Fall 2011

Association Mining in Database Machine

Jindou Jiao
San Jose State University

Association Mining in Database Machine

A Writing project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirement for the Degree

Master of Science

By

Jindou Jiao

Fall 2011

Approved by: Department of Computer Science
College of Science
San Jose State University
San Jose, CA

____

Dr. Tsau Young Lin

____

Dr. Chris Tseng

____

Dr. Howard Ho, IBM Almaden Research Center

Acknowledgements

I thank my advisor Dr. T.Y. Lin, whose support me to complete my project. Dr. Lin is always there to give me technical advice and help me find the best solution to the problems I met during the project. I also like to thank Dr. Chris Tseng and Dr. Howard Ho for participation as my thesis committee members.

Very special thanks to my family and friends for their love and encourage.

Thank you so much!

Abstract

Association rule is wildly used in most of the data mining technologies. Apriori algorithm is the fundamental association rule mining algorithm. FP-growth tree algorithm improves the performance by reduce the generation of the frequent item sets. Simplex algorithm is a advanced FP-growth algorithm by using bitmap structure with the simplex concept in geometry. The bitmap structure implementation is particular designed for storing the data in database machines to support parallel computing the association rule mining.

Keywords: Data mining, Association rule mining, Database Machine, Simplex algorithm

Table of Contents

List of Tables

List of Figures

1 Introduction

This section state the basic concept of data mining and the technologies data mining used. Data mining discovers and exacts knowledge and information from big amount of data. Database, data warehouse and other data storage can provide data source for data mining usage.

1.1 Data mining and related concepts

The main components for data mining are categorized as:

- Database, data warehouse or other data storage
- Database or data warehouse server
- Knowledge base
- Data mining engine
- Pattern analysis module
- User interface

By principle, data mining can be applied to any type of data, which includes: relational database, data warehouse, transaction database, advance database system and web data. There are two stages of data mining: knowledge discovery and prediction. Knowledge discovery is used to analysis data and find out the general pattern from the data. Prediction uses the result of knowledge discovery to predict the further stuff. Usually, the main tasks for data mining are association analysis, clustering, classification, predication, time-series pattern and deviation.

- Association analysis

  Association rules are the important information of the database which is discoverable. The goal of association analysis is to find out the implicit relation of the data. Support and confidence are used to measure the association between or among two or more variables.

- Clustering

  Clustering classifies the data into different categories based on their similarity. The data in the same category are similar with each other. Clustering analysis can find out the disturbing pattern of the data and the possible relationship of data.

- Classification

  Classification is to find out the description of a category. This description stands for the information of the whole category and also be used to create a pattern based on it.

- Predication

  Predication is to use the data of the past to find out the changing rules and create models. Then use the rules and models to predict the data and its characters will happen in future. Accuracy and uncertainty are important for predication.

- Time-series pattern

  Time-series pattern is based on the time order find out the repeat happened high probability pattern. Same as the predication, it also uses the known data to predict the value of data in future.

- Deviation

  Deviation contains much useful knowledge. Data in the database always have some abnormal cases, to discover these abnormal cases are important. The basic method to do the deviation checking is to compare the data with the pattern.

1.2 Data Mining Core Technology

1.2.1 Artificial neural network

Artificial neural network is a mathematic model for information processing which is inspired by the structure of neural synapse networks. Artificial neural network is a computational model composed by a group of connected artificial neural cells. Each cell stands for a kind of output function which is called activation function. The interconnection between every two nodes stands for the weight value of this connection signal. The output of the network is based on how the cells connected the weight values and activation function. Artificial neural network is a system can automatic learn and summarize the known data and find out the patterns among them. ANN can be used for inference based on that[1].

1.2.2 Genetic Algorithm

Genetic algorithm is a random search algorithm inspired by the natural selection, inheritance and other evolution mechanisms in biology. It is one of the evolution algorithms to solve the mathematical optimization problems. For an optimization problem, the abstract of amount of candidate solutions represent the group could evolve to a better solution. Binary strings usually are used to present a solution as 0s and 1s. Evolution starts from a totally random individually group and evolving by every generation. For every generation, the fitness of the whole group will be evaluated. Based on the fitness value, choose multiple individuals and generate new live group by natural selection and mutation. This new group will be the current group for next iteration in the algorithm. Generic algorithm is proved by Sunil can be used for data mining.

1.2.3 Decision Tree

Decision Tree is an algorithm for predictive model. It consist a decision diagram and the possible results. It classifies a big amount of data, and find out the valuable potential information from them.

1.2.4 Rough Set

In the theory of rough set, crisp set is the traditional set, but the rough set is used to get the approximation of the original set as give the upper approximation and the lower approximation set for the crisp set. The upper and lower approximation sets are both crisp set. The rough set processes the two dimensional data table as its object. The relational data base management system and the data warehousing management system provide the data as the foundation to rough set method for data mining.

2 Association rule mining

Single transaction of the shopping cart can tell us the shopping behavior of a particular costumer. After getting a large amount of records of the shopping cart transactions, the behavior of the whole costumers can be analyzed and predicted. Association rule mining helps to find out the relationship amount the items costumers purchased in their shopping carts. The well known example of association rule mining is the “diaper and beer”. The analysis of the transactions of a grocery store discovered that young men who buy diapers may also buy beer. This indicates an unexpected relation between two products in costumers’ daily shopping behaviors.

2.1 Definition of association rule mining

In the transaction record, every transaction contains the transaction id and the items number purchased by the costumers. The set of the items happened in a transaction is called the “Itemset.” A transaction is called T [2].

