

# *Compress and Mine: An Efficient Graph Based Algorithm to Generate Frequent Itemsets*

P. Deepa Shenoy, Srinivasa K. G, Achint O Thomas  
Venugopal K. R, L. M. Patnaik\*

Department of Computer Science and Engineering, University Visvesvaraya College of Engineering.

\* Microprocessor Applications Laboratory, Indian Institute of Science, Bangalore.

{shenoypd@yahoo.com, kgsrinivas@msrit.edu, achnit@ieee.org, vkrajuk@vsnl.com, lalit@micro.iisc.ernet.in}

---

## **Abstract**

Data mining is the extraction of hidden knowledge and frequent patterns from very large databases. Associations among items of the same transaction leads to correlation and identification of frequent itemsets. Generating frequent itemsets at multiple levels provides more specific and concrete knowledge from data. Once the data becomes huge, it increases number of I/O scans. Therefore, the usage of compression techniques on the databases minimizes the number of I/O scans and also reduces space complexity. The objective of this paper is to apply compression on the databases and identify frequent itemsets at single and multiple levels, in a transaction database using an efficient graph based approach. This algorithm scans the database once to find frequent one items and scans it again to construct an association graph to derive the path matrix, then traverses the path matrix to generate frequent itemsets. We observe that this technique is an improvement over the earlier graph based method, both in time complexity and space complexity.

**Keywords:** Data Mining, Frequent Itemsets, Multiple Level Frequent Itemsets, Graph Based Approach, Compression.

---

## **1. Introduction**

Data mining has attracted a great deal of attention in the information industry in the recent years due to wide availability of huge amount of data. The information and knowledge gained from such data can be used for applications ranging from business management, production control and market analysis to engineering design and science exploration. Mining frequent patterns has been a focused topic in data mining research in recent years, with the development of numerous interesting algorithms for mining associations, partial periodicity, constraint-based frequent mining, associative classification and emerging patterns. The popular area of application is market basket analysis, which studies the buying habits of customers by searching for sets of items that frequently appear together.

It is important for the transaction database worked on to be as large as possible. However due to hardware restrictions a breakeven point has to be decided. If we can store more information in the same amount of space, our model constructed would be more accurate and predictive. Thus, we propose to compress the database so that it occupies less space and can reside in the main memory. This serves two purposes. First, the database is compressed, so it occupies less space and hence more transactions can be included. Second, the compressed database stored in main memory, is much faster than secondary memory and access time improves.

A Graph can be applied to represent any physical situation involving discrete objects and a relationship among them. Graph representation can be used efficiently to associate items in a transaction to generate frequent itemsets in a very large database. The objective of this paper is to generate frequent itemsets, from very large databases using an Efficient Graph Based Approach (EGBA) and to reduce space complexity of the algorithm using compression techniques. EGBA method adopted in this paper is not only computationally efficient but also requires very less storage space because of

compression. Again since the graphs in these scenarios involve hundreds of millions of nodes and even more edges, highly space-efficient data structures are needed to fit the data in memory. In this paper we also store the path matrix in a compressed form.

This paper is organised as follows. In section 2 terms related to frequent itemsets and compression are introduced. Section 3 presents a review of related work. The problem of mining associations using Graph Based approach and the algorithm is addressed in section 4. Examples illustrating the algorithm are given in section 5. Numerical results are analysed in section 6. Section 7 presents the conclusions.

## 2. Definitions

**Frequent Itemsets:** Let  $I = \{i_1, i_2, i_3, \dots, i_m\}$  be a set of literals called items. Let  $DB$  denote a set of transactions where each transaction  $T$  is a set of items, such that  $T$  is a subset of  $I$ . Associated with each transaction is a unique identifier, called Transaction Identifier (TID). A transaction  $T$  is said to contain  $A$  iff  $A \subseteq T$ . A set of items is referred to as an itemset. An itemset that contains  $K$  items is a  $K$ -itemset. The frequency of occurrence or support count of an itemset is the number of transactions that contain the itemset. An itemset satisfies minimum support ( $min\_sup$ ) if the frequency of occurrence of the itemset is greater than or equal to threshold. If an itemset satisfies  $min\_sup$ , then it is a frequent itemset.

