Middle-East Journal of Scientific Research 23 (Sensing, Signal Processing and Security): 222-231, 2015
ISSN 1990-9233
© IDOSI Publications, 2015
DOI: 10.5829/idosi.mejsr.2015.23.ssps.57

Frequent Itemsets Generation from Transaction Database and Rule Generation using Quine-McCluskey Method

[1]*M. Krishnamurthy,* [2]*K. Manivannan,* [3]*A. Chilambuchelvan,* [4]*E. Rajalakshmi and* [5]*A. Kannan*

[1]Professor in CSE, KCG College of Technology, Chennai, Tamil Nadu, India
[2]Professor in IT, RMK Engineering College Kavaraipettai, Tamil Nadu, India
[3]Professor & Head RMK Engineering College Kavaraipettai, Tamil Nadu, India
[3]SriVenkateswara College of Engineering, Sriperumbudur, Tamil Nadu, India
[4]Anna University, Chennai, Tamil Nadu, India

Abstract: Data mining is the process of extracting useful information from databases. Finding frequent itemsets is one of the most investigated fields of data mining. The significant feature is to find new techniques to reduce candidate itemsets in order to generate frequent itemsets efficiently. It is a challenging task and generating rules in order to identify the purchase behaviour of the customer to progress the business. In this paper, an efficient and optimized algorithm called Customer Purchase Behaviour (CPB) has been introduced to find frequent itemsets using least scans, time and memory and the rules are generated. The subset discovery method is used for the generation of frequent itemsets which reduces the intermediate tables. This approach diminishes main memory constraint because it considers only a small cluster in given period of time. The purchase behaviour of the customer can be easily evaluated by using the Quine-McCluskey method. This algorithm is very proficient because of redundancy removal and rule generation compared with Apriori, Cluster-Based Bit Vector Mining for Association Rule Generation (CBVAR) and Improved Cluster-based Bit vector mining algorithm for Frequent Itemsets Generation (ICBV). Thus the proposed algorithm reduces the scanning time, processing time and the storage space respectively.

Key words: Cluster based Mining · Bit Vector Mining · Redundancy Removal · Frequent Itemsets Generation · Customer Purchase Behaviour · Quine-McCluskey

INTRODUCTION

Computerized data gathering tools and grown-up database technology lead to amazing amount of data store in databases, data warehouses and other information repositories. The absolute answer for knowledge novelty can be given by data mining technology. Data mining is the process of extraction of interesting information or patterns from data in large databases. Data mining tools predict behaviours and future trends, allowing businesses to make proactive, knowledge-driven decisions. Frequent itemsets mining is a popular and important field in data mining and it plays an essential role in many important data mining tasks. Frequent patterns are itemsets that appear in a dataset with frequency greater than or equal to a user-specified threshold called support. If the support-count of an itemset I satisfies the minimum support threshold, then the itemset I is a frequent itemset (Jiawei 2009).

Related Work: There are many works in the literature that discuss about Association rules, mining and Frequent Itemsets. The Association Rule mining raised by R.Agarwal (1993) is an important research in data mining field [1]. His Apriori algorithm can discover meaningful itemsets and build association rules. Different strategies were proposed after Apriori as in FP-growth (2000), which outperforms all candidate generations but still have problems in the case of no common prefixes within the data items [2]. Jie Dong *et al.,* (2007) address an algorithm

Corresponding Author: M. Krishnamurthy, Professor in CSE, KCG College of Technology, Chennai, Tamil Nadu, India.
E-mail: mkrish@kcgcollege.com.

which uses a special data structure called BitTable for compressing the database and for the candidate itemsets generation [3]. M. Krishnamurthy *et al.,* (2011) proposed a new association rule mining algorithm called Hash Based Frequent Itemsets-Quadratic Probing in which hashing technology has been proposed and implemented to store the database in vertical data format [4]. Saquer *et al*., (2010) presented a new approach for generating representative association rules that use only a subset of the set of frequent itemsets called frequent closed itemsets [5].

M. Krishnamurthy *et al.,* (2011) a new algorithm called Temporal Pattern Mining has been proposed to find the frequent temporal pattern based on Clustering, Bit Vector and Variable Threshold [6]. M. Krishnamurthy *et al*, (2010), an improved algorithm is proposed for generating frequent k-itemsets [7]. The improvement is mainly to reduce query frequencies and storage resources. M. Krishnamurthy *et al.,* (2009) proposed a work called mining frequent itemsets using temporal association rule in which the algorithm discovers all possible temporal association rules with necessary support and confidence from a set of time stamped transactions [8]. M. Krishnamurthy *et al.,* (2009) proposed a work which incorporates flexible threshold during mining and it accomplishes two goals: it saves a lot of computations in the post-processing phase and it suits the interactive need of the users [9]. CBVAR by M. Krishnamurthy *et al.,* (2011) for frequent itemsets generation uses single scan towards the database [10, 11].

