Indian Journal of Science and Technology, Vol 9(44), DOI: 10.17485/ijst/2016/v9i44/102646, November 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

# Mining Frequent ItemSet Based on Clustering of Bit Vectors

**R. Velmurugan[1*], C. Jothi Venkateswaran[2] and M. Krishna murthy[3]**

[1]Department of Computer Science, L. N. Government College, Ponneri - 601204, Tamil Nadu,India; vel_ram@yahoo.com
[2]Research Department of Computer Science, Presidency College, Chennai - 600005, Tamil Nadu, India; jothivenkateswaran@yahoo.co.in
[3]Department of Computer Science, K.C.G. College of Technology, Chennai - 600097, Tamil Nadu, India; mkrish@kcgcollege.com

## Abstract

**Objectives:** In data mining, finding frequent item set from voluminous databases is an important role. It is a challenging task to find item sets frequently arise in the exponential growth of the databases. The number of scans to find the frequent item set is assessed as more and created redundancy. **Methods/Statistical analysis**: In this research work, a new technique is applied to improve the reduction in number of scans and minimize the redundancy. An algorithm is developed to mine the frequent item set using clustering techniques. A cluster table of frequent item set is created with minimal number of scanning. **Findings:** The support count is fixed to eliminate the duplicates which minimize the redundancy. This algorithm is appreciable, because the intermediate data in the dataset can be reused. The efficiency of algorithm can be identified by seeing the experimental results that is significantly performed well than the exiting algorithms. **Applications/ Improvements:** In future, other algorithms are applied for the same data set to test its effectiveness in order to minimize the redundancy which avoids wastage of the memory space.

**Keywords:** Association Rule Mining, Bit Vector, Cluster, Frequent Itemset

## 1. Introduction

In data mining, association rule is more powerful to discover interesting patterns between items in the large set of databases. The patterns which are occur frequently in transactional database, known as Frequent patterns which has a itemsets appear together frequently, for example milk and sugar. There is a challenging task to find all the sets of items frequently arise in the exponential growth of the databases. To mine the frequent item set, more number of scans needed and created redundancy. The some of the purchase pattern which sequence frequently such as milk followed by sugar. Forming group of similar objects and dissimilar objects which known as cluster analysis.

In this research work focused on bit vector in which all transactions are converted into $0^s$ or $1^s$ instead of storing an actual Item name which occupies less memory space and fast accessing. Discovering Frequent Itemsets(FI) is the most crucial process in the mining of association rules. The Basic association rule mining algorithm for all the algorithms is Apriori algorithm which can find meaningful pattern of items and derive interesting association rules. This algorithm scans the database repeatedly and produces a huge number of candidate sets[1].TheFI-growth has some limitations in generating candidate generation and which has no prefixes within the data items[2] commonly.

The Divide and Conquer technique is used to construct and traversing of Frequent Pattern (FP) tree which is decomposing large mining task into set of different smaller task. In this method however reduces the search space it is better only for dense data. Reducing the search space is advantage of this method but it is better only for dense data. The method of Matrix which is applied to find frequent items which reduce both scanning and execution time but the algorithm proposed only for limited transactions[3]. In paper[4], which uses special data structure

*Author for correspondence

called Bit Table to generate candidate itemsets and compress the database. The issue of finding frequent itemsets which can be obtained solution by building a candidate set generation and then distinguishing itemsets which satisfy the requirement of the frequent itemset inside candidate set.

To reduce size of the candidate items sets, there is no pruning technique used in BitTableFI algorithm, so it consumes more times for scanning the database.

The Boolean Matrix array is an improved method is proposed to store data in which AND operation performed to replace non frequent items can be removed from matrix. This method takes more storage space[5]. The association rule can be generated by Boolean algorithm in which logical operations OR and AND are applied to find frequent itemset[6]. To derive association rule, AND and XOR are performed. It occupies more memory and consumes more computational time.

In[7], an algorithm is proposed to mine Association rules to form clusters and its length. The graph construction for each cluster takes time is not appreciable. The Algorithm CBAR which uses cluster table for loading the databases into main memory[8]. The cluster table in which support count is calculated and there is no necessity to scan the transitions in the cluster table, hence the time takes counting of support count is saved.

