An Algorithm for Finding Frequent Itemset based on Lattice Approach for Lower Cardinality Dense and Sparse Dataset

Ajay Acharya
CSE,SRIT
Jabalpur, M.P.[INDIA]

Shweta Modi
HOD,CSE,SRIT
Jabalpur, M.P.[INDIA]

Abstract

Whenever mining association rules work for large data sets frequently itemset always play an important role and enhance the performance. Apriori algorithm is widely used for mining association rule which uses frequent item set but its performance can be improved by enhancing the performance of frequent itemsets. This paper proposes a new novel approach to finding frequent itemsets. The approach reduces a number of passes through an input data set in this paper from the study of data mining technology An Algorithm for Finding Frequent Itemset based on Lattice Approach for Lower Cardinality Dense and Sparse Dataset developed, by making variation in Apriori which improves performance over Apriori for lower cardinality. It does not follow generation of candidate-and-test method. It also reduces the scanning of database and needs only two scanning of database. The paper presents the results of experiments conducted to find how performance of association rule mining algorithm depends on the values of parameters i.e. number of transaction, cardinality and minimum support.

Key Words - Data Mining, Apriori Algorithm, Lattice.

1. Introduction

With the development of information technology, there are many different kinds of information databases, such as scientific data, medical data, financial data, and marketing transaction data. How to effectively analyze and apply these data and find the critical hidden information from these databases have become very important issues. Data mining technique has been the most widely discussed and frequently applied tool in recent decades. Data mining has been successfully applied in the areas of scientific analysis, business application and medical research. Not only are its applications getting broader, but its computational efficiency and accuracy are also improving. Data mining can be categorized into several models, including association rules, clustering and classification. Among these models, association rule mining is the most widely applied method. The Apriori algorithm is the most representative algorithm. It consists of many modified algorithms that focus on improving its efficiency and accuracy. However, two parameters minimal sup-port and confidence are always determined by the decision-maker him/herself or through trial-and-error and thus the algorithm lacks both objectiveness and efficiency. Therefore, the main purpose of this study is to propose an improved algorithm that can provide feasible threshold values for minimal support and confidence. The simulation results show that the proposed algorithm has better computational performance than Apriori Algorithm. Moreover, a real-world data provided by a well-known security firm also indicate that the proposed algorithm can mine the relationship between investors transaction behavior in different industrial categories.[5][15]

Association Rule Mining

Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.[09][10]

Applications

Market Basket Data Analysis, Cross-Marketing, Catalog Design, Loss-Leader Analysis, Clustering, Classification, etc.

Examples

Rule form: "Body $\rightarrow$ Head [support, confidence]".
buys(x, "diapers") $\rightarrow$ buys(x, "beers") [0.5%, 60%]
major(x, "CS") ^ takes(x, "DB") $\rightarrow$ grade(x, "A") [1%, 75%]

Association Rule Mining– Problem Description

The formal description of association rule mining is largely based on the description of the problem. Formally, the problem can ie stated as follows: Let Z = $\{i_l, i_2, . . . , i_m\}$ be a set of m. distinct literals called items. D is a set of variable length transactions over I. Each transaction contains a set of items i_i, i_j,….$i_k \subset$ 1. A transaction also has an associated unique identifier called TID. An association rule is an implication of the form

X=>Y, where X, Y $\subset$ I, and X $\cap$ Y = φ. X is called the antecedent and Y is called the consequent of the rule. [6][11]

There are two basic measures for association rules support(s) and confidence(c). Each item set has an associated measure of statistical significance called support. A rule has a measure of its strength called confidence.

The X=>Y holds in transaction set D with **support s**, where s is the percentage of transaction in D that contain X U Y (i.e., both X and Y). This is taken to be probability, P (X U Y). Support(s) = support (X=>Y) = P(X U Y)

Thus, support(s) of an association rule is defined as the percentage / fraction of records that contain X U Y to the total number of records in the database.

