

## Association Rule Mining Using Frequent Itemsets Generation by Anti- Mirroring Of Bit Vectors

D Sathyanarayanan<sup>1</sup>, M Krishnamurthy<sup>2</sup>

<sup>1</sup>Research Scholar, AMET University, Chennai, Tamil Nadu, India.

<sup>2</sup>Professor Department of CSE, KCG College of Technology, Chennai, Tamil Nadu, India.

[dsathya@rediffmail.com](mailto:dsathya@rediffmail.com), [mkrish@kcgcollege.com](mailto:mkrish@kcgcollege.com)

### Abstract

Data mining is the key step in Knowledge Discovery (KDD) Process. It is increasingly becoming an important tool in extracting interesting knowledge from large databases. The project introduces an efficient algorithm using a new technique to find frequent itemsets from a huge set of itemsets and to generate association rules. The existing algorithms for the frequent item sets generation will have more number of scans of database and more number of candidate generations. Also, the existing methods use more number of intermediate tables and the duplicate transactions are not eliminated. The solution for this problem is given in the proposed system by converting the transactions in the database into bit vectors by using counters and by the anti-mirroring concept. Counters are fixed to eliminate the duplicates. Anti-mirroring concept is used to reduce the storage space so that the scanning time and computing time is reduced. . In this work, all the items in a transaction are converted into bit Vectors (0 or 1).

*Keywords: Frequent Itemsets; Association Rule; Counter; Bit vectors; Anti-Mirrors;*

### 1. Introduction

#### 1.1 Data Mining

Data mining is the key step in Knowledge Discovery (KDD) process. It is increasingly becoming important tool in extracting interesting knowledge from large databases. Moreover, many data mining problems involve frequent itemset generation, with examples ranging from engineering to scientific research, finance and medicine.

#### 1.2 Frequent Itemsets

Finding out frequent itemsets is an important issue in many data management systems. The discovery of association rules have been discussed in the past using two steps namely finding the frequent itemsets and generating association rules [1].The existing system contains more number of zeros while

generating frequent itemsets,this leads to major issue.

#### 1.3 Proposed Work

The existing methods which are so traditional which is followed for storing the pattern of purchase will have more number of zeros. For example when a customer purchase an item it will store all its occurrences and non occurrences .In order to store only one bit it stores more number of bits i.e. the number of zeros are more in storage.Consider an example; in a store 1000-items are available. When a customer purchases only 1-item, the storage pattern will contain 999-zeros and only 1-one.It leads to the wastage of storage space. This in turn leads to more scanning time and processing time.

## 2. Related works

There are many works in the literature that discuss about Association rules, Temporal Mining and Frequent Itemsets.

The Association Rule mining raised by R. Agarwal [1] is an important research in data mining field. His Apriori algorithm can discover meaningful itemsets and build association rules. However, a large number of Candidate sets are generated and the database needs repeated scanning.

In paper [4], mining frequent itemsets using Matrix reduces scanning cost and execution times, but the algorithm works only for nine transactions. The partition algorithm [2], to further improve the efficiency, it does so by reducing the number of database scans, however, considerable time is still wasted in scanning infrequent candidate itemsets.

In [11], the proposed Boolean algorithm mines association rules in two steps. In the first step logical OR and AND operations are used to compute frequent itemsets. In the second step, logical AND and XOR operations are applied to derive all interesting association rules based on the computed frequent itemsets. But the computational time is more as well as it occupies more memory.

In [6] proposed an algorithm using Boolean matrix array and using AND operation to replace transaction database therefore non frequent items can be removed from matrix. Algorithm since takes boolean matrix it takes more storage space. In paper [8] proposed a algorithm to find effective time sensitive algorithm using H-struct called Temporal H-mine. It is faster in traversing and can be scaled up to large database by database partitioning but it requires more scanning and processing time.