In[9], CGBAR (Clustering and Graph based Association Rule), To create cluster table by scanning the database, and process with neat clusters depending on its length. The performance of this approach is somewhat good but the cost of constructing graphs is more.

A new method is proposed in[10] to generate the cluster which consists of only Bit vectors. This Algorithm generates frequent itemsets from the database in single scan. The approach uses multiple set of tables and the redundancy is the issue during rule generation.

In this approach[11], the transactions are converted into decimal numbers which is horizontal representation that reduces the scanning, execution time and consumes less memory space. but it is not applied for high utility dataset and for dense data set.

## 2. Background

When mining the FI from the large set of databases, it is difficult to store the entire database in memory. The existing algorithms proposed by the various researchers to reduce the complexity of space and time are not appreciable. Hence, the utilization of memory space for discovering frequent itemsets generation with cost effective is very much needed. In this research paper, an improved Cluster based Bit Vectors for Association Rule mining (CBVAR)is proposed to mine frequent itemsets efficiently.

Finding interesting frequent itemsets and deriving association rules are the two important steps in the association rules discovery. The similar items are grouped together to form a cluster that helps in making decisions faster and to search data efficiently. The process of creating the cluster which is iterative that consumes time and occupies more space. This issue is solved by an approach to add bit vector with each cluster.

The frequent itemset which is an itemset $X$, whose support(x) => s, where s is assigned by the user that indicates minimum support count. Any subset of frequent itemset is also known as frequent itemset. Let I={item1, item2, item3..item$_m$} be a collection of distinct items. Let $D$ be the Collection of transactions, here each transaction $T \subseteq I$. A transaction can be called as support count of an itemset $X \subseteq I$ if it has all items of $X$ ,ie., $X \subseteq T$.

There is an itemset X which occurs in the number of transactions is known as support. A frequent itemset, its support checks with minimum support threshold assigned by user, if it results greater than or equal then it is called as Min_support. The Table 1 representing 18 transactions

**Table 1.** Grocery dataset

| Transactions | Items |
|---|---|
| T1 | Milk, Diaper, Beer |
| T2 | Diaper, Beer |
| T3 | Beer, Coke |
| T4 | Milk, Beer, Breed, Coke |
| T5 | Milk, Beer |
| T6 | Milk, Beer, Coke |
| T7 | Beer, Coke |
| T8 | Diaper, Beer, Coke |
| T9 | Milk, Diaper, Beer, Breed |
| T10 | Milk, Breed |
| T11 | Milk, Diaper, Breed |
| T12 | Beer, Coke |
| T13 | Milk, Diaper, Beer,Coke |
| T14 | Beer, Breed |
| T15 | Diaper, Beer, Breed |
| T16 | Milk, Breed, Coke |
| T17 | Diaper, Breed, Coke |
| T18 | Milk, Beer, Breed |

of Grocery database. Each transaction represents the different items which are purchased in the dataset.

## 3. Materials and Methods

Finding frequent itemset is not an easy task nowadays. A number of algorithms utilised for the removal of redundancy by using different data sets and some other purposes in same context. Hereafter, few algorithms which are used in this research are discussed.

### 3.1 Apriori Algorithm

The Apriori Algorithm is basic algorithm for finding frequent itemsets in a given database. There is more number of times scanning the database and it uses breadth-first search which is iterative, to discover (k+1)-itemset, where, k-itemsets.

In this algorithm, transaction database is input and a parameter called min_sup that is a value in [0,1] and frequent patterns is output. A pattern, its supports higher or equal to min_sup is frequent pattern. The presence an itemset is the more number occurs in the number of transactions, frequent itemset which called as support count(SC).

$$SC = \sum_{i=1}^{n} A \cap B / N$$

where,

$A$ & $B$ are the items in the dataset

$i$ is the tuple

$N$ is the number of tuples

The support of a pattern that can be called frequency is the more number of transactions occur the pattern which divided by the total number of transactions in the transactional database is referred to as Support Threshold (ST).