Assume X is a itemset, if all items in X are all included in transaction T, then it is called transaction T supports itemset X. The support count of itemset X is defined as the total amount of the transactions which support itemset X. the support of the itemset X is the percentage of the number of transactions which support itemset X in the total number of all transactions.

The association rule is in X -> Y[suport], X and Y are two itemsets. The support of the rule X->Y is the support of the union set of X and Y which is XUY. The confidence of the rule X -> Y is the percentage of the support if the rule X -> Y in the support of the item set X.

Confidence = support (X U Y) / support X.

Association rule must satisfy two pre-defined valued: the minimum support and the minimum confidence. The minimum support count is the minimum support multiplies the total amount of the transactions in the record.

The problem of the association rule mining can be split into two sub problem. The first one is to find out all item sets which support is equal or greater than the minimum support. These item sets are large item sets. The second one is to find out the association rules which confidence is equal or greater than the minimum confidence from the large item sets. Obviously, as long as the large item sets have been found, we can get the association rules directly from the large item sets.

2.2 Apriori Algorithm

An item set which contains k items is called k item set. Lk is the union set of all large k item sets. Apriori algorithm uses the large item sets found in pervious step to find out the large item set for next step. Which means, first find out the large item set L1 and then based on L1 to find out the large item set L2, and so on, until no more large item set can be found. To reduce the calculation time to generate the large item set, Apriori uses the breath-first method to count the support and generates the large item sets by the down closure property. The down closure property of the item set is any sub-set of a large item set must be a large item set [3].

Assuming {A, B} is a large item set, based on the definition of the large item, the support of {A, B} must equal or greater than the minimum support. For the sub-set of {A, B}: {A} and {B}, in the transactions record, the support of item set {A} must equal or greater than the support of the large item set {A, B}. Thus, the item set {A} is a large item set. Similar for item set {B}, {B} is a large item set.

Apriori algorithm uses the down closure property to generate candidate item sets. There are two steps to generate the candidate item sets:

- Join procedure
- Prune procedure

Generate candidate k item sets is based on the (k-1) large item set. Ck stands for union set of all candidate k item sets. X1 and X2 are two (k0 - 1) large item sets, Xi[j] stands for the jth item in item set Xi. It is assumed all items in the item set are in the ascending order. If the first k-2 items in X1 and X2 are all the same, and $X_1[k\text{-}1] < X_2[k\text{-}1]$, then X1 and X2 can be combined together to get a k candidate item set, which is $\{X_1[1], X_1[2], \ldots, X_1[k\text{-}1], X_2[k\text{-}1]\}$. Using the condition $X_1[k\text{-}1] < X_2[k\text{-}1]$ is to avoid re-generating the duplicated K item set.

Delete candidate item sets which do not satisfy the Apriori property. Candidate k-item set C_k includes L_k, but may not equal to L_k. If $X \in C_k$, while Apriori algorithm querying each transaction from the record, it will add 1 to the support of X if the transaction support item set X. Reduce the size of C_k by applying Apriori property: if any sub-set of size (k-1) of item set X is not large (k-1) item set, then X must not be a large k item set, then delete X from C_k.

Pseudo code of Apriori Algorithm:

```
$L_1$ = Union set of large 1 item sets;
For ($k$ = 2; $L_{k-1} \neq \emptyset$; $k$++)
    $C_k$ = Candidate_generate ($L_{k-1}$);
    for each transaction t
        For each item set c in $C_k$
            If t contains c
                Support of C ++;
    For each candidate sub-set $C_i$ of $C_k$
        If support of $C_i$ >= minimum support
            $L_k$ += $C_i$;
        End;
    Return L = all large candidate item sets;
```

2.3 Simplex

In geometry, simplex or n-simplex is an n dimensional polytope which is a convex hull n + 1 vertices. N starts from 1 to arbitrary. 1-simplex is a line segment. 2-simplex is a triangle. 3-simplex is a tetrahedron. 4-simplex is a heptapeton. A simplex is the smallest convex set of the given vertices. A regular simplex is a simplex which is a regular polytope. An n-simplex can be built by connecting a new vertex to a (n - 1)-simplex using the common edge length.