In improved Apriori algorithm [10], the mining efficiency is very unsatisfactory when memory for database is considered. In paper [9] proposed a new algorithm called CBVAR for generating frequent itemsets using single scan towards the database but not

concentrating on redundancy elimination and rule generation. It uses multiple set of tables to find the frequent itemsets.

Different strategies were proposed after Apriori as in FP-growth [5], which outperforms all candidate generations but still have problems in the case of no common prefixes within the data items. Temporal FP tree uses divide and conquer technique for construction and traversing of tree which is used to decompose the mining task into a set of smaller task which reduces the search space. However, Temporal FP Tree technique is better only when the data is dense.

In paper [7] which uses a special data structure called Bit Table for compressing the database and for the candidate itemsets generation. The problem of mining frequent itemsets can be solved by constructing a candidate set of itemsets and then, identifying those itemsets that meet the frequent itemset requirement within this candidate set. The BitTableFI Procedure does not take any pruning or other technique to reduce the size of candidate item sets and the times of scanning the transaction database.

## 3. Finding frequent pattern:

The method of frequent pattern mining is to find the complete set of frequent patterns in a given transaction database with respect to a given support threshold. Let  $I = \{I_1, I_2, I_3, \dots, I_m\}$  be a set of items. Let  $D$  be the transactional database, where each transaction  $T$  is a set of items. Each transaction is associated with an identifier TID. A set of items is referred as itemset. An itemset that contains  $K$  items is a  $K$ -itemset. The number of transactions in which a particular itemset exists gives the support or frequency count or count of the itemset. If the support of an itemset  $I$  satisfies the minimum support threshold, then the itemset  $I$  is a frequent itemset.

For example, when we have only 3-items, we have  $2^3=8$  combinations of purchase. The following Table 1 illustrates the pattern of purchase.

**Table 1: Bit Vector for 3-items**

| I1 | I2 | I3 |
|----|----|----|
| 0  | 0  | 0  |
| 0  | 0  | 1  |
| 0  | 1  | 0  |
| 0  | 1  | 1  |
| 1  | 0  | 0  |
| 1  | 0  | 1  |
| 1  | 1  | 0  |
| 1  | 1  | 1  |

The above table 1 contains 8 kinds of purchase. The first row contains 000 transactions and it is always not considered since no items are purchased.

### 3.1 Disadvantage of above method:

The above method for storing the pattern of purchase will have more number of zeros. For example, when a customer purchases an item I3, the pattern of storing is 001. In order to store only one bit, we have to store 3-bits.i.e. The number of zeros is more in storage.

### 4. Anti-Mirroring Algorithm:

The work is to reduce the number of zeros stored in the memory and to reduce the scanning time and the processing time.

**Input:** Dataset

**Output:** K-Frequent Itemset

**Step 1:** Form a table for given transaction dataset

**Step 2:** Convert the given transaction dataset into bit vectors

**Step 3:** Check for duplicates, if two rows are similar write as single transaction and increment the counter.

**Step 4:** Get the minimum threshold

**Step 5:** Check if each transaction has anti-mirror

**Step 5.1:** If present replace the transaction with its corresponding anti-mirror and update CNT1 and CNT2

**Step 5.2:** Else If transaction starts with 0 or 1

**Step 5.2.1:** If it is 0, create anti-mirror and replace the same in new table and increment only CNT2

**Step 5.2.2:** Else write the original transaction in new table and update CNT1.

Frequent itemset are generated

**Step 6:** Calculate the sup\_cnt for each item in new table

$$\text{Sup\_cnt} = \frac{\text{Number of 1's in transactions}}{\text{Total number of transaction}}$$

**Step 7:** If sup-cnt > MST

**Step 7.1:** Display frequent itemset

**Step 7.2:** Else delete the item

**Step 8:** Calculate for k-itemsets.

### 4.1 Implementation of Anti-mirroring:

Anti-mirroring is the concept of eliminating number of zeros stored in memory and to reduce the scanning time and processing time respectively.