**Multiple Level Frequent Itemsets:** Generating frequent itemsets at different levels of abstraction is multiple level frequent itemsets. This requires progressively deepening the mining process for finding refined knowledge from data. The multiple level frequent itemset is used to find frequent itemsets at the top-most level and then deepen the mining process to extract frequent descedents at the lower levels.

**Graph Based Approach:** A graph  $G = (V, E)$  consists of a set of objects  $V = \{v_1, v_2, \dots\}$  called vertices and another set  $E = \{e_1, e_2, \dots\}$ , whose elements are called edges such that each edge  $e_k$  is identified with an ordered pair  $(v_i, v_j)$  of vertices. In the graph based approach vertices represent the items and edges represent the association between various items to generate frequent itemsets. Consider an example of a transaction database shown in Fig. 1. The first column represents the transaction TID and the second column gives the items purchased in a particular transaction.

TID	Items
1	a b c
2	b c

Fig. 1. Transaction database.

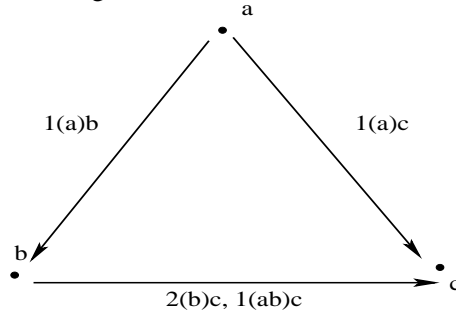


Fig. 2. Graph of the transaction database shown in Fig. 1.

In the graph shown in Fig. 2, vertices  $a$ ,  $b$ , and  $c$  represent various items in a transaction database. Edges give the association between these items. For the transaction TID 1, the item  $a$  is associated with the item  $b$ , so an edge is drawn between the vertices  $a$  and  $b$  with a label  $1(a)b$ . In this label

$1(a)b$ , 1 gives the count of occurrence of  $a$  and  $b$ ,  $b$  is the destination vertex and  $a$  is the source vertex. The item  $b$  is associated with the item  $c$ , an edge is drawn between  $b$  and  $c$  with a label  $1(b)c$ , and  $a$  is associated with  $c$ , so an edge is drawn from  $a$  to  $c$  with a label  $1(a)c$ . In the same transaction  $a$  is associated with  $c$  through the item  $b$ , so the edge between  $b$  and  $c$  is appended with a sequence  $1(ab)c$ . The transaction TID 2 begins with  $b$  and is associated  $c$ , the count in the sequence  $1(b)c$  of the edge between  $b$  and  $c$  is updated to 2.

**Path Matrix:** The path matrix of a graph  $G$  with  $n$  vertices is an  $n$  by  $n$  symmetric matrix  $X = [x_{ij}]$  where every element  $x_{ij}$  will have an entry, if there is an edge between vertex  $i$  and vertex  $j$ . In our graph based approach the path matrix is used to represent the association between various items in the database and it is traversed to generate frequent itemsets. The path matrix for the graph of Fig. 2 is represented in Fig. 3. In path matrix memory is allocated only when the itemset is present. For example, in Fig. 2 lower diagonal matrix is empty. So the memory is allocated only for upper diagonal matrix. Therefore memory is allocated only for  $\frac{n}{2} * \frac{n}{2}$  matrix.

Items	a	b	c
a		$1(a)b$	$1(a)c$
b			$2(b)c, 1(ab)c$
c			

Fig. 3. Path matrix of the graph Fig. 2.

**Compression:** It is a technique of making data occupy less space than required, by means of certain algorithms. The compression algorithm employed in this paper uses the concept of fourier mathematics to transform data between the amplitude and frequency domains.