Jayalakshmi, N. *et al.,* (2012) given an algorithm called DHBFI [12]. It takes the database in the vertical format and applies double hashing technique to reduce the storage space and to avoid collision and primary as well as secondary clustering problem. ICBV by E. Rajalakshmi *et al.,* (2013) for frequent itemsets generation uses single scan of database and eliminates redundant transactions by fixing an attribute called count [13]. Ya-Han Hua *et al.,* (2013) gave a study which includes the concepts of recency, frequency and monetary (RFM) analysis in the sequential pattern mining process [14]. Mining hybrid sequential patterns and sequential rules by Yen-Liang Chen *et al.,* (2002) defined a new kind of patterns, called hybrid pattern, which is the combination of continuous patterns and discontinuous patterns [15]. Ion Railean et al published a paper in 2013 called closeness preference for sequential rules mining [16, 17, 5].

Proposed Work: In order to reduce the time and space in generating the frequent itemsets and also to predict the customer purchase behaviour we introduce a new algorithm called Customer Purchase Behaviour (CPB). The algorithm contains two phases – Generating frequent itemsets and forecasting the customer behaviour on purchase of frequent itemsets.

The existing algorithms for the generation of frequent itemsets are taking k-tables to generate k-itemsets and there is no elimination of duplicate data as well as mining of rules. This leads to more processing time and space. The proposed algorithm is using less than k-tables in order to generate the frequent k-itemsets. All redundant transactions are eliminated before generating the frequent itemsets. The subset generation process plays a significant role in the frequent itemsets invention. Quine-McCluskey Method is used to generate the rules which are very famous extension of k-map technique.

Proposed Algorithm: This section explains the algorithm CPB used for the implementation. The algorithm takes transaction database as the input and produces frequent itemsets and rules for customer purchase pattern as output.

```
Input: Database D
Output: Frequent Itemsets
begin
Form a cluster table from the given transaction database
Convert the given transaction database into bit vectors
Get the minimum threshold (min_thresh)
Determine frequent k-itemsets
freq_cnt = sup_thresh = 0, k=1
$L_k$: frequent itemset of size k
$C_k$: candidate itemset of size k
CNT: Transaction Count
Rule_CNT: Similar transactions differing with one and more items
//Frequent Itemsets Generation
Combine the Identical rows into one row and increment CNT accordingly
Generate candidate itemset, $C_k$ by seeing the 1's in the corresponding item position.
Until all the rows are unique proceed
Implement i loop from 1 to NT times
Implement j loop from 1 to NI times
if (itembit[j] && itembit[j+1] || itembit[j]&& itembit[j+2]||……||itembit[j]&&itembit[j+NI]) Generate all the subsets related to the given itemset.
```

```
    Implement i loop from 1 to NT times
    {
        freq_count = freq_count + CNT
    sup_thresh = freq_count * NI
    Implement j loop from 1 to NI times
    if sup_thresh > min_thresh[j]
    {
    Print the frequent itemset
    Return $L_J$
    //reset freq_count and sup_thresh
    }
    else
    Delete that item(s)
    //reset freq_count and sup_thresh
}
/*Customer Purchase Behaviour. Consider the frequent itemsets from the previous procedure*/
Implement i loop from 1 to NT times
Implement j loop from 1 to NI times
{
/* any two itemset with one itemset differ from only one item can be combined. The unmatched variable can be replaced as - (hyphen)*/
Rule_Cnt_[i]++;
//Place a tick mark for the combined rules
}
Print the unticked transactions as rules
/*the unticked transactions are considered as the Purchase behaviour of the customers.*/
end
```

The given database is converted in to bit vectors and the cluster table is formed. If the cluster table contains any duplicate transaction, those transactions will be eliminated after incrementing the CNT value. When all the rows are unique, the frequency of each item (F_i) will be calculated by using the following Equation 4.1

$$F_i = \Sigma ((C_i)*(CNT)) \tag{4.1}$$

Where,
F_i – Frequency of ith item
C_i – 1's in each column
CNT – Count of similar transactions (similar rows)

The frequent 1-itemsets are calculated using the following Equation 4.

$$St_i = (F_i) * NI \tag{4.2}$$

where, ST_i - Support Threshold for ith item.
NI - Total number of items in the database.

After generating the frequent 1-itemsets, the infrequent items are removed from the table. This may lead to some duplicate transactions. Thus the duplicate values are eliminated after incrementing the CNT value and the frequent k-itemsets are calculated at this stage by using the same Equation 4.1 and 4.2. The algorithm will terminate after generating the k-itemsets when all the transactions are unique and the unique rules for predicting the purchase behavior of customers are generated. In order to understand the algorithm still more in a clear way, an example is explained in the succeeding section.

Subset Generation: A subset is a part of a set. A is a subset of B if and only if every element of A is a element of B. This can be denoted as A ⊂ B.

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n \tag{4.3}$$

A Set of n- elements has 2^n subsets (contains an empty set). This can be given by using binomial sum. When n=1, 2, 3 ... the total number of subsets are 2, 4, 8 ... When n=1 (a set contain only one element), the possible subsets are {ø}, {1} i.e. 2^1 in our case we are not going to consider any null values. Therefore the total number of subsets used here are 2^n-1 in number.

