2010

Geometrical Structure and Analysis of Association Rules

Sar Rajat
San Jose State University

Geometrical Structure and Analysis of Association Rules

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Rajat Sar

Spring 2010

ABSTRACT

Association rule mining helps us to identify the association between items from a large transactional data set. It has always been a time consuming process because of repeatedly scanning of the data set. Apriori Algorithm [1] and FP-Tree Algorithm [2] are the two methods to find out the association of items in a large transactional item set. Both the above algorithm works differently (Apriori follows Bottom-Up Approach & FP-Tree follows Top-Down Approach) in order to get the association. Associations of items generated from the above two algorithms can be represented in geometry. The geometrical form of associations is called Simplical Complex. By exploring the FP-Tree method and using the bit pattern of records we can quickly indentify the possible longest associations on transactional data. The proposed Bitmap method in using the FP-Tree method is a new approach which helps quickly by using Human-Aide to find out the longest association. This method quickly finds out the longest association of items by looking the bit pattern of items in a large item set. By aligning the similar bits of records and arranging the attributes in order of highest frequency first are the underline logics of the above algorithm to get the longest association of items.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. T. Y. Lin, whose guidance, support, and instructions are priceless. Dr. Lin is an educator in the truest sense of the word. I very much appreciate the invaluable support I got from the San Jose State library for finding the valuable resources relevant to my project. I also thankful to the machine learning repository of University of California, Irvine for their invaluable data set to test the data. I would also like to thank my defense committee members Prof. Soon Tee Teoh and Mr. Eric Louie for their kind support and useful comments on the project.

It has been a challenging, yet rewarding journey which I could not have completed alone and I am grateful for your support.

Thank you.

Table of Contents

List of Figures

List of Tables

INTRODUCTION

Data mining is a technique to process different types of data such as structured, unstructured and find out hidden information from it. Data mining can be done in many different ways such as Classification, Clustering, and Association Rule Mining. Classification helps us to find out useful information from a structured data set. Clustering derives meaningful information from unstructured data. Association rule mining defines the association of items from a large set of transactional data.

The transactional data set mainly contains transaction number with the name of the items sold in that transaction. By using this sales' pattern, we can derive the association between items. This derived association of items helps to put the right items close to each other to maximize the sales. Due to the large size of the transactional data set, it always takes time to get the association of items by using the algorithms such as Apriori and FP-Tree Algorithms. Sometimes the storage of this large number of transactions is a problem in memory.

We can visualize the association of items in a geometrical structure called Simplical Complex. The Bitmap approach is an easy way to derive the association between items. It can be processed on small parts and then stored the result in memory to compare with new records. It follows the FP-Tree approach to get the records in order then run a logical operation on it to produce the final association of items.

DATA MINING

Data Mining came on top of data base processing to find out the hidden knowledge from data which is not possible in data base processing. Database has certain limitations to store and process information.

Database can only store very much required and limited information into it. In data mining the size of the data set has no limit and it can be extended to terabytes. Suppose you have a web system and there are millions of users using the system, in a query we want to see the user information who logged into the system since last one month. Inside database it is hard to maintain and store information about each individual user's activities to the system. If we store this kind of user's information in a database it will increase the size of the table very rapidly and fetching information from it will be very slow as it will dig into the whole table of structure data. If we store information of the above into the database, it will make the database architecture complicated.

DATA MINING Vs. DATABASE

Database supports well defined queries while data mining queries are poorly defined [10]. SQL is running on the database to get a result from it. In data mining there is no precise or exact query language to find out information from it. Database technique is useful on operational data and data mining is much more effective on non-operational data. Database output is very much precise to the query request to it and data mining output is quite fuzzy.

DATA MINING METHODS

Data mining methods are broadly divided into three basic categories:

- Classification
- Clustering
- Association Rule Mining

CLASSIFICATION

Classification is one of the data mining techniques that apply on structured data to find out meaningful information from it. Classification is called supervised learning method and it helps us to recognize the pattern and makes a prediction about the class of a new entity to the system.

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Table 1: List of Objects for Classification
Source [3]

The table-1 describes the characteristics of animals which are falling into two different classes such as mammals and non-mammals. Now based on the table-1 we can predict the class of the following given characteristics of an entity.

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Table 2: Entity Attributes to test the Class

A – Attributes

M – Mammals

N – Non-Mammals

By using Naïve Bayes classification algorithm the probability of the given entity would be:

Probability of [A/M] = P (A/M) = (6/7) × (6/7) × (2/7) × (2/7) = .06

P (A/N) = (1/13) × (10/13) × (3/13) × (4/13) = .0042

P (A/M) P (M) = (0.06) × (7/20) = 0.021

P (A/N) P (N) = (0.0042) × (13/20) = .0027

The class of the new entity would be **mammals.**

Classification derives the class of new entities by using previous predefined classes. Algorithms such as C4.5, CART (Classification and Regression Tree), Rule based classifiers, Naïve Bayes classifiers, K-NN (k-nearest neighborhoods), and Artificial neural networks are coming under classification.

CLUSTERING

Clustering is a data mining technique to find out groups of similar characteristics within it from large amount of data. On different way it is also a technique to find out densely connected sub-graph form a graph. Clustering is called unsupervised learning. Clustering uses segmentation and partitioning approaches to find out similar groups. In this method we need to define the size

of the cluster to stop the algorithm running on a data set. K-Means, Density based clustering, and Hierarchical clustering are the basic clustering algorithms.

ASSOCIATION RULE

Association rule in data mining is the method to find out the close relationships between items. Dr. Rakesh Agrawal first introduced Apriori algorithm for finding out association between items and later Dr. Jiawei Han introduced FP-Tree method to find out the same association from items. Both the methods derive the longest association of items from large transactional data but the approach towards the derivation of longest association is different. These methods are described in detail in the coming sections. There are some key definitions about the Association Rule [10].

- ***Set if items:*** $I=\{I_1,I_2,\ldots,I_m\}$
- ***Transactions:*** $T=\{t_1,t_2, \ldots, t_n\}$, $t_j \subseteq I$
- ***Itemset:*** $\{I_{i1},I_{i2}, \ldots, I_{ik}\} \subseteq I$
- ***Support of an itemset:*** Percentage of transactions which contain that itemset.
- ***Large (Frequent) itemset:*** Itemset whose number of occurrences is above a threshold.

Itemset contains different combinations of items which generally satisfy the threshold value decided by the support of an itemset.

PROBLEM ADDRESSED

In order to process a large item set by using the above defined Apriori or Association Rule mining algorithm takes a lot of time for computation to produce the associations between items. On the other hand FP-Tree algorithm approach also needs lot of memory to get the

association. In the proposed bitmap approach the storage of items requires only one bit in memory to store into it.

Human can easily compute the logical operations on bits to calculate the association pattern of records in transaction but it is not possible to compute the association by using human aide in either Association Rule or FP-Tree algorithm. Association Rule mining starts from combination of small number of items and it converges later to produce the longest association. This approach scans the data set repeatedly for compute the association of item sets. The FP-Tree approach calculates the frequency of each item and the order of the items based on their frequency and construct a tree as the items appear in it. This approach is also very much computationally expensive.

RELATED STUDY

There are many different approaches have given to this field such as Apriori Algorithm, FP-Tree Algorithm, etc. Association Rule mining finds out the longest association by calculating the support and confidence of items from transactions. In FP-Tree algorithm each unique record can be represented as a branch of the tree with the highest frequency item could be the root of the tree. In order to visualize items inside a transaction in geometry, we need **Simplical Complex** structure to represent it. In the following sections the above mentioned algorithms (Apriori and FP-Tree) and the geometry of Association Rule are defined elaborately.

GEOMETRY OF ASSOCIATION RULE

The geometry of Association Rule can be defined in the form of ***Simplical Complex.*** In order get the detail of Simplical Complex, we need to know the high level definition of Simplex as well as Complex.

