Generation of Frequent Itemset with Bit Stream Mask Search and Sparse Bit Mask Search

Einstein and Young

Abstract—In general, computer systems are often stored as large amounts of data from which a particular record must be retrieved in accordance with some search criterion. As a result, the well-organized storage scheme to smooth the progress of fast searching is a vital issue. Frequent pattern mining was initially proposed by Agrawal et al. for the purpose of market basket analysis in the form of Association Rule Mining (ARM). Researchers have formulated several algorithms for the generation of frequent itemsets. Frequent itemsets are found mainly from the dataset through a number of searching algorithmic schemes. The novel bit search technique is implemented in the existing ARM algorithms. Frequent itemsets are generated with the assistance of apriori based bit search technique is known as Bit Stream Mask Search and eclat based search technique is branded as Sparse Bit Mask Search. These two algorithms are executed on six datasets namely T10100K, T4010100K, Pump, connect-4, mushroom and chess. These six datasets again executed on Apriori Trie and FP-Growth algorithms. All the algorithms are executed in 5% to 25% support level and the results are compared for the purpose of performance evaluation of the proposed scheme. Efficiency is proved through performance analysis.

Keywords—Association Rules, Frequent Itemset Mining, Bit Search, Bit Stream Mask Search, Sparse Bit Mask Search.

I. INTRODUCTION

DATA mining is such a technique that extracts nontrivial, implicit, previously unknown and potentially useful information from data in databases. Association rule mining searches for interesting correlations among items in a given data set. It was originally proposed almost a decade ago by Agarwal et al. and has since then attracted enormous attention in both academia and industry. Frequent Itemset Mining (FIM) is a data analysis method, which was originally developed for market basket analysis and which aims at finding regularities in the shopping behavior of the customers of supermarkets, mail-order companies and online shops. In particular, it tries to identify sets of products that are frequently bought together.

Efficient Mining of frequent itemsets is a fundamental problem for mining association rules. It also plays an important role in other data mining tasks such as sequential patterns, episodes, multidimensional patterns, The description of the problem is as follows: Let I = {i1,i2,.....in} be a set of items and D be multiset of transactions, where each transaction T is a set of items such that T ⊆ I for X ⊆ I, say that T contains X if X∈T. The set X is called an itemset.

II. RULE MINING ALGORITHMS

In the last two decades, a lot of algorithms are developed for frequent itemset generation. Among all, Apriori is candidate itemset generation, FP Growth is without candidate itemsets and Eclat is vertical data layout are played a very good role in frequent itemset mining. There is more number of data structure used to find frequent itemsets. For frequent itemset generation, more number of searching techniques emerged. Each and every algorithms have their own method of finding frequent itemsets process is analyzed.

A. Apriori Trie

The data structure trie used in the Apriori algorithm is a root (downward) directed tree like a hash tree. The root is defined to be at depth 0, and a node at depth d can point to nodes at depth d+1. A pointer is also called edge or link which is labeled by a letter. There exists a special letter * which represents an “end” character. If node ‘u’ points to node ‘v’, then well can ‘u’ the parent of ‘v’, and ‘v’ is a child node of ‘u’.

Every leaf ‘l’ represents a word which is the concatenation of the letters in the path from the root to ‘l’. Note that if the first k letters are the same in two words, then the first k steps on their paths are the same as well.

Patel et al have proposed parallel algorithm for the mining of frequent itemsets. This is an algorithm for mining frequent itemsets from those databases, whose size is very large and have high data skewness. Other algorithms which adopt the data parallelism include CD (PDM by Park et al.), DMA by Cheung et al., CCPD by Zaki et al. and Lattice based algorithm by Sharma et al. These algorithms differ in whether further candidate pruning or candidate counting techniques are employed or not.

B. Eclat Algorithm

In Eclat algorithm implementation the set of transactions as a (sparse) bit matrix and intersects rows to determine the support of item sets. The search space of Eclat algorithm is based on depth first traversal of a prefix tree.