Bit Vectors: The Given transaction database is converted in to bit vectors. Consider when the database contains only 2 items; there is possible transaction types can be represented in $2^2 = 4$ combinations. The combinations are 00,01,10,11 Here the first position represent the item1 transaction, second position represents the item2 transaction. 00 says that no items are transacted. 01 represents item1 is not transacted and item2 is transacted. Here 00 is not considered anyway for the transaction database. Therefore, the total number of transaction for n-items can be represented as 2^n-1.

Sample Transaction: The implementation of this algorithm has been tested by considering the following Pharmacy billing details in Table 4.1 as transaction database. The Table 4.1 is converted in to bit vectors (0 or 1). If an item is present in the transaction, that will be represented as 1(one) else it will be represented as 0(zero). After conversion, depending upon the number of 1's grouping will be made among the transactions. This forms the cluster table. For convenience, the real time items Benoquin, Dialyte, Ibuprofen, Nutradrops and Veetids have been named as A, B, C, D and E respectively in Table 4.2. The Bit Vectors (BV) for the items A, B, C, D

Table 4.1: Pharmacy Transaction Database

TID	Items	TID	Items
T1	Benoquin, Dialyte, Ibuprofen	T10	Benoquin, Nutradrops
T2	Dialyte, Ibuprofen	T11	Benoquin, Dialyte, Nutradrops
T3	Ibuprofen, Veetids	T12	Ibuprofen, Veetids
T4	Benoquin ,Ibuprofen, Nutradrops, Veetids	T13	Benoquin, Dialyte, Ibuprofen, Veetids
T5	Benoquin, Ibuprofen	T14	Ibuprofen, Nutradrops
T6	Benoquin, Ibuprofen, Veetids	T15	Dialyte, Ibuprofen, Nutradrops
T7	Ibuprofen, Veetids	T16	Benoquin, Nutradrops, Veetids
T8	Dialyte, Ibuprofen, Veetids	T17	Dialyte, Nutradrops, Veetids
T9	Benoquin, Dialyte, Ibuprofen, Nutradrops	T18	Benoquin, Ibuprofen, Nutradrops

Table 4.2: Cluster Table Without Duplicates

Item/TID	A	B	C	D	E	Count
T2	0	1	1	0	0	1
T3, T7, T12	0	0	1	0	1	3
T5	1	0	1	0	0	1
T10	1	0	0	1	0	1
T14	0	0	1	1	0	1
T1	1	1	1	0	0	1
T6	1	0	1	0	1	1
T8	0	1	1	0	1	1
T11	1	1	0	1	0	1
T15	0	1	1	1	0	1
T16	1	0	0	1	1	1
T17	0	1	0	1	1	1
T18	1	0	1	1	0	1
T4	1	0	1	1	1	1
T9	1	1	1	1	0	1
T13	1	1	1	0	1	1

Table 4.3: Itemsets Generation

TID	Itemsets
T2	{B,C}=>{B},{C}
T3, T7, T12	{C,E}=>{C},{E}
T5	{A,C}=>{A},{C}
T10	{A,D}=>{A},{D}
T14	{C,D}=>{C},{D}
T1	{A,B,C}=>{A},{B},{C},{A,B}{A,C},{B,C}
T6	{A,C,E}=>{A},{C},{E},{A,C},{A,E},{C,E}
T8	{B,C,E}=>{B},{C},{E},{B,C},{B,E},{C,E}
T11	{A,B,D}=>{A},{B},{D},{A,B}{A,D}, {B,D}
T15	{B,C,D}=>{B},{C},{D},{B,C}{B,D}, {C,D}
T16	{A,D,E}=>{A},{D},{E},{A,D}{A,E}, {D,E}
T17	{B,D,E}=>{B},{D},{E},{B,D}{B,E},{D,E}
T18	{A,C,D}=>{A},{C},{D},{A,C}{A,D},{C,D}
T4	{A,C,D,E}=>{A},{C},{D},{E},{A,C}{A,D},{A,E},{C,D},{C,E},{D,E},{A,C,D}{A,C,E},{A,D,E}{C,D,E}
T9	{A,B,C,D}=>{A},{B},{C},{D},{A,B}{A,C},{A,D},{B,C},{B,D},{C,D},{A,B,C}{A,B,D},{A,C,D}{B,C,D}
T13	{A,B,C,E}=>{A},{B},{C},{E},{A,B}{A,C},{A,E},{B,C},{B,E},{C,E},{A,B,C},{A,B,E},{A,C,E},{B,C,E}

and E are shown in Table 4.2. From Table 4.2 if the transaction pattern (occurrences) is same for multiple transactions, then the transactions are in to a single transaction and also the count value is updated accordingly.

