Journal of Theoretical and Applied Information Technology
15 July 2012. Vol. 41 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# FREQUENT ITEMSET MINING WITH BIT SEARCH

[1]N. VENKATESAN, [2]RAMARAJ

[1]Research Scholar, Madurai Kamaraj University, Madurai, INDIA
[2]Technology Advisor, Madurai Kamaraj University, Madurai, INDIA

*Email: Email: [1]envenki@gmail.com, envenki@sify.com, [2]dr_ramaraj@yahoo.co.in*

## ABSTRACT

Computer systems are often used to store large amounts of data from which individual records must be retrieved according to some search criterion. Thus the efficient storage of data to facilitate fast searching is an important issue. Frequent pattern mining was first proposed by Agrawal et al. for market basket analysis in the form of association rule mining. It analyses customer buying behavior by finding associations between the different items that customers place in their shopping baskets. Researchers have proposed several algorithms for generating frequent itemsets. Frequent itemsets are found from the dataset through several searching algorithmic approaches. The novel bit search technique is implemented in the existing association rule mining algorithms. Frequent itemsets are generated with the help of apriori based bit search technique is known as Bit Stream Mask Search and eclat based bit search technique is branded as Sparse Bit Mask Search. These two algorithms are implemented in six datasets namely T10100K, T40I10100K, Pump, connect-4, mushroom and chess. These six datasets again run in AprioriTrie and FP-Growth algorithms. All the algorithms are executed in 5% to 25% support level and the results are compared. Efficiency is proved through performance analysis.

**Keywords:** *Association Rules, Frequent Itemset Mining, Bit Search, Bit Stream Mask Search, Sparse Bit Mask Search*

## 1. 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. [1], and has since then attracted enormous attention in both academia and industry.

Frequent Itemset Mining (FIM) [12] 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 [13]. It also plays an important role in other data mining tasks such as sequential patterns, episodes, multidimensional patterns [5], etc. The description of the problem is as follows: Let I = $\{i_1, i_2, \ldots .i_n\}$ be a set of items and D be multiset of transactions, where each transaction T is a set of items such that $T \subseteq I$ for $X \subseteq I$, say that T contains X if $X \subseteq T$. The set X is called an itemset.

Frequent itemset Rule Mining Algorithms are discussed in the Section 2. Section 3 describes the problem definition and new search solution. New search technique is implemented in the existing algorithms and new algorithm is proposed in the Section 4. Section 5 contains results of proposed algorithm. Section 6 discussed about the Performance analysis of new algorithms compared with existing algorithms. Section 7 is concluded along with future work.

## 2. 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.

## 2.1. APRIOR ITRIE

The data structure trie used in the Apriori [9] 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.

Tries are suitable to store and retrieve not only words, but any finite ordered sets. In this setting a link is labeled by an element of the set, and the trie contains a set if there exists a path where the links are labeled by the elements of the set, in increasing order.

Patel *et al.* [14] 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.*) [6], DMA by Cheung *et al.*, [5], CCPD by Zaki *et al.*, [8] and Lattice based algorithm by Sharma *et al.* [7]. These algorithms differ in whether further candidate pruning or candidate counting techniques are employed or not.

## 2.2. ECLAT ALGORITHM

In Eclat algorithm [3] 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 [2].

**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.

The item corresponding to the row is intersected with the following rows to form the common prefix of the item sets, processed in the corresponding child node. Of course, rows corresponding to infrequent item sets should be discarded from the constructed matrix, which can be done most conveniently if it stores with each row the corresponding item identifier rather than relying on an implicit coding of this item identifier in the row index.

## 2.3. FP-GROWTH

The FP tree algorithm [4] 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.

For instance, one record {A,B,C) has a common prefix with {A,B,D} namely {A,B}. Nodes are not repeated, but the counts for A and B nodes are incremented. When the C node is reached, a new at the same level for C is created with the value D. Note that non frequent items are ignored in the FP-tree construction. In addition, a linked list of frequent items is also maintained, thus every occurrences of A is linked to every other node.