Figure 1 - 0-simplex or a point

Figure 2 - 1-simplex or a line segment

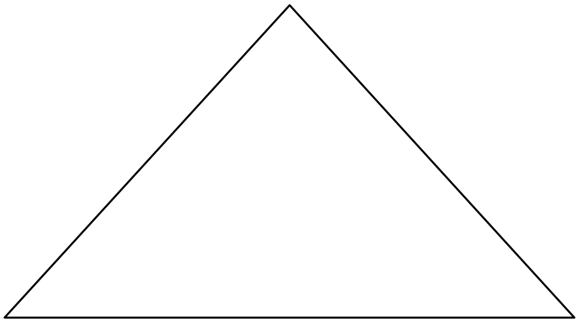

Figure 3 - 2-simplex or a triangle

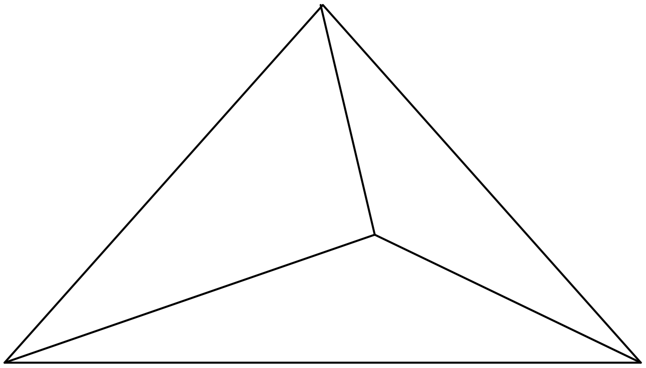

Figure 4 - 3-simplex or a tetrahedron

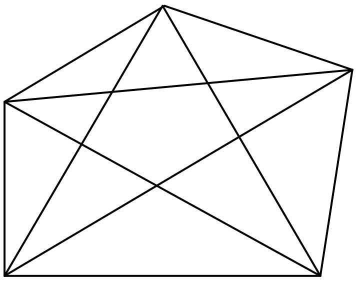

Figure 5 - 4-simplex

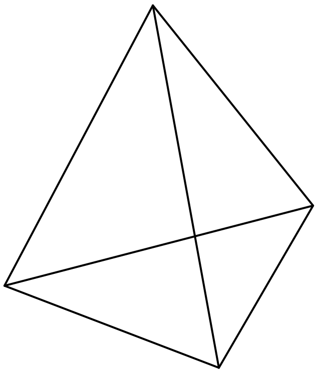

Figure 6 - Regular 3-simplex

3 Database machine – shared nothing architecture

3.1 Shared nothing architecture

Shared nothing architecture is a distributing architecture for computing. Each node in the architecture has its own resources to complete its tasks. The resources include memory and hard disk for storage. The nodes in the architecture are independent from each other and all self sufficient. There is no single point of contention in the share nothing architecture.

For processing the information gathered from each distributed node, there is a place to get all the information integrate together and generate the final result.

3.2 Master and slave module

I used the master and slave module to implement the data warehousing system. Master machine is the center point of the system. It sends the query request parallel to each slave machine. Slave nodes use shared noting architecture to process the query request from Master independently. After completing the query, each node sends the result back to the master machine. Master machine gathers the result from each node and then integrate them together.

3.3 Bitmap structure in Database machine

Bitmap structure is used to store the data in my data warehouse experiment [2]. It's the core technology implemented for providing data used by Database machine. Bitmap was converted data in multiple database tables from different database sources. I first compressed the bitmap by compressing every 32 1s and or 0s into one integer. Then store the integers into files which simulated the hard drive structure. The I/O operations on these files can access the whole segment bitmap data by a single read or write operation. The bitmap data is evenly split and stored on different slave machines responding to master machine's request.

4 Bitmap simplex algorithm

Jia implemented Bitmap FP-growth algorithm by applying bitmap technology to improve the FP-growth tree algorithm[3]. Bitmap simplex algorithm is designed for parallel computing on database machines based on bitmap FP-growth algorithm. I use the same example to describe the bitmap simplex algorithm as Jia[3]. The transactions in the table are picked from several data sets. The minimum support is 3[4][5].

ID	Items in Basket
1	5,10,14,16,17,19,34,38,39
2	5,7,23,26,31,40,47,59,63
3	5,13,19,23,26,29,36,40,75
4	5,19,27,29,30,33,36,38,54
5	5,7,10,12,13,20,54,59,90
6	5,13,14,24,27,30,32,43,48
7	6,7,14,17,19,21,34,38,46
8	6,10,21,22,24,28,29,35,39
9	7,16,17,21,40,43,46,53,70
10	7,13,34,35,38,46,48,54,68

Table 1- Transaction Table

The qualified item set is {10, 46, 40, 21, 17, 38, 13, 19, 7, 5} with the minimum support of 3.

Item ID	Frequency Count(Support)
10	3
46	3
40	3
21	3
17	3
38	4
13	4

19	4
7	5
5	6

Table 2 – Support for qualified items

The bitmap table of this item set is as follows:

10	46	40	21	17	38	13	19	7	5
1	0	0	0	1	1	0	1	0	1
0	0	1	0	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	1
0	0	0	0	0	1	0	1	0	1
1	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	1	0	0	1
0	1	0	1	1	1	0	1	1	0
1	0	0	1	0	0	0	0	0	0
0	1	1	1	1	0	0	0	1	0
0	1	0	0	0	1	1	0	1	0

Table 3 – Bitmap table for item set

Sort the item set in descending order and get {5, 7, 19, 13, 38, 17, 21, 40, 46, 10}, then pick the first item's bitmap to do the AND operation with all other items' bitmap.

5
1
1
1
1
1

AND

10	46	40	21	17	38	13	19	7
1	0	0	0	1	1	0	1	0
0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	1	1	0
0	0	0	0	0	1	0	1	0
1	0	0	0	0	0	1	0	1

1
0
0
0
0

0	0	0	0	0	0	1	0	0
0	1	0	1	1	1	0	1	1
1	0	0	1	0	0	0	0	0
0	1	1	1	1	0	0	0	1
0	1	0	0	0	1	1	0	1

Table 4 – Intermediate step 1 for item 5

The result bitmap of this operation is:

10	46	40	21	17	38	13	19	7
1	0	0	0	1	1	0	1	0
0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	1	1	0
0	0	0	0	0	1	0	1	0
1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0

Table 5 – Intermediate bitmap 1

Sort this table in descending order, the last three items will be ignored in the following steps because they could not qualify the minimum support**:**

Item	Amount
19	3
13	3
7	2
38	2
40	2
10	2

17	1
46	0
21	0

Table 6 – First step sorted result

Pick the first item from the table above: 19 to do the AND operation with the first item in the original table 5:

5		19		Result
1		1		1
1		0		0
1		1		1
1	AND	1	=>	1
1		0		0
1		0		0
0		1		0
0		0		0
0		0		0
0		0		0

Table 7 – Intermediate step 2 for item 5

The frequency of the result R1 is 2, which is qualified with the minimum support of 3. So set {5,19} is a solution.