After incrementing the count, the duplicate transactions are deleted except the transaction having updated count value. The amount of memory consumed by the database is reduced here. Thus, instead of storing the transaction pattern which is similar to the already existing transaction, we are going to store it as a single transaction. Based on the Table 4.3, support threshold is computed using Equations 4.1 and 4.2. For example the support threshold for item C is computed by taking the count column value wherever the item C is present (i.e the value of item C is 1).

Depending upon the number the number of ones in the item, the set and the corresponding subset values are generated. The set and the subsets are shown in the Table 4.3. Support threshold for items A, B, C, D and E are as follows.

$BV_{\{A\}} = 10 * 5 = 50\%$ $BV_{\{B\}} = 8 * 5 = 40\%$
$BV_{\{C\}} = 14 * 5 = 70\%$ $BV_{\{D\}} = 9 * 5 = 45\%$
$BV_{\{E\}} = 9 * 5 = 45\%$

For example, if the minimum threshold for single itemset is 45%, the support threshold of item B is less than 45% and it is not used for further processing. Since, the support thresholds of A , C, D and E are greater than are equal to 45% the frequent 1-itemsets are {A}, {C}, {D} and {E}

$BV_{\{A, C\}} = 7*5 = 35\%$ $BV_{\{A, D\}} = 6*5 = 30\%$
$BV_{\{A, E\}} = 4*5 = 20\%$ $BV_{\{C, D\}} = 5*5 = 25\%$
$BV_{\{C, E\}} = 7*5 = 35\%$ $BV_{\{D, E\}} = 3*5 = 15\%$
$BV_{\{A,C,E\}} = 3*5 = 15\%$ $BV_{\{A,D,E\}} = 2*5 = 10\%$
$BV_{\{A,C,D\}} = 3*5 = 15\%$ $BV_{\{C,D,E\}} = 1*5 = 5\%$

The 2, 3-itemsets are formed by seeing the 1's in the corresponding item position. For example in transaction T18, the items A,C,D are having 1's in their position, therefore the Itemset {A,C,D} is formed and E is exempted from this set because it is having a zero entry in its position. Since the itemset is having more than 3-items, its subsets are also generated. For example, the Itemset {A, C, D} provides the subsets {A, C}, {A, D} and {C, D} as output.

If frequent itemset generation needs a support threshold, let the threshold support be 30% for 2-itemset and it is 15% for the 3-itemset. In this process, the summation of the count values is formed to find the frequency of occurrence of particular itemset when the itemset occurs in the database. For example, the itemset {A, C} has the count values in four places. Therefore, the count value is computed as 1+1+1+1+1+1+1=7 from Table 4.3. Now the support threshold is equal to 7*5=35%. Since the support threshold is greater than that of minimum support threshold 30%, the itemset is said to be a frequent itemset. The same procedure is applicable for the generation of all the other itemset and for finding the frequency. Instead of generating newer tables, the same table can be used for generating itemsets. Thus, the frequent 2-itemsets are {C, E}, {A, D}, {A, C} and the frequent 3-itemsets are {A, C, E}, {A, C, D}.

Table 4.4 Frequent Itemsets Generation

Item / Transaction	A	C	D	E	Count
T2	0	1	0	0	1
T3,T7, T12,T8	0	1	0	1	4
T5,T1	1	1	0	0	2
T10,T11	1	0	1	0	2
T14,T15	0	1	1	0	2
T17	0	0	1	1	1
T6,T13	1	1	0	1	2
T16	1	0	1	1	1
T18,T9	1	1	1	0	2
T4	1	1	1	1	1

Predicting Customer Purchase Behaviour: The conditional probability plays a major role in the generation of rules in transaction databases normally. When an itemset satisfying both minimum support threshold and minimum confidence threshold will be considered as the strong itemset and the association rules may be generated from the itemset.

Here the conditional probability plays a major role. When a customer purchases items *A* and *B*, the conditional probability of *A* given *B* is defined as joint probability of *A* and *B* and the probability of *B*:

$$P (A/B) = P (AnB) / P (B) \tag{4.4}$$

The equation 4.4 is saying that the space for judging the frequent itemset or the strong itemset is reduced because of B. The following Table 4.4 is generated after removing the infrequent item B and combining similar rows together updating the count value.

By applying the Quine- McCluskey method, the rule generation for predicting the customer purchase behaviour is done. For that, Group the itemsets according to the number of ones. This is called as primary clustering. Each itemset in one cluster is compared with every other cluster. This technique is called as matching process. Any two itemset with one itemset differ from only one item can be combined. The unmatched variable can be replaced as - (hyphen). This shows that the item may or may not be purchased. The itemsets in one cluster are combined with the next following cluster only. Any two itemsets differing by more than one bit cannot match. If any two itemsets are the same in every position except a position, a tick mark is placed to the right of both the itemsets to show that they are transacted under combination.

This procedure is repeated until the itemsets in one cluster cannot make a match with any other itemsets.