$C1 = A(X)$ is the set of all one-itemsets, $k = 1$
While $C_k \neq 0$;
do
scan database to determine support of all $ay$ with $y \in C_k$
extract frequent itemsets from $C_k$ into $L_k$
generate$C_{k+1}$
$k$:=$k + 1$.
end while

It has some disadvantages to find the frequent itemsets:

- It requires too many database scans.
- It consumes large amount of time.
- It generates redundant item-sets.

A novel frequent pattern method is needed to solve the above issues. The efficiency of mining is focus on a big database which has to compressed into smaller as possible as, consumes minimum execution time, minimize cost, avoids repetition of scanning database and generation of candidate sets.

### 3.2 CBVAR Algorithm

CBVAR is an algorithm which uses a new approach for mining frequent itemsets from large amount of database

**Step 1:** Form a table for a given transaction data set
**Step 2:** Convert the itemsets in transaction data set into ($0^s$ or $1^s$) bit vectors
**Step 3:** read minimum threshold,min_thres
**Step 4:** Calculate frequent item sets for 1..m items
(Perform logical AND operation to find frequent items of pairs of items and etc.,
**Step 5:** Calculate the sup_cnt for 1..mitems in new table
sup_cnt=(occurrence of 1's ) * (Total Number of items)
**Step 6:** If sup-cnt>min_supthreshold then print frequent item set
Else remove the item
**Step 7**: Create table for the retained items
**Step 8:** Repeat from step 4 until there are no items in the table.

In this algorithm, a transaction in which itemsare converted into bits(zeros or ones).

Then only once read the database, a table is cluster which is created from which 1-frquent itemsets extracted. The frequent k-itemsets can be extracted by doing AND operation in the table. Minimum memory since it uses a cluster which is small at a time and as scalable for any large amount of database. CBVAR outperforms the Apriori but it fails to avoid the duplications while constructing cluster table.

### 3.3 ICBVAR Algorithm

In this section the improved CBVAR algorithm is proposed. The cluster table is loaded to the main memory, from which the Improved Cluster Based Big Vector Association Rule (ICBVAR) algorithm can fetch the clusters. The support count is calculated on the cluster table and there is no necessity to read all the transactions in the cluster table.

**Step 1:** Form a table for given transaction data set
**Step 2:** Convert the given transaction data set into bit vectors

$$X_{ij=}\begin{cases}1 \text{ if } i\text{th transaction list has } j\text{th item}\\ 0 \text{ otherwise}\end{cases}$$

**Step 3:** $C_i$= *Count of* $i^{th}$ *transaction list*
*For(1, 2,….m) item combination* $1 \le m \le R$ *where R be the number of items.*

$$SC[1,2...m] == \sum_{i=1}^{L}(\prod_{j=1}^{m} X_{ij})C_i$$

*Where SC-Support Count, L be number of list*
*ST[1,2..m]=SC[1,2..m]/ T.*
*Where T be the number of transactions and ST stands for Support Threshold*

**Step 5:** ST= SC/N * 100
**Step 6:** Frequent item set are generated
**Step 7:** If ST>min_sup_threshold then print frequent item set
Else remove the item
**Step 8:** repeat from step 3until there are no items in the table.

The Improved Cluster based Bit Vector Association Rule (ICBVAR) mining algorithm is addressed, which is the enhancement of CBVAR algorithm. In this section, it is generated a Bit table to improve the performance of discovering frequent itemsets. Bit table contains set of integers for representing an item. This table which is used for compressing the candidate itemsets, for which elements in the bit table's is denoted as either one or zero.

Each transaction as specified in the Table1 is converted into bit vector format. For example, the transaction*T15*-contains 3 items Diaper, Beer, Breed and it is represented as 01110 in bit vector format. The Table 2 represents the bit table of Grocery database.