Use the result R1 to do the AND operation with the left 8 items in the item set:

{7, 13, 38, 21, 40, 46} and get the following result:

R1
1

10	40	38	13	7
1	0	1	0	0

0
1
1
0
0
0
0
0
0

AND

0	1	0	0	1
0	1	0	1	0
0	0	1	0	0
1	0	0	1	1
0	0	0	1	0
0	0	1	0	1
1	0	0	0	0
0	1	0	0	1
0	0	1	1	1

=>

10	40	38	13	7
1	0	1	0	0
0	1	0	1	0
0	0	1	0	0

Table 8 – Intermediate step 2 result

The result table shows only item 38 is still qualify the minimum support after AND.

Now find a solution set {5, 19, 38}.

After getting the first solution, use item 5 to do the AND operation with the second top item 13 from table step 1 sorted result:

5
1
1
1

13
0
0
1

Result
0
0
1

1	AND	0	=>	0
1		1		1
1		1		1
0		0		0
0		0		0
0		0		0
0		1		0

Table 8 – Intermediate step 3 for item 5

The result of this step R2 has 3 of 1s, so set {5, 13} is a solution. Using result R2 to do the AND operation with item set: {7, 38, 40, 10}

R2		10	40	38	7
0		1	0	1	0
0		0	1	0	1
1		0	1	0	0
0	AND	0	0	1	0
1		1	0	0	1
1		0	0	0	0
0		0	0	1	1
0		1	0	0	0
0		0	1	0	1
0		0	0	1	1

= >

10	40	38	7

0	1	0	0
1	0	0	1
0	0	0	0

Table 9 – Intermediate step 3 result

The table above shows there is no qualified solution for the AND operation.

Then, we need to go back to use the first item of the original table 5 and do the AND operation with the third item from the step 1 result set: item 7:

5		7		Result
1		0		0
1		1		1
1		0		0
1	AND	0	=>	0
1		1		1
1		0		0
0		1		0
0		0		0
0		1		0
0		1		0

Table 10 – Intermediate step 4 for item 5

There are two 1s in the result R3, so it is another solution: {5, 7}.

Then use the result R3 to add the item set {10, 40, 38}:

R3
0
1
0
0
1
0
0
0
0
0

AND

10	40	38
1	0	1
0	1	0
0	1	0
0	0	1
1	0	0
0	0	0
0	0	1
1	0	0
0	1	0
0	0	1

= >

10	40	38
0	1	0
1	0	0

Table 11 – Intermediate step 4 result

The result table shows there is no solution.

Then add item 5 and 38:

5
1
1
1

38
1
0
0

Result
1
0
0

1	AND	1	=>	1
1		0		0
1		0		0
0		1		0
0		0		0
0		0		0
0		1		0

Table 12 – Intermediate step 5 for item 5

Result R4 is a solution: {5, 38}

And R4 with item set: {10, 40}:

R4		10	40
1		1	0
0		0	1
0		0	1
1	AND	0	0
0		1	0
0		0	0
0		0	0
0		1	0
0		0	1
0		0	0

= >

10	40
1	0
0	0

Table 13 – Intermediate step 5 result

There is no solution in this step.

Add item 5 with item 40:

5		40		Result
1		0		0
1		1		1
1		1		1
1	AND	0	=>	0
1		0		0
1		0		0
0		0		0
0		0		0
0		1		0
0		0		0

Table 14 – intermediate step 5 result

Result R6 is a solution: {5, 40}

And R6 with the last item left 10:

R6		10
0		1
1		0
1		0
0	AND	0
0		1
0		0
0		0
0		1
0		0
0		0

= >

10
0
0

Table 15 – Intermediate step 6

There is no solution in this step.

Then add item 5 and item 10:

5	10	Result
1	1	1
1	0	0
1	0	0

1	AND	0	=>	0
1		1		1
1		0		0
0		0		0
0		1		0
0		0		0
0		0		0

Table 16 – intermediate step 7

Result R6 is a solution: {5, 10}

Summarize all steps above, we found all solutions item 5 involved:

{5,19}
{5, 19, 38}
{5, 13}
{5, 7}
{5, 38}
{5, 40}
{5, 10}

Table 17 - Solutions for item 5

The table above can be described in simplex as follows:

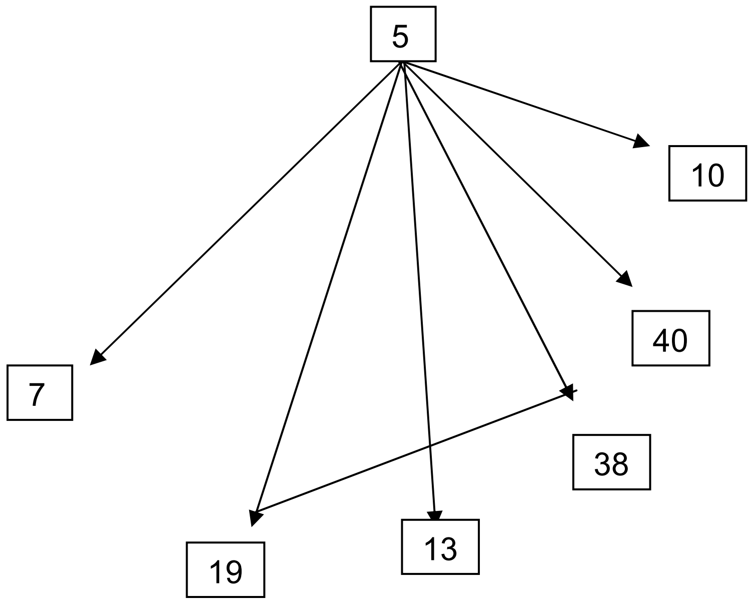

Figure 7 – simplex diagram for item 5

Repeat the above steps for the item in the second position sorted item 7, I got all solutions for it:

{7, 46}
{7, 46, 38}
{7, 46, 17}
{7, 46, 21}
{7, 46, 38, 17}
{7, 46, 38, 21}
{7, 13}
{7, 38}
{7, 17}
{7, 21}
{7, 40}

Table 18 – Solutions for item 7

The table above can be described in simplex as follows:

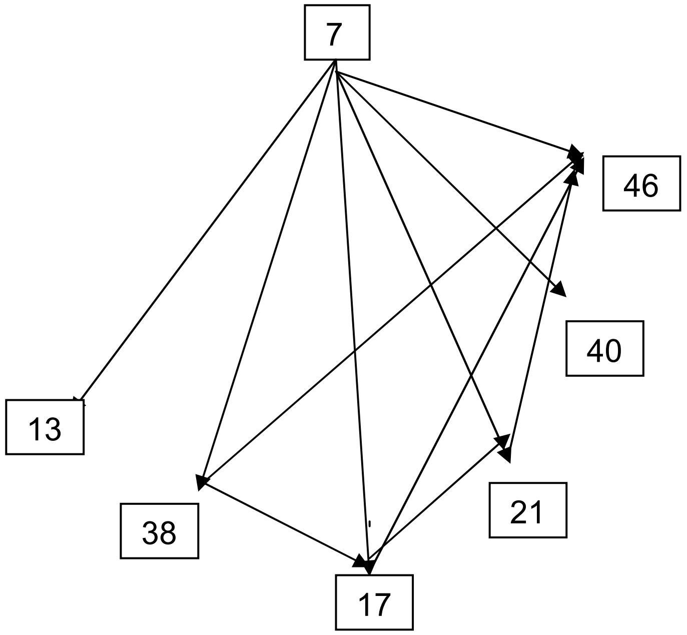

Figure 8 – Simplex diagram for item 7

For item 19, the solutions are as follows:

{19, 38}
{19, 38, 17}

Table 19 – Solutions for item 19

The table above can be described in simplex as follows:

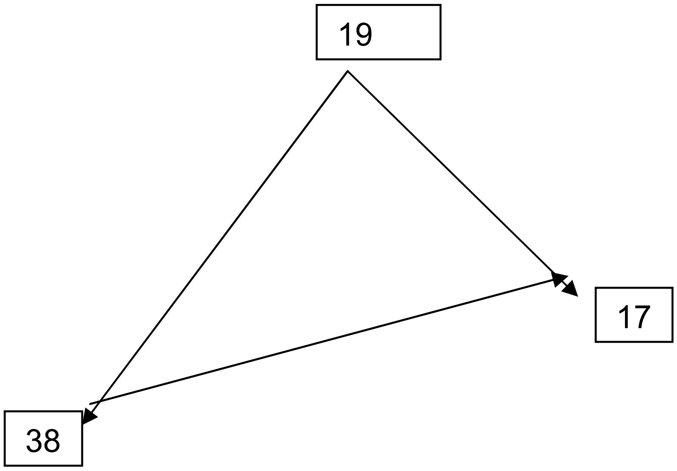

Figure 9 – Simplex diagram for item 19

For item 13, there is no valid solution.

For item 38, the solutions are as follows:

{38, 17}
{38, 17, 46}

Table 20 – Solutions for item 38

The table above can be described in simplex as follows:

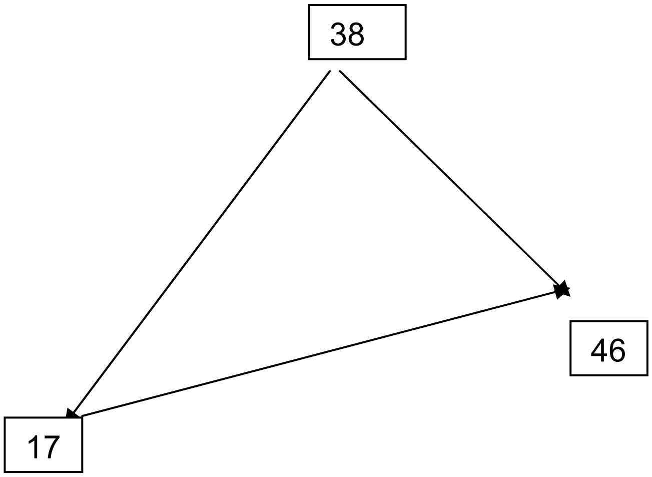

Figure 10 – Simplex diagram for item 38

For item 17, there is no valid solution.

For item 21, the solutions are as follows:

{21, 46}

Table 21 – Solution for item 21

The table above can be described in simplex as follows:

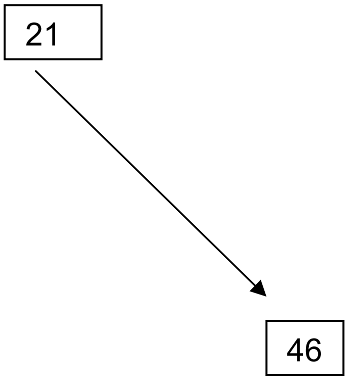

Figure 11 – Simplex diagram for item 21

For item 40, 46 and 10, there is no more solutions.

