57, 2004.
[10]Pei J., Han J., Lu H., Nishio S., Tang S. and Yang D., Hmine: Hyper-structure mining of frequent patterns in large databases, In Proc. Int'l Conf. Data Mining **(2001)**
[11]Borgelt C., Efficient Implementations of Apriori and Eclat, In Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations **(2003)**
[12]Borgelt C., SaM: Simple Algorithms for Frequent Item Set Mining, IFSA/EUSFLAT 2009 conference **(2009)**
[13]Patel Tushar S. and Amin Kiran R. A New Approach to Mine Frequent Itemsets ISCA Journal of Engineering Sciences Vol. **1(1),** 14-18, July **(2012)**
[14] C.Berge, North Holland Mathematical Library-Hypergraphs, Volume 45, 1989.
[15]Blake C.L. and Merz C.J., UCI Repository of Machine Learning Databases, Dept. of Information and Computer Science, University of California at Irvine, CA, USA **(1998)**