**Table 2 :Anti-mirrors of itemsets with counter**

| I1 | I2 | I3 | CNT |
|----|----|----|-----|
| 0  | 0  | 0  | 1   |
| 0  | 0  | 1  | 1   |
| 0  | 1  | 0  | 1   |
| 0  | 1  | 1  | 1   |
| 1  | 0  | 0  | 1   |
| 1  | 0  | 1  | 1   |
| 1  | 1  | 0  | 1   |
| 1  | 1  | 1  | 1   |

Total transactions =8

In the table 2 the above part contains more number of zeros. The number of ones are more in the below part. The below part can be

chosen for computation, since the zeros are less in number.

The reflection of zeros in above part is anti-mirrored as ones in below part and vice-versa. Hence the table 3 can be reduced further as given below.

**Table 3 : Simplified form of Table 2 with counter**

| I1 | I2 | I3 | CNT1 | CNT2 |
|----|----|----|------|------|
| 1  | 0  | 0  | 1    | 1    |
| 1  | 0  | 1  | 1    | 1    |
| 1  | 1  | 0  | 1    | 1    |
| 1  | 1  | 1  | 1    | 1    |

$$\text{Total transaction} = 4(\text{CNT1}) + 4(\text{CNT2}) = 8$$

The CNT 1 is the counter for the existing bit vectors and CNT 2 is the counter for the anti-mirrors respectively. For example, Consider the transaction 100, It shows that I1 is purchased and is existing in table 2 for 1 time. Therefore the CNT 1 value is 1. When I1 is purchased for the second time, the CNT 1 is incremented by 1 and is not be stored as a separate transaction row. Here, the duplicate transactions are eliminated.

For example, when I2 and I3 are purchased, the transaction can be given as 011 as bit vectors. This transaction 011 is the anti-mirror of 100, since the zeros are reflected as ones and vice-versa in these transactions. In such cases, the CNT2 will be used, in order to represent 011. The interesting concept over here is that, the transaction 011 is not stored as a separate row, but CNT2 is incremented by 1. If I2 and I3 are purchased for the second time, the CNT 2 is incremented by 1.

Consider the following pharmacy database which contains 18-transactions.

**Table 4: Pharmacy Database**

| TID | ITEMS                                             | TID | ITEMS                                          |
|-----|---------------------------------------------------|-----|------------------------------------------------|
| T1  | Benoquin,<br>Dialyte,<br>Ibuprofen                | T10 | Benoquin,<br>Nutradrops                        |
| T2  | Dialyte,<br>Ibuprofen                             | T11 | Benoquin,<br>Dialyte,<br>Nutradrops            |
| T3  | Ibuprofen,<br>Veetids                             | T12 | Ibuprofen,<br>Veetids                          |
| T4  | Benoquin,<br>Ibuprofen,<br>Nutradrops,<br>Veetids | T13 | Benoquin,<br>Dialyte,<br>Ibuprofen,<br>Veetids |
| T5  | Benoquin,<br>Ibuprofen                            | T14 | Ibuprofen,<br>Nutradrops                       |
| T6  | Benoquin,<br>Ibuprofen,<br>Veetids                | T15 | Dialyte,<br>Ibuprofen,<br>Nutradrops           |
| T7  | Ibuprofen,<br>Veetids                             | T16 | Benoquin,<br>Nutradrops,<br>Veetids            |
| T8  | Dialyte,<br>Ibuprofen,<br>Veetids                 | T17 | Dialyte,<br>Nutradrops,<br>Veetids             |
| T9  | Benoquin,<br>Dialyte,<br>Ibuprofen,<br>Nutradrops | T18 | Benoquin,<br>Ibuprofen,<br>Nutradrops          |

The above transactions are converted into bit vector. For example, consider T15 it contains 3 items Dialyte, Ibuprofen, Nutradrops. The items which are present are marked as 1 and the items which are absent are marked as 0. Hence, transaction T15 contains 01110 as bit vectors.