Simplex is a geometrical structure to represent n-dimensional geometrical shapes or images. We can use the simplex and Simplical complex to represent the association of items. An n-simplex contains n+1 no. of vertices with all the vertices connected with each other to represent geometrical shapes of that simplex. Simplex certifies the convex hull properties, i.e. all the lower dimensional simplex are within the highest dimensional simplex. N-Simplex is made by connecting one vertex to the (n-1) simplex and all existing vertex has connection to the new vertex. The convex hull of non-empty subset of the n+1 points that defines the simplex (n) is called faces of the simplex [4]. M-faces of an n-simplex can be represented by using Binomial Co-efficient (n+1, m+1).

The geometrical structure of simplex can be represented mathematically in two ways such as by using Cartesian coordinates and by geometric properties.

A Simplical complex is a combination of simplexes with the following properties:

1. Any face of a simplex belongs to a complex is in that complex [5].
2. The intersection of any two simplexes belongs to the Simplical complex structure is a face of that complex.

Important terms related to a Simplical Complex:

Closure: The closure is the smallest Simplical sub-complex of a large complex.

Star: The star is all the faces link to a point in a complex.

Link: The link is the boundary drawn by all the links from a point in a complex.

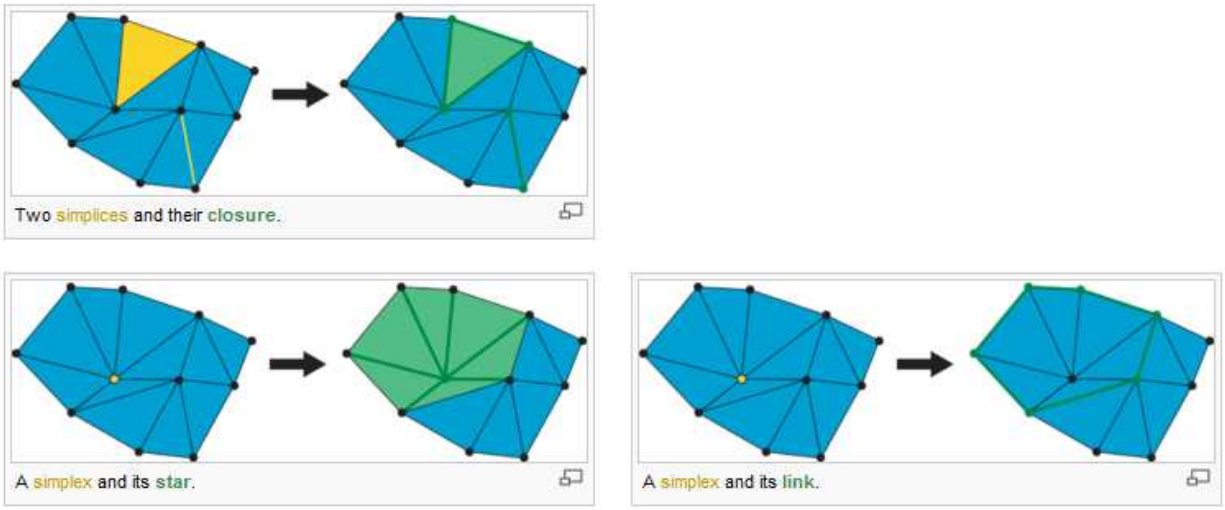


Figure 1: Closure, Star, and Link in a Simplical Complex
Source:[5]

This above geometrical structure gives an idea to represent the items' in association.

SIMPLICAL COMPLEX

A Simplical Complex K consists of a set {v} of vertices and a set {s} of finite nonempty subsets of {v} called simplexes such that: [9]

a) Any set consisting of exactly one vertex is a simplex.

b) Any nonempty subset of a simplex is a simplex.

Some interesting properties about the Simplical Complex: [9]

- The empty set of simplexes is a Simplical Complex denoted by □.
- For any set A the set of all finite nonempty subsets of A is a Simplical Complex.
- If '**S**' is a simplex of Simplical Complex **'K'**, the set of all faces of '**S**' is a Simplical Complex denoted by Ŝ.
- If '**S**' is a simplex of a Simplical Complex **K**, the set of all proper faces of **S** is a Simplical Complex denoted by Ŝ.

- If **K** is a Simplical Complex, its q-dimensional skeleton K^q is defined to be the Simplical complex consisting of all p-simplexes of K for $p \leq q$.
- If K_1 and K_2 are Simplical complexes, their join $K_1 * K_2$ is the Simplical complex defined by

  $$K_1 * K_2 = K_1 \square K_2 \cup \{s_1 \square s_2 \mid s_1 \varepsilon k_1, s_2 \varepsilon k_2\}$$

  Thus the set of vertices of $K_1 * K_2$ is the set sum of the set of vertices of K_1 and the set of vertices of K_2.
- There is Simplical Complex whose set of vertices is **Z** and whose set of simplexes is:

  $\{\{n\} \mid n \varepsilon Z\} \cup \{\{n, n+1\} \mid n \varepsilon Z\}$.

The above features about a Simplical Complex are defined by Edwin H. Spanier in his book Algebraic Topology [9].

APRIORI ALGORITHM

This algorithm produces the *bottom-up* approach to derive the longest association between items. Based on the two different measures (**Support & Confidence)**, it decides whether to take that item for finding the longest association in it. In a relationship between **X→ Y,** Below are the formulas to find out the association:

Support (S) = $(X \cup Y) / | T |$

| T | - Total number of transactions

Confidence (C) = $(X \cup Y) / | X |$

Example:

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Table 3: Apriori Algorithm Item Set
Source [3]

The table on the above represents the transactions occurred in a store. Based on the two factors **Support (S) & Confidence (C)** we need to find out the largest association between items.

Example: { Milk, Diaper} => Beer

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

By using the table-3, we can derive the candidates and largest itemsets from it. Consider the threshold for support is **20%** and confidence is **50%**.

Pass	Candidates	Large Itemsets
1	**{Bread}, {Milk}, {Diaper}, {Beer}, {Eggs}, {Coke},**	**{Bread}, {Milk}, {Diaper}, {Beer}, {Coke}**
2	**{Bread, Milk}, {Braed, Diaper}, {Bread, Beer}, {Bread, Coke}, {Milk, Diaper}, {Milk, Beer}, {Milk, Coke}, {Diaper, Beer}, {Diaper, Coke}, {Beer, Coke}**	**{Bread, Milk}, {Bread, Diaper}, {Bread, Beer}, {Milk, Diaper}, {Milk, Coke}, {Diaper, Beer}, {Diaper, Coke}**
3	**{Bread, Milk, Diaper}, {Bread, Milk, Beer}, {Milk, Diaper, Coke}, {Diaper, Beer, Coke}**	**{Bread, Milk, Diaper}, {Milk, Diaper, Coke}**
4	**{Bread, Milk, Diaper, Coke}**	**Φ**

Table 4: Candidate and Large Itemsets generate from Table-3.

APRIORI ALGORITHM [10]:

1. C_1 = Itemsets of size one in I;
2. Determine all large itemsets of size 1, $L_{1;}$
3. i = 1;
4. Repeat
5. i = i + 1;
6. C_i = Apriori-Gen(L_{i-1});
7. Count C_i to determine $L_{i;}$
8. until no more large itemsets found;

The detail execution of the Apriori algorithm is as follow. In order to find out the minimum support to qualify an item take part for finding the longest association we need to calculate the frequency of each individual item inside transactions. The first would be the unique items with their counts in transaction.

ITEM	SUPPORT
Bread	**4**
Milk	**4**
Diaper	**4**
Beer	**3**
Coke	**2**
Eggs	**1**

Table 5: One Item with Frequency (Minsup ≥ 2)

On the above table we calculate the support for each individual item inside the transaction. If we consider the minimum support would be **'2'** to qualify the item for association then item eggs will not be considered for association. The next step will take the combination of items from the above table whose minimum support is ≥ 2.

ITEM	SUPPORT
{ Bread, Milk}	3
{Bread, Diaper}	3
{Bread, Beer}	2
{Bread, Coke}	1
{Milk, Diaper}	3
{Milk, Beer}	1
{Milk, Coke}	2
{Diaper, Beer}	3
{Diaper, Coke}	2
{Beer, Coke}	1

Table 6: Combination of Two items

In the above table the combinations all possible two items from the table-3 is represented with its frequencies. The next step would follow the same rule with the table-4 to make the combination of three items.