Table 4.5: Predicting the Customer Purchase Behaviour

Item / Transacti on	A	C	D	E	Count	Transaction Status
T2,T3,T7,T12,T8	0	1	0	-	5	√
T2,T5,T1	-	1	0	0	3	√
T2,T14, T15	0	1	-	0	3	√
T3,T7 , T12,T8,T6,T13	-	1	0	1	6	√
T5,T1, T6,T13	1	1	0	-	4	√
T5,T1, T18,T9	1	1	-	0	4	√
T10, T11, T16	1	0	1	-	3	√
T10,T11, T18,T9	1	-	1	0	4	√
T14,T15, T18,T9	-	1	1	0	4	√
T17,T16	-	0	1	1	2	
T6,T13,T4	1	1	-	1	3	√
T16,T4	1	-	1	1	2	√
T18,T4,T9	1	1	1	-	3	√

Table 4.6: Matching - 1-itemsets

Item / Transaction	A	C	D	E	Count	Trans. Status
T2	0	1	0	0	1	√
T3,T7,T12,T8	0	1	0	1	4	√
T5,T1	1	1	0	0	2	√
T10, T11	1	0	1	0	2	√
T14, T15	0	1	1	0	2	√
T17	0	0	1	1	1	√
T6,T13	1	1	0	1	2	√
T16	1	0	1	1	1	√
T18,T9	1	1	1	0	2	√
T4	1	1	1	1	1	√

Table 4.7: Matching-2 Itemsets

Item / Transaction	A	C	D	E	Count
T2,T3,T7, T12,T8,T5, T1,T6,T13	-	1	0	-	9
T2,T5,T1,T3,T7,T12,T8, T6,T13	-	1	0	-	9
T2,T5,T1, T14,T15, T18,T9	-	1	-	0	7
T2,T14,T15,T5,T1, T18,T9	-	1	-	0	7
T5,T1,T6, T13,T18,T4,T9	1	1	-	-	7
T5,T1,T18, T6,T13,T4, T9	1	1	-	-	7
T10,T11,T16,T18,T4,T9	1	-	1	-	6
T10,T11,T18,T9,T16,T4	1	-	1	-	6

The unchecked itemsets in the tables are used for the analysis of the purchase behaviour of customers. A 1(one) under the item shows that the item is purchased and 0(zero) under the item shows that the item is not purchased. The – (Hyphen) shows that the item may or may not be purchased. For example if a transaction represents 0 – 1 1 for the itemset {A, B, C, D}, it shows that When item A is not purchased then items C and D are purchased together. Here item B may or may not be purchased.

The above Table 4.5 shows the frequent itemsets transaction after subsets generation. Clusters are formed based on the number of one's (1's). Here there are 4- clusters. Now, Cluster-1 is combined with cluster- 2 using matching process. Transaction T2 - 0 1 0 0 can be combined with the transaction with Ids T3, T7, T12, T8 - 0 1 0 1, since they differ in one bit position. The resultant transaction is in the form 0 1 0 – and the count value is 5. Here for T2, the count is 1 and for the T3, T7, T12, T8, the count value is 4. Place a tick mark at the right hand side to show that the transaction where combined together. Similarly, combine all the other itemsets in one cluster with other cluster. The matching process must be done with the next following cluster only. The unchecked itemsets in the Table 4.5 and Table 4.6 are used for the analysis of the purchase behaviour of customers in the shop. Since all the rows in the Table 4.7 are unique, the procedure stops. Now the rows used for analysis are the unchecked itemsets in the two Tables 4.6 and 4.7. In the above Table 4.6, first two rows show the same pattern of transactions.

Similarly, third and fourth rows, fifth and sixth, seventh and eighth are same respectively. Instead of two we can consider only one of each of two transaction pattern. Now the patterns for analysis are as follows,

Since there are 18 transactions, the support count for each pattern in Table 4.8 can be calculated by count/ (total number of transactions). The results are as follows:

From the analysis of the transaction - 1 0, When a customer X, purchases item C, he will not purchase item D and he may/may not purchase A and E. The % of support is 50% for this type of pattern of purchase.

From the above analysis, the following rules can be framed.

Rule 1: Buys (X,"C") ➔ ¬ Buys (X,"D") [Purchase Nature: 50%]

Rule 2: Buys (X,"C") ^ ¬ Buys (X,"D") ➔ Buys(X, "A, E") ^ ¬Buys (X,"A, E") [Purchase Nature: 50%]

Similarly, the transaction from Table 4.8 - 1 - 0, it is understood that:

When a customer X, purchases item C, he will not purchase item E and he may or may not purchase items A and D. The support count is 39% for this type of pattern.

