CHAPTER 3

SIMPLE ALGORITHM FOR MINING FREQUENT
ITEMSETS

3.1 OVERVIEW

Mining frequent itemsets is instrumental for mining association
rules, correlations, multi-dimensional patterns, etc. Most of the existing works
focus on mining all frequent itemsets. However, since any subset of a frequent
set also is frequent, it is sufficient to mine only the set of maximal frequent
itemsets [31].

3.2 PROPOSED ALGORITHM

A fast and an efficient algorithm is developed based on the
occurrences of items as well as by performing logical AND operation as
detailed below:

Step 1:
Scan the Transactional database to construct the Support Count
Table (SCT) and Bit Form Table (BFT).

Step 2:
In SCT, consider the group of nodes which satisfies the minimum
support count. And find the group of nodes which are fully connected (G)
with each other.
Step 3:
For all \( G \),

(i) Perform logical ‘AND’ operation on BFT.

(ii) Find the total value of the resultant value of logical ‘AND’ operation.

(iii) If the sum of resultant value of logical ‘AND’ operation satisfies the \text{min\_support} count then add ‘G’ to the frequent itemsets \( L_k \).

3.3 IMPLEMENTATION OF THE PROPOSED ALGORITHM

<table>
<thead>
<tr>
<th>Trans.Id</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>abe</td>
</tr>
<tr>
<td>2</td>
<td>bd</td>
</tr>
<tr>
<td>3</td>
<td>bc</td>
</tr>
<tr>
<td>4</td>
<td>abd</td>
</tr>
<tr>
<td>5</td>
<td>ac</td>
</tr>
<tr>
<td>6</td>
<td>bc</td>
</tr>
<tr>
<td>7</td>
<td>ac</td>
</tr>
<tr>
<td>8</td>
<td>abcc</td>
</tr>
<tr>
<td>9</td>
<td>abc</td>
</tr>
</tbody>
</table>

Table 3.1 Transaction table for the proposed algorithm

Table 3.1 contains an entry for every possible set of items which are sold. In the proposed algorithm, the tables SCT and BFT are created by scanning every itemset of the transaction table. The table SCT consists of separate rows and columns for the items found in the transaction table. The value in all locations of SCT are initialized to zero.
Creation of SCT

The rows and columns for the table SCT depending on the number items present in the transaction table. To start with the table SCT is initialized to zero.

The table SCT is created by applying the following procedure.

If the sold itemset is ‘abcd’ then the following entries are made in SCT:

- All the elements of SCT are initialized to zero.
- For the row ‘a’, under the columns ‘b’, ‘c’ and ‘d’, the value will be incremented by one. That is in SCT[a,b], SCT[a,c] and SCT[a,d] the value is incremented by 1.
- For the row ‘b’, under the columns ‘c’ and ‘d’, the value will be incremented by one. That is in SCT[b,c] and SCT[b,d] the value is incremented by 1.
- And for the row ‘c’, under the column ‘d’, the value will be incremented by one. That is in SCT[c, d] the value is incremented by 1.

In this way, for all the itemsets of the given transaction table, the entries of the table SCT will be incremented by one.

Creation of BFT

The table BFT is created by getting binary form of the item. That means if the itemset is abd, then to indicate the presence of items under the columns a, b and d, 1 is used and to indicate the absence of c and e, 0 is used. Based on the items present in the transaction table, columns will be created in
BFT. For example, in the above transaction table maximum 5 items are used. Therefore 5 columns will be created in BFT.

*Implementation of Proposed Algorithm*

**Step 1:**

After scanning the first record of the transaction table that is meant for the first transaction abe, the counter value of the following positions in the table SCT will be incremented by 1.

\[
\begin{align*}
\text{SCT} & \quad [a,b] \\
& \quad [a,e] \\
& \quad [b,e]
\end{align*}
\]

Now the SCT becomes,

\[
\begin{array}{ccccc}
\text{SCT} \\
a & b & c & d & e \\
a & 1 & 0 & 0 & 1 \\
b & 0 & 0 & 1 \\
c & 0 & 0 \\
d & 0 & 0 & 0 \\
\end{array}
\]

And the BFT becomes,

\[
\begin{array}{ccccc}
\text{BFT} \\
a & b & c & d & e \\
1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

**Step 2:**

After scanning the next record of the transaction table that is meant for the next transaction bd, the counter value of the following position in the table SCT will be incremented by 1.

\[
\begin{align*}
\text{SCT} & \quad [b, d]
\end{align*}
\]
Now the SCT becomes,

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

And the BFT becomes,

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Step 3:**

After scanning the next record of the transaction table, that is meant for the next transaction $bc$, the counter value of the following position in the table SCT will be incremented by 1.

SCT \([b, c]\)

Now the SCT becomes,

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
And the BFT becomes,

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Step 4:

After scanning the next record of the transaction table that is meant for the next transaction \( abd \), the counter value of the following positions in the table SCT will be incremented by 1.

\[
\text{SCT \{a, b\}} \\
\text{SCT \{a, d\}} \\
\text{SCT \{b, d\}}
\]

Now the SCT becomes,

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

And the BFT becomes,

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Step 5:

After scanning the next record of the transaction table that is meant for the next transaction \( ac \), the counter value of the following position in the table SCT will be incremented by 1.