ITEM	SUPPORT
{Bread, Milk, Diaper}	2
{Bread, Milk, Beer}	1
{Milk, Diaper, Coke}	2
{Diaper, Beer, Coke}	1

Table 7: Three items with its Support

{Bread, Milk, Diaper, Coke} → 1 (Support)

So the longest associations are from table-2 **{Bread, Milk, and Diaper} & {Milk, Diaper, and Coke}.**

In this method we are approaching the single item first then the combinations of items so this process is a bottom-up approach to derive the associations. It means we derive the longest association of items at the end of the process.

The process running time of Apriori algorithm is too high because in a case of 100 items in transactional data set will set 2^{100} possible combinations and it will take 2^{100} ns to calculate the largest association of items. Computationally, this process is slow but it could be problem of NP-hard to find out the largest association from this type of data set using Apriori algorithm.

In order to solve the NP-hard problem of Apriori algorithm, we need to explore the structure of the data so that we will make the running time less for this algorithm. There are some advantages and disadvantages of the Apriori algorithm on below [10]:

Advantages:

- Uses large itemset property.
- Easily parallelized
- Easy to implement.

Disadvantages:

- Assumes transaction database is memory resident.
- Requires up to m database scans.

FP-TREE

FP-Tree derives the association of items by getting the longest association first from the transactions. This is called a top-down approach to derive the longest association. Below are some transactions to derive the longest association/s from items.

TID	ITEMS SOLD
100	F,A,I,M,P,D,G,C
200	K,A,F,B,C.M
300	F,L,S,B
400	T,P,B,C
500	Z,A.M.P.C,F

Table 8: FP-Tree Transaction Data

In order to define the structure of the tree, we need to calculate the frequency of each item in transactions.

ITEM	FREQUENCY
F	4
C	4
A	3
B	3
M	3
P	3
I, D, G, K,L,S,T,Z	1

Table 9: Frequency of Items

By using the table-9, we can rearrange the items as per frequency from the original transactional table (Table-8). The threshold to consider an item for FP-Tree in the above transactions is greater than 1 so item with frequency 1 will not be considered. The F-list for items is F→C→A→B→M→P. By using the F-list we can rearrange the items from transaction from highest frequent items first then the lowest frequent items.

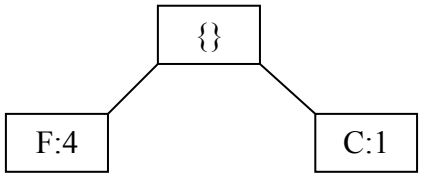
TID	ORDERED ITEMS
100	F,C,A,M,P
200	F,C,A,B,M
300	F,B
400	C,B,P
500	F,C,A,M.P

Table 10: Ordered Items From Table-8

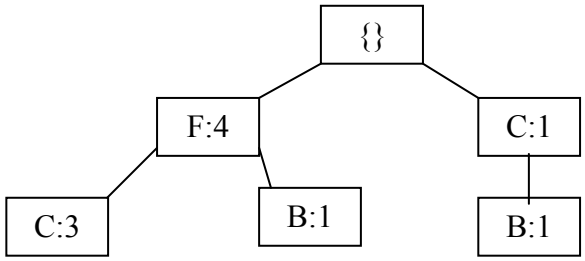
Now, we need to construct the tree based on the ordered items on the table-10. So the tree would be the following:

Construct of FP-Tree from Table-10:

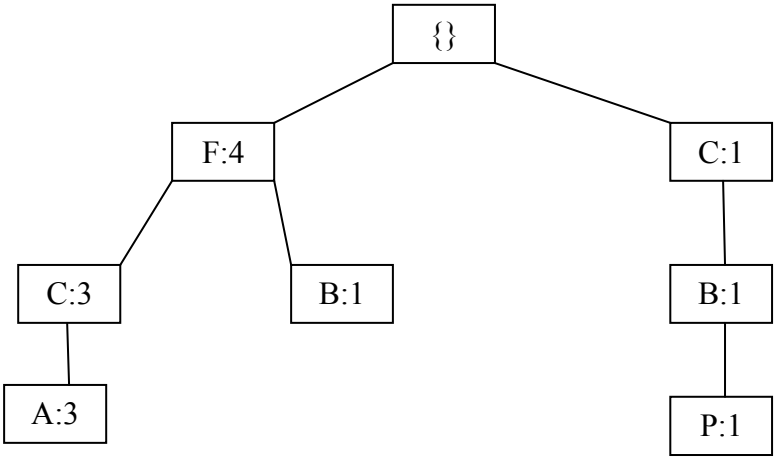
The tree construction starts from left to write by parsing the records column wise. In first column we have items **F & C,** so the root of the tree would be {}. F appears in four records while C appears only one record so F will go the left side of the parent and C will go the right side of the parent. The tree will look like the following:



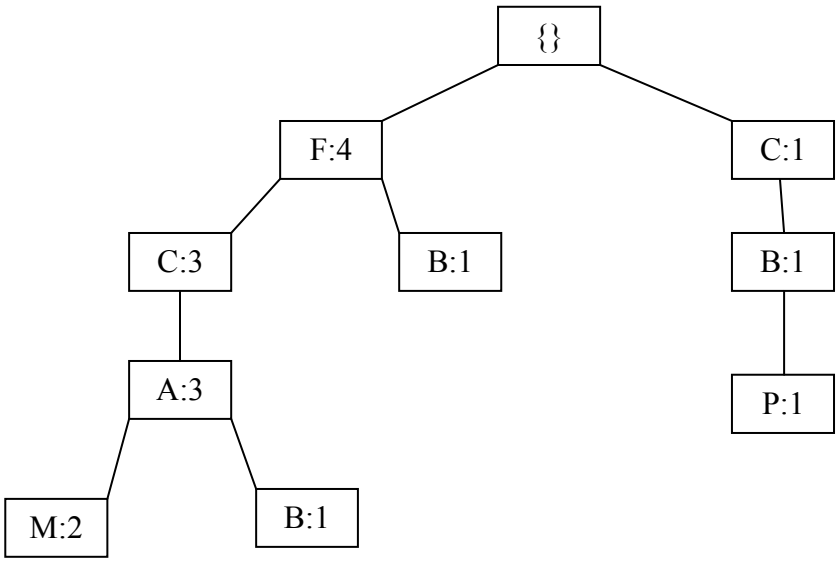
The next column contains **C & B**. C appears thrice after F, B appears once after F, and B appears once after C. So C will go left side of 'F' and B will go right side of 'F'. B will also appear left side of 'C' in the tree.



The next column contains **A & P**. We will follow the same logic like the higher number item will go the left and the lower number will go to the right side of parent.



The next column contains **M & B**.



The next column consists of **P & M**. At the end of construction of the tree we need to draw links to all similar items to different branches.

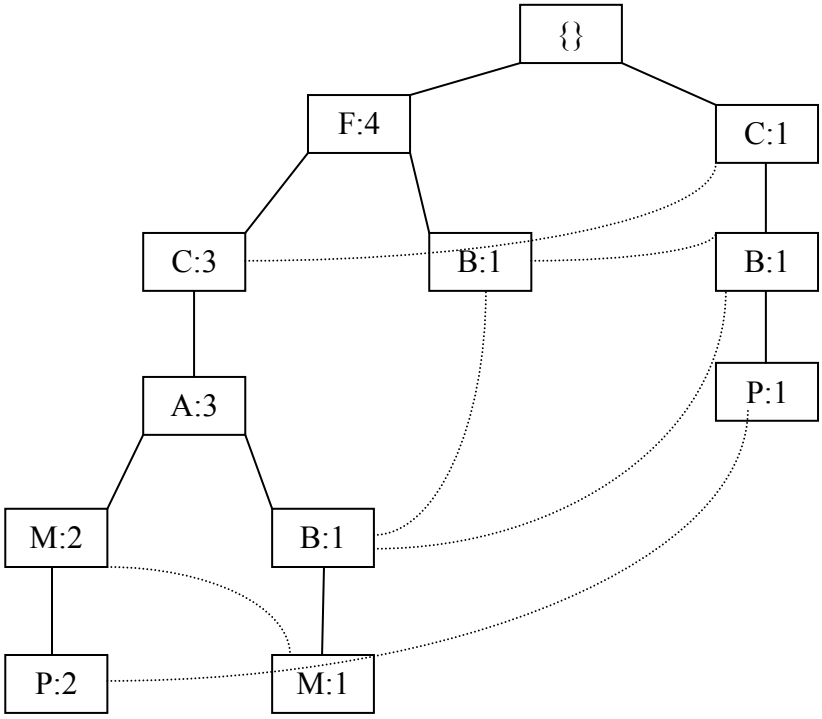


Figure 2: FP-Tree

This tree concludes that each unique individual branch of the tree would be the possible longest association of items.