The rule X=>Y has **confidence c** in the transaction set D if c is the percentage of transaction in D containing X that also contain Y. This is taken to be the conditional probability, P(Y\X). That is

Confidence(c) = confidence(X=>Y) =
P(Y\X) = support_count(X U Y) / support_count(X)

In other words, confidence of an association rule is defined as the percentage / fraction of number of transactions that contain X U Y to the total number of records that contains X, where if the percentage exceeds the threshold of confidence association rule X=>Y can be generated. [1][2]

Problem Decomposition: The problem of mining association rules is to generate all rules that have support and confidence greater than some user specified minimum support and minimum confidence thresholds, respectively. This problem can be decomposed into the following sub-problems:

i. All item sets that have support above the user specified minimum supports are generated. These itemset are called the frequent itemsets or large itemset.
ii. For each large itemsets, all the rules that have minimum confidence are generated as follows: for a large itemset X and any Y$\subset$ X, if support(X)/support(X - Y) $\geq$ minimum- confidence, then the rule X - Y => Y is a valid rule.[17]

Dense & Sparse Data Set

Dense Data A dataset is dense if it contain many long pattern even those the support threshold is relatively high.

Sparse Data A dataset is sparse if it contain rare long pattern even those the support threshold is relatively low.

Finding Associations in Dense Data Sets

Since association rule algorithms work by iterative enumeration, they work best for sparse data sets, that is, data sets where each record contains only a small fraction of the total number of possible items (if the total number of items is very large). Algorithm performance degrades exponentially with increasing number of frequent items per record. Therefore, to get good runtime performance, one of the following conditions should hold

- If the data set is dense, the number of possible items is small.
- If the number of possible items is large, the data set is sparse.
- The data set becomes progressively sparser with increasing item set length due to the application of the minimum support threshold.

The last condition holds for higher minimum support values. Typical data sets in many bioinformatics applications are dense with large number of attributes. In order to use association rules effectively for such problems, careful planning is required. One option is to start with a high minimum support threshold and repeat it for lower values until desirable results are obtained. Another option is to recode some of the uninteresting attribute values to NULL, if such recoding makes the data set sparse.[17]

The Apriori Algorithm

Apriori was proposed by Agrawal and Srikant in 1994. It is also called the level-wise algorithm. It is the most popular and influent algorithm to find all the frequent sets. It makes the use of downward closure property. As the name suggests, the algorithm is a bottom-up search, moving upward level-wise in the lattice.[1][2]

First the set of frequent1-itemset is found, this set is denoted L1. L1 is used to find L2, the set of frequent 2- itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found.

The finding of each L_k requires one full scan of the database. This algorithm uses prior knowledge of frequent item set properties. It is an iterative approach where K item sets used to explore K+1 itemsets.

Apriori Candidate Generation - Monotonically Property, All subsets of a frequent set are frequent Given L_{k-1}, C_k can be generated in two steps [5][11]

i. **The Join Step** Join L_{k-1} with L_{k-1}, with the join condition that the first k-1 items should be the same i.e. members l_1 and l_2 of L_{k-1} are joined if $(I_1 [1]= I_2 [1]) \wedge \ldots (I_1 [k- 2])]= I_2 [k-2])\wedge(I_1 [k-1]< I_2 [k-1])$.
ii. **The Prune Step** The pruning step eliminates the extensions of (k-1)- itemsets which are not found to be frequent, from being considered for counting support.

It is the basic algorithm; many variations of the Apriori algorithm have been proposed that focus on improving efficiency of the original algorithm. These variations are Hash-based technique, Partitioning, Sampling and Dynamic itemset counting etc.