In Table 2, the purchase pattern of transactions T3,T7,T12are having the same items, these three transactions are considered as a single transaction and the CNT has updated accordingly. The same pattern of purchase behavior is treated as duplication and it is considered as single transaction. In this way, a new table is formed which consists of duplicate transactions to be eliminated. To easily identify, replace these items Milk, Diaper, Beer, Breed, Coke with A, B, C, D and E respectively in tables. Consider the Table3, let the MST for 1-frequent itemset is 45%. Support Threshold (ST) for items A(Milk), B(Diaper), C(Beer), D(Breed) and E(Coke) are calculated the frequency as:

**Table 2.** Bit vectors of grocery database

| TID | Milk | Diaper | Beer | Breed | Coke | CNT |
|---|---|---|---|---|---|---|
| T7 | 0 | 0 | 1 | 0 | 1 | 1 |
| T3 | 0 | 0 | 1 | 0 | 1 | 1 |
| T12 | 0 | 0 | 1 | 0 | 1 | 1 |
| T14 | 0 | 0 | 1 | 1 | 0 | 1 |
| T17 | 0 | 1 | 0 | 1 | 1 | 1 |
| T2 | 0 | 1 | 1 | 0 | 0 | 1 |
| T8 | 0 | 1 | 1 | 0 | 1 | 1 |
| T15 | 0 | 1 | 1 | 1 | 0 | 1 |
| T10 | 1 | 0 | 0 | 1 | 0 | 1 |
| T16 | 1 | 0 | 0 | 1 | 1 | 1 |
| T5 | 1 | 0 | 1 | 0 | 0 | 1 |
| T6 | 1 | 0 | 1 | 0 | 1 | 1 |
| T18 | 1 | 0 | 1 | 1 | 0 | 1 |
| T4 | 1 | 0 | 1 | 1 | 1 | 1 |
| T11 | 1 | 1 | 0 | 1 | 0 | 1 |
| T1 | 1 | 1 | 1 | 0 | 0 | 1 |
| T13 | 1 | 1 | 1 | 0 | 1 | 1 |
| T9 | 1 | 1 | 1 | 1 | 0 | 1 |

**Table 3.** Elimination of duplicate transactions

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

A:$BV_A$=*10 / 18=55%*
B: $BV_B$= *8 / 18=44%*
C: $BV_C$= *14 / 18=77%*
D:$BV_D$=*9 / 18=50%*
E: $BV_E$=*9 / 18=50*

The item B is removed from the database since its support threshold is less than 45%.

The items A, C, D and E are the frequent 1-itemsets whose support thresholds are greater than or equal to 45%. With each pair of frequent 1-item sets. The frequent 2-itemsets are performed by using logical AND operation. Add CNT If C and E both contains 1and divide by total number of transactions i.e.,

$BV_{\{C, E\}} = 7/18 = 38\%$
$BV_{\{A, C\}} = 7/18 = 38\%$
$BV_{\{A, D\}} = 6/18 = 33\%$
$BV_{\{C, D\}} = 5/18 = 27\%$
$BV_{\{D, E\}} = 3/18 = 16\%$
$BV_{\{A, E\}} = 4/18 = 22\%$

Items BV{C, D}, BV {D, E}, BV {A, E} does not satisfy the MST=30%, so they are eliminated. Therefore frequent-2 itemsets will be {A, C}, {A, D} and {C, E}

$BV_{\{A, C, E\}} = 3/18 = 16\%$
$BV_{\{A, D, E\}} = 2/18 = 11\%$
$BV_{\{A, C, D\}} = 3/18 = 16\%$

BV{A, D, E} does not satisfy the MST=15%, so it is eliminated. The frequent 3-itemsets obtained are {A, C, D} and {A, C, E}. The frequent itemset finally obtained as

| | |
|---|---|
| 1 | {A}, {C}, {D}, {E} |
| 2 | {A, D}, {A, C}, {C, E} |
| 3 | {A, C, D}, {A, C, E} |

In real time the data generated is enormous and analyzing the data becoming very difficult. This algorithm helps to eliminate the redundant data there by reducing the machine time as well as memory requirements.

## 4. Performance Analysis

The performance of Apriori, CBVAR and ICBVAR Algorithms are analyzed by means of a comparison hereafter. The experiments are performed in Intel i5 Core processor with 4 GB RAM, with grocery dataset which consists of 10000 transactions of customers' buying behavior and the results are reported in Table 4. Python, a high level programming language, is used for the implementation of the proposed ICBVAR, CBVAR, and its parent Apriori algorithm. The Groceries dataset contains 10000records with each record representing single transaction. In each transaction, the products that are bought during that transaction is listed by comma separated values, it means that during one transaction an anonymous customer buys these products. Similarly, 10000 of those transactions are recorded and form the dataset. The Table 4 presents the run time performances of these algorithms.