**Harmonic Analysis to Compress Data:** Consider a function  $f(x)$  specified through a table of values of  $x$  and  $y = f(x)$ . According to the property of definite integrals, the mean value of  $\phi(x)$  over the interval  $(a, b)$ , is given by  $[\phi(x)] = \frac{1}{b-a} \int_a^b \phi(x) dx$ . In Fourier mathematics, the Euler formulae for Fourier co-efficients are given as, with the integral limits ranging from  $a$  to  $a+2l$ .

$$a_0 = (1/l) \int f(x) dx = 2[[f(x)]], \quad a_n = (1/l) \int f(x) \cdot \cos\left(\frac{n\pi}{l}x\right) dx = 2[[f(x) \cdot \cos\left(\frac{n\pi}{l}x\right)]]$$

$$b_n = (1/l) \int f(x) \cdot \sin\left(\frac{n\pi}{l}x\right) dx = 2[[f(x) \cdot \sin\left(\frac{n\pi}{l}x\right)]]$$
, Where  $n = 1, 2, 3, \dots$

Here  $[[\phi(x)]]$  is the mean value of  $\phi(x)$  over interval  $(a, a+2l)$ , which is of length  $2l$ . The fourier expansion of a function  $f(x)$  is given as,  $f(x) = \frac{a_0}{2} + \sum a_n \cdot \cos\left(\frac{n\pi}{l}x\right) + \sum b_n \cdot \sin\left(\frac{n\pi}{l}x\right)$ . Where  $(1 \leq n \leq \infty)$ , In this expansion of  $f(x)$ , the term  $(a_1 \cos(x) + b_1 \sin(x))$  is called the fundamental harmonic. Since  $f(x)$  is the summation of series of harmonics, it is thus possible to break a function  $f(x)$  into a series of sine and cosine terms.

### 3. Related Work

The problem of mining association between set of items in large databases was first introduced in [4]. The Apriori algorithm in [5] entails number of iterations to compute large itemsets. Hash based algorithm described in [6] is effective in generating of candidate sets for large two itemsets. Study of mining association rules from single level to multiple levels was introduced in [7]. These methods adopt candidate generation-and-test approach which is expensive, especially when encountering long and numerous patterns. The problem of mining association rules on dynamic databases using evolutionary approach was introduced in [1]. The application of an efficient hash based algorithm to generate frequent web access patterns was discussed in [2]. Dynamic subspace clustering on high dimensional databases was applied in [3].

A new methodology called frequent pattern growth introduced by J. Han and J. Pei [8, 9] mines frequent patterns without generating candidates. It adopts a divide and conquer method to project

databases based on the recently generated frequent patterns and grow longer patterns. Efficient data structures developed for database compression tries to eliminate the candidate sets which reduces the size of the database to be scanned iteratively. The FP-tree is extended to mine frequent closed itemsets in the algorithm CLOSET [10]. Pincer-search method for discovering the maximum frequent set [11], combines bottom-up and top-down searches. The method detects a maximal frequent set very early and hence may reduce the number of passes.

The tree projection method introduced in [12] proposes a database projection technique for generating frequent itemsets. The constraint pushing technique developed for Apriori-Based mining [13] can be applied for pattern growth mining. The frequent pattern mining has been improved to incorporate interesting constraints in [14]. Constrained frequent pattern growth integrates constraint pushing into mining process. A Graph Based approach to analyse a large database to generate frequent itemsets using bit vector for each item in the database is proposed in [15]. This approach is simple, but the memory requirement becomes very large when the size of the database increases.

#### 4. Efficient Graph Based Algorithm (EGBA) for Mining Frequent Itemsets

Given a very large database  $DB$  consisting of a set of transactions  $\{t_1, t_2, t_3, \dots, t_m\}$  where a transaction  $t_i$  is represented by a set of characters for a single level and by a set of encoded strings for multiple level, the objectives are to (i) Use compression technique to accommodate large database, (ii) Generate frequent itemsets at single and multiple levels.

##### 4.1 Algorithm EGBA

1. a) Partition the database  $DB$  into  $x$  segments. Compress all segments and store in main memory.  
b) Decompress each segment one at a time.
2. The decompressed segment is scanned to generate frequent one-items using minimum support. The minimum support is based on the probability of occurrence of the items in the database. Compress the database segment wise and store it in the main memory.
3. The database is decompressed and scanned to construct the association graph and path matrix.
4. The path matrix is traversed to find frequent itemsets.
5. Repeat steps (2), (3) and (4) to generate multiple level frequent itemsets.

