A NEW ASSOCIATION MINING ALGORITHM, AMOEBA FOR FINDING CHAIN OF FREQUENT ITEMS USING FUNCTIONAL DEPENDENCY AND PROBABILITY

The research in mining association rules has been expanding extensively in many areas of applications. Among the existing association rule mining algorithms APRIORI and FP-Growth are dominating, besides having few obstacles like time complexity and space complexity. Also these algorithms have a prerequisite of constructing transactional database from the given database before mining frequent patterns. Many algorithms were proposed to overcome these obstacles. Despite of these improved algorithms this chapter introduces a new algorithm, named Amoeba. The algorithm Amoeba is free from constructing transactional database and usage of prior mentioned brink values like support and confidence. This algorithm uses the concept of functional dependency and the probability of an attribute value using functional dependency.

5.1 Introduction

Knowledge discovery is multidisciplinary in the field of computer science. This aims in discovering of knowledge from large amounts of existing data. Mining knowledge can be done by using data mining techniques. Among the existing mining techniques association rule mining
has emerged at-most in providing solutions for many areas of application. Discovering frequent patterns becomes tough if there are more number of patterns in database.

The outcomes of any association rule mining algorithm are rules which take of the form “if then”. These rules identify frequent items between two groups of features. One group is defined as antecedent (if). The other group is defined as consequent (then). An antecedent determines consequent. In contrast consequent is determined by antecedent i.e. consequent is dependent on antecedent.

As association rules are simple “if then” statements, these rules are defined by using two pre-calculative measures. These are: support and confidence. The number of occurrences of an item in transactional database is called support whereas the number of occurrences of a valid rule is called confidence. The predefined value support may or may not discover interesting frequent items. Moreover few algorithms prior to discovering of frequent items are involved in constructing transactional database or frequent pattern trees.

Among the existing association rule mining algorithms APRIORI and F-P Growth are the two dominating algorithms for discovering frequent items.

The APRIORI [133] algorithm has two pre-requisites. They are: construction of transactional database and defining the measures support and confidence. For a predefined support value, frequent item set size varies for iteration of algorithm. Moreover, in every iteration of the algorithm some transactions are eliminated. By eliminating some transactions few items also
gets eliminated. These eliminated items may become frequent items for another execution of same algorithm with altered support and confidence values. Also construction of data set for a given database is not static. The outcome of this algorithm is set of frequent items. For every re-execution of this algorithm with varied support and confidence values and varied transactional database results in change of association rules. Also set of association rules generated by this algorithm changes when there is a change in database, because database is not static.

Another competent algorithm for discovering frequent patterns is FP-Growth [134]. This uses pattern fragment growth and divide-and-conquer policy for mining frequent items. This method stores the compacted and decisive information of frequent items using a special data structure, frequent pattern tree (FP Tree). The algorithm FP-Growth is more reliable than Apriori in discovering frequent items because FP-Growth discovers frequent items without generating candidates. The vital slice of this algorithm is the special structure FP Tree which holds the details of frequent patterns.

Most of the dominating association rule mining algorithms have some nuts and bolts before discovering frequent patterns. Few of them are: building of transactional database; suppositions of support and confidence values; building of FP Tree. These preconditions are to be met by these algorithms for discovering frequent patterns. Additionally data set doesn’t remain static with change in time. A different algorithm AMOEBA is proposed based on the locomotive movements of unicellular organism called amoeba [135]. This algorithm bypasses the nuts and bolts of some existing association rule algorithms.
The rest of this chapter is organized as follows:

Section 5.2 exposes the evolution of proposed association rule mining algorithm, Amoeba.

Section 5.3 states the basic principle of the proposed algorithm, Amoeba.

Section 5.4 illustrates the proposed algorithm, Amoeba

Section 5.5 demonstrates the working of the algorithm, Amoeba, using an instance data set.