Rule 1: Buys (X,"C") ➔ ¬ Buys (X,"E") [Purchase Nature: 39%]

Rule 2: Buys (X,"C") ^ ¬ Buys (X,"E") ➔ Buys(X, "A, D") ^ ¬Buys (X,"A, D") [Purchase Nature: 39%]

Table 4.8: Analysis of Purchase Behaviour

Item / Transaction	A	C	D	E	Count	Purchase Nature in %
T2,T3, T7,T12,T8,T5, T1,T6, T13	-	1	0	-	9	9/18 = 0.5 =50%
T2,T5, T1,T14,T15, T18,T9	-	1	-	0	7	7/18=0.39=39%
T5,T1, T6,T13,T18,T4,T9	1	1	-	-	7	7/18=0.39=39%
T10, T11, T16, T18,T4, T9	1	-	1	-	6	6/18=0.33=33%
T17, T16	-	0	1	1	2	2/18=0.11=11%

From the above two analysis, we can come to the judgement that when a customer buys an item in C, there is no guarantee that the customer will purchase all the other items. Thus the rule is,

Rule: Buys (X,"C") ➜ ¬Buys (X,"A, D, E")

From the Table 4.8, the pattern 1 1 - - indicating, When a customer X, purchases item A and item C, the customer may or may not purchase D and E. The support is 39% for this pattern.

Rule 1: Buys (X,"A") ➜ Buys (X,"C") [Purchase Nature: 39%]

Rule 2: Buys (X,"A") ^ Buys (X,"C") ➜ Buys(X, "D, E") ^ ¬Buys (X,"D, E") [Purchase Nature: 39%]

Similarly, when a customer purchases item A and item D, he may or may not purchase items C and E. It is indicated from 1 – 1 - from Table V.

Rule 1: Buys (X,"A") ➜ Buys (X,"D") [Purchase Nature: 33%]

Rule 2: Buys(X,"A") ^ Buys(X,"D") ➜ Buys(X, "C,E") ^ ¬Buys(X,"C,E") [Supp : 33%]

From the above two analysis we can definitely say that if a customer buys item A, there is a guarantee that he will purchase either C or D.

Rule: Buys (X,"A") ➜ Buys (X,"C") ^ Buys(X, "D")

Performance Analysis: The performance of our algorithm can be analysed easily by comparing the execution t ime and the total number of lines scanned while generating the frequent itemsets generation.

Time Complexity of Proposed Algorithm for Frequent Itemsets Generation: The complexity of the algorithm is $C = ?_k\, m_k k$ where $m_k = |C_k|$. It is observed that $m_1 = d$ as one needs to consider all single items. Since, it requires only one database scan and also the database is updated after finding the frequent itemsets, $m_2 = d\text{-}1$. Thus, the lower bound for C is obtained as:

$$C \le m_1 + 2m_2 = d. \tag{5.1}$$

So, the time complexity of ICBV is less than that of Apriori algorithm which is:

$$T = O\,(dn). \tag{5.2}$$

If d = 10, 000 items and n = 1,000, 000 data records and the speed of the computations is such that ô = 1ns the Apriori algorithm would require 10 seconds. Thus, the time spent for the algorithm is clearly considerable.

RESULTS

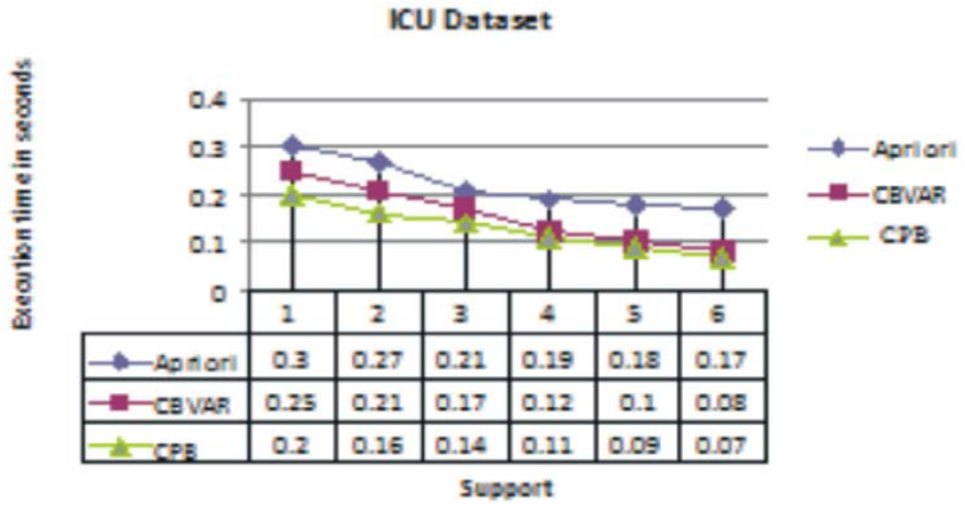
In order to test the proposed algorithm, the space utilization and the time to compute the frequent itemsets were compared. The transaction dataset of pharmacy with 18 transactions is taken with 5-items (as shown in the example section 4.3) for testing.

The following table compares three algorithms by taking the given example as a reference.

By comparing the previous algorithms like apriori and CBVAR algorithm, this algorithm reduces the computation time and computation work. In this work, two types of input datasets have been used for the implementation of the Frequent Itemsets generation. First, the ICU dataset is used as the input dataset for testing the frequent itemset mining algorithms. Second, the transaction dataset is used as the input dataset for validating the frequent itemset generation using the Bit Vector Mining.