Summarize solutions for all items:

{5,19}, {5, 19, 38}, {5, 13}, {5, 7}, {5, 38}, {5, 40}, {5,10}
{7, 46}, {7, 46, 38}, {7, 46, 17}, {7, 46, 21}, {7, 46, 38, 17}, {7, 46, 38, 21}, {7, 13}, {7, 38}, {7, 17}, {7, 21}, {}
{19, 38}, {19, 38, 17}
{38, 17}, {38, 17, 46}
{21, 46}

Table 22 – Solutions for all items

We can use the following simplex diagram describe it:

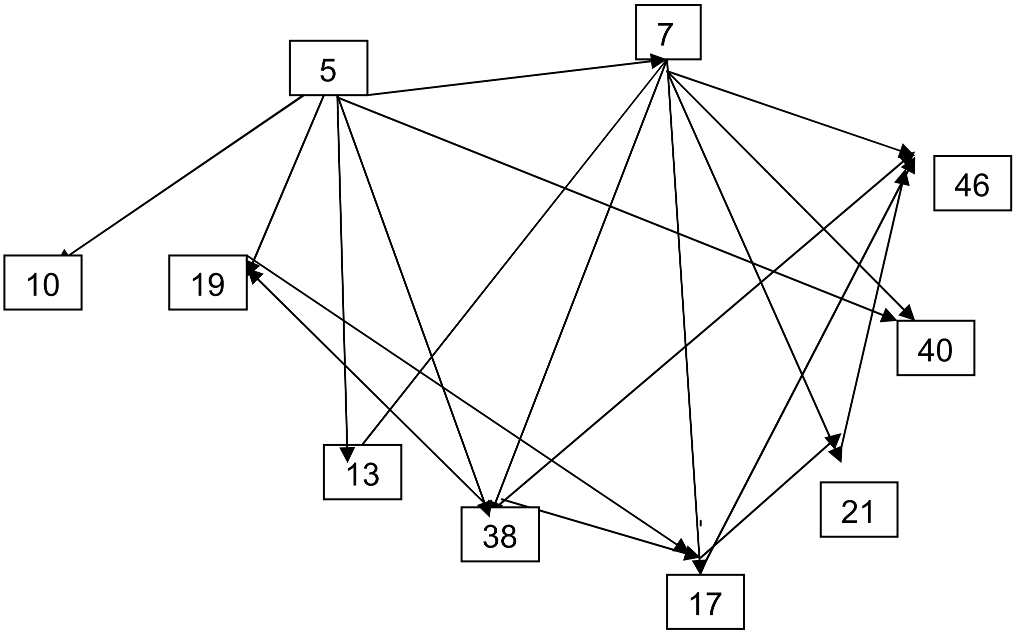

Figure 12 – Simplex diagram for all items

This simplex diagram is same as the one generated by bitmap FP-growth algorithm for this example. It helps to verify the accuracy of the simplex algorithm.

5 Experiment and results

I did an experiment by using the bitmap structure I created to do the association rule mining on 4 different computers parallel.

The original bitmap converted from the database has 123500 columns. My experiment calculated two items frequently sets and three items frequently sets with minimum support 10% and 5%. For two items, it took less than one second to complete the 10% calculating found 94 solutions and four seconds to complete 5% found 294 solutions. For three items, it took about 2 minutes for 10% found 94 solutions and about 7 minutes for 5% found 971 solutions. I also did the experiment to get four items frequently sets with 5% support. The experiment took about 18 minutes to complete and found 2114 solutions.

I have the solution sets result attached in Appendix.

6 Conclusion and Future work

In this paper, I combined the bitmap structure to improve the performance of association rule mining on database machines. I also created a simplex algorithm using the bitmap structure to achieve parallel computing. I proofed the association rule solutions are accurate as from FP-growth tree algorithm and Bitmap FP-growth algorithm. In the experiment, the performance of calculating association rule mining on simulated master – slave architecture database machines are impressive. In future, I plan to use the bitmap structure I implemented for different kinds of association rule mining algorithms to compare the performance of them and improve the structure if needed.

7 Reference

[1] Artificial neural network concept – wikipeida.org
http://en.wikipedia.org/wiki/Artificial_neural_network

[2] Ramakrishnan, R., Gehrke, J. (2002) Database Management Systems, 2nd Edition, Mc Graw Hall.

[3] Agrawal, R., Imielinski, T., Swami, A. (1993) Mining association rules between sets of items in large databases. *ACM SIGMOD Record*.

[4] Jia, D. (2011) Association Rule Mining – Geometry and Parallel Computing Approach. San Jose State University.

[5] Cheung, D., Han, J., Ng, V., Fu, A., Fu, Y. (1996) A fast distributed algorithm for mining association rules, International Conference on Parallel and Distributed Information Systems, Miami Beach, Florida.