##### Step 1a) Algorithm for Compression:

1. Initialization  
 $f\_len = \text{Length of the database}; \theta = 2\pi / (f\_len - 7), count = 1.$
2. Generation of Harmonics  
Scan the database transaction by transaction.  
For each transaction, use ASCII values of all the items to form  $a_0$ .  
*While* (not eof)  
    Scan the four items of next transaction, Use the ASCII values of all the items to form  $a_1$ .  
    Scan the next four items of the transaction, Use the ASCII values of all the items to form  $b_1$ .  
     $count = count + 1; hrnm = count$  {set the number of harmonics}.
3. Generation of data  
 $count = 0; n = 0.$   
*While* ( $count < hrnm$ )  
    (a)  $icnt = 1, p\_sum = a_0.$   
    (b) *While* ( $icnt < 8$ )  
        (1)  $p\_sum += a[icnt - 1] * \cos[icnt * n].$   
        (2)  $p\_sum += b[icnt - 1] * \sin[icnt * n].$   
        (3)  $n += \theta.$   
    (c) Save  $p\_sum$  across four characters to compressed data file.

### Step 1b) Algorithm for Decompression:

1. Initialization

$f\_len = \text{Length of the compressed file}; \theta = 2\pi / (f\_len * 7), a_0 = count = n = 0;$

For all  $i$  ( $1 \leq i \leq n$ ),  $a[i] = b[i] = 0$ .

2. Generation of Harmonics

While (not eof)

Scan the four items  $c_1, c_2, c_3, c_4$ . Use the ASCII values  $c_1$  through  $c_4$  to form  $c$ .

$a_0 += c; count = 1;$

While ( $count < 8$ )

(a)  $a[count-1] += c * \cos(count * n)$ .

(b)  $b[count-1] += c * \sin(count * n)$ .

(c)  $count = count + 1;$

$n += \theta$ .

For  $i$  ( $1 \leq i \leq n$ ),  $a[i] = b[i] / = 2\pi / \theta$ .

3. Construction of original database

Convert  $a_0$  to four characters and store in database file.

For  $i$  ( $1 \leq i \leq n$ ), Convert  $a[i]$  and  $b[i]$  to corresponding four characters and store in the database file.

### Step 2: Generation of Frequent Items.

[a] Scan the database to find frequent one-items and store them in a set called as Frequent item Set.

[b] Count Total number of Frequent Items ( $n$ ).

[c] Assign unique identification number (token) to each frequent one-item starting from 0.

[d] Compression algorithm is applied on these frequent one-item sets.

### Step 3: Construction of Association Graph to Derive Path Matrix.

[a] Decompress the database and scan the database again, to prune non frequent items.

[b] Construct a graph  $G$ , having the vertex set  $V = \{v_0, v_1, \dots, v_{(n-1)}\}$  in which every vertex corresponds to a frequent item having  $id = \{0, 1, \dots, (n - 1)\}$ . Declare a path matrix of size  $n \times n$ .

[c] Add the transaction having only frequent items to the graph  $G$ . For every frequent item in the transaction, starting from the first frequent item,

(i) Add an edge from source vertex to target vertex. ( source vertex corresponds to the first item in the transaction and target vertex corresponds to the second the vertex in the transaction).

(ii) Label the edge with {count, (source vertex) target vertex}.

(iii) Generate all the subsequences for every transaction and appropriately append the labels.

(iv) Represent the above information in the path matrix.

(v) Repeat the steps from (i) to (iv) till all the transactions are covered.

[d] Once the database has been scanned for frequent itemsets, a number of counters have to be maintained to keep track of the items. The compression algorithm can be applied on these counters to reduce the space requirements.

### Step 4: Mining Frequent Itemsets from the Path Matrix.

Traverse the path matrix rowwise, if the count associated with a sequence is greater than or equal to the minimum support, display it as the frequent itemset.

## 5. Example

Consider an example of a transaction database shown in Fig. 4. The first column represents the transaction TID. The various items purchased in a particular transaction are given in the column 2. This algorithm scans the database one transaction at a time.

TID	Items
1	a c d
2	b c e
3	a b c e
4	b e

Fig. 4. Transaction Database

Frequent Items	Support	Token
a	2	0
b	3	1
c	3	2
e	3	3

Fig. 5. Frequent Items ( $min\_sup = 2$ )

In step 2, the database is scanned to find frequent one-items. Taking minimum support as 2, the frequent one-items generated are  $a, b, c$  and  $e$ . These frequent one-items are given unique identification numbers(token) starting from zero. Fig. 5. gives frequent one-items, their support and token.