Apriori algorithm shows good performance with sparse datasets such as market basket data, where the frequent patterns are very short. However, with the dense datasets such as telecommunications, web log data and census data, where there are many, long frequent patterns.[7] The performance of this algorithm degrades incredibly. This degradation is due to the following reasons:

- ❖ Algorithm performs as many passes over the database as the length of the longest frequent pattern. This creates high I/O overhead for scanning large disk-resident databases many times.
- ❖ It is computationally expensive to check a large set of candidates by pattern matching. [14]

The characters of Apriori Algorithm

1. Apriori algorithm is an iterative algorithm for the first excavation generated L_1, and then by the L_1 generation C_2, from C_2 scan Affairs database by L_2; again L_2 generated by the C_3, from C_3 scanning services database be L_3, until all have C_K is empty Frequent Itemsets.
2. data by the level of organization, the so-called level of organization that is in accordance with the data (TID, IS), that is, # (Services, the project sets) This form of organization.
3. Apriori optimizes the use of the method, the called Apriori is to optimize the use of Apriori nature of the optimization.
4. Service database for the mining association rules.
5. For sparse data sets, based on previous studies, this method can only sparse data sets for the mining association rules, which is frequently set the length of the project smaller data sets. [13]

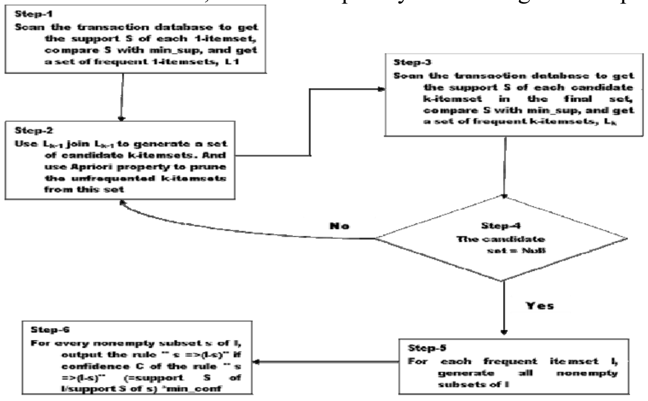

Step-1
Scan the transaction database to get the support S of each 1-itemset, compare S with min_sup, and get a set of frequent 1-itemsets, L1
Step-2
Use L_{k-1} join L_{k-1} to generate a set of candidate k-itemsets. And use Apriori property to prune the unfrequented k-itemsets from this set
Step-3
Scan the transaction database to get the support S of each candidate k-itemset in the final set, compare S with min_sup, and get a set of frequent k-itemsets, L_k
Step-4
The candidate set = Null
No
Yes
Step-5
For each frequent itemset l, generate all nonempty subsets of l
Step-6
For every nonempty subset s of l, output the rule "s =>(l-s)" if confidence C of the rule "s =>(l-s)" (=support S of l/support S of s) "min_conf

Fig.I. Flow Chart of Apriori Algorithm

2. Previous Work

All classical methods [1,2] adopt the downward closure property of frequent itemset with respect to set inclusion. It is proved that **if an itemset is frequent than all its subsets must be frequent**. All classical methods are based on this property. The recent algorithm are **variation/modification in apriori [4]** in this paper a new method towards automatic personalized recommendation based on the behavior of a single user in accordance with all other users in web-based information systems is introduced. The proposal applies a modified version of the well-known Apriori data mining algorithm to the log files of a web site (primarily, an e-commerce or an e-learning site) to help the users to the selection of the best user-tailored links. The paper mainly analyzes the process of discovering association rules in this kind of big repositories and of transforming them into user-adapted recommendations by the two-step modified Apriori technique, which may be described as follows. A first pass of the modified Apriori algorithm verifies the existence of association rules in order to obtain a new repository of transactions that reflect the observed rules. A second pass of the proposed Apriori mechanism aims in discovering the rules that are really inter-associated. This way the behavior of a user is not determined by ''what he does'' but by ''how he does''. Furthermore, an efficient implementation has been performed to obtain results in real-time. As soon as a user closes his session in the web system, all data are recalculated to take the recent interaction into account for the next recommendations. Early results have shown that it is possible to run this model in web sites of medium size. And based on lattice approach **[3]** Data mining has recently attracted considerable attention from database practitioners and researches because of its applicability in many areas, such as decision supports, market strategy and financial forecasts combing techniques from the fields of machine learning, statistics and database, data mining enables us to find out useful and invaluable information from huge database. Mining of association rules has received much attention among the various data mining problems. Most algorithms for association rule mining are the variants of the basic apriori algorithm. One characteristic of this apriori-based algorithm is that candidate itemsets are generated in rounds, with the size of the itemsets incremented by one per round. The number of database scan required by apriori based algorithms thus depends on the size of the largest itemsets. In this research we proposed a new candidate set generation algorithm, which generates candidate itemsets of multiple sizes at each iteration by taking input as suggested large itemsets.