For our convenience, Let us replace these real time items Benoquin, Dialyte, Ibuprofen, Nutradrops and Veetids with **A** (Benoquin), **B**(Dialyte), **C**(Ibuprofen) **D**(Nutradrops), and **E**(Veetids) respectively

**Table 5 :Bit vectors of Table 4**

| TID | A | B | C | D | E | CNT |
|-----|---|---|---|---|---|-----|
| T3  | 0 | 0 | 1 | 0 | 1 | 1   |
| T7  | 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   |

Transactions T3, T7, T12 in table 5 are having same pattern, hence these three transactions are considered as single transaction and the CNT cell is updated as three instead of one in this way the duplicate transactions are removed.

**Table 6 : Elimination of Duplicate Transactions**

| TID       | A | B | C | D | E | CNT |
|-----------|---|---|---|---|---|-----|
| T3,T7,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   |

Now from the above table 6 we have to check for mirrors. T3, T7, T12 have T11 as

anti-mirror. for example, consider transaction T17 it contains 01011 its mirror image is 10100. Further transactions are listed in table 7.

**TABLE 7 : Mirrors and their concerned Transactions**

| Transactions | Mirror Transactions |
|--------------|---------------------|
| T3,T7,T12    | T11                 |
| T17          | T5                  |
| T2           | T16                 |
| T8           | T10                 |

Now in new table 8 create two counters CNT1 and CNT2 and replace the mirror transactions in the place of original transaction and update CNT1 and CNT2.

T14 and T15 have no anti-mirrors. Create anti-mirrors for those transactions since they start with zeros and replace the created transaction in table 8 and increment only CNT2.

T6, T18, T4, T1, T13, T9 transactions start with 1. Therefore, these transactions need not be represented as anti-mirrors.

**Table 8 : Simplified table after antimirroring**

| TID | A | B | C | D | E | CNT1 | CNT2 |
|-----|---|---|---|---|---|------|------|
| T15 | 1 | 0 | 0 | 0 | 1 | 0    | 1    |
| T10 | 1 | 0 | 0 | 1 | 0 | 1    | 1    |
| T16 | 1 | 0 | 0 | 1 | 1 | 1    | 1    |
| T5  | 1 | 0 | 1 | 0 | 0 | 1    | 1    |
| T6  | 1 | 0 | 1 | 0 | 1 | 1    | 0    |
| T18 | 1 | 0 | 1 | 1 | 0 | 1    | 0    |
| T4  | 1 | 0 | 1 | 1 | 1 | 1    | 0    |
| T14 | 1 | 1 | 0 | 0 | 1 | 0    | 1    |
| T11 | 1 | 1 | 0 | 1 | 0 | 1    | 3    |
| T1  | 1 | 1 | 1 | 0 | 0 | 1    | 0    |
| T13 | 1 | 1 | 1 | 0 | 1 | 1    | 0    |
| T9  | 1 | 1 | 1 | 1 | 0 | 1    | 0    |

Total transaction  $CNT1 + CNT2 = 10 + 8 = 18$

In above table for example take transaction T11, since T11 is the mirror image of T3, T7 and T12 and these transactions start with zero, T11 is replaced in table8 CNT 1 is updated with 1 and CNT2 is updated with 3.

#### 4.2 Finding frequent itemsets:

Support count for an item (SC)  

$$= \sum ((\text{Presence of '1' of the Item}) / (\text{Count of that TID}))$$

Support threshold for an item =  $(SC) / (\text{Total No. of items})$

The MST for 1-frequent itemset is 45%

Support threshold for items A (Benoquin), B(Dialyte), C( Ibuprofen), D( Nutradrops) and E(Veetids) are:

Frequency of A (Benoquin):

Take CNT1 if A contains 1 and take CNT2 if A contains 0 and add them and divide by total number of transactions.

$$\text{i.e. } 10 + 0 = 10 / 18 = 55\%$$

Calculate for remaining items

$$BV_A = 10 / 18 = 55\%$$

$$BV_B = 8 / 18 = 44\%$$

$$BV_C = 14 / 18 = 77\%$$

$$BV_D = 9 / 18 = 50\%$$

$$BV_E = 9 / 18 = 50\%$$

The support threshold of item B(Dialyte) is less than 45% and hence removed from the database.