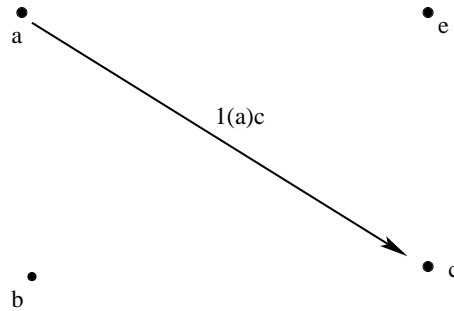


Fig. 6. Graph for the transaction TID 1.

In step 3, the association graph is constructed to compute the path matrix. Fig. 6 shows the graph for the transaction with TID 1 and the corresponding path matrix is shown in Fig. 7. The four frequent items  $a, b, c$  and  $e$ , are assigned four vertices in the graph. Since  $a$  is the beginning of the transaction and it is associated with  $c$ , an edge is drawn between vertices  $a$  and  $c$  with a label  $1(a)c$ . In this label, 1 represents the count,  $c$  represents the destination and  $a$  represents the source. Fig. 7 shows the path matrix for the transaction TID 1.

	a	b	c	e
a			$1(a)c$	
b				
c				
e				

Fig. 7. Path matrix for the transaction TID 1.

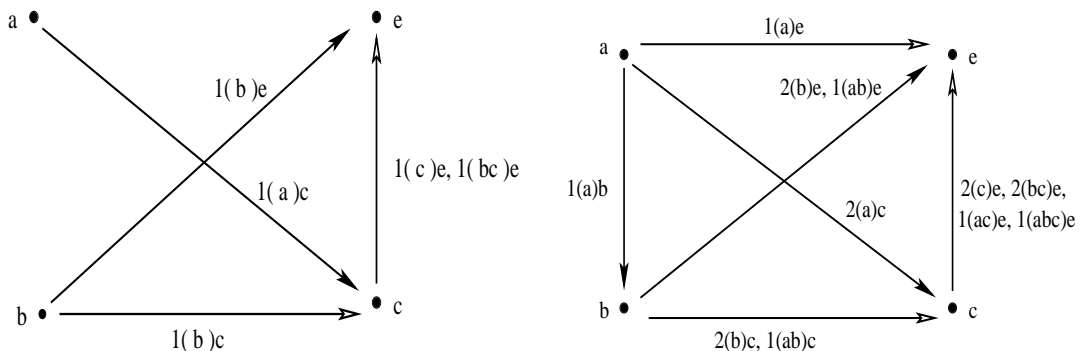


Fig. 8. Graph for the transaction TID 1 and TID 2. Fig. 10. Graph for the transaction TID 1, TID 2 and TID 3.

	a	b	c	e
a			1(a)c	
b			1(b)c	1(b)e
c				1(c)e,1(bc)e
e				

Fig. 9. Path matrix for TID 1 and TID 2.

	a	b	c	e
a		1(a)b	2(a)c	1(a)e
b			2(b)c,1(ab)c	2(b)e,1(ab)e
c				2(c)e,2(bc)e, 1(ac)e,1(abc)e
e				

Fig. 11. Path matrix for TID 1, TID 2 and TID 3.

Fig. 8 shows the graph for the transactions TID 1 and TID 2. The first item is  $b$  and it is associated with the item  $c$ , an edge is drawn between  $b$  and  $c$  with a label  $1(b)c$  and the item  $b$  is also associated with the item  $e$ , so an edge is drawn from  $b$  to  $e$  with a label  $1(b)e$ . The item  $c$  in this transaction is associated with the item  $e$ , so an edge is drawn between  $c$  and  $e$  with a label  $1(c)e$ . The items  $b$ ,  $c$  and  $e$  are associated, which is represented by a label on the edge from  $c$  to  $e$  as  $1(bc)e$ . The corresponding path matrix is shown in the Fig. 9. The transaction TID 3, starts with the item  $a$ , and is associated with the item  $b$ . An edge is drawn from  $a$  to  $b$  with a label  $1(a)b$ . The item  $b$  is associated with item  $c$ , so the edge between  $b$  and  $c$  is incremented to  $2(b)c$  and so on. The transaction is covered for all the subsequences and the graph is appropriately updated. The graph and the corresponding path matrix are shown in Fig. 10 and Fig. 11. The transaction TID 4 has two items  $b$  and  $e$ . The edge between  $b$  and  $e$  is updated to  $3(b)e$ . The graph and the corresponding path matrix are shown in Fig. 12 and Fig. 13.