8 Appendix – Running Result of two item frequent sets with 5% support

```
Thread hold = 6175

Start from: 21:43:23

index1=4, index2=13, amt=10524

index1=4, index2=23, amt=27498

index1=4, index2=29, amt=9413

index1=4, index2=39, amt=8396

index1=4, index2=151, amt=6720

index1=4, index2=161, amt=7894

index1=4, index2=318, amt=8492

index1=4, index2=410, amt=6733

index1=4, index2=421, amt=7907

index1=4, index2=553, amt=21726

index1=4, index2=574, amt=67866

index1=4, index2=614, amt=27086

index1=4, index2=624, amt=12744

index1=4, index2=625, amt=38109

index1=4, index2=633, amt=27547

index1=4, index2=634, amt=24779

index1=4, index2=691, amt=6657

index1=4, index2=692, amt=15625

index1=4, index2=756, amt=54172

index1=4, index2=762, amt=6472

index1=4, index2=775, amt=8696

index1=4, index2=789, amt=38490

index1=4, index2=815, amt=13743
```

```
index1=4, index2=819, amt=29667

index1=4, index2=823, amt=12422

index1=4, index2=826, amt=36784

index1=4, index2=1002, amt=51924

index1=4, index2=1003, amt=8039

index1=4, index2=1018, amt=65731

index1=4, index2=1019, amt=7092

index1=4, index2=1356, amt=33783

index1=4, index2=1363, amt=33057

index1=4, index2=1369, amt=33049

index1=4, index2=1375, amt=33056

index1=13, index2=23, amt=12834

index1=13, index2=574, amt=12333

index1=13, index2=625, amt=7740

index1=13, index2=756, amt=9110

index1=13, index2=789, amt=8472

index1=13, index2=819, amt=6967

index1=13, index2=826, amt=8695

index1=13, index2=1002, amt=8781

index1=13, index2=1018, amt=11249

index1=23, index2=553, amt=10214

index1=23, index2=574, amt=32896

index1=23, index2=614, amt=13387

index1=23, index2=625, amt=23716

index1=23, index2=633, amt=15268

index1=23, index2=692, amt=8485

index1=23, index2=756, amt=24194
```

```
index1=23, index2=789, amt=20041
index1=23, index2=819, amt=18853
index1=23, index2=826, amt=24039
index1=23, index2=1002, amt=23942
index1=23, index2=1018, amt=32130
index1=23, index2=1356, amt=14778
index1=23, index2=1363, amt=14296
index1=23, index2=1369, amt=14295
index1=23, index2=1375, amt=14295
index1=29, index2=574, amt=10699
index1=29, index2=634, amt=7701
index1=29, index2=756, amt=10394
index1=29, index2=1002, amt=8512
index1=29, index2=1018, amt=10085
index1=39, index2=574, amt=9309
index1=39, index2=634, amt=9914
index1=39, index2=756, amt=10547
index1=39, index2=1002, amt=7609
index1=39, index2=1018, amt=8796
index1=145, index2=405, amt=8054
index1=145, index2=574, amt=6703
index1=145, index2=1018, amt=6696
index1=151, index2=410, amt=9640
index1=151, index2=574, amt=7856
index1=151, index2=756, amt=6304
index1=151, index2=1002, amt=6261
index1=151, index2=1018, amt=7877
```

```
index1=161, index2=421, amt=11187

index1=161, index2=574, amt=9381

index1=161, index2=756, amt=7616

index1=161, index2=1002, amt=7119

index1=161, index2=1018, amt=8977

index1=318, index2=574, amt=9923

index1=318, index2=634, amt=11904

index1=318, index2=756, amt=12141

index1=318, index2=1002, amt=8616

index1=318, index2=1018, amt=9631

index1=405, index2=574, amt=6714

index1=405, index2=1018, amt=6701

index1=410, index2=574, amt=7873

index1=410, index2=756, amt=6298

index1=410, index2=1002, amt=6302

index1=410, index2=1018, amt=7911

index1=421, index2=574, amt=9412

index1=421, index2=756, amt=7619

index1=421, index2=1002, amt=7164

index1=421, index2=1018, amt=9025

index1=553, index2=574, amt=25651

index1=553, index2=614, amt=13974

index1=553, index2=625, amt=12595

index1=553, index2=633, amt=8100

index1=553, index2=634, amt=11502

index1=553, index2=692, amt=6737

index1=553, index2=756, amt=21143
```

```
index1=553, index2=789, amt=13669

index1=553, index2=819, amt=9695

index1=553, index2=826, amt=15149

index1=553, index2=1002, amt=26045

index1=553, index2=1018, amt=30837

index1=553, index2=1356, amt=17678

index1=553, index2=1363, amt=17132

index1=553, index2=1369, amt=17131

index1=553, index2=1375, amt=17131

index1=574, index2=609, amt=9257

index1=574, index2=614, amt=15467

index1=574, index2=624, amt=17291

index1=574, index2=625, amt=57110

index1=574, index2=633, amt=39223

index1=574, index2=634, amt=30604

index1=574, index2=691, amt=9854

index1=574, index2=692, amt=23042

index1=574, index2=755, amt=7050

index1=574, index2=756, amt=74294

index1=574, index2=762, amt=10736

index1=574, index2=775, amt=13481

index1=574, index2=789, amt=56869

index1=574, index2=815, amt=20800

index1=574, index2=819, amt=44348

index1=574, index2=823, amt=11802

index1=574, index2=826, amt=47240

index1=574, index2=1002, amt=63175
```

```
index1=574, index2=1003, amt=9993
index1=574, index2=1018, amt=79931
index1=574, index2=1019, amt=8166
index1=574, index2=1356, amt=41950
index1=574, index2=1363, amt=41061
index1=574, index2=1369, amt=41005
index1=574, index2=1375, amt=41042
index1=574, index2=1598, amt=6734
index1=574, index2=1604, amt=6734
index1=574, index2=1611, amt=6750
index1=575, index2=614, amt=6728
index1=609, index2=756, amt=7816
index1=609, index2=1018, amt=6912
index1=614, index2=634, amt=8921
index1=614, index2=756, amt=9016
index1=614, index2=826, amt=15835
index1=614, index2=1002, amt=26419
index1=614, index2=1018, amt=32001