Since, the support thresholds of A(Benoquin), C(Ibuprofen), D(Nutradrops) and E(Veetids) are greater than are equal to 45% the frequent 1-itemsets are A(Benoquin), C(Ibuprofen), D(Nutradrops) and E(Veetids).

**Table 9 : Elimination of Duplicate Transactions**

| TID            | A            | C            | D            | E            | CNT1         | CNT2         |
|----------------|--------------|--------------|--------------|--------------|--------------|--------------|
| T15            | 1            | 0            | 0            | 1            | 0            | 1            |
| T10            | 1            | 0            | 1            | 0            | 1            | 1            |
| T16            | 1            | 0            | 1            | 1            | 1            | 1            |
| T5             | 1            | 1            | 0            | 0            | 1            | 1            |
| T6             | 1            | 1            | 0            | 1            | 1            | 0            |
| T18            | 1            | 1            | 1            | 0            | 1            | 0            |
| T4             | 1            | 1            | 1            | 1            | 1            | 0            |
| <del>T14</del> | <del>1</del> | <del>0</del> | <del>0</del> | <del>1</del> | <del>0</del> | <del>1</del> |
| <del>T11</del> | <del>1</del> | <del>0</del> | <del>1</del> | <del>0</del> | <del>1</del> | <del>3</del> |
| <del>T1</del>  | <del>1</del> | <del>1</del> | <del>0</del> | <del>0</del> | <del>1</del> | <del>0</del> |
| <del>T13</del> | <del>1</del> | <del>1</del> | <del>0</del> | <del>1</del> | <del>1</del> | <del>0</del> |
| <del>T9</del>  | <del>1</del> | <del>1</del> | <del>1</del> | <del>0</del> | <del>1</del> | <del>0</del> |

The above table eliminates the duplicate transactions. Since T14, T11, T1, T13, T9 and T15 are similar make it as unique transaction and enter in table 10. Both counters CNT1 and CNT2 are updated accordingly. In table 9 consider transaction T10, it is similar to transaction T11. In table 10 transaction T10 is entered and CNT1 is updated as  $1 + 1 = 2$  and CNT2 is updated as  $1 + 3 = 4$ .

**Table 10:**

| TID | A | C | D | E | CNT1 | CNT2 |
|-----|---|---|---|---|------|------|
| T15 | 1 | 0 | 0 | 1 | 0    | 2    |
| T10 | 1 | 0 | 1 | 0 | 2    | 4    |
| T16 | 1 | 0 | 1 | 1 | 1    | 1    |
| T5  | 1 | 1 | 0 | 0 | 2    | 1    |
| T6  | 1 | 1 | 0 | 1 | 2    | 0    |
| T18 | 1 | 1 | 1 | 0 | 2    | 0    |
| T4  | 1 | 1 | 1 | 1 | 1    | 0    |

From the table 8 calculate the 2-frequent itemset, 3-itemset and 4-frequent itemset.

Let the support threshold be **30%** for 2-itemset and it is **15%** for the 3-itemset and **10%** for 4-itemset. Calculate the support count for the generated sets and subsets.

Frequent 2-itemset

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

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 **{Benoquin, Ibuprofen}**, **{Benoquin, Nutradrops}** and **{Ibuprofen, Veetids}**

Frequent 3-itemsets-

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

$BV_{\{A,D,E\}}$  does not satisfy the  $MST=15\%$ , so it is eliminated. The frequent 3-itemsets obtained are **{Benoquin, Ibuprofen, Nutradrops}** and **{Benoquin, Ibuprofen, Veetids}**.