Bit AssocRule

The Bit AssocRule was proposed by Dr. T.Y. Lin in his paper "A Fast Association Rule Algorithm Based on Bitmap and Granular Computing [6]." A granule is a clump of objects drown together by indistinguishability, similarity, proximity or functionality. The equivalent relations are the granules of the relation. Each unique attribute value in the relational table is a granule, and each granule is a list of tuples that have the same attribute value [6]. In his paper he described how a single bit patter to represent item and perform logical AND, OR, & NOT

operations on it can be faster to find out an association. The bit is set ‘1’for the tuple with attribute value present in it and ‘0’ otherwise.

By using the above two approaches FP-Tree and Bit AssocRule, I defined the new proposed bitmap algorithm to solve the association. FP-Tree gives an idea about the similar tuples form a branch and the later Bit AssocRule gives an idea how to represent the records in granules.

GEOMETRICAL INTERPRETATATION OF ASSOCIATION RULE

Dr. T. Y. Lin proposed some of the key terminologies in his paper “A Simplical Complex, a hyper-graph, structure in the latent semantic space of document clustering” to represent an association in terms of geometry such as CONCEPTS, IDEA, PRIMITIVE CONCEPTS, and MAXIMAL PRIMITIVE CONCEPTS [8]. A CONCEPT is the term associated with connected components of term association. An IDEA consists of many CONCEPTS and that consists of many PRIMITIVE CONCEPTS (in the form of maximal simplexes) [8]. The maximal simplexes of highest dimension is called MAXIMAL PRIMITIVE CONCEPT [8]. A simplex is said to be a maximal if no other simplex in the complex is a superset of it.

In order to represent the association of items in geometrical form, transactions can be represented in the structure of *Simplical Complex*. Each association in the data can be represented as a *Simplex*. The combination of all different size Simplexes can be formed the shape of Simplical Complex.

The transactions from table-3 can be represented in the following geometrical shapes by using the structure of *Simplical Complex*.

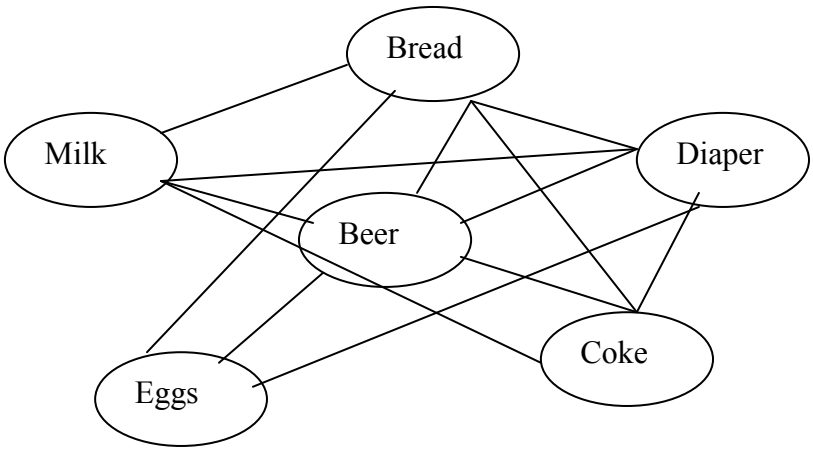
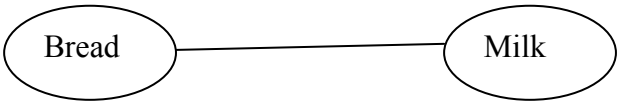
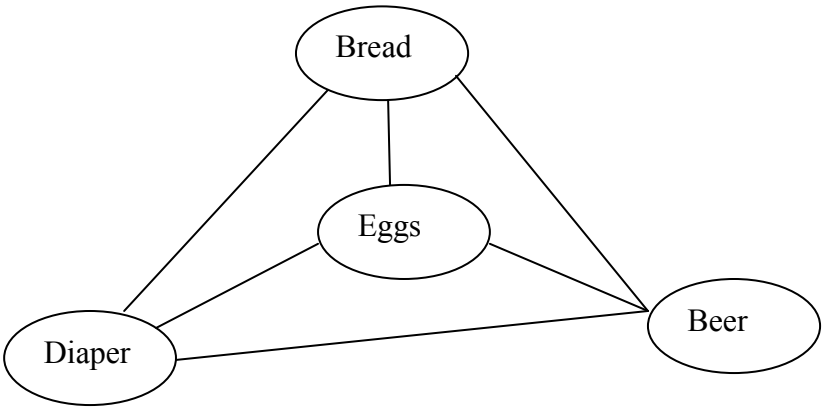


Figure 3: Simplical Complex form of Association from Table-3.

The above figure is a combination of the following simplexes.



{Bread, Milk} is 1-*Simplex*.



{Bread, Diaper, Beer, Eggs} is 3-*Simplexes*.

Similarly, we can represent all others transactions in the form of *Simplexes*. The figure-3 is a combination of all the individual association in the form of *Simplexes*.

The Simplex in the geometry of Simplical Complex doesn't contain by any higher dimension Simplex is called *Maximal Primitive concept*.

Example:

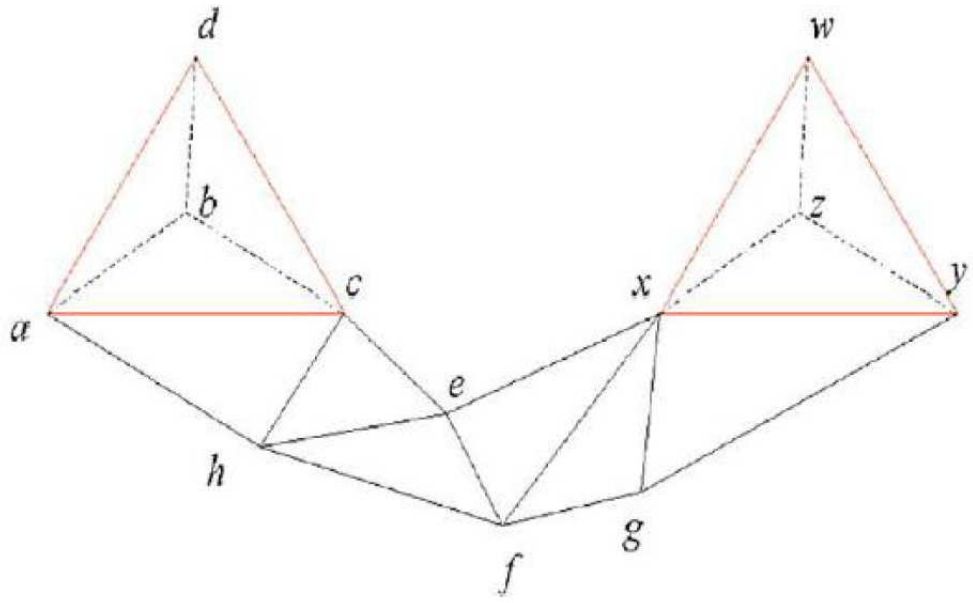


Figure 4: A Complex with 12 Vertexes
Source [8]

In the above figure consists of an IDEA that consists of 12 terms that organized in the forms of 3-complex, denoted by S3. Simplex (a, b, c, d) and Simplex (w, x, y, z) are two maximal simplexes of 3, the highest dimension. Let us consider S3 1, it is the leftover from the removal of all 0-simplexes from S3: Simplex (a, b, c, d) and its 10 faces [8]:

Simplex (a, b, c)

Simplex (a, b, d)

Simplex (a, c, d)

Simplex (b, c, d)