The Figure 1 shows the performance of execution time for different size of transactions(1000..10000) per iteration for executing the proposed ICBVAR, CBVAR and traditional Apriori algorithm. The support count decreases which leads to increase the number of frequent itemsets. Apriori which performs poorly because of more number of scanning over the database. The time taken for executing transactions 1000..5000 by Apriori and CBVAR has minimum differences. The major time differences can be seen when the size of transactions increases. The CBVAR and ICBVAR has major time differences for 7000..10000 transactions. The ICBVAR experimented in order to avoid redundancy in CBVAR has memory size in less than CBVAR which needs minimum execution time.

**Table 4.** Time complexity of APRIORI,CBVAR and ICBVAR algorithms

| S.No | Transactions | Computing time (Time/sec) | | |
|---|---|---|---|---|
| | | APRIORI | CBVAR | ICBVAR |
| 1. | 1K | 17.80 | 16.00 | 14.9 |
| 2. | 2K | 18.16 | 16.32 | 15.198 |
| 3. | 3K | 19.06 | 16.97 | 15.80592 |
| 4. | 4K | 20.78 | 17.99 | 16.75428 |
| 5. | 5K | 23.48 | 19.43 | 18.09462 |
| 6. | 6K | 27.47 | 21.57 | 19.90408 |
| 7. | 7K | 33.52 | 24.59 | 22.29257 |
| 8. | 8K | 42.57 | 29.01 | 25.63645 |
| 9. | 9K | 55.76 | 35.40 | 30.25102 |
| 10. | 10K | 75.80 | 44.95 | 36.90624 |

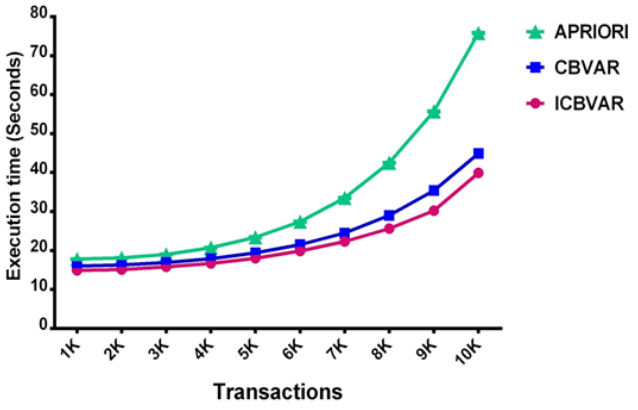


**Figure 1.** Time complexity of algorithms.

The resultant graph it can be absorbed that time consumes for ICBVAR is considerably reduced from 75.8 to 44.95 to 36.90 secs for 10000 transactions with respect to APRIORI, CBVAR and ICBVAR algorithms respectively.

The Figure 2 shows, the space occupied for storing different size of transactions(1000..10000) per iteration for executing the proposed ICBVAR, CBVAR and traditional Apriori algorithm. It shows more differences between Apriori and CBVAR from transactions 1000..10000. The Apriori algorithm takes more memory space in order to store items as it is given but CBVAR has items which are converted into 0 's and 1's for storing.

ICBVAR experimented in order to avoid redundancy takes less space than CBVAR. The major differences can be seen from transactions 7000 to 10000because huge number of redundancy can occur in large transactions.