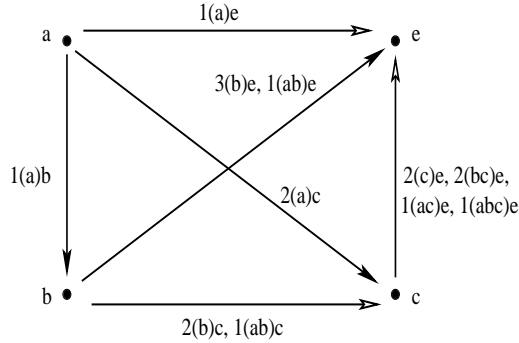


Fig. 12. Graph for the transaction TID 1, TID 2, TID 3 and TID 4.

	a	b	c	e
a		1(a)b	2(a)c	1(a)e
b			2(b)c,1(ab)c	3(b)e,1(ab)e
c				2(c)e,2(bc)e, 1(ac)e,1(abc)e
e				

Fig. 13. Path matrix for the transaction TID 1, TID 2, TID 3 and TID 4.

In step 4, mining is performed to generate frequent itemsets. The elements of the path matrix in Fig. 13 are traversed along the rows to extract the sequences which satisfy the minimum support. The element extracted from the first row of the path matrix is  $2(a)c$  as it satisfies the minimum support of 2 and  $ac$  is a frequent two itemset. The elements extracted from the second row are  $2(b)c$  and  $3(b)e$ , so  $bc$  and  $be$  are frequent two itemsets. The elements extracted from the third row are  $2(c)e$  and  $2(bc)e$ , so  $ce$  is a frequent two itemset and  $bce$  is a frequent three itemset.

**Multiple Level:** In multiple level, each item is represented by a set of relevant attributes. Each attribute describes a certain concept and these relevant attributes form a set of concepts. For the

database shown in the Fig. 14, food items can be described by category, content and brand. Here, category is the first level, content is the second level and brand is the third level. For example, in 'Amul salted butter', butter is category, salted is content and Amul is the brand. These items can be represented by an encoded string as 111, 121 etc. In the string 111, the first bit is level 1 representing the category, the second bit is level 2 representing the content and the third bit is level 3 representing the brand. The algorithm EGBA performs steps 2 to 4 three times to generate frequent itemsets at three levels. The frequent items at level one as shown in Fig. 15 are represented as 1\*\*, 2\*\*, 3\*\*. They are assigned tokens as 0, 1 and 2 respectively to construct the graph and path matrix at level one. The same procedure is adopted for level two and three in which the various items are represented as 11\*, 12\*, 22\* and 32\* for level two and 111, 121, 222 and 324 for level three.

TID	Items
1	111 121 222 324
2	121 222 324

Fig. 14. Database for multiple level

Frequent items	Support	Token
1**	2	0
2**	2	1
3**	2	2

Fig. 15. Frequent items at first level with  $min\_sup = 2$

## 6. Performance analysis.

The algorithms Primitive Association Pattern Growth (PAPG) and Multiple Level Association Pattern Growth (MLAPG) to generate single and multiple level association patterns are used to compare with our algorithm EGBA. Simulations were conducted on a system of physical memory capacity 128 MB and a PentiumIV processor of speed 1.5 GHZ. The transaction data in the form of set of characters for single level and binary encoded strings of desired length for multiple level was generated using a synthetic data generator. It is used to generate transactions upto one million, the number of items are 1000, the average size of the transactions is eight, the maximum number of items in a transaction are 25 and average size of maximal potentially large itemsets is four. For multiple level, the number of levels used is three. The code is implemented in java. Transaction database of required size is stored in a text file and is accessed by PAPG, MLAPG and EGBA algorithms.