Éclat principle: A convenient way to represent the transactions for the Eclat Algorithm is a bit matrix, in which each row corresponds to an item, each column to a transaction. A bit is set in this matrix if the item corresponding to the row is contained in the transaction corresponding to the column, otherwise it is cleared. Eclat searches a prefix tree. The transition of a node to its first child consists in constructing a
new bit matrix by intersecting the first row with all following rows. For the second child, the second row is intersected with all following rows and so on.

C. FP-Growth

The FP tree algorithm scans the database twice. In the first time it determines the frequent items that will be used to create the FP-tree and sorts them in frequency order. The top node of the graph is the root. The first node, underneath the tool, is the most frequent item for each record scanned along with a count. Similarly many records are sorted and the most frequent items identified. The basic process involves laying out each record in a frequent order and creating a node for each item under the root. As more items are added, there will be common prefixes.

The inherent advantages of this structure are the relatively compact representation of the database and the exclusion of non-frequent items. This makes it easy to fit the FP-tree into memory and this is easy to scan for rule development.

III. PROBLEM DEFINITION

A. Transaction Bit Array

Let N be the number of transactions of the data set. Let M be the total number of items in the datasets. Convert the dataset items into N x M sparse matrix. Substitute all non-zero elements of sparse matrix as 1 and Mask the matrix as sparse bit matrix. Hence, keep all the transactions of the dataset as transaction bit array.

B. Subset Bit Array

Let I be a set of items. A set X = \{i1, . . . , ik\} is the subset of I is called an itemset, or a k-itemset if it contains k items. All the k-itemsets are converted into bit array by substituting the presence of items as 1 and absence as 0. All subset itemsets are converted into subset bit array.

C. Bitwise AND

Bitwise AND operation is a novel searching technique used to find the frequent itemsets. The AND can be used to find the result value for subset bit array with transaction bit array of dataset sparse bit matrix. If the result value is as same as the subset bit array value, the kitemsets are present in the transaction. This operation is applicable and done for all the subset k-itemsets (where k = 1,2,3,…n) and find the result in a single search. If the result value is not same as the subset bit array value, the items are not present in the transactions.

D. Bit Mask

A pattern of binary values which is combined with some value using bit values 1 for presence of items and 0 for absence of items. The transaction with 0 and 1 combination for searching process is called Bit Mask.

In this research work, the new data structure for searching k-itemsets for frequent itemset mining is implemented. One of the important contributions of this work is a novel searching technique used special data structure, called Sparse Bit Matrix. In the newly proposed algorithms, the role of transaction bit array and subset bit array are explained with examples on datasets. Bit Search has been shown to be a very efficient data structure for searching k-itemsets which search time is reduced to one. Bit Search is implemented in the existing frequent itemset mining algorithms. Bit Search technique is classified into two types. First one is Bit Search Item and second one is Bit Search Tid. Both types of Bit search are implemented in the existing Apriori and eclat respectively. Experimental results are carried out from various dataset implementations for proposed algorithms and also compared with existing AprioriTrie and FP Growth.

IV. PROPOSED FREQUENT ITEMSET MINING ALGORITHMS

Searching an itemset from the dataset with bit search is implemented in the existing Apriori is known as Bit Stream Mask Search (BSMS). This BSMS algorithm is developed for candidate itemset based Apriori algorithm itemsets search is done with the help of Bit Search. Vertical data layout representation of the given dataset is implemented by the bit search is known as Sparse Bit Mask Search.

A. Apriori with Bit Search

Bit Stream Mask Search is a novel approach in which the input file is first transformed into numerical data. After this the transaction file is compressed into an array for further processing. 16 items are stored in one single memory location of the two dimensional array. This technique is divided into two major procedures. Bit Stream Mask procedure is used to compress the 16 data elements in one array location as 16 bit value. MIP Search is used for searching the elements in the above mentioned array. All the processes are done by with this 16 bit storage only. This approach increases the overall efficiency of the apriori algorithm in terms of time and space.

Bit Stream Mask Algorithm

This algorithm reads the transaction file generated, for each transaction it takes items 1 to n and transformed it into Bit Stream format which makes the overall checking of item combinations for all itemsets (1 to n) optimized.