SCT \[ a, c \]

Now the SCT becomes,

\[
\begin{array}{cccccc}
  & a & b & c & d & e \\
 a & 2 & 1 & 1 & 1 \\
b & 1 & 2 & 1 \\
c & 0 & 0 \\
d & 0 \\
\end{array}
\]

And the BFT becomes,

\[
\begin{array}{cccccc}
  & a & b & c & d & e \\
 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 \\
\end{array}
\]

Step 6:

After scanning the next record of the transaction table, that is meant for the next transaction \( bc \), the counter value of the following position in the table SCT will be incremented by 1.

SCT \[ b, c \]
Now the SCT becomes,

**SCT**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

And the BFT becomes,

**BFT**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Step 7:**

After scanning the next record of the transaction table that is meant for the next transaction ac the counter value of the following position in the table SCT will be incremented by 1.

SCT [a,c]

Now the SCT becomes,

**SCT**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
And the BFT becomes,

\[
\begin{array}{cccccc}
\text{BFT} \\
\hline
a & b & c & d & e \\
1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 \\
\end{array}
\]

**Step 8:**

After scanning the next record of the transaction table that is meant for the next transaction \(abce\), the counter value of the following positions in the table \(SCT\) will be incremented by 1.

- \(SCT\ [a,b]\)
- \(SCT\ [a,c]\)
- \(SCT\ [a,e]\)
- \(SCT\ [b,c]\)
- \(SCT\ [b,e]\)
- \(SCT\ [c,e]\)

Now the \(SCT\) becomes,

\[
\begin{array}{cccccc}
\text{SCT} \\
\hline
a & b & c & d & e \\
a & 3 & 3 & 1 & 2 \\
b & 3 & 2 & 2 &   \\
c &   & 0 & 1 &   \\
d &   &   & 0 &   \\
\end{array}
\]
And the BFT becomes,

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
 & a & b & c & d & e \\
\hline
1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 \\
\hline
\end{array}
\]

**Step 9:**

After scanning the last record of the transaction table, that is meant for the next transaction \( abc \), the counter value of the following positions in the table SCT will be incremented by 1.

- SCT \([a,b]\)
- SCT \([a,c]\)
- SCT \([b,c]\)

Now the SCT becomes,

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
 & a & b & c & d & e \\
\hline
a & 4 & 4 & 1 & 2 & 0 \\
b & 4 & 2 & 2 & 0 & 0 \\
c & 0 & 1 & 0 & 0 & 0 \\
d & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
\]
And the BFT becomes,

\[
\begin{array}{ccccc}
\text{BFT} \\
\begin{array}{ccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} \\
1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0
\end{array}
\end{array}
\]

After scanning all the records of the transaction table D, the tables
SCT and BFT become,

\[
\begin{array}{ccccc}
\text{SCT} \\
\begin{array}{ccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} \\
1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0
\end{array}
\end{array}
\]

Table 3.2 (a) Support Count Table  Table 3.2 (b) Bit Form Table
After getting the table SCT, find the locations in which the entries of it do not satisfy the minimum support count value and make such entries as zero. For this implementation, the minimum support is taken as 22% and the minimum support count becomes: 22/100 *9=2.

In SCT table, the following locations’ entries are not satisfying the minimum support count value.

SCT [a, d]
SCT [c, e]

So put SCT [a, d] =0 and SCT [c, e] =0. Now the content of SCT is as shown in Table 3.3.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 3.3 Support Count Table with min_support value**

The implementation of the proposed Algorithm is shown in Figure 3.1. According to the proposed algorithm, the frequent itemsets are nothing but the itemsets which are fully connected (connectedness) and their logical AND operation’s result must meet the min_support count value.
Figure 3.1 Implementation of Proposed Algorithm

Connectedness (fully connected)

An itemset **abcd** is said to have connectedness (fully connected) only if the non-zero values are occurred in the following position of SCT.

In the row ‘a’, under the columns ‘b’, ‘c’ and ‘d’
In the row ‘b’, under the column ‘c’ & ‘d’
And in the row ‘c’, under the column ‘d’.

After getting the table SCT, for each row, group all non-zero items together (itemsets) and check them for connectedness (fully connected). For the Table 3.3, the following itemsets are checked for connectedness.

In the first row, the itemset **abce** and
In the second row, the itemset **bcde**
And no itemsets corresponding to third and fourth row.
Now for the first row, consider the itemset **abce**.

In the row a, the columns b, c and e have non-zero values.
In the row b, the columns c and e have non-zero values.
But in the row c, the column e has zero value. This means that the items c and e cannot occur together.

Therefore the itemset **abce** is not fully connected. But the itemsets **abc** and **abe** may be fully connected. So the connectedness of the itemsets **abc** and **abe** have to be checked.

*Itemset abc:*
In the row a, the columns b and c are non-zero values.
In the row b, the column c has non-zero value.
Therefore the itemset **abc** is fully connected.

*Itemset abe:*
In the row a, the columns b and e are non-zero values.
In the row b, the column e has non-zero value. Therefore the itemset **abe** is fully connected.