3. Proposed Work

In this paper propose an algorithms (variation in apriori) based on lattices approach. Our aim is to develop an algorithm for Finding Frequent Itemset based on Lattice Approach for Lower Cardinality Dataset which gives the improved performance over a priori.[12]

Association Rule Mining is generally performed in two steps:

- ❑ Generation of frequent item sets / Large item sets
- ❑ Rule generation

Apriori algorithm is based on the candidate generation- and-test method. In many cases it reduces the size of candidate set significantly and leads to good performance gain. However, it may suffer from two non-trivial costs.

1. It may need to generate huge number of candidate sets.
2. It may need to repeatedly scan the database and check a large set of candidates by pattern matching.

In this paper propose an algorithm which is not on the based on candidate generation-and-test method. It is based on Lattices / Partial Ordered Set and used Upward Closure Property.**[8]**

It reduces the computation cost of candidate set generation. It also tries to overcome the problem of pruning of candidate set at each level. Therefore algorithm work efficiently and provide good result. Which give the improved performance over apriori for lower cardinality dense & sparse dataset.

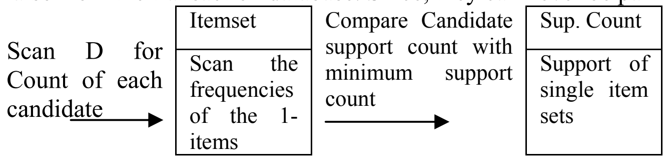
The steps of Proposed "Algorithm"

1. In initial database scan the frequencies of the 1-items are determined and discard the infrequent item set.
2. Find the cardinality k (number of items) for 1-temsets. And generate the maximum large itemset for carnality k.
3. Generate **all possible POSETs of Maximum Large** Itemset for cardinality k.
4. In second database scan for each transaction, search in POSETs and increment the counter by 1 for found itemset.
5. Frequent itemset can get by selecting the item which satisfied the minimum support.

The algorithm works as follows

Step-I In initial scan the frequencies of the 1-items (support of single item sets) are determined. All infrequent itemsets – that are all items that appear in fewer transactions than a user specified minimum number are discarded from i-items and

also from the transaction database. Since, they can never be part of the frequent itemset.


Scan D for Count of each candidate
Itemset
Scan the frequencies of the 1-items
Compare Candidate support count with minimum support count
Sup. Count
Support of single item sets

Find the cardinality K (number of items) for 1-itemset. And find the Maximum Large Itemset for carnality k.[18]

General concept

The diagram of the poset ℘(L) for L = {A,B,C,D}A line connecting a lower node to an upper node means the lower node is ⊆ the upper. Note that not all sets are ordered by ⊆.[8]