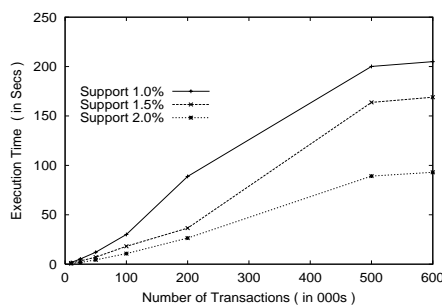


Fig. 16. Transaction Database Vs Execution Time.

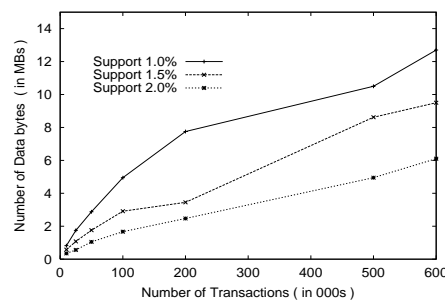


Fig. 17. Transaction Database Vs Data bytes.

The performance of the algorithm is compared with the standard Apriori algorithm and PAPG for mining frequent item sets. Algorithm EGBA shows better performance than Apriori in terms of execution time. Apriori takes more time for mining multiple level frequent itemsets because of more number of I/O scans and the number of I/O scans is reduced in EGBA using compression technique. The database *DB* of various sizes were used. The execution time for various databases of size 10k to 600k transactions for supports of 1%, 1.5% and 2.0% are shown in Fig.16. The execution time for the support 1%, increases steeply as the database size changes from 100k to 200k. After 200k, the increase

in the execution time is linear with the change in the database size. The execution time decreases as the support increases from 1% to 2%, and the execution time increases linearly for various databases at support 2%.

Fig. 17 shows the memory requirements in data bytes as the database size varies from 10k to 600k for various supports. The memory requirement at 2% support increases linearly whereas at 1.5% support, the memory requirement changes rapidly after 200k. The relative execution time of PAPG/EGBA, and the relative execution time of Apriori/EGBA for generating frequent itemsets at single level for various databases at 1% support is shown in Fig. 18. From Fig. 18 it is evident that EGBA outperforms both Apriori and PAPG with respect to execution time.

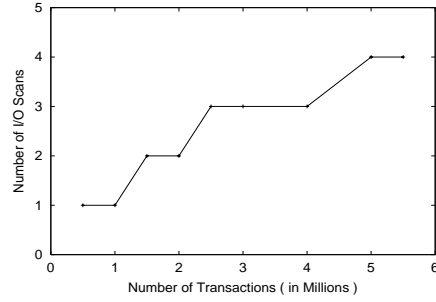
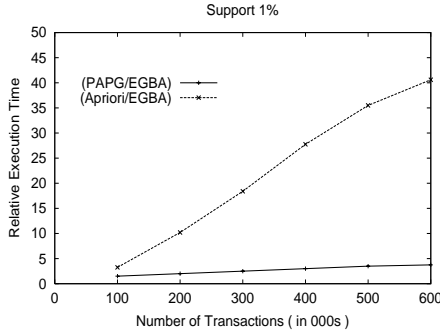


Fig. 18. Transactions Vs Relative Execution Time. Fig. 19. Number of disk reads versus transactions.

It is evident that the algorithm EGBA is faster than PAPG for various sizes of database. In PAPG, if  $L_1$  is the number of frequent one-items, bit vector for each item, where number of bits is equal to the number of transactions is constructed. It performs  $|L_1| \times |L_1 - 1| / 2$  logical AND operations on the bit vectors to construct an association graph to generate two-itemsets. In the  $k^{th}$  iteration ( $k > 2$ ), PAPG performs  $(k - 1) \times |L_k - 1| \times q$  logical AND operations to find frequent  $k$ -itemsets, where  $q$  is the average out-degree of each item in the association graph. EGBA scans the database for the second time to construct an association graph from which it builds the path matrix, and the size of the path matrix is  $n \times n$  in which various associations are stored. The size of the path matrix is fixed by the frequent one-items, so the memory requirement of the path matrix of EGBA is smaller than that for PAPG. The number of I/O scans decreases significantly due to the compression of the database. This allows a considerable number of transactions to reside in the main memory and thereby reducing the number of disk reads leading to low time complexity. Figure 19 shows the number of disk reads versus the number of transactions. Thus our algorithm EGBA takes care of reducing time complexity, space complexity and I/O complexity.