MIP Search Algorithm

This algorithm is used to check whether the subsets formed in the subset are frequent or not. This is done to make sure that an itemset is frequent only if its subsets are frequent. If subsets are found to be frequent then the corresponding itemset is added to the candidate itemset else it is discarded thus, reducing the search space. For searching k-itemsets, the following new searching concept is introduced. This algorithm searches k-itemsets in one time search. This search procedure is called Masked Item Processing (MIP) Search. This technique uses code to search the number of occurrences of a particular subset in itemset. In Masked Itemset Processing (MIP) search, Bitwise AND is

Steps for Bit Stream Mask Search

This represents the numeric data items which are converted from transaction database. Algorithm compresses the 16 data items into one memory storage place. First step of the algorithm is to allocate the memory for the given dataset is known as MIP array or transaction bit array. If the number of unique items in the database is N, and Number of transaction in database is T, then the BitStreamMask array is declared as
BitStreamMask \([T]\) \([(N -1)/16]\). It optimizes the dataset process memory.

In Step 1 read each item in transaction 1 to N. In the step 2 each 16 data items are compressed into one value in single memory location. Consider the following example for masked process.

Bit Stream Mask Array storage In the given example transaction \((T10)\) the numbers are in the range of 1 to 16, 17 to 32 and 33 to 48 are compressed and the result values are stored in the array location.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Sample Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>T2</td>
<td>1 2 4 17</td>
</tr>
<tr>
<td>T3</td>
<td>5 1 6 4 14</td>
</tr>
<tr>
<td>T4</td>
<td>2 7 15</td>
</tr>
<tr>
<td>T5</td>
<td>8 9 10 16</td>
</tr>
<tr>
<td>T6</td>
<td>2 11 12 18</td>
</tr>
<tr>
<td>T7</td>
<td>4 13 2 9 1 21 28</td>
</tr>
<tr>
<td>T8</td>
<td>4 8 32</td>
</tr>
<tr>
<td>T9</td>
<td>14 18 25 33</td>
</tr>
<tr>
<td>T10</td>
<td>1 3 6 9 12 21 25 28</td>
</tr>
</tbody>
</table>

Frequent Itemset Finding: In Step 1, mask the item subset (Masked subset). Consider the 2 item subset \((2, 3)\) this is masked as follows

\[
2 \times 2 - 1 + 2 \times 3 - 1 = 2 + 4 = 6 \quad \text{and position to search in MIP array is 0 because the items are between 1 to 16.}
\]

Bitwise AND operation between Masked subset and each transaction in MIP array are performed to the all transactions. First, check whether the items are present in Transaction \((T1)\).

\[
\begin{align*}
2 & 3 \\
\text{=1+4+32+2048} & =2320 \\
\text{=2085} & =9
\end{align*}
\]

V. EXPERIMENT AND RESULTS

A. Experimental Setup

Bit Stream Mask Search and Sparse Bit Mask Search algorithms were experimented on six data sets, which exhibit different characteristics and the results evaluated. The data sets used were:

- \(T10100K\), \(T40I200200K\), pump, chess, connect and mushroom obtained from FIMI web site [119]. For the experiments, Intel Pentium 2.5 GHz processor, Windows XP with 256 MB RAM was used.

Both Bit Stream Mask-Search and Sparse Bit Mask-Search algorithms were implemented to the above mentioned six datasets with the various support levels. Experimental evaluation is done with C++ language. Further, AprioriTrie and FP Growth algorithms are also implemented and their results are also tabulated. Table 5.1: characteristics of experiment data sets

<table>
<thead>
<tr>
<th>Data</th>
<th>No. of items</th>
<th>Avg. trans. length</th>
<th>Total No. of transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T10100K</td>
<td>500</td>
<td>10</td>
<td>50,000</td>
</tr>
<tr>
<td>Pump</td>
<td>500</td>
<td>50</td>
<td>29,219</td>
</tr>
<tr>
<td>T40I200200K</td>
<td>942</td>
<td>39</td>
<td>50,000</td>
</tr>
<tr>
<td>Mushroom</td>
<td>120</td>
<td>23</td>
<td>8,124</td>
</tr>
<tr>
<td>Connect-4</td>
<td>130</td>
<td>43</td>
<td>67,557</td>
</tr>
<tr>
<td>Chess</td>
<td>75</td>
<td>40</td>
<td>3,325</td>
</tr>
</tbody>
</table>

B. Performance Analysis of Frequent Itemset with Bit Search

Bit Search is implemented in the existing Apriori and eclat algorithms for the both type representation. Both Bit Stream Mask Search and Sparse Bit Mask Search results are carried out and compared.