Section 5.6 elucidates the results of the algorithm over a standard data set

Section 5.7 concludes with the summary of the chapter

5.2 Evolution of Algorithm Amoeba

In biological classification of organisms protozoa is a group of organisms which contains one cell. One among those is Amoeba. This organism doesn’t have a fixed shape [136]. The first microscopic view of Amoeba was viewed by August Johann Rosel in 1757 [137]. In Greek the word amoibe means change. Thus Bory de Saint Vincentt named it as “amibe” [138]. The locomotive motion of Amoeba is characterized by “false feet” called pseudopodia. Though many postulates were proposed, the movement mechanism of Amoeba is still unknown. The movement characteristic feature of Amoeba is analyzed, thus resulted in progression of a new algorithm for mining frequent patterns, Amoeba.
The locomotive motion of Amoeba if not definite because it contains pseudopodia. This behavior leads in developing the algorithm Amoeba. This behavior is studied as determining an attribute value. Determining an attribute value is determined by functional dependency. In a functional dependency an attribute value determines another attribute value. This determination comprises the percentage of determining an attribute value by another attribute value.

5.3 Principle of The Algorithm Amoeba

The algorithm Amoeba works on the following two principles:

- Determining an attribute value using another attribute value in a data set.

Or

Determining another attribute value in a data set which determined the attribute value.

- Probability of an attribute value being determined by an attribute value.

5.4 The Algorithm Amoeba

Algorithm:

Input: An attribute from discrete dataset

Output: A frequent set of associated attributes with uppermost probability.

Variables: Flag, Input_Attrib, Result, Data_Point
Procedure Amoeba()

Input_Attrib:=read_value

Data_Point := gain row from dataset at random

Index:= discovery the index of the attribute

At random call any one of the following procedures

Feed_left(Data_Point, Input_Attrib, Index)

Feed_Right(Data_Point, Input_Attrib, Index)

End procedure Amoeba

Algorithm Feed_Right:

Input: A row at random from the dataset with an attribute.

Output: chain of frequent attributes towards right in the dataset.

Procedure Feed_Right()

if ! Flag

Find all the associated attributes related to Input_Attrib.

Temp_Attrib:= save all associated attribute

j:= column index of attribute

for each i:=j to max number of columns in dataset do
Find max probability among temp.

Result:= Temp_Attrib

Break

end for

feedRight()

Else

Return

End if

End procedure feedRight

Algorithm Feed_Left

Input: A row at random from the dataset with an attribute.

Output: chain of frequent attributes towards right in the dataset.

Procedure Feed_Left()

if ! Flag

Find all the associated attributes related to Input_Attrib.

Temp_Attrib:= save all related attribute

j:= column index of attribute
for each i:=0 to j do

Find max probability among temp.

Result:= Temp_Attrib

Break

End for

Feed_Left()

Else

Return

End if

End procedure Feed_Left

5.5 Working of Algorithm Amoeba using an instance data set

The working of algorithm Amoeba is demonstrated by using an instant data set as shown in Figure 5.1. This data set is assumed with five discrete attributes. The demonstration of the algorithm Amoeba is as follows:
An attribute at random say attribute 3 is selected. Also, a value at random from the selected attribute (attribute 3) says \( A \) is selected. This is shown in Figure 5.2.

Figure 5.1: A data set at an instance with five discrete attributes

<table>
<thead>
<tr>
<th>Attribute 1</th>
<th>Attribute 2</th>
<th>Attribute 3</th>
<th>Attribute 4</th>
<th>Attribute 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>J</td>
<td>A</td>
<td>D</td>
<td>G</td>
</tr>
<tr>
<td>M</td>
<td>J</td>
<td>A</td>
<td>D</td>
<td>G</td>
</tr>
<tr>
<td>O</td>
<td>K</td>
<td>B</td>
<td>D</td>
<td>H</td>
</tr>
<tr>
<td>N</td>
<td>L</td>
<td>A</td>
<td>E</td>
<td>H</td>
</tr>
<tr>
<td>O</td>
<td>L</td>
<td>B</td>
<td>F</td>
<td>I</td>
</tr>
<tr>
<td>N</td>
<td>K</td>
<td>C</td>
<td>F</td>
<td>I</td>
</tr>
<tr>
<td>M</td>
<td>J</td>
<td>A</td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td>M</td>
<td>J</td>
<td>B</td>
<td>E</td>
<td>G</td>
</tr>
</tbody>
</table>

Figure 5.2: Selection of value of random attribute
- Attribute 3 with discrete value $A$ determines discrete values $D$, $E$ and $F$ of Attribute 4. Among $D$, $E$ and $F$ discrete values the discrete value $A$ determines $D$ with maximum probability as shown in Figure 5.3. The discrete value $A$ of attribute 3 determines the discrete value $D$ of Attribute 4 in two tuples. Similarly, discrete values $E$ and $F$ of Attribute 4 are determined by discrete value $A$ in one and one tuple respectively.