**Table 11 :Frequent Itemsets**

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

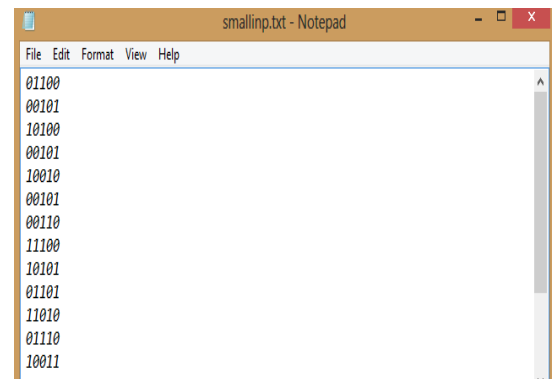
Where,

A – **Benoquin**,  
C – **Ibuprofen**,  
D – **Veetids**  
E – **Nutradrops**

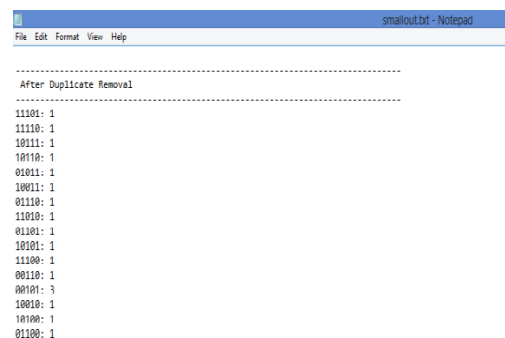
### 4.3 SCREEN SHOTS

The following are the screen shots of various datasets such as pharmacy dataset and bakery dataset and their mining which includes frequent itemset generation and formation of rules.

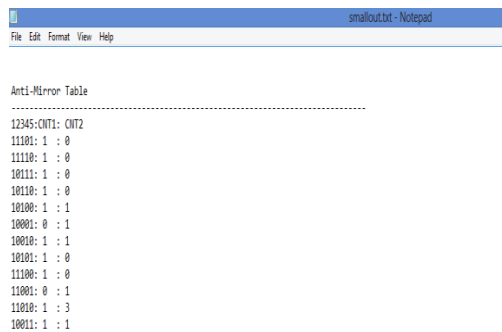
Figure 4.3.1 to Figure 4.3.5 shows the output of frequent itemsets generation by taking smallinp.txt as the input which contains 18-transactions of 5-items.



**FIGURE 4.3.1 Input File Contains 18Transactions Of 5-Items**



**FIGURE 4.3.2 Output File Contains Transactions after duplicate removal**

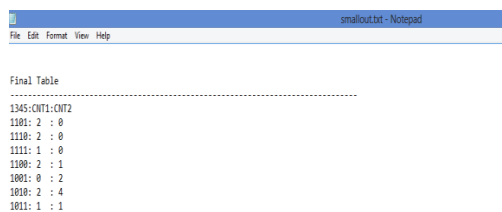


```

Anti-Mirror Table
-----
12345: CNT1: CNT2
11101: 1 : 0
11110: 1 : 0
10111: 1 : 0
10110: 1 : 0
10100: 1 : 1
10001: 0 : 1
10010: 1 : 1
10101: 1 : 0
11100: 1 : 0
11001: 0 : 1
11010: 1 : 3
10011: 1 : 1

```

**FIGURE 4.3.3 Output File Contains Transactions Anti-Mirror table**

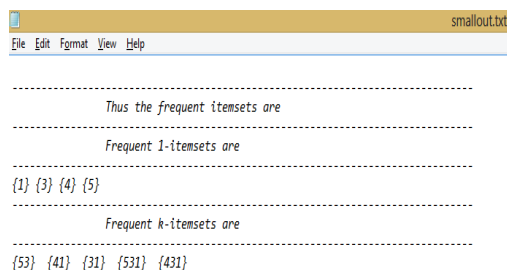


```

Final Table
-----
12345: CNT1: CNT2
1101: 2 : 0
1110: 2 : 0
1111: 1 : 0
1100: 2 : 1
1001: 0 : 2
1010: 2 : 4
1011: 1 : 1