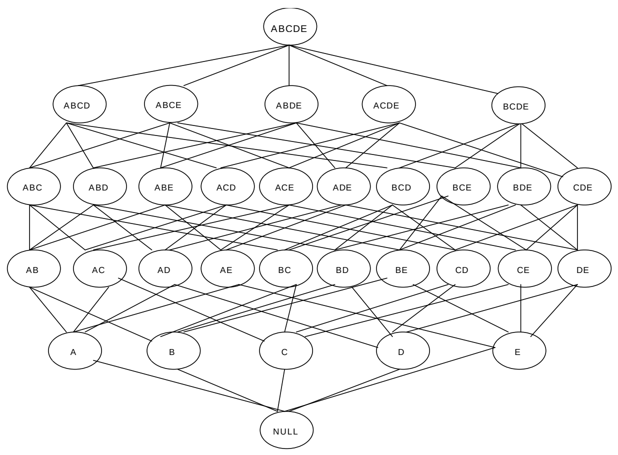


Fig.II Diagram of POSET

Step-II Generate Itemset L = all POSETs of Max. Large Itemset for cardinality k that are singles, doublets, triplets, ….k-itemsets and these are kept in lexicographical ascending order. The maximum number of itemsets will be $2^K - 1$ where k is the cardinality (number of items in the 1-itemsets).

	Itemset		Itemset	Sup. Count		Itemset	Sup. Count
Generate C_2 candidates from L_1 $\longrightarrow$	All POSets of Max. Large Itemset	Scan D for Count of each candidate $\longrightarrow$	For each transaction, search in L and	Increment the counter of found itemset ∀ I ∈ L by 1.	Compare candidate support count with minimum support count $\longrightarrow$	Frequent itemset = Pruned itemset L	L by user Specified minimum support.

Generate Frequent Item = Pruned itemset L by user specified minimum support.[18]

4. Experimental Results

We studied the performance of our implementation on four transaction datasets created randomly. The details of these datasets are given in Table I. We used 10 distinct items and a maximum of 8 items in each. Transaction for all the datasets. We created our own datasets due to the fact that there are no benchmark data available for this problem. The experiments were performed on a Pentium 4, 3.2 GHz system running Windows XP Professional with 1 GB of main memory. The preprocessing, algorithm and other interface are coded in ASP.NET & C#. All the experiments were on the synthetic student database. Fig.III shows the running time comparison between the Apriori algorithm and the New Proposed algorithm for the Dataset 1. It can be seen that the New Proposed Algorithm always outperforms the Apriori for all values of minimum support. This was the case in all the four datasets tested.

TABLE- I. TEST DATASETS

Datasets	Number of Transactions	Number of Item
Dataset 1	250	We used 10 distinct items and a maximum of 8 items & minimum of 4 item in each
Dataset 2	500	
Dataset 3	750	
Dataset 4	1000	

Fig.III. shows the running time comparison between the Apriori & New Proposed Algorithm(Modify Apriori).
Fig.IV. shows the Result of total execution time between the Apriori & New Proposed Algorithm(Modify Apriori).

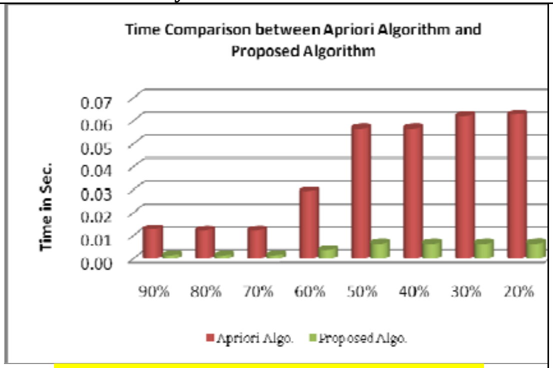
In order to evaluate the performance of New Proposed Algorithm, we have tested various dataset take FIMI dataset[16] and synthetic dataset and compared them with Apriori, we used our own implementation to compare with new proposed algorithm.