## 7. Conclusions

We propose an efficient graph based algorithm EGBA to generate frequent itemsets at single and multiple levels for very large databases. Extensive simulations have been conducted using synthetic data. The approach includes four phases, compression/decompression, generation of frequent one-items, construction of association graph and building path matrix and lastly traversing the path matrix (mining) to generate frequent itemsets. The algorithm is efficient in generating frequent itemsets and it is faster than Apriori and PAPG. It uses very small storage space and can handle very large databases. The usage of compression in mining reduces both time complexity as well as number of disk reads which in turn increases the efficiency of the algorithm. The concept used in our algorithm(EGBA) can be extended to dynamic and distributed databases.

## References

- [1] P. Deepa shenoy, Srinivasa K. G, Venugopal K. R and L. M. Patnaik, “*An Evolutionary Approach for Mining Association Rules on Dynamic Databases*”, In Proc. of PAKDD - 2003, Advances in Knowledge Discovery and Data Mining, LNAI, April 2003, pp. 325 -336.
- [2] P. Deepa shenoy, Srinivasa K. G, K. R. S. Sharma, Pratap R, Venugopal K. R and L. M. Patnaik, “*Hash-Mine: An Efficient Web Log Mining Algorithm to Discover Frequent Access Patterns*”, In Proc. of ITPC - 2003, Vol. 1, Nepal, May 2003, pp.160-167.
- [3] P. Deepa shenoy, Srinivasa K. G, Mithun M. P, Venugopal K. R and L. M. Patnaik, “*Dynamic Sub-space Clustering for Very Large High Dimensional Databases*”, In Proc. of IDEAL - 2003, LNCS 2690, Hongkong, March 2003, pp. 850-854.
- [4] Agrawal R, Imielinski T and Swami A., “*Mining Association Rules between Sets of Items in Large Databases*”, In Proc. of ACM-SIGMOD Intl. Conf. on Management of Data. Washington D.C., USA. May 1993. pp. 207-216.
- [5] Agrawal R and Srikant R., “*Fast Algorithms for Mining Association Rules*”, In Proc. of the Intl. Conf. on Very Large Data Bases. Santiago, Chile. Sept. 94. pp. 478-499.
- [6] Park J S, Chen M S and Yu P., “*An Effective Hash based Method for Mining Association Rules*”, In Proc. of the ACM-SIGMOD Intl. Conf. on Management of Data. San Jose, CA. May 1995.
- [7] Jiawei Han and Yongjian Fu., “*Discovery of Multiple-level Association Rules from Large Databases*”, In Proc. of VLDB Conf., Zurich, Seitzerland. September 1995. pp. 420-431.
- [8] Jiawei Han, J.Pei and Y. Yin., “*Mining Frequent Patterns without Candidate Generation*” In ACM-SIGMOD Conf., Dallas, Texas, May 2000, pp. 1-12.
- [9] Jiawei Han and Jian Pei., “*Mining Frequent Patterns by Pattern-Growth: Methodology and Implication*”, In Proc. of Intl. Conf. on SIGKDD. December 2000. pp. 30-36.
- [10] J. Pei, J. Han and R. Mao, “*CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets*” In DMKD’00, Dallas, TX, May 2000. pp. 11-20.
- [11] Lin D and Kedem Z. M., “*Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set*” In Proc. of sixth European Conf. on Extending Database Technology, 1998
- [12] R. Agrawal, C.Agrawal and V.V.V. prasad., “*A Tree Projection Algorithm for Generation of Frequent Itemsets*”, In Journal of Parallel and Distributed Computing, 2000.
- [13] R. Ng, L. V. S. Lakshmanan, J. Han and A. Pang., “*Exploratory Mining and Pruning Optimizations of Constrained Association Rules*”, In SIGMOD’98 Seattle, WA, June 1998, pp. 13-24.
- [14] J. Pei, J.Han and L. V. S. Lakshmanan., “*Mining Frequent Itemsets with Convertible Constraints*”, In ICDE’01, Heidelberg, Germany, April 2001.
- [15] Show-Jane and Arbee L P Chen., “*A Graph Based Approach for Discovering Various Types of Association Rules*”, IEEE Trans. on Knowledge and Data Engineering. Sept/Oct 2001.