Simplex (a, b)

Simplex (a, c)

Simplex (b, c)

Simplex (a, d)

Simplex (b, d)

Simplex (c, d)

Simplex (a, c, h) and its three faces: Simplex (a, c)

Simplex (a, h)

Simplex (c, h)

Simplex (c, h, e) and its three faces: Simplex (c, h)

Simplex (h, e)

Simplex (c, e)

Simplex (e, h, f) and its three faces: Simplex (e, h)

Simplex (h, f)

Simplex (e, f)

Simplex (e, f, x) and its three faces: Simplex (e, f)

Simplex (e, x)

Simplex (f, x)

Simplex (f, g, x) and its three faces: Simplex (f, g)

Simplex (g, x)

Simplex (f, x)

Simplex (g, x, y) and its three faces: Simplex (g, x)

Simplex (g, y)

Simplex (x, y)

From figure-4:

The Simplex (w, x, y, z) and its 10 faces: Simplex (w, x, y)

Simplex (w, x, z)

Simplex (w, y, z)

Simplex (x, y, z)

Simplex (w, x)

Simplex (w, y)

Simplex (w, z)

Simplex (x, z)

Simplex (x, y)

Simplex (y, z)

The Simplex (a, c), Simplex (c, h), Simplex (h, e), Simplex (e, f), Simplex (f, x), Simplex (g, x), and Simplex (x, y) all have common faces. So they generate a connected path from Simplex (a, b, c, d) to Simplex (w, x, y, z), and sub-paths [8]. Therefore the S3 1 complex is connected. This assertion also implies that S3 is connected. Hence the IDEA consists of a single CONCEPT [8].

Let us consider the (3, 2)-skeleton S3 2, by removing all 0-simplexes from S3: Simplex (a, b, c, d) and its four faces [8]:

Simplex (a, b, c)

Simplex (a, b, d)

Simplex (a, c, d)

Simplex (b, c, d)

Simplex (a, c, h)

Simplex (c, h, e)

Simplex (e, h, f)

Simplex (e, f, x)

Simplex (f, g, x)

Simplex (g, x, y)

Simplex (w, x, y, z) and its four faces: Simplex (w, x, y)

Simplex (w, x, z)

Simplex (w, y, z)

Simplex (x, y, z)

In the figure-4, there are no common faces between any two simplexes, so S3 2 has eight connected components, or eight CONCEPTS [8]. For S3 3, it consists of two non-connected 3-simplexes or two MAXIMAL PRIMITIVE CONCEPTS [8]. A complex, connected component or simplex of a skeleton represent a more technically refined IDEA, CONCEPT or PRIMITIVE CONCEPT [8]. If a maximal connected component of a skeleton contains only one simplex, this component is said to organize a PRIMITIVE CONCEPT [8].

A set of maximal connected component is said to be independent if there are no common faces between any two maximal connected components [8].

As, we saw a skeletal structure of Simplical Complex of higher order consists of combination of all lower order skeletal structures.

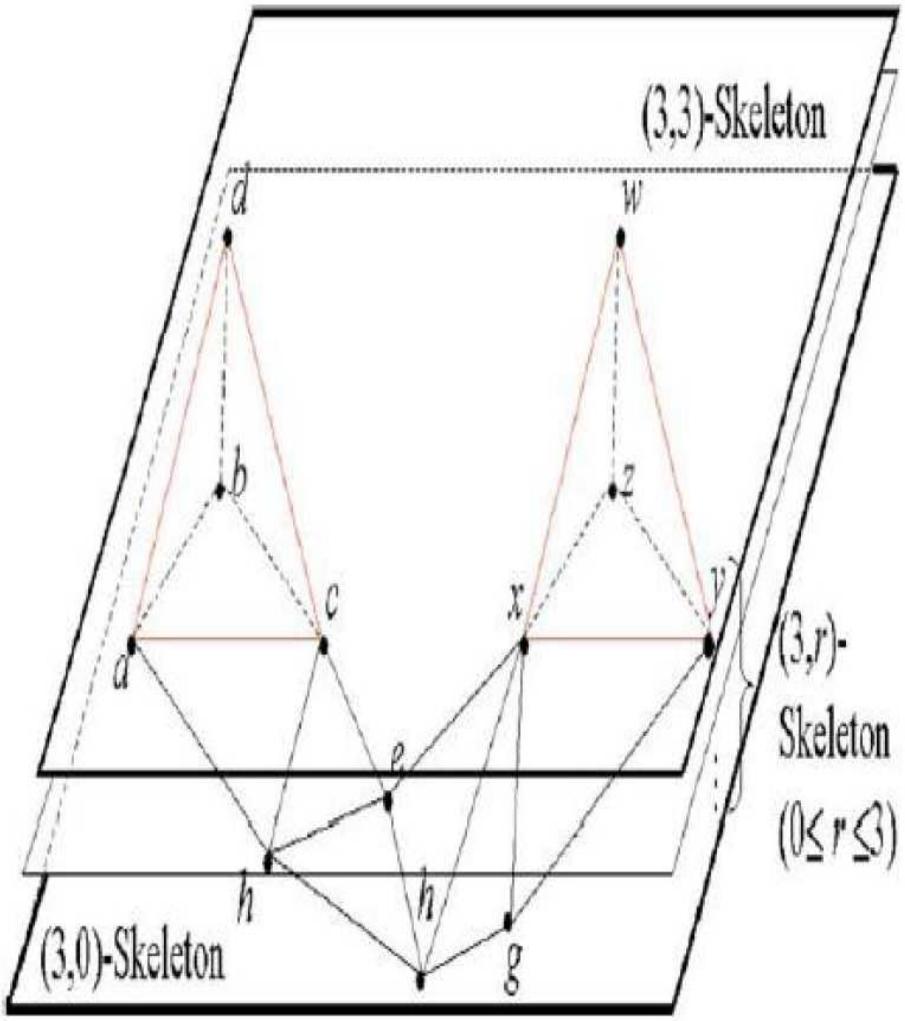


Figure 5: Skeletal Structure of Simplical Complex
Source: [8]

In the above figure if we can say skeletal structure of (3, 3) consists of [3, 0], [3, 1], and [3, 0]. The above description clearly indicates that we can make an orientation of association by using Simplical Complex.

FP-TREE EXPLORATION

The conditional pattern to represent an association can be represented by using the F-list in FP-Tree. The F-list we get on figure-2 is F→ C → A → B → M → P. The conditional pattern base of the FP-tree from figure-2 would be the following.

ITEM	CONDITIONAL PATTERN BASE
C	F:3
A	FC:3
B	FCA:1, F:1, C:1
M	FCA:2, FCAB:1
P	FCAM:2, CB:1

By looking the conditional pattern base we can pick up the possible longest association from the transaction which represents the tree in figure-2. On the above conditional pattern base of different items we can say the conditional pattern base for the item ‘P’ is the longest one.

BITMAP APPROACH

Bitmap is a very noble approach to represent records or tuples from a transactional data set into bit pattern of '1' or '0' inside the database. Bits are suitable to run any logical operations on it. The following table contains some transactional data with items.

TID	ITEMS
100	F,A,I,M,P,D,G,C
200	K,A,F,B,C.M
300	F,L,S,B
400	T,P,B,C
500	Z,A.M.P.C,F

Table 11: Transactional Data Set for Bitmap

Now, we need to convert this into bitmap by using the attribute present in a particular transaction or not.

TID	F	A	I	M	P	D	G	L	S	Z	B	T	C
100	1	1	1	1	1	1	1	0	0	0	0	0	1
200	1	1	0	1	0	0	0	0	0	0	1	0	1
300	1	0	0	0	0	0	0	1	1	0	1	0	0
400	0	0	0	0	1	0	0	0	0	0	1	1	1
500	1	1	0	1	1	0	0	0	0	1	0	0	1

Table 12: Bitmap Table for transactional data from table-11

FREQUENCY OF ITEMS

Frequency of item can get the largest selling items together and it helps to define the stronger association of items. We will rearrange the attribute based on the highest frequency attribute first and then the lowest frequency Items. Bitmap table-11 will become the following table.