- This demonstrates that discrete value $A$ of Attribute 3 determines the discrete value $D$ of attribute 4 with maximum probability.

<table>
<thead>
<tr>
<th>Attribute 1</th>
<th>Attribute 2</th>
<th>Attribute 3</th>
<th>Attribute 4</th>
<th>Attribute 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$J$</td>
<td>$A$</td>
<td>$D$</td>
<td>$G$</td>
</tr>
<tr>
<td>$M$</td>
<td>$J$</td>
<td>$A$</td>
<td>$D$</td>
<td>$G$</td>
</tr>
<tr>
<td>$O$</td>
<td>$K$</td>
<td>$B$</td>
<td>$D$</td>
<td>$H$</td>
</tr>
<tr>
<td>$N$</td>
<td>$L$</td>
<td>$A$</td>
<td>$E$</td>
<td>$H$</td>
</tr>
<tr>
<td>$O$</td>
<td>$L$</td>
<td>$B$</td>
<td>$F$</td>
<td>$I$</td>
</tr>
<tr>
<td>$N$</td>
<td>$K$</td>
<td>$C$</td>
<td>$F$</td>
<td>$I$</td>
</tr>
<tr>
<td>$M$</td>
<td>$J$</td>
<td>$A$</td>
<td>$F$</td>
<td>$G$</td>
</tr>
<tr>
<td>$M$</td>
<td>$J$</td>
<td>$B$</td>
<td>$E$</td>
<td>$G$</td>
</tr>
</tbody>
</table>

Figure 5.3: Determining adjacent attribute value

- Attribute 3 discrete value $A$ and Attribute 4 discrete value $D$ together determines discrete value $G$ of Attribute 5 with maximum probability as shown in Figure 5.4.
\begin{figure}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Attribute 1 & Attribute 2 & Attribute 3 & Attribute 4 & Attribute 5 \\
\hline
N & J & A & D & G \\
M & J & A & D & G \\
O & K & B & D & H \\
N & L & A & E & H \\
O & L & B & F & I \\
N & K & C & F & I \\
M & J & A & F & G \\
M & J & B & E & G \\
\hline
\end{tabular}
\caption{Determining remaining attribute value}
\end{figure}

- \( \{A, D, G\} \) gets determined by the discrete value \( J \) of Attribute 2 with maximum probability as shown in Figure 5.5.

\begin{figure}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Attribute 1 & Attribute 2 & Attribute 3 & Attribute 4 & Attribute 5 \\
\hline
N & J & A & D & G \\
M & J & A & D & G \\
O & K & B & D & H \\
N & L & A & E & H \\
O & L & B & F & I \\
N & K & C & F & I \\
M & J & A & F & G \\
M & J & B & E & G \\
\hline
\end{tabular}
\caption{Get determining remaining attribute value}
\end{figure}
• Attribute 3 discrete value \( A \), Attribute 4 discrete value \( D \), Attribute 5 discrete value \( G \) and Attribute 2 discrete value \( J \) together gets determined with equal probability by each discrete value of Attribute 1. Thus resulted with chain of frequent patterns (items). This frequent chain contains attribute values in the order; determining values followed by get determined by values. In the above demonstration frequent items chain is \( \{A, D, G, J\} \). At all times an attribute value probably determine or get determined by its neighboring attribute values.

5.6 Results

Time Complexity of algorithms Apriori, FP-Growth and Amoeba is shown in Table 5.1.