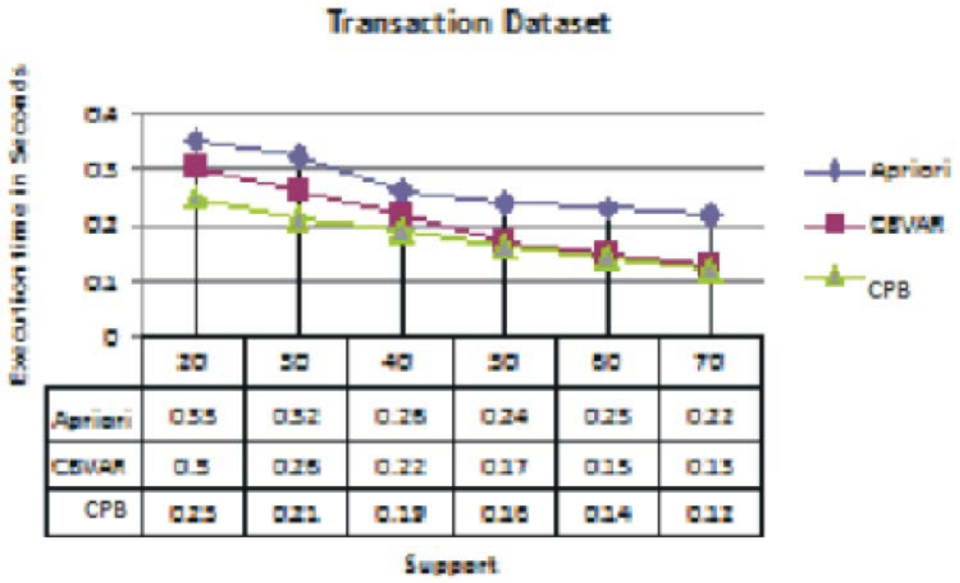
Figure 5.1 shows the execution time for all the algorithms with different support threshold for ICU dataset. The time of execution is decreased with the increase support threshold.

It is also observed that the execution time for the Apriori algorithm is high with the small support threshold and it decreases with the decrease in support using ICU dataset. It is also analyzed that the execution time for CPB algorithm is less as compared to Apriori and CBVAR algorithm.



	1	2	3	4	5	6
Apriori	0.3	0.27	0.21	0.19	0.18	0.17
CBVAR	0.25	0.21	0.17	0.12	0.1	0.08
CPB	0.2	0.16	0.14	0.11	0.09	0.07

Fig. 5.1: Execution Time for ICU Dataset



	20	30	40	50	60	70
Apriori	0.35	0.32	0.26	0.24	0.23	0.22
CBVAR	0.3	0.26	0.22	0.17	0.15	0.13
CPB	0.25	0.21	0.19	0.16	0.14	0.12

Fig. 5.2: Execution time for Transaction Dataset

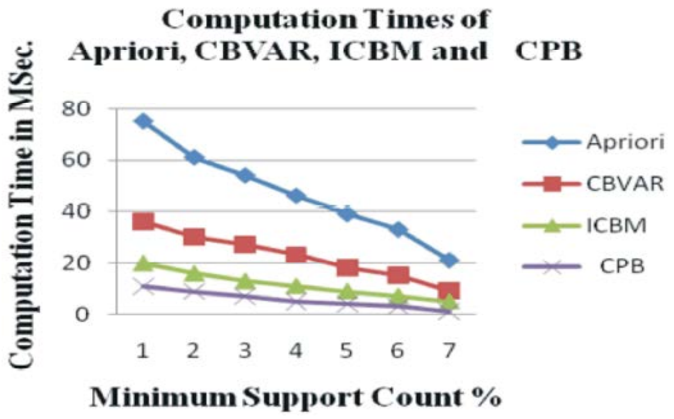


Fig. 5.3: Computation Times of Apriori, CBVAR, ICBM and CPB

Table 5.1: Comparison of Algorithms

Parameter	APRIORI	CBVAR	CPB
Duplicate Removal	No	No	Yes
Number of scans	3	1	1
Total Number of Record Scanning	3*18=54 (scans original database)	3*18=54 (scans intermediate tables)	18+16 +10 =44 (scans the reduced table)

Table 5.2: Datasets for Testing

Name of the Dataset	Number of Records	Input Columns(Items)
ICU Data	50,000	04
Transaction Data	1,000	50

Table 5.3 Comparision of Execution Time

Algorithm	Seconds	Minutes	Hours	Day
Apriori	100000	1666.66	27.77	1.15
CPB	10	0.01666	0.000277	0.04

Figure 5.2 shows the execution time in seconds for the algorithms with different support threshold for Transaction dataset. The time of execution is decreased with the increase in support threshold.

From the Figure 5.2 it is observed that the execution time for the Apriori algorithm is high with the small support threshold and it decreases with the decrease in support using Transaction dataset. It is also analysed that the execution time for CPB algorithm is less as compared to other two algorithms.