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. After completion of construction, the tree is mined for frequent pattern as

a) Deriving a set of conditional paths. These are suffix patterns from the FP-tree.

b) Constructing a conditional FP-tree for the conditional paths.

c) Exploring the conditional tree recursively to find the Frequent Patterns and determine the support level for each pattern.

Note that the tree contains only frequent items. No step is wasted with non-frequent items. In addition, since the most frequent items are near the top or root of the tree, the mining algorithm works well, but there are some limitations.

a) The databases must be scanned twice.

b) Updating of the database requires a complete repetition of the scan process and construction of a new tree, because the frequent items may change with database update.

c) Lowering the minimum support level requires complete rescan and construction of a new tree.

d) The mining algorithm is designed to work in memory and performs poorly if a higher memory paging is required.

## 3. PROBLEM DEFINITION

**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.

**Subset bit array**
Let I be a set of items. A set X = {$i_1$, . . . , $i_k$} 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.

**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 k-itemsets 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.

**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.

## 4. PROPOSED FREQUENT ITEMSET MIINING 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.

## 4.1. 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.

### 4.1.1. 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.

**Algorithm 4.1: BitStreamMask**

```
Input: Numerical dataset
// allocate Memory for storing the Masked
information

BitStreamMask( )
{
BitStreamMask [ no of Transaction] [((Maxitem-
1)/16)+1]
 for each transaction in input file {
     for each item in transaction  {
          pos=(item -1)/16;
            if (item%16=0) then
              item = 16;
           else
                item = item %16
BitStreamMask [transaction][pos] + =
power(2,item)
        }
    }
        return(BitStreamMask array)
 }
```

### 4.1.2. 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 introduced to search the k-itemsets in a single time matching. This MIP search algorithm supports k-itemsets (where k=1,2,…n) in a single time search.

**Algorithm 4.2: MIP Search**

```
Searching of itemset $_k$ Masked Item processing
[MIP]

MIPSearch (i$^{th}$ item combination in itemset $_k$ ,
minimum support)
{
    for each  transaction in MIP array {
     if ( MIP Search [0,1..n] & (BIT AND)
    MIPSearch [transaction][0,1, ..n] =  MIP

    Search [0,1,..n] )
         itemset_count = count++;
         }
         if  ( itemset_count >= minsupport )
         add itemset$_i$
         else
         delete  itemset$_i$
     }
```

### 4.1.3. STEPS FOR BIT STREAM MASK SEARCH

Table 4.1 given below represents the numeric data items which are converted from transaction database. Algorithm 4.1 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.

Journal of Theoretical and Applied Information Technology
15 July 2012. Vol. 41 No.1
© 2005 - 2012 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

*Table 4.1: Transaction dataset*

| Transaction | Sample Dataset |
|---|---|
| T1 | 1 2 3 |
| T2 | 1 2 4 17 |
| T3 | 5 1 6 4 14 |
| T4 | 2 7 15 |
| T5 | 8 9 10 16 |
| T6 | 2 11 12 18 |
| T7 | 4 13 2 9 1 21 28 |
| T8 | 4 8 32 |
| T9 | 14,18, 25 33 |
| T10 | 1 3 6 9 12 21 25 28 33 36 |

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.

In the above table 4.1, consider the transaction (T10) having items 1 3 6 9 12 21 25 28 33 36 and to convert into Bit Stream Mask array. This can be stored in the Bit Stream Mask array as in table 4.2 format.

Table 4.2 Bit Stream Mask Array storage

| **BitStream Mask [0][0]** | **BitStream Mask [0][1]** | **BitStream Mask[0][2]** |
|---|---|---|
| $(2^{1-1}+2^{3-1}+2^{6-1}+2^{12-1}$ $=1+4+32+2048)$ $=2085$ | $(2^{5-1}+2^{9-1}+2^{12-1}$ $=16+256+2048)$ $=2320$ | $(2^{1-1}+2^{4-1}$ $=1+8)$ $=9$ |

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.

In BitStreamMask [0][0] , the items 1 to 16 are masked. In BitStreamMask [0][1], the items 17 to 32 is masked, where 17,18,19,…,32 is taken as 1,2,3,…..,16. In BitStreamMask [0][2] , the items 33 to 48 is masked, where 33,34,35,…,48 is taken as 1,2,3,…..,16 for each transaction the above transformation is done.