TID	F	C	A	B	M	P	I	D	G	L	S	Z	T
100	1	1	1	0	1	1	1	1	1	0	0	0	0
200	1	1	1	1	1	0	0	0	0	0	0	0	0
300	1	0	0	1	0	0	0	0	0	1	1	0	0
400	0	1	0	1	0	1	0	0	0	0	0	0	1
500	1	1	1	0	1	1	0	0	0	0	0	1	0

Table 13: Bitmap Table with highest Frequency First

By sing the bit pattern matching algorithm with highest number of bits matched between records can be placed together can help to compute the association easier. On the above table, the TID-100 has two highest number of bits matches (4) with 200 and 500. Now we can rearrange the tuple by 100, 200, 500, 300, and 400. Rearrangement of the table makes easier to compute the logical operations on it.

TID	F	C	A	B	M	P	I	D	G	L	S	Z	T
100	1	1	1	0	1	1	1	1	1	0	0	0	0
200	1	1	1	1	1	0	0	0	0	0	0	0	0
500	1	1	1	0	1	1	0	0	0	0	0	1	0
300	1	0	0	1	0	0	0	0	0	1	1	0	0
400	0	1	0	1	0	1	0	0	0	0	0	0	1

Table 14: Rearrange of Bitmap table with Tuple Similarity

In order to get the longest association from the above table we need to see all the set bits in each row.

100 → (F, C, A, M, P, I, D, G)

200 → (F, C, A, B, M)

500 → (F, C, A, M, P, Z)

300 → (F, B, L, S)

400 → (C, B, P, T)

Every transaction is a unique transaction on the above table so the most common subset would be the longest association in this table. By using the help of FP-Tree the items with lowest frequency will not be considered so the items F→C→A→M→P would be the longest association in this item set.

BITMAP VISUAL GRAPH

The Bitmap visual graph can be drawn by using the attributes from transactions. This graph can help human to get the longest association quickly than the machine. By using the above transactions the visual graph would be the following.

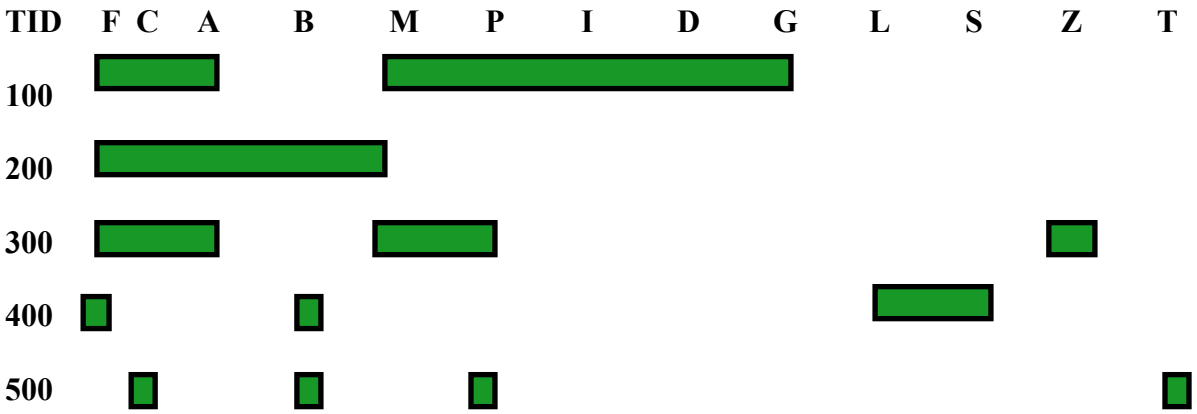


Figure 6: Visual Bitmap of Transaction from Table-14

IMPLEMENTATION

Implementation of the Bitmap algorithm is divided into the following steps:

- Data Store into a table from files
- Sort Table using Highest frequency item first
- Cluster similar records
- Compute the similarity by using the number of bit matches
- Derived the longest association
- Software used

DATA STORAGE

In this process the files are accessing from University of California Machine learning Laboratory with various sizes (records and attributes). In order to store the file into a database we need to calculate the number of unique attributes inside the file. Once we finish the calculation, we need to create a table with the number of unique attributes from files to store the data into it. I choose the **cell-cycle data** and **dauxic-shift** data to run the algorithm. Each file has certain number of unique attributes with lot of missing values in it.

The above two files have different number of attributes with different number of tuples in it. The algorithm counts the attribute presents inside a tuple or not. Based on the attribute's value present it sets the bit fro that position and store the records into database.

e.g.: code snippet:

```
// Code to store data file into database.
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
```

```
public class Connect
{       static int count = 0;
            static String str;
            /* This below code reads a data file and store in strings */
            static public String getContents(File aFile) {
                StringBuilder contents = new StringBuilder();
               try {
                    BufferedReader input =  new BufferedReader(new FileReader(aFile));
                 try {
                   String line = null;
                    while (( line = input.readLine()) != null){
                     contents.append(line);
                     contents.append(System.getProperty("line.separator"));
                     count++;
                   }
                 }
                 finally {
                   input.close();
                 }
               }
               catch (IOException ex){
                 ex.printStackTrace();
               }
```

```
return contents.toString();
                }
   public static void main (String[] args)throws IOException
   {
      Connection conn = null;
      try
      {
         String userName = "root";
         String password = "puspalata";
         String url = "jdbc:mysql://localhost/data";
         Class.forName ("com.mysql.jdbc.Driver").newInstance ();
         conn = DriverManager.getConnection (url, userName, password);
         System.out.println ("Database connection established");
      }
      catch (Exception e)
      {
         System.err.println ("Cannot connect to database server");
      }
      finally
      {
         if (conn != null)
         {
            try
```

```
        {
            Statement stmt = conn.createStatement();
                        String sql = "DELETE FROM data_store WHERE id>=1";
                        int rows = stmt.executeUpdate(sql);
            System.out.println (rows + " record(s) deleted.");
        }
        catch (Exception e) { /* ignore close errors */
            e.printStackTrace();
        }
    }
}
File testFile = new File("C:\\Users\\RAJAT SAR\\Desktop\\cell_cycle-2.txt");
        str = (getContents(testFile));
        String[] record = str.split("\n");
        String[][] value = new String[record.length][];
        // This code below checks the value present inside an attribute.
        int[] count = new int[100];
        for (int j = 1; j < record.length; j++) {
            int a = 0;
            value[j] = record[j].split("\t");
            for (int k=1; k < value[j].length; k++) {
                if (value[j][k].equalsIgnoreCase("") || value[j][k].equalsIgnoreCase("0"))
                    count[a++] = 0;
```

```
            else
                    count[a++] = 1;
}
// Code below stores the data file into database.
        try {
                Statement stmt = conn.createStatement();
                String sql = "INSERT INTO data_store VALUES (";
                sql = sql + "'"+ j + "', ";
                for (int i=0; i<a; i++) {
                        sql = sql + "'" + count[i] + "', ";
                }
                for (int i=a; i<99; i++){
                        sql = sql + "'" + 0 + "', ";
                }
                sql = sql + "'" + 0 +"')";
                //String sql = "INSERT INTO data_store VALUES ('7','0','1')";
                //System.out.println(sql);
                stmt.executeUpdate(sql);
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
```

```
        }
        try {
                conn.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
}
```

DATA ARRANGEMENT WITH FREQUENCY

This process calculates the frequency of each unique attributes from the file and sorted the attribute columns from highest frequency to lowest frequency. At the end of this process we can set aside all the lowest frequency items. The lowest frequency items have very less chance to be an item of longest association as they have very less support.