```

**Figure 4.3.4 Showing anti mirror final table**



```

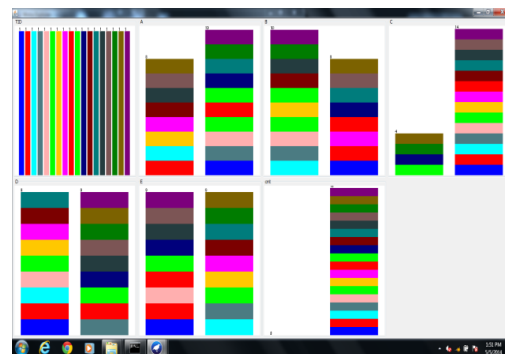
-----
Thus the frequent itemsets are
-----
Frequent 1-itemsets are
-----
{1} {3} {4} {5}
-----
Frequent k-itemsets are
-----
{53} {41} {31} {531} {431}

```

**FIGURE 4.3.5 Frequent K-Itemsets**

## 4.4 Experimental Results

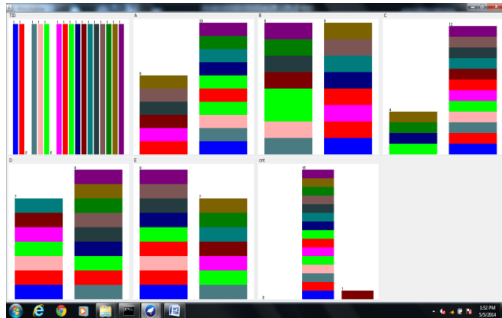
In order to test the proposed algorithm, the data mining tool, Weka is utilized. The transaction dataset of pharmacy with 18 transactions is taken with 5-items for testing. Figure 4.4.1 shows the original dataset of pharmacy. This dataset is given as the input for the Apriori algorithm, CBVAR, Anti-mirror algorithm and our proposed method for the frequent itemsets generation. The Apriori algorithm and CBVAR are not removing the redundant transactions thus leads to processing of lengthier database. The Anti-mirror algorithm is removing the duplicates but it is taking more tables to generate the frequent itemsets. The first panel of Figure 4.4.1 shows the TID from 1 to 18, the second panel shows transaction of the medicine Benquin. Among 18- transactions, 10-transactions contain this item and 8-transactions doesn't. In the same manner, all the other items like Dialyte, Ibuprofen, Nutrodrops, Veetids were shown in the sequential panels.



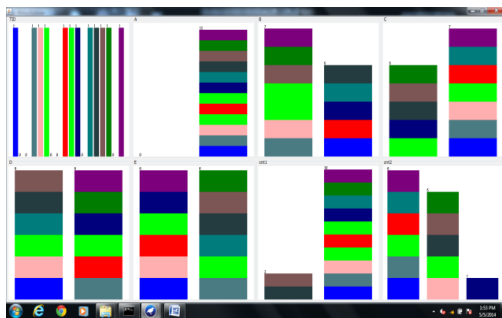
**FIGURE 4.4.1 Pharmacy Dataset with 18- Transactions of 5-Items**

Figure 4.4.2 shows the output of weka after removing the duplicate transactions by the

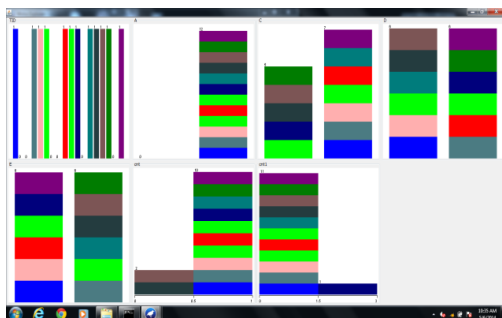
proposed algorithm. But, the Apriori algorithm is not producing such output since it utilizes the original 18-transactions without considering the redundant transactions