Normally, search algorithms explore the whole database for each combination of itemsets to gather the required itemsets. But BitStreamMask-Search picks out the required itemsets at a single glance.

**Frequent Itemset Finding**

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

$2^{2-1} + 2^{3-1} = 2 + 4 = 6$ 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).

For example, (2,3) ➔ Masked_subset = 6 and position is 0.

6 AND MIP[1,2,…n][0] of T1 = 6

The result is same as the value 6 (Masked subset value i.e., 6), so the item subset is present in that transaction. If the bit operation result value is not equal to Masked Subset value, then the items are not present in that particular transaction.

Similarly, frequent itemsets are generated for all subsets of the transaction dataset are searched

## 4.2. ÉCLAT WITH BIT SEARCH

Normally, eclat procedure is implemented through tree data structure for searching the itemsets. Bit Search procedure is implemented in the existing eclat algorithm is known as Sparse Bit Mask Search. First of all, the transactions are converted into Sparse Matrix and again converted into Sparse Bit Matrix. All the search operations are done only with the help of Sparse Bit Matrix.

### 4.2.1. BIT ARRAY ECLAT ALGORITHM

Bit Array Eclat is a new algorithm that mines association rules from the transaction file which array is transformed as sparse bit matrix for processing. This processing is done from numerically transferred input file.

This algorithm shows the usage of sparse matrix to mine association rules with bit array data structure. This produces only positive association rules.

Algorithm 4.3 converts all the data items in the transaction database into sparse bit matrix. With the help of sparse bit matrix {item, tid} representation, 1-itemset count is done quickly and efficiently. Bit Mask Tid algorithmic procedure is followed to find the frequent itemsets.

*Algorithm 4.3: Bit_Array_Eclat*

```
1. Initialize the matrix bit[n][m] where n-> number of itemsets m-> no. of transactions
2. For each item in the transaction repeat the steps 3,4,5
3. for each transaction in the input file repeat the step 4
4. Check whether the bit [item] [m] is not equal to zero. If yes the increment the total count (tcount)
5. Calculate the support using total count divided by total no. of transactions
```

### 4.2.2. K-ITEMSET BIT SEARCH ALGORITHM

The k-itemset (where k = 2, 3,…, n) combination subset items are searched with the algorithm 4.4 using bit search. Bitwise AND operation is used to find the k-itemsets in a single search

*Algorithm 4.4: Searching of k- itemset using Bit Search*

```
Procedure (k^th item combination in itemset )
    1. for each transaction in Bit array repeat the step 2
    2. Bitwise AND can be used to find the result value for subset with transaction dataset. If the result value is as same as the subset value, the k-itemsets are present in the transaction
    3. increment Bit_itemset count
    4. Check whether bit_item_count is greater than or equal to minsup. If yes add the frequent bit_itemsets otherwise delete item
```

### 4.2.3. SPARSE BIT MASK SEARCH EXPLANATION

First of all, the frequent itemset is identified the itemset whose support is greater than or equal to the minimum threshold value. In the database the every itemsets are found by using the bit array structure. The above numeric transactions are given in the table 4.3 as sparse bit matrix format.

Journal of Theoretical and Applied Information Technology
15 July 2012. Vol. 41 No.1
© 2005 - 2012 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 4.3: Sparse Bit Matrix

| **Item sets** | **Tids** | | | | | | | | | | Count |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 9 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 10 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 4 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 4 |
| 5 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 5 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 |
| 8 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 2 |
| 10 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 4 |

From the above table 4.3, sparse bits occupy only 100 bits for 10 x 10 matrix data representation.

Column wise itemset bits are bitwise AND operated to find whether elements are found or not in one search.

For example, the subset {2, 10} is used to search in the transactions 4 and 6. From the subset, bit representation of array is 1000000010.
Searching the subset bit array in the transaction 4.
Transaction bit array of T4 is 0000100111.
Subset bit array of {2, 10} AND Transaction of bit array {T4}
1000000010 AND 0000100111 = 0000000100
The result value is 0000000100 which is not equal to the subset bit array {2, 10}. Hence the items are not present in the transaction 6.