<table>
<thead>
<tr>
<th>Apriori</th>
<th>FP-Growth</th>
<th>Amoeba</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(nm^m) )</td>
<td>( O(n^2) )</td>
<td>( O(m^m) )</td>
</tr>
</tbody>
</table>

Table 5.1: Time Complexities of algorithms Apriori, FP-Growth and Amoeba, where \( n \) is number of transactions, \( m \) is number of items

The algorithm AMOeba is implemented on standard data set supermarket.arff. This data set is taken from www.cs.waikato.ac.nz. Three instances of this standard data set with cardinality 3627, 1689 and 941 are considered for implementing the proposed algorithm. These three instances are taken into account for paralleling the algorithms Apriori, FP-Growth and Amoeba. The execution speeds of algorithm Apriori, FP-Growth and Amoeba over the mentioned three instances is shown in Table 5.2.
<table>
<thead>
<tr>
<th>No. of Instances</th>
<th>Apriori</th>
<th>FP Growth</th>
<th>Amoeba</th>
</tr>
</thead>
<tbody>
<tr>
<td>3627</td>
<td>47</td>
<td>3</td>
<td>22.3</td>
</tr>
<tr>
<td>1689</td>
<td>25</td>
<td>2</td>
<td>13.7</td>
</tr>
<tr>
<td>941</td>
<td>8</td>
<td>1</td>
<td>5.2</td>
</tr>
</tbody>
</table>

Table 5.2: Execution times of algorithms Apriori, FP-Growth and Amoeba in seconds.

The algorithm Amoeba is executed on weather, diabetes, iris and hypertension data sets. The cardinality of these data sets are; 14, 768, 150 and 50. The execution speeds of algorithm Amoeba over these data sets is compared with Apriori and FP Growth and is depicted in Figure 5.6.

![Figure 5.6](image)

Figure 5.6: Execution Time (in seconds) of algorithms Amoeba, Apriori and FP Growth over weather, diabetes, iris and hypertension data sets
5.7 Summary

The advantage of algorithm Amoeba is free from construction of FP Growth tree and doesn’t include the values of factors like support and confidence. The algorithm Amoeba doesn’t require a transaction data set. Moreover the algorithm is not limited in dimension of frequent pattern set.

The algorithm AMOEBA results in a chain of frequent items set with a minimum of size 2. The chain of frequent items is not identical for all times, it varies with time as data set is not constant. The probability of frequent items in the frequent item chain decreases if the frequent item set size increases. In the above mentioned example frequent items chain is \{ A, D, G, J \}. The frequent items sets for this chain are \{ A, D \}, \{ A, D, G \} and \{ A, D, G, J \}. The probability of frequent items of these frequent items sets decreases with increase in size of frequent item set i.e. frequent item set of size 2 has highest probability and frequent item set with greater size has less probability.

Algorithm AMOEBA is free from construction of transaction data set, calculation factors like support and confidence and construction of frequent pattern trees. The limitation of this algorithm AMOEBA is, input data set must be discretized because determination through probability can be defined on discrete values. Selection of initial attribute value, influence the evolution of frequent items chain for the algorithm AMOEBA. This is due to, if the determination values of other attributes by initial attribute value are of lowest probability or zero then such initial attribute value becomes void for finding frequent items chain. Selection of such initial attribute value whose probability of determining other attribute values is zero, results in
identifying infrequent items in a data set. This attribute value cannot be included in frequent item set. The algorithms, Apriori and FP Growth cost more, when compared with the algorithm Amoeba in terms of disk usage. More disk space is utilized by Apriori and FP Growth algorithms. The Table 5.3 summaries the algorithms Apriori, FP Growth and Amoeba based on performance parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Apriori</th>
<th>FP Growth</th>
<th>Amoeba</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>Slow</td>
<td>Fast</td>
<td>Moderate</td>
</tr>
<tr>
<td>Dimension of association rule</td>
<td>Required</td>
<td>Not required</td>
<td>Not required</td>
</tr>
<tr>
<td>Construction of transactional data set</td>
<td>Required</td>
<td>Not required</td>
<td>Not required</td>
</tr>
<tr>
<td>Support and Confidence values</td>
<td>Required</td>
<td>Not required</td>
<td>Not required</td>
</tr>
<tr>
<td>Construction of FP Tree</td>
<td>Not required</td>
<td>Required</td>
<td>Not required</td>
</tr>
<tr>
<td>Memory cost</td>
<td>High</td>
<td>Very high</td>
<td>Low</td>
</tr>
<tr>
<td>Input data set</td>
<td>Discrete</td>
<td>Discrete</td>
<td>Discrete</td>
</tr>
<tr>
<td>Size of frequent item pattern</td>
<td>Depends on iteration</td>
<td>Not limited</td>
<td>Not limited</td>
</tr>
</tbody>
</table>

Table 5.3: Summary of algorithms Apriori, FP-Growth and Amoeba.