Comparative Study of- Total Execution Time Comparison between Apriori & New Proposed Algorithm(Modify Apriori)

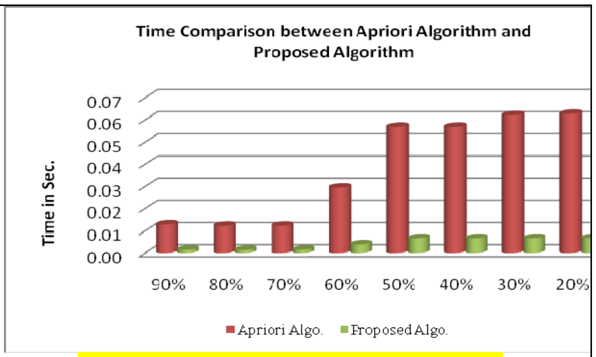
TRAN	ITEMS	MINSUPP	AprioriTIME	LowerCostTIME
250	4	90	00:00:00.8583839	00:00:00.0971634
250	4	80	00:00:00.8344610	00:00:00.0887040
250	4	70	00:00:00.8332190	00:00:00.0974176
250	4	60	00:00:01.5723096	00:00:00.1717508
250	4	50	00:00:02.8049866	00:00:00.2805521
250	4	40	00:00:02.8005400	00:00:00.2802420
250	4	30	00:00:02.8413259	00:00:00.2817394
250	4	20	00:00:02.8371116	00:00:00.2848498
250	5	90	00:00:01.1111082	00:00:00.1242395
250	5	80	00:00:01.0576081	00:00:00.1183781
250	5	70	00:00:01.0590790	00:00:00.1271510
250	5	60	00:00:02.5443060	00:00:00.3303872
250	5	50	00:00:04.9081992	00:00:00.5738849
250	5	40	00:00:04.9088023	00:00:00.5711005
250	5	30	00:00:05.3672704	00:00:00.5716061
250	5	20	00:00:05.4221575	00:00:00.5788417
250	6	90	00:00:01.3283059	00:00:00.1573605
250	6	80	00:00:01.3043012	00:00:00.1483258
250	6	70	00:00:01.2814005	00:00:00.1586050
250	6	60	00:00:05.8690420	00:00:00.7021177
250	6	50	00:00:06.5951996	00:00:01.3828494
250	6	40	00:00:08.6219329	00:00:01.3780159
250	6	30	00:00:09.0993444	00:00:01.3783645
250	6	20	00:00:09.1281076	00:00:01.3833730
250	7	90	00:00:01.5255181	00:00:00.1879562
250	7	80	00:00:01.4946430	00:00:00.1820259
250	7	70	00:00:01.4963580	00:00:00.1942461
250	7	60	00:00:06.0164129	00:00:01.6037236
250	7	50	00:00:11.0938353	00:00:03.4414440
250	7	40	00:00:12.2548234	00:00:03.4938013
250	7	30	00:00:12.4228177	00:00:03.4652457
250	7	20	00:00:12.4365769	00:00:03.4579375
250	8	90	00:00:01.7618987	00:00:00.2314604
250	8	80	00:00:01.7627843	00:00:00.2174709
250	8	70	00:00:01.8290525	00:00:00.2760607
250	8	60	00:00:09.6121146	00:00:04.5058743
250	8	50	00:00:18.7392593	00:00:11.4763284
250	8	40	00:00:18.3467932	00:00:11.4674334
250	8	30	00:00:19.2165976	00:00:11.5634892
250	8	20	00:00:19.2252761	00:00:11.4915828
250	9	90	00:00:01.9564777	00:00:00.2605339
250	9	80	00:00:01.9613422	00:00:00.2512241
250	9	70	00:00:02.0320607	00:00:00.3164150
250	9	60	00:00:09.8314241	00:00:04.5587712
250	9	50	00:00:27.5989331	00:00:33.2855910
250	9	40	00:00:27.4808898	00:00:33.1931767
250	9	30	00:00:30.3239062	00:00:33.1359960
250	9	20	00:00:30.3366553	00:00:33.4602707