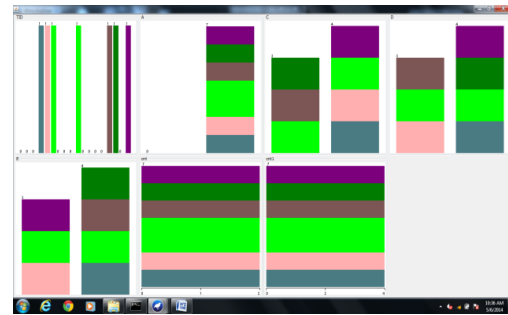
**Figure 4.4.2** Shows after duplicate removal



**Figure 4.4.3** Shows Anti-mirror with counters



**FIGURE 4.4.4** Pruning the Infrequent Item – Dialyte And Removing Duplicates With 10-Transactions of 4-Items



**Figure 4.4.5** Showing anti-mirror table after removing duplicate

Figure. 4.4.5 shows the output of 10-transactions of 4-items after pruning the infrequent item – Dialyte. Here, the support threshold fixed for frequency is 45%. We can observe the graph is so clear when compared to the Figure 4.4.1. The extraction of frequent itemsets is very easy from Figure 4.4.5 rather than Figure 4.4.1 and 4.4.2. The graph shown in the Figure 4.4.4 and 4.4.5 cannot be produced by Apriori and Apriori-like algorithms.

By comparing the previous algorithms like apriori, and CBVAR algorithm, this algorithm reduces the computation time and computation work.

## 5. Conclusion:

The algorithm needs only one database scan. By using the Anti-mirroring method the storage space is reduced so that the scanning time and the computing time are also reduced. Here we are using the counter which is used to eliminate the duplicate present in the transaction database. It reduces the candidate generation and also reduces the time and space complexity.

## References:

- [1] R.Agarwal, T.Imielinski, and A.Swami, "Mining Association Rules Between Sets of Items in Large Databases", In Proceedings of the ACM SIGMOD Conference on Management of Data, Washington, DC, 1993, pp 207-216.

- [2] Ashok Savasere, Edward Omiecinski, and Shamkanth Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases" In VLDB 1994, Zurich, Switzerland, pp.432-443.
- [3] Bay Vo, Tzung-Pei Hong, Bac Le , " DBV-Miner : A Dynamic Bit Vector approach for Fast Mining Frequent Closed Itemsets", In Journal of Elsevier on Expert Systems with Applications, Vol. 39, pp. 7196-7206,2012.
- [4] Chaohui Liu, Jiancheng an, The Software Engineering School, China "Fast Mining and Updating Frequent Itemsets", 2008 ISECS International Colloquium on Computing, Communication, Control and Management, Vol.1, pp. 365-368.
- [5] 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, pp. 1-12.
- [6]Honglie Yu, Jun Web, Hongmei Wang, Li Jun,"An Improved Apriori Algorithm Based On the Boolean Matrix and Hadoop", 2011.
- [7]Jie Dong, Min Han. (2007), "BitTableFI: An Efficient Mining Frequent Itemsets Algorithm", In Journal of Elsevier on Knowledge-Based Systems, Vol. 20, pp.329-335.
- [8] Keshari Verma, O.P.Vyas, "Efficient calendar based association rule", SIGMOD record, Vol.34, No.3, Sept. 2005.
- [9]M. Krishnamurthy, A. Kannan, R. Baskaran , M. Kavitha. (2011) "Cluster based Bit Vector Mining Algorithm for Finding Frequent Itemsets in Temporal Databases", In Journal of Elsevier on Procedia Computer Science , Vol. 3, pp. 513-523.
- [10] Li Chao, Yu Zhao-ping, "Improved Method of Apriori Algorithm based on Matrix[j]", In the Proceedings of Computer Engineering of China", 2006, Vol. 23, pp.68-69.
- [11] Suh-Ying Wur and Yungho Leu, "An Efficient Boolean Algorithm for Mining Association Rules in Large Databases". 6th International Conference on Database Systems for Advanced Applications, 1999, pp: 179.-186.