Bit Search Performance Evaluation

In theoretical analysis of algorithms it is common to estimate their complexity in the asymptotic sense, i.e., to
estimate the complexity function or arbitrarily large input. Big O notation, omega notation is used to the new bit search technique. Performance analysis is measured through finding the time complexity of the algorithms. The bit search complexity is the complexity for all numeric values either presence or absence as a single bit.

Complexity analyses of the new proposed algorithms are defined as follows: For both horizontal and vertical sparse bit representation of the dataset are same processes.

Let number of items in dataset is $M$ and let the number of items in transaction be $N$. During the searching process, all the items in the transactions are converted as bit storage. The required memory allocation is to represent the array as bit elements. So the memory occupation was reduced. Approximate number of bytes required to the represent items and searching is calculated as $\log M/2$.

Time required to search any $k$-itemset ($k=1, 2, \ldots$) in a single transaction is $1$ (one). For $N$ number of transaction is $O(N)=N$ which is the lowest one while compared to any other Association Rule searching technique. In Best case, searching 1-itemset search space time is 1 and also in the worst case of $k$-itemset search space time is also reduced to 1. This algorithm implies its best performance for all itemset combination from 1 to $k$ search time is reduced to 1 (one).

Figure 6.1 to 6.4 shows the comparison of six datasets for various support levels execution time for AprioriTrie, FP-Growth, BitStream Mask Search and Sparse Bit Mask Search respectively.

Figure 6.1: Run Time Comparison of AprioriTrie
Figure 6.2: Run Time Comparison of FP Growth
Figure 6.3: Run Time Comparison of Bit Stream Mask Search for Various Datasets
Figure 6.4: Sparse Bit Mask Search Run time

Figure 6.1 and Figure 6.2 show the comparison of AprioriTrie and FP Growth algorithms respectively. These two algorithms execution time is high.
Figure 6.4 represents the Sparse Bit Mask Search algorithm time comparison for the datasets T10100K, T40120200K, Pump, Connect-4, Mushroom and chess datasets respectively.

Figure 6.5 shows the Comparison of BSMS, SBMS, AprioriTrie and FP-Growth with datasets. In the figure 6.5, the new algorithms Bit Stream Mask Search and Sparse Bit Mask Search compared with the existing AprioriTrie and FP-Growth algorithms.

This figure further compared with six different datasets and also various support level execution times. For the dataset T40120020K, Sparse Bit Mask Search execution is less while compared with others in all support levels. SBMS run time is faster than other algorithms for the dataset T10100K in all support levels. For Pump dataset, Bit Stream Mask Search algorithm execution time is less than compare with other algorithms. Sparse Bit Mask Search algorithm run time is faster than other algorithms for Connect-4 dataset. For Mushroom dataset, SBMS execution time is faster than FP Growth and AprioriTrie in all support levels. Sparse Bit Mask Search is superior in execution for the dataset Chess in all support levels while compared with other algorithms.

From the above comparison figure, it proves the efficiency of Bit Search techniques reduced the execution time more. Hence, Bit Stream Mask Search and Sparse Bit Mask Search algorithms are superior to the existing AprioriTrie and FP Growth algorithms. Sparse Bit Mask Search run time is less in all datasets and its support levels.

VI. CONCLUSION

The above discussions are made with the algorithms which are developed by the new bit search technique. From these comparisons and analysis conclude that the bit search produced only low execution time in all datasets compared with the existing algorithms. Both Bit Stream Mask Search and Sparse Bit Mask Search run time are very low while comparing with other algorithms. Memory space allocation for

REFERENCES