Searching the subset bit array in the transaction 6.
Transaction bit array of T6 is 1010001011
Subset bit array of {2, 10} AND Transaction of bit array {T6}
1000000010 AND 10100010111 = 1000000010.

The result value is 1000000010 which are equal to subset bit value. Therefore, all items in the subset are present in that transaction 6 (T6).

In this manner, all the frequent itemsets are generated.

## 5. EXPERIMENT AND RESULTS

### 5.1. 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.

The six data sets were often used in the previous study of association rule mining and were downloaded from the websites [119] [120]. Some characteristics of these datasets are shown in table 4.6.

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

| **Data** | **No. of items** | **Avg. trans. length** | **Total No. of transacti ons** |
|---|---|---|---|
| T10100K | 500 | 10 | 50,000 |
| Pump | 500 | 50 | 29,219 |
| T40I200200 K | 942 | 39 | 50,000 |
| Mushroom | 120 | 23 | 8,124 |
| Connect-4 | 130 | 43 | 67,557 |
| Chess | 75 | 40 | 3,225 |

## 6. PERFORMANCE ANALYSIS OF FREQUENT ITEMSET WITH BIT SEARCH

In Section 4, 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.

### 6.1. 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,….) 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 ApriroriTrie, FP-Growth, BitStreamMaskSearch and SparseBitMaskSearch respectively.

Journal of Theoretical and Applied Information Technology
15 July 2012. Vol. 41 No.1
© 2005 - 2012 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

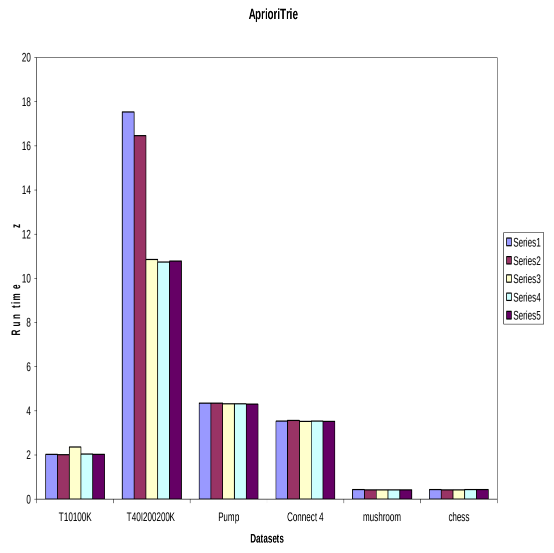

*Figure 6.1: Run time comparison of AprioriTrie*

Figure 6.1 and Figure 6.2 show the comparison of AprioriTrie and FP Growth algorithms respectively. These two algorithms execution time is high.

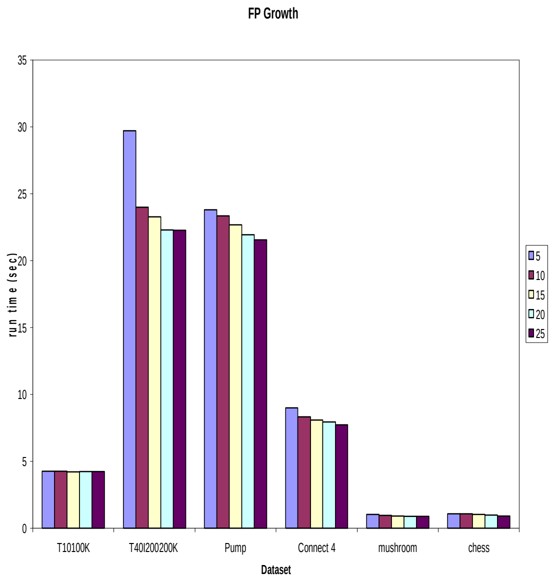

*Figure 6.3: Run time comparison of Bit Stream Mask Search for various datasets*

Figure 6.3 shows the Bit Stream Mask Search algorithm execution time comparison with six datasets.