**Table 5.** Space complexity of APRIORI, CBVAR and ICBVAR algorithms

| S.No | Transactions | Memory (KB) | | |
|---|---|---|---|---|
| | | APRIORI | CBVAR | ICBVAR |
| 1. | 1K | 52.00 | 15.08 | 9.05 |
| 2. | 2K | 59.00 | 17.68 | 10.61 |
| 3. | 3K | 67.00 | 19.00 | 11.37 |
| 4. | 4K | 76.00 | 22.00 | 13.18 |
| 5. | 5K | 90.00 | 27.30 | 16.38 |
| 6. | 6K | 100.00 | 34.00 | 20.42 |
| 7. | 7K | 111.00 | 43.00 | 25.76 |
| 8. | 8K | 123.00 | 55.00 | 33.86 |
| 9. | 9K | 136.00 | 65.00 | 39.12 |
| 10. | 10K | 149.00 | 76.70 | 46.02 |

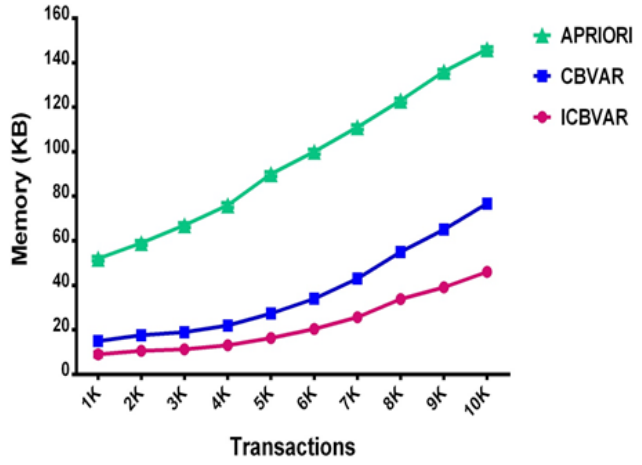


**Figure 2.** Space complexity of algorithms.

## 5. Conclusion

In the generation of big data era reducing the data storage itself will help in improvising the system performance. In this paper, it is studied the mining of frequent itemsets which occurred in the exponential growth of the databases. The existing algorithms are analyzed with respect to number of scans and redundancy to mine the frequent itemset. A new algorithm ICBVAR is developed which is the improvement of CBVAR algorithm. These two algorithms are compared with the Apriori algorithm. The Experiments are conducted to test the time and the space complexity. The ICBVAR algorithm performs lesser memory space and time. These improvements are very much needed to speed up the process of analyzing and decision making. The experimental results indicate more than order of magnitude improvements over the previous algorithms. The future work in the directions would be the use of association rules in past data to predict the future.

## 6. References

1. Agarwal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. Proceedings of the ACM SIGMOD Conference on Management of Data; 1993. p. 207–16.
2. Han J, Pei J, Yin Y.Mining frequent patterns without candidate generation. Proceedings of the ACM SIGMOD International Conference on Management of Data. New York, ACM Press; 2000. p. 1–12.
3. Liu C, Cheng J. The software engineering school, China. Fast mining and updating frequent itemsets. ISECS International Colloquium on Computing, Communication, Control and Management. 2008; 1:365–8.
4. Dong J, Han M.BitTableFI: An efficient mining frequent itemsets algorithm. Journal of Elsevier on Knowledge-Based Systems. 2007; 20:329–35.
5. Yu H, Web J, Wang H, Jun L. An improved Apriori algorithm based on the Boolean Matrix and Hadoop;2011.
6. Wur SH, Leu Y. An efficient Boolean Algorithm forming association rules in large databases. 6thInternational Conference on Database Systems for Advanced Applications; 1999. p. 179–86.
7. Chao L, Zhao-Ping Y. Improved method of Apriori algorithm based on Matrix[j]. Proceedings of Computer Engineering of China. 2006; 23:68–9.
8. Tsay YJ,Chiang JY. CBAR:An efficient method for mining association rules. Knowledge-Based Systems. 2005; 18(2–3):99–105.

9. Alzoubi WA, Bakar AA, Omar K. Scalable and efficient method for mining association rules. International Conference on Electrical Engineering and Informatics; 2009 Aug. p. 5–7.
10. Krishnamurthy M, Kannan A, Baskaran R, Kavitha M.Cluster based bit vector mining algorithm for finding frequent itemsets in temporal databases. Journal of Elsevier on Procedia Computer Science. 2001; 3:513–23.
11. Krishnamurthy M,Manivannan K, ChilambuchelvanA, RajalakshmiE, Kannan A. Enhanced candidate generation for frequent item set generation. Indian Journal of Science and Technology. 2015 Jul; 8(13).DOI:10.17485/ijst/2015/v8i13/60756.