Below are the some code snippets to compute the above process in a system.

```
// Code to process the file to get the frequency of items in it.
try {
                Statement stmt = conn.createStatement();
                String sql = "SELECT * FROM data_store";
                ResultSet res =         stmt.executeQuery(sql);
                 while (res.next()) {
                        int i = 1;
```

```
                    value[i] = record[i].split("\t");
                    if (max < value[i].length)
                        max = value[i].length;
                    for (int j=1; j<value[i].length; j++) {
                        if ((res.getInt("A-" + (j))== 1))
                            count[j-1] = count[j-1] + 1;

                    }
                }
            }
// code sorts with the frequency of items
for (int i = 0; i< count.length; i++){
            for (int j = i+1; j< count.length; j++) {
                if (count[i] < count[j]){
                    int temp = count[i];
                    count[i] = count [j];
                    count[j] = temp;
                    temp = att[i];
                    att[i]= att[j];
                    att[j] = temp;
                }
            }
        }
```

CLUSTER SIMILAR RECORDS

Cluster similar records based on the similarity of bits can make the process faster. It certainly helps us to group all the relevant records to a particular record. This clustering helps us to process records by running a window on it. In this way we can store the data into memory.

```
// Order the items in a tuple based on F-list.
        System.out.println("\n\n");
            int[][] orditems = new int[matrix.length][max];
            for(int i=0; i<matrix.length; i++){
                int k =0;
                for (int j=0; j < max; j++){
                    if (matrix[i][j] != 0){
                        orditems[i][k++] = att[j];
                    }
                }
                for (int j=k; j<max; j++)
                    orditems[i][j++] = -1;
            }
```

SIMILARITY MATRIX

The similarity matrix is a square matrix can be defined the number of bit matches from one record to other. Below is the similarity matrix from table-13.

	100	200	500	300	400
100	8	4	5	1	2
200	4	5	4	2	2
500	5	4	6	1	1
300	1	2	1	4	1
400	2	2	1	1	4

Table 15: Similarity of bit table from table-13

We can also find out the common items by using the similarity matrix.

DERIVED LONGEST ASSOCIATION

In order to produce the longest association from transaction we need to consider the frequency of items as well as their matches to other records. We also can select the largest number of common subset between records would be the longest association. In order to derive the longest association we need help of FP-Tree to define the possible longest associations.

//FP-Tree code for finding out the possible longest Association

```
for (int i=0; i<matrix.length; i++) {
                    int flag = 0;
                    for (int j=0; j<cn; j++){
                        if (point[j] == att[i]){
```

```
                    flag = 1;
                    break;
                }
            }
            if (flag == 0) {
                point[cn++] = att[i];
                reg++;
                int n = 0;
                for (int l= i+1; l<matrix.length; l++){
                    for (n=max-1; n>=0; n--) {
                        if (orditems[i][n] != -1)
                            break;
                    }
                    if((orditems[i][n] != -1) && (orditems[l][n] != -1) ||
(orditems[i][n++] == -1) && (orditems[l][n++] == -1)){
                        for (int p =0; p<orditems[i].length; p++){
                            if (orditems[i][p] != orditems[l][p]) {
                                flag = 1;
                                break;
                            }
                        }
                        if (flag == 0){
                            reg++;
```

```
                                point[cn++] = att[l];
                            }
                        }
                    }
                    System.out.println("\n\nThe branch of FP-Tree is: ");
                    for (int m=0; m<max; m++){
                        if (orditems[i][m] == -1) {
                            if (scale < m){
                                scale = m;
                                rec = i;
                            }
                            break;
                        }
                        System.out.print((orditems[i][m] + 1) + "->");
                    }if (reg > 1){
                        recfre[t][0] = i;
                        recfre[t++][1] = reg;
                        reg = 0;
                    }
                }
            }
```

SOFTWARE USED

The software used to develop the system are Java (Version1.5), MySQL(Version 5.3) , and Eclipse IDE. Java has very good file parsing apis which really help to quickly extract the contains from a file and store into the database. MySQL is helpful for storing the records in database. Eclipse IDE provides good java development environment to develop and debug the system.

RESULTS

The result of the experiment shows the branches of an FP-Tree and their attributes. This also can show the longest branch of an FP-Tree.

The F-List:

5-52-53-56-57-61-82-14-17-18-19-21-22-23-29-31-35-36-38-40-43-4-1-10-11-60-12-62-63-69-70-78-79-80-81-13-45-9-30-55-74-75-76-77-8-39-59-32-33-67-7-37-72-73-2-16-24-58-34-44-20-46-51-54-66-47-42-15-71-49-27-48-65-28-64-41-26-87-

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->24->58->34->44->20->46->51->54->66->42->15->71->49->27->48->65->28->64->41->26->87->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75-

>76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->64->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->49->27->48->65->28->64->26->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->7->37->72->73->2->16->24->58->34->44->20->46->51->54->47->42->15->71->65->64->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->49->27->48->65->28->26->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->8->39->59->32->33->67->7->37->72->73->2->16->24->58->44->20->46->51->54->66->47->42->15->71->64->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->7->37->72->73->2->16->24->58->34->44->20->46->51->47->42->15->71->49->27->48->65->28->64->26->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->46->51->54->66->42->15->71->49->27->64->41->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->49->27->48->65->28->26->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->7->37->72->73->2->16->24->58->34->44->20->46->51->54->47->42->15->71->65->64->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->

>76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->49->27->48->65->28->26->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->30->55->74->76->77->8->39->32->33->67->7->37->72->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->65->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->44->20->46->51->54->66->47->42->15->71->49->27->48->28->64->41->26->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->74->75->76->77->8->39->59->32->33->67->37->72->73->2->16->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->28->64->41->

The Longest branch of FP-Tree is: 19

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->64->41->26->

Result is measured by the correctness of the process with other existing algorithm to derive the longest association. The correctness of the process can also be derived from the error rate of each individual process.

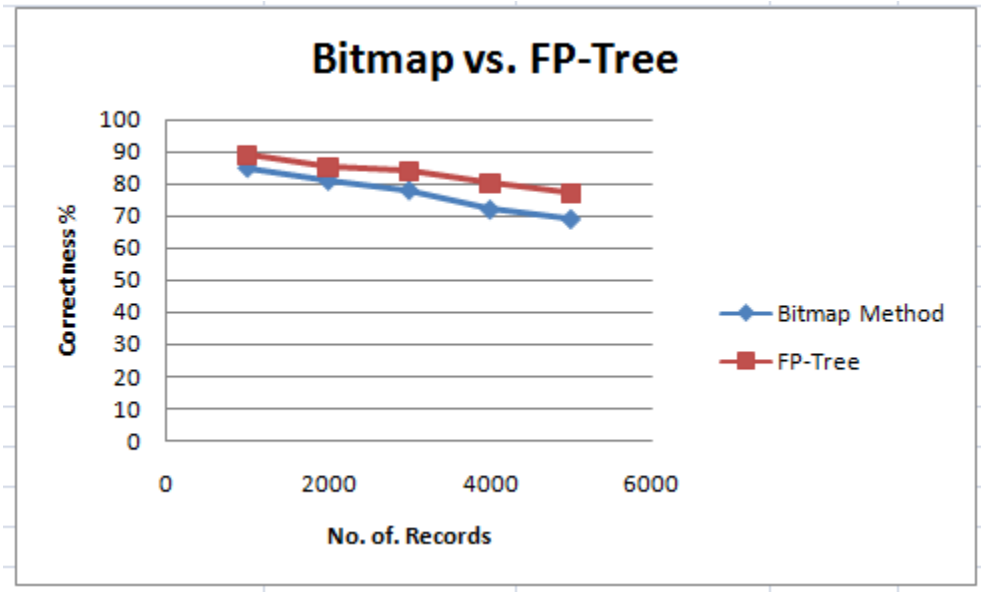


Figure 7: Bitmap Vs. FP-Tree (Correctness)

The above figure represents the accuracy of result of the Bitmap algorithm vs. the FP-Tree algorithm in terms of accuracy of result to find out the exact longest association of items.

The Bitmap algorithm runs almost double speed than the traditional FP-Tree algorithm.