Fig. III. Comparative Study on the Synthetic Dataset

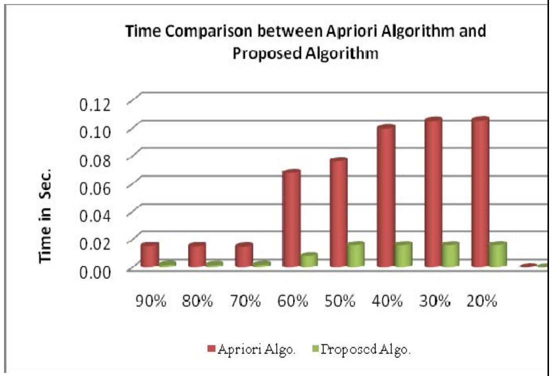
Results on the Synthetic Dataset



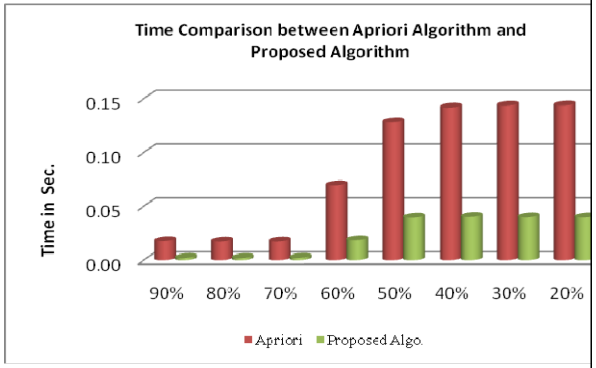
(Max. 4 items,250Trn. Min.Support 90% to 20%).



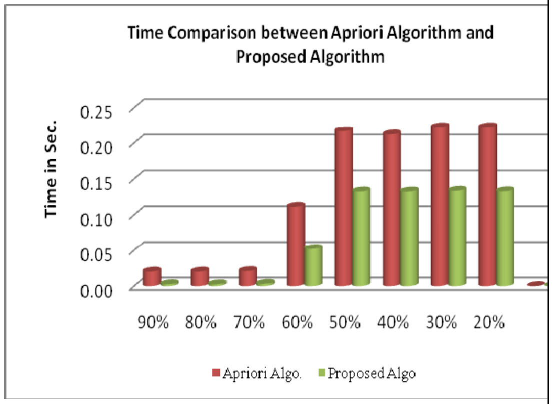
(Max. 5 items,250Trn. Min.Support 90% to 20%)



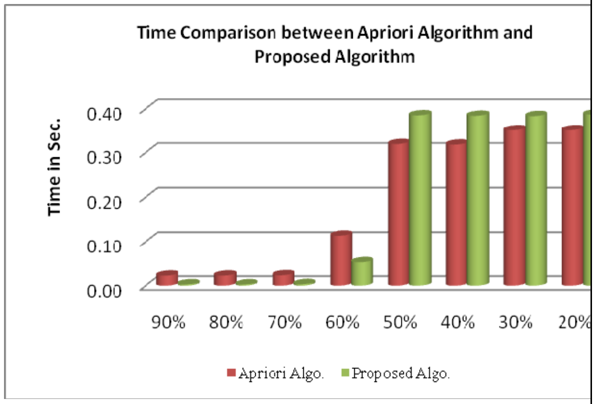
(Max. 6 items,250Trn. Min.Support 90% to 20%).



(Max. 7 items,250Trn. Min.Support 90% to 20%).



(Max. 8 items,250Trn. Min.Support 90% to 20%).



(Max. 9 items,250Trn. Min.Support 90% to 20%).

Fig. IV. Results on the Synthetic Datase

5. Conclusion and Future Work

In this research work on the basis of experimental result and its observation shows that the Classical Apriori Algorithm require multiple database scan i.e. 1-item, 2-item, 3-item,……n-items, but New Proposed modified Algorithm requires only two database scan first for 1-item and second for all the POSETs. The efficiency of Apriori is independent of cardinality, but the efficiency of proposed algorithm depends on the cardinality in case of lower cardinality it provides better result for both sparse and dense dataset and good performance then apriori, as the cardinality of the dataset increases performance of the proposed algorithm will degrade specially for dense dataset. The classical apriori algorithm suffers due to implementation complexity for JOIN & PRUNING, but new proposed algorithm does not follow generation of candidate-and-test method so it is free from join & prune process.