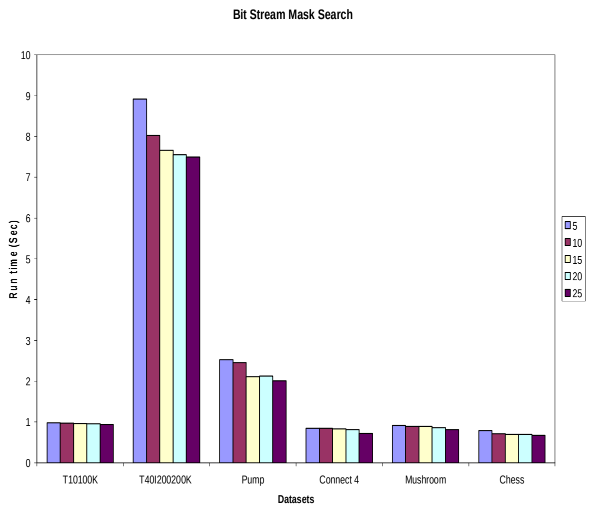

*Figure 6.2: Run time comparison of FP Growth*

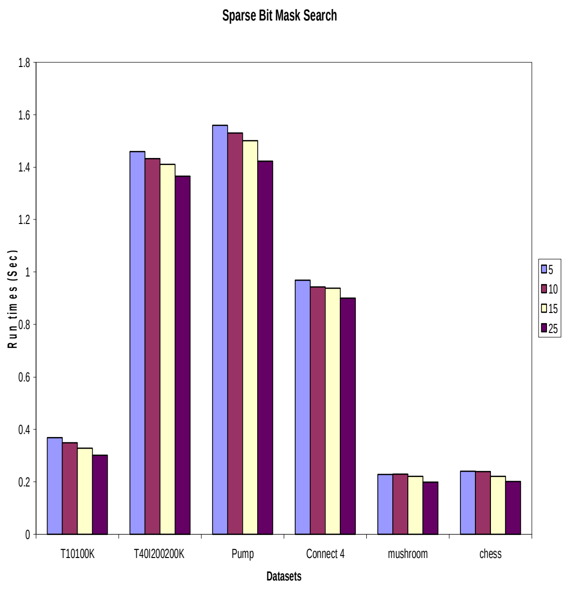

*Figure 6.4: Sparse Bit Mask Search Run time comparison*

Figure 6.4 represents the Sparse Bit Mask Search algorithm time comparison for the datasets T10100K, T40I20200K, Pump, Connect-4, Mushroom and chess datasets respectively.