For the second row,
Consider the itemset **bce**, which are having non-zero values.

*Check for full connectivity for the itemset bce*
In the row b, the columns c and e are non-zero values. But in the row c, the column e is having zero value. Therefore the itemset **bce** is not fully connected as well as not a frequent itemset.
And there is no itemset corresponding to third and fourth row. After checking all the rows of the table SCT for fully connected, perform logical AND operation on the fully connected itemsets.

In the proposed algorithm, a logical AND operation is performed on fully connected itemsets to get the frequent itemsets. In this example, the fully connected itemsets are found to be abc and abe. Now perform logical AND operations to check whether these two fully connected itemsets are frequent item sets or not using the Bit Form Table (BFT) of the transaction table.

<table>
<thead>
<tr>
<th>Trans.Id</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>Logical AND For the itemsets a b c</th>
<th>Logical AND For the itemsets a b e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Total for Logical AND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 3.4 Logical AND for fully connected itemsets abc and abe**

Since the result of logical AND of the fully connected itemsets abc and abe meet the min_support value (2) in Table 3.4, they are frequent itemsets with support count value of 2 (Result of Logical AND operation).
### Table 3.5 Frequent itemsets

<table>
<thead>
<tr>
<th>The frequent itemsets</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a, b, c}</td>
<td>2</td>
</tr>
<tr>
<td>{a, b, e}</td>
<td>2</td>
</tr>
</tbody>
</table>

#### 3.4 EXPERIMENTAL RESULTS

The screen shot shown below is the proposed algorithm’s window. In this screen shot, with the help of the buttons *settings, dataset, scan Dset, SCT Conv, Set MCS, Check FC, List and Time Diff* the frequent itemsets had been obtained.

By clicking on ‘settings’ button the settings toolbar (through which the minimum support count can be set) and delay timing tool bar will be displayed as shown in the following screen shot. After clicking on ‘settings’ button the following screen shot will be displayed.
To get the transaction table for which the frequent itemsets to be found, click on ‘DataSet’ button. While clicking the ‘DataSet’ button, the following screen shot will be displayed, from which we can select the transaction table.

After selecting the transaction table, the content of that table will be displayed as shown below.

Each and every record (itemset) of the transaction table has been scanned by clicking the button ‘Scan Dset’ through which the tables SCT and BFT have been created.
The content of the tables SCT and BFT are displayed in the list boxes SCT Conversion Table and BFT by clicking on the button 'SCT Conv'.

The SCT elements, whose value is less than the min_support, will be made zero by clicking the button 'Set MCS'. The respective screen shot is shown below:
With the help of the buttons ‘Check FC’ and ‘List’ the frequent itemsets are found and displayed along with their counts in the respective list boxes as shown in the screen shot given below.

By clicking on the button Time Diff the execution time of the current transaction table is obtained. Through the button Compare Alg the execution time for the existing algorithms Apriori and FP-Growth and the proposed algorithm have been obtained. The screen shot given below shows the time taken by the proposed algorithm and Apriori algorithm in seconds and FP-Growth algorithm in milliseconds.
The time taken by the Apriori algorithm, FP-Growth and proposed algorithm are shown with the file size in the following table.

<table>
<thead>
<tr>
<th>File size (KB)</th>
<th>Proposed (Sec)</th>
<th>Apriori (Sec)</th>
<th>FP-Growth (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>0.782</td>
<td>14.128</td>
<td>0.015</td>
</tr>
<tr>
<td>57</td>
<td>0.798</td>
<td>18.753</td>
<td>0.172</td>
</tr>
<tr>
<td>62</td>
<td>0.798</td>
<td>23.879</td>
<td>0.015</td>
</tr>
<tr>
<td>70</td>
<td>0.782</td>
<td>12.158</td>
<td>0.016</td>
</tr>
<tr>
<td>84</td>
<td>1.079</td>
<td>29.3</td>
<td>0.297</td>
</tr>
</tbody>
</table>

Based on the time taken and the file size, the comparative study chart is drawn.
Figure 3.2 Comparative Study of Proposed and other Algorithms

The graph in Figure 3.2 represents the results of comparative study of the proposed algorithm with Apriori and FP-Growth algorithms.

3.5 CONCLUSION

Traditionally, the frequent pattern mining has been kept as an offline analytical task, where the frequent patterns are found on the data captured for a specific time period, few weeks, months or even years. An offline approach to data mining reflects sound practice because the data have to be cleaned, checked for accuracy, etc [3]. The execution time taken by the proposed algorithm is very less compared to Apriori algorithm and slightly more than the FP-Growth algorithm. Since FP-growth algorithm had been implemented in Java, Java client alone can access. Even though the proposed
algorithm has been implemented in Java, using the proposed interoperability tool PIS, both Java and .NET clients can access.

The proposed algorithm is simple and easy to understand. In the proposed algorithm, number of scanning has been reduced compared to Apriori algorithm and there is no need of tree construction as in FP-Growth algorithm. The Subsequent chapter describes the introduction of component based development systems, need of interoperability and the standards or protocols for providing interoperability to integrate the components.