In future, would like to work on an algorithm for discovers the hidden relationship for higher cardinality dense dataset based on lattice approach.

6. Reference

[1] R. Agrawal, T. Imielinski, and A. Swami. “Mining association rules between sets of items in large databases.” in Proceedings of the International Conference on Management of Data ACM 1993 SIGMOD Washington DC, May 26- 28 1993, pages 207-216.

[2] R. Agrawal and R.Shrikanth.“Fast Algorithm for Mining Association Rules.” In Proceeding of VLBD Conference, Santigo, Chile- 1994,pp 487-494.

[3] Sanjeev Sharma, Akhilesh Tiwari and K.R.Pardasani. “Design of Algorithm for Frequent Pattern Discovery Using Lattice Approach.”, Asian Journal of Information Management 1(1): 11-18,2007

[4] Enrique Lazcrorreta, Federico Botella, Antonio Fernandez-Caballero. “Towards personalized recommendation by Two-step modified Apriori data mining algorithm.” ScienceDirect Expert Systems ELSEVIER with Applications 35,1422-1429 (2008)

[5] Goswami D.N.Chaturvedi and Anshu Raghuvanshi. “An Algorithm for Frequent Pattern Mining Based on Apriori.” (IJCSE) International Journal on Computer Science and Engineering Vol.02, No. 04, 942-947,2010

[6] Y. Wang, Y. He and J. Han. “Mining Frequent Item Sets Using Support Constraints.” in Proc. IC- VLDB’00 Egypt Sep.2000, Page 43-52.

[7] Ashok Savasere, Edward Omiecinski, and Shamkant Navathe.“An Efficient Algorithm for Mining Association Rules in Large Database.” In Proceedings of the 21st Conference Zurich, Swizerland, 1995.

[8] Zinaida Trybulec, Properties of Subsets, Journal of Formalized Mathematics Inst. of Computer Science, Univ. of Białystok , Volume 1, Released 1989, Published 2003

[9] J. Han and M. Kamber. Data Mining: Concepts and techniques, Morgan Kaufmann Publishers, Elsevier India, 2001, pp- 225-235.

[10] A.K. Pujari. Data Mining Techniques, University Press 2001, pp -69-81.

[11] Pratima Gautam, K. R. Pardasani. “A Fast Algorithm for Mining Multilevel Association Rule Based on Boolean Matrix.” (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 03, 746-752,2010

[12] K.K. Loo*, Chi Lap Yip, Ben Kao, David Cheung. “A lattice-based approach for I/O efficient association rule mining.”, Information Systems 27 (2002), 41–74, 2002

[13] HAN Feng, ZHANG Shu-mao, DU Ying-shuang. “The analysis and improvement of Apriori algorithm.” Journal of Communication and Computer, USA Volume 5, No.9 (Serial No.46), Sep2008

[14] S.Shankar,T.Purusothaman. “Utility Sentient Frequent Itemset Mining and Association Rule Mining: A Literature Survey and Comparative Study”, International Journal of Soft Computing Applications, Issue 4 (2009), pp.81-95,2009.

[15] Viktor Jovanoski and Nada Lavrac, J Stefan, “Classification Rule Learning with APRIORI C”, Institute Jamo-39,1000, Ljubljana Slovenia

[16] FIMI Dataset – http://fimi.cs.helsinki.fi/

[17] N. Badal, Shruti Tripathi. “Frequent Data Itemset Mining Using VS_Apriori Algorithms.”, (IJCSE) International Journal on Computer Science and Engineering, Vol. 02, No. 04, pp1111-1118,2010.

[18] Ajay Acharya, Shweta Modi , Vivek Badhe, “An Algorithm for Finding Frequent Itemset based on Lattice Approach for Lower Cardinality Dataset”, International Journal of Mathematical Archive 1(1), pp16-19 Oct.-2010,