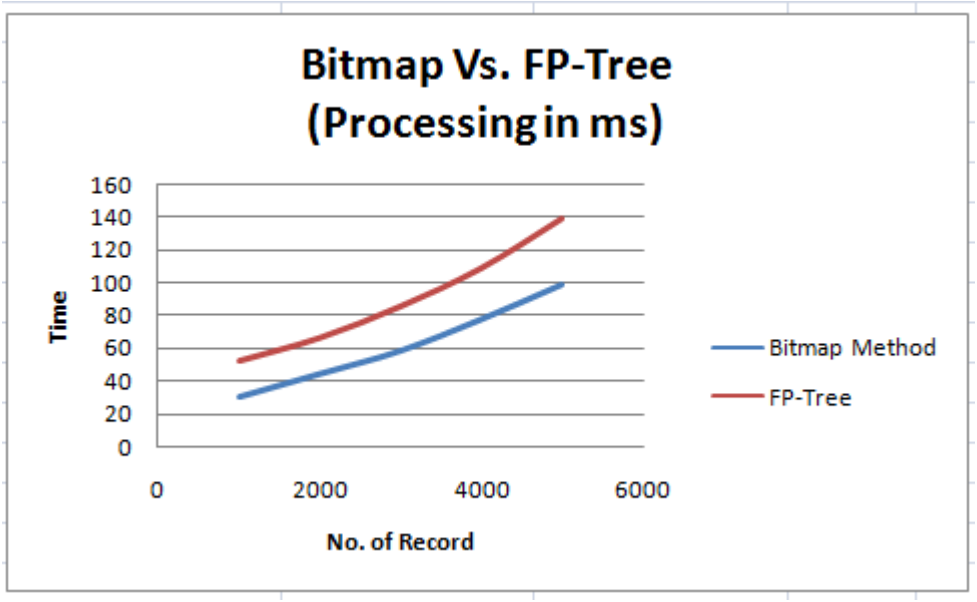


Figure 8: Bitmap Vs. FP-Tree (Processing in ms)

As we can see the above figures (7 & 8) and conclude that Bitmap algorithm runs faster with approximately good result to produce the longest association from item set.

CONCLUSION

The Bitmap algorithm quickly and effectively generates the longest association on item sets. The nature of the algorithm solves the memory as well as processing speed problem in a system. This algorithm takes less number of scanning on transactions to produce the result on it. Visual graph drawn by Bitmap can help the human to quickly select a longest association from items. This approach solves the association problem effectively as compared to other algorithms. This algorithm enhances the performance of the association mining.

FUTURE WORK

The dropping of columns from the bitmap table is still not perfectly solved to derive the association. We can select multiple columns to drop from the table and run logical operations on rest of the columns inside the records to find out quickly the longest association but selection of columns needs to be perfect to pick the appropriate columns to be dropped from the table. Also, we can make the execution parallel on different small sizes data.

REFERENCES

1. Agrawal, Rakesh, & Srikant, Ramakrishnan. Mining Generalized Association Rules. IEEE Transaction on Very Large Databases. 1995.

2. Han, Jiawei, & Kamber Micheline. Data Mining Concepts and Techniques. Morgan Kaufmann, 2006.

3. Tan, Pang-Ning, Steinbach, Michael, & Kumar, Vipin. Introduction to Data Mining. Addison Wesley, 2006.

4. http://en.wikipedia.org/wiki/Simplex (Simplex in Geometry)

5. http://en.wikipedia.org/wiki/Simplicial_complex (Simplical Complex)

6. Lin, T.Y., Hu, Xiaohua, & Louie, Eric. A Fast Association Rule Algorithm Based on Bitmap and Granular Computing. IEEE International Conference Fuzzy Systems*, May 25-28, 2003, St Louis, Missouri, 678-683.*

7. http://archive.ics.uci.edu/ml/datasets.html (UCI Machine Learning Repository)

8. Lin, T.Y., & Chiang, I-Jen. A Simplical complex, a hyper-graph, structure in the latent semantic space of document clustering. Internat. J. Approx. Reason. (40) 2005.

9. Spanier, Edwin H. Algebraic Topology. McGraw-HILL series in higher mathematics.

10. Dunham, Margaret H. Data Mining: Introductory and Advanced Topics, Prentice Hall, 2003.

APPENDIX

The results of the experiment:

Database connection established
43 record(s) deleted.
83
100

No.of items: 78

Attribute no.	Frequency
5	43
52	43
53	43
56	43
57	43
61	43
82	43
14	42
17	42
18	42
19	42
21	42
22	42
23	42
29	42
31	42
35	42
36	42
38	42
40	42
43	42
4	42
1	42
10	42
11	42
60	42
12	42
62	42
63	42
69	42
70	42
78	42

79	42
80	42
81	42
13	42
45	41
9	41
30	41
55	41
74	41
75	41
76	41
77	41
8	41
39	41
59	41
32	41
33	41
67	40
7	40
37	40
72	40
73	40
2	40
16	40
24	40
58	40
34	40
44	40
20	40
46	40
51	40
54	39
66	39
47	39
42	39
15	37
71	34
49	33
27	33
48	32
65	31
28	30
64	25
41	23
26	23
87	1

The F-List:
5-52-53-56-57-61-82-14-17-18-19-21-22-23-29-31-35-36-38-40-43-4-1-10-11-60-12-62-63-69-70-78-79-80-81-13-45-9-30-55-74-75-76-77-8-39-59-32-33-67-7-37-72-73-2-16-24-58-34-44-20-46-51-54-66-47-42-15-71-49-27-48-65-28-64-41-26-87-

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->24->58->34->44->20->46->51->54->66->42->15->71->49->27->48->65->28->64->41->26->87->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->64->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->49->27->48->65->28->64->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->7->37->72->73->2->16->24->58->34->44->20->46->51->54->47->42->15->71->65->64->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->49->27->48->65->28->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->8->39->59->32->33->67->7->37->72->73->2->16->24->58->44->20->46->51->54->66->47->42->15->71->64->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->7->37->72->73->2->16->24->58->34->44->20->46->51->47->42->15->71->49->27->48->65->28->64->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->46->51->54->66->42->15->71->49->27->64->41->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->49->27->48->65->28->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->7->37->72->73->2->16->24->58->34->44->20->46->51->54->47->42->15->71->65->64->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->49->27->48->65->28->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->30->55->74->76->77->8->39->32->33->67->7->37->72->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->65->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->44->20->46->51->54->66->47->42->15->71->49->27->48->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->74->75->76->77->8->39->59->32->33->67->37->72->73->2->16->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->28->64->41->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->66->47->42->15->71->49->27->48->65->28->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->58->34->44->20->46->54->66->47->42->15->71->49->27->48->28->64->41->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->59->32->33->67->7->72->73->16->24->58->34->44->20->51->54->66->47->71->49->27->48->65->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->71->49->27->48->65->28->41->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->32->33->67->7->37->72->2->16->24->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->41->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->

>76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->41->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->15->71->49->27->48->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->41->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->66->47->42->15->49->27->48->65->28->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->64->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->54->47->42->15->71->65->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->46->51->54->66->47->42->15->49->27->48->65->28->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->35->4->1->10->11->60->12->62->69->70->78->79->80->81->13->9->55->74->75->76->77->8->39->59->67->7->72->73->2->16->24->58->20->51->54->66->15->71->65->64->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->49->27->48->65->28->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->65->64->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->49->27->48->65->28->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->73->2->16->24->58->34->44->20->46->51->54->66->47->42->71->49->27->48->65->64->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->29->31->35->36->38->40->43->1->60->62->63->55->39->59->33->67->34->51->54->66->47->49->27->48->65->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->65->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->66->47->42->15->71->49->27->48->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->28->64->41->

The branch of FP-Tree is:

5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->42->15->71->49->27->48->65->41->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->24->58->34->20->46->54->66->47->42->15->71->49->27->48->28->64->41->26->

The branch of FP-Tree is:
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->41->

The Longest branch of FP-Tree is: 19
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->60->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->64->41->26->

Bit-Map Aid for FP-Tree Growth:

Total no. of unique records: 21

Unique Record Number	Frequency
1	2
3	2
5	2
7	2
9	2
11	2
13	2
15	2
17	2
19	2
21	2
23	2
25	2
27	2
29	2
31	2
33	2
35	2
37	2
39	2
41	2

The threshold to consider an attribute for finding longest association: 3

The longest association is:38
5->52->53->56->57->61->82->14->17->18->19->21->22->23->29->31->35->36->38->40->43->4->1->10->11->12->62->63->69->70->78->79->80->81->13->45->9->30->55->74->75->76->77->8->39->59->32->33->67->7->37->72->73->2->16->24->58->34->44->20->46->51->54->66->47->42->15->71->49->27->48->65->28->64->41->26->

1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->0->1->1->

>1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->

>1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->1->0->