index1=614, index2=1356, amt=16422
index1=614, index2=1363, amt=15987
index1=614, index2=1369, amt=15986
index1=614, index2=1375, amt=15986
index1=624, index2=634, amt=14767
index1=624, index2=756, amt=16265
index1=624, index2=789, amt=10393
index1=624, index2=815, amt=6339
index1=624, index2=1002, amt=11528
```

```
index1=624, index2=1018, amt=13632

index1=624, index2=1356, amt=8034

index1=624, index2=1363, amt=7863

index1=624, index2=1369, amt=7863

index1=624, index2=1375, amt=7864

index1=625, index2=633, amt=37140

index1=625, index2=691, amt=9154

index1=625, index2=692, amt=17132

index1=625, index2=755, amt=6727

index1=625, index2=756, amt=45949

index1=625, index2=762, amt=10010

index1=625, index2=775, amt=7589

index1=625, index2=789, amt=40557

index1=625, index2=815, amt=10789

index1=625, index2=819, amt=42329

index1=625, index2=826, amt=39968

index1=625, index2=1002, amt=34560

index1=625, index2=1003, amt=6489

index1=625, index2=1018, amt=45842

index1=625, index2=1356, amt=22419

index1=625, index2=1363, amt=21962

index1=625, index2=1369, amt=21964

index1=625, index2=1375, amt=21964

index1=633, index2=691, amt=8540

index1=633, index2=692, amt=8820

index1=633, index2=756, amt=30231

index1=633, index2=762, amt=6729
```

```
index1=633, index2=789, amt=26682

index1=633, index2=815, amt=8578

index1=633, index2=819, amt=28017

index1=633, index2=826, amt=27210

index1=633, index2=1002, amt=23733

index1=633, index2=1018, amt=31557

index1=633, index2=1356, amt=15619

index1=633, index2=1363, amt=15390

index1=633, index2=1369, amt=15390

index1=633, index2=1375, amt=15390

index1=634, index2=692, amt=10247

index1=634, index2=756, amt=36121

index1=634, index2=789, amt=16832

index1=634, index2=815, amt=8390

index1=634, index2=1002, amt=24464

index1=634, index2=1018, amt=28628

index1=634, index2=1356, amt=15384

index1=634, index2=1363, amt=14859

index1=634, index2=1369, amt=14859

index1=634, index2=1375, amt=14859

index1=691, index2=1018, amt=7832

index1=692, index2=756, amt=23027

index1=692, index2=789, amt=16696

index1=692, index2=819, amt=13260

index1=692, index2=826, amt=12914

index1=692, index2=1002, amt=15156

index1=692, index2=1018, amt=19523
```

```
index1=692, index2=1356, amt=9961

index1=692, index2=1363, amt=9619

index1=692, index2=1369, amt=9619

index1=692, index2=1375, amt=9619

index1=756, index2=762, amt=8406

index1=756, index2=775, amt=10251

index1=756, index2=789, amt=48951

index1=756, index2=815, amt=16003

index1=756, index2=819, amt=36155

index1=756, index2=823, amt=8897

index1=756, index2=826, amt=33485

index1=756, index2=1002, amt=51031

index1=756, index2=1003, amt=7280

index1=756, index2=1018, amt=64008

index1=756, index2=1356, amt=33109

index1=756, index2=1363, amt=32285

index1=756, index2=1369, amt=32287

index1=756, index2=1375, amt=32287

index1=762, index2=789, amt=7274

index1=762, index2=819, amt=7664

index1=762, index2=826, amt=7671

index1=762, index2=1002, amt=7176

index1=762, index2=1018, amt=9113

index1=775, index2=815, amt=11661

index1=775, index2=1002, amt=7572

index1=775, index2=1018, amt=10177

index1=789, index2=819, amt=42225
```

```
index1=789, index2=826, amt=33534
index1=789, index2=1002, amt=36804
index1=789, index2=1018, amt=45795
index1=789, index2=1356, amt=22734
index1=789, index2=1363, amt=22521
index1=789, index2=1369, amt=22521
index1=789, index2=1375, amt=22521
index1=815, index2=1002, amt=11578
index1=815, index2=1018, amt=15659
index1=815, index2=1356, amt=8940
index1=815, index2=1363, amt=8686
index1=815, index2=1369, amt=8686
index1=815, index2=1375, amt=8687
index1=819, index2=826, amt=34454
index1=819, index2=1002, amt=26902
index1=819, index2=1018, amt=35305
index1=819, index2=1356, amt=16725
index1=819, index2=1363, amt=16621
index1=819, index2=1369, amt=16623
index1=819, index2=1375, amt=16623
index1=823, index2=1002, amt=7539
index1=823, index2=1018, amt=11186
index1=823, index2=1356, amt=6830
index1=823, index2=1363, amt=6559
index1=823, index2=1369, amt=6559
index1=823, index2=1375, amt=6559
index1=826, index2=1002, amt=35604
```

```
index1=826, index2=1003, amt=6505

index1=826, index2=1018, amt=45895

index1=826, index2=1356, amt=21465

index1=826, index2=1363, amt=21142

index1=826, index2=1369, amt=21141

index1=826, index2=1375, amt=21141

index1=1002, index2=1018, amt=76913

index1=1002, index2=1356, amt=41484

index1=1002, index2=1363, amt=40643

index1=1002, index2=1369, amt=40643

index1=1002, index2=1375, amt=40644

index1=1003, index2=1018, amt=9351

index1=1018, index2=1356, amt=47464

index1=1018, index2=1363, amt=46326

index1=1018, index2=1369, amt=46303

index1=1018, index2=1375, amt=46317

index1=1018, index2=1598, amt=6586

index1=1018, index2=1604, amt=6585

index1=1018, index2=1611, amt=6599

index1=1356, index2=1363, amt=49565

index1=1356, index2=1369, amt=49564

index1=1356, index2=1375, amt=49564

index1=1363, index2=1369, amt=49656

index1=1363, index2=1375, amt=49693

index1=1369, index2=1375, amt=49658

index1=1598, index2=1604, amt=8283

index1=1598, index2=1611, amt=8278
```

```
index1=1604, index2=1611, amt=8280

End at: 21:43:27
```