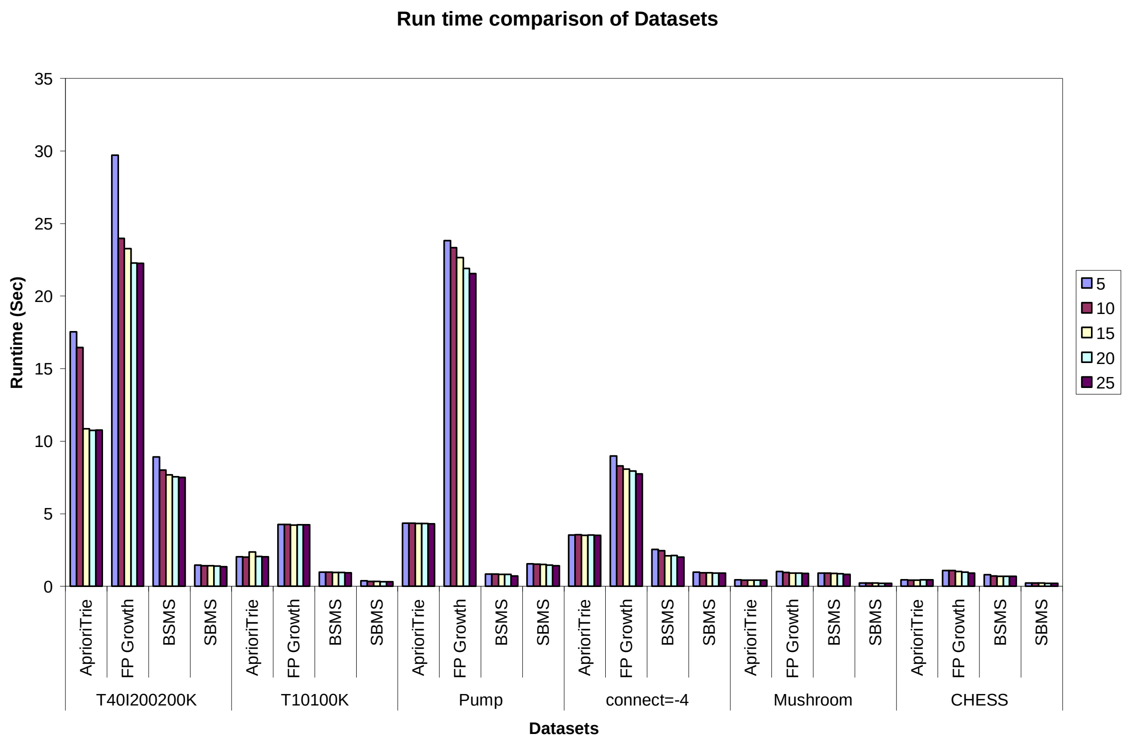

*Figure 6.5: Comparison of BSMS, SBMS, AprioriTrie and FP-Growth with datasets*

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 T40I20020K, 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.

## 7. 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 Bit Search is also less whereas GP-Growth and AprioriTrie. Interesting application will be developed as future enhancement.

## REFERENCE:

[1]. Agrawal, R., Imielinski, T., and Swami, A. N. 'Mining association rules between sets of items in large databases'. In Proceedings of the 1993 ACM SIGMOD International

Conference on Management of Data, 1993, pp: 207-216.

[2]. Bayardo, R.J., 'Efficiently mining long patterns from databases' . Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, June 1-4, 1998. New York, USA., pp: 85-93.

[3]. Bucila, C., J. Gehrke, D. Kifer and W. White, 'DualMiner: A dual-pruning algorithm for itemsets with constraints'. Data Mining and Knowledge Discovery, 7 2003. 241-272.

[4]. Calders, T. and B. Goethals, 'Depth-first non-derivable itemset mining'. Proc. SIAM Int. Conf. Data Min., 119 2005. pp: 250-261.

[5]. Cheung, D.W., J. Han, V.T. Ng and C.Y. Wong, 'Maintenance of discovered association rules in large databases: An incremental updating technique'. Proceedings of International Conference on Data Engineering, Feb. 26-Mar. 1, 1996. New Orleans, Louisiana, pp: 106-114.

[6]. Park, J.S., Chen M.S.and Yu, P.S. 'Efficient parallel mining for association rules'. Proceedings of the 4th International Conference on Information and Knowledge Management, Nov. 29-Dec. 2, 1995. Baltimore, MD., pp: 31-36.

[7]. Sharma, S., Tiwari, A. Sharma S. and Pardasani, K.R. 'Design of algorithm for frequent pattern discovery using lattice approach.' Asian J. Inform. Manage, 2007. 1: pp: 11-18.

[8]. Zaki, M.J., Mitsunori, O. Parthasarathy S. and Wei, L. 'Parallel data mining for association rules on shared memory multiprocessors'. Proceedings of the 1996 ACM/IEEE Conference on High Performance Networking and Computing, Jan. 01, 1996. IEEE Computer Society, Washington, DC., USA., pp: 1-25.

[9]. Bart Goethals 'Survey on Frequent Patten Mining' 2004

[10]. FIMI dataset – http://fimi.cs.helsinki.fi/

[11]. http://miles.cnuce.cnr.it/ palmeri/datam/DCI /datasets.php

[12]. Margret H. Dunham, Data Mining: Introductory and Advanced Topics , Pearson Edition , 2004.

[13]. Agrawal, R. and R. Srikant, 'Mining sequential patterns'. Proceedings of the 11th International Conference on Data Engineering, March 6-10, 1995 Taipei, Taiwan, pp: 3-14.

[14]. Pei, J., Han J. and Wang, W. 'Constraint-based sequential pattern mining in large databases'. Proceeding of the 2002 International Conference on Information and Knowledge Management, Nov. 4-9, 2002. McLean, VA., pp: 18-25.