The Performance of various algorithms to perform mining of frequent itemsets in various minimum support thresholds is shown in Figure 5.3 (time in seconds). The graph in Figure 5.3 shows that FCPB algorithm has better performance than Apriori, CBVAR and ICBM in all cases of minimum support. Moreover, FCPB algorithm is effective because space consumption is reduced and it takes only one database scan.

CONCLUSION AND FUTURE WORK

The Proposed algorithm uses single scan, which reduces the database scans and hence the computation time taken is also very less for the frequent itemsets generation. It uses fewer amounts of steps to generate the frequent itemsets with duplicate transaction elimination. The strong rules for judging the customer purchase behaviour has been generated using Quine- McCluskey method which is a new concept in Association Rule Mining. Future work in this direction could be the use of minimum space and scan in the customer purchase behaviour, since an item in one cluster is compared with all the other itemsets in the next cluster leads to more time in the generation of associated rules.

REFERENCES

1. Ashok Savasere, Edward Omiecinski and Shamkanth Navathe, 1994. "An Efficient Algorithm for Mining AssociationRules in Large Databases" In VLDB, Zurich, Switzerland, pp: 432-443.
2. Han, J., J. Pei and Y. Yin, 2000. "Mining Frequent Patterns without Candidate Generation" Proceedings of the ACM SIGMOD International Conference on Management of Data, New York, ACM Press, pp: 1-12.
3. Jie Dong, Min Han, 2007. "BitTableFI: An Efficient MiningFrequent Itemsets Algorithm", In Journal of Elsevier on Knowledge-Based Systems, 20: 329-335.
4. Krishnamurthy, M., A. Kannan, R. Baskaran and R. Deepalakshmi, 2011. "Frequent Itemset Generation Using Hashing-Quadratic Probing Technique" European Journal of Scientific Res., 50(4): 523-532.
5. Muhammad Shaheena, Muhammad Shahbaz and Aziz Guergachi, 2013. "Context based positive and negative spatio-temporal association rule mining" In Journal of Elsevier on Procedia Knowledge Based Systems, 37: 261-273.
6. Krishnamurthy, M., A. Kannan, R. Baskaran and G. Bhuvaneswari, 2011. "Hybrid Temporal Mining for Finding out Frequent Itemsets in Temporal Databases Using Clustering and Bit Vector Methods" Communications in Computer and Information Science, Springer, 141(5): 245-255.
7. Krishnamurthy, M., Kannan, R. Baskaran and V. Malini, 2010. "An Escalated Space Preservation Mining Algorithm for Finding Frequent Itemsets" International Journal of Computer Engineering, 2(2): 133-138.
8. Krishnamurthy, M., A. Kannan, R. Baskaran and S. Kanmanirajan, 2009. "Mining Frequent Itemsets using Temporal Association Rule" CiiT International Journal of Data Mining Knowledge Engineering, 1(1): 40-44.
9. Krishnamurthy, M., A. Kannan, R. Baskaran and K. Mythili, 2009. "Temporal-Hmine-rev algorithm for Mining Frequent Patterns in Temporal Databases" CiiT International Journal of Automation and Autonomous System, 1(1): 10-14.

10. Krishnamurthy, M., A. Kannan, R. Baskaran and M. Kavitha, 2011. ”Cluster based Bit Vector Mining Algorithm for Finding Frequent Itemsets in Temporal Databases”, In Journal of Elsevier on Procedia Computer Science, 3: 513-523.
11. Saquer and S. Jitender, 2010. “Using Closed Itemsets for Discovering Representative Association Rules” Foundations Of Intelligent Systems LNCS, Springer, 1932, pp: 495-504.
12. Ion Railean, Philippe Lenca, Sorin Moga, Monica Borda, 2013. ”Closeness Preference - A new interestingness measure for sequential rules mining”, In Journal of Elsevier on Procedia Knowledge Based Systems, 44: 48-56.
13. Krishnamurthy, M., E. Rajalakshmi, A. Kannan and R. Baskaran, 2013. “Improved Cluster Based Mining Algorithm for Frequent Itemsets Generation using Bit Vectors”, TQ-Research Journal.
14. Ya-Han Hu, Tony Cheng-Kui Huang and Yu-Hua Kao, 2013. “Knowledge discovery of weighted RFM sequential patterns from customer sequence databases”, In the Proceedings of The Journal of Elsevier on Procedia Systems and Software, 86: 779-788.
15. Yen-Liang Chen, Shih-Sheng Chen and Ping-Yu Hsu, 2002. “Mining hybrid sequential patterns and sequential rules”, In Journal of Elsevier on Procedia Information System, 27: 345-362.
16. Jayalakshmi, N., V. Vidhya, M. Krishnamurthy and A. Kannan, 2012. “Frequent Itemset Generation Using Double Hashing” Accepted in International Journal of Procedia Engineering, Elsevier, April 2012.
17. Li Chao, Yu Zhao-ping, 2006. “Improved Method of Apriori Algorithm based on Matrix[j]”, In the Proceedings o f Computer Engineering of China”, 23: 68-69.