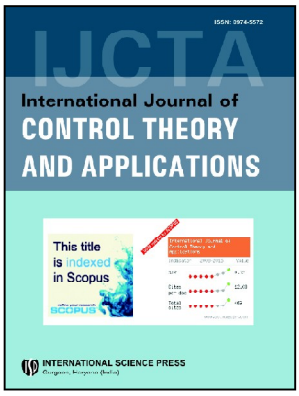

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 11 • 2017

Frequent Pattern Sub-space Clustering Optimization (FPSSCO) Algorithm for Datamining From Large Data Base

T. Sheik Yousuf[1] and M. Indra Devi[2]

[1] *Assistant Professor, Department of Computer Science Engineering, Mohammed Sathak Engineering College, Kilakarai, Ramanathapuram, Tamil Nadu*
[2] *Professor, Department of Computer Science Engineering, Kamaraj college of Engineering and Technology, Virudhunagar, Tamil Nadu*

Abstract: Data mining environment give a quick response to the user by fast and correctly pick-out the item from the large database is a very challenging task. Previously multiple algorithms were proposed to identify the frequent item since they are scanning database at multiple times. To overcome those problems we proposed Rehashing based Apriori Technique in which hashing technology is used to store the data in horizontal and vertical formats. Rehash Based Apriori uses hashing function to reduce the size of candidate item set and scanning of database, eliminate non frequent items and avoid hash collision. After finding frequent item sets perform level wise subspace. We instigate Generalized Self Organized Tree based (GSTB) mechanism to adaptively selecting root to construct the path from the cluster head to neighbors when constructing the tree. Our experimental results showthat our proposed mechanisms reduce the computational time of overall process.

Index terms: Sub-space clustering, GSTB (Generalized Self organized Tree Based Cluster Head selection).

I. INTRODUCTION

In recent years all industries, government sector and entertainment applications are manipulate in database for storing and retrieving their information in a large-scale database and also currently the growth of the World Wide Web rate has increased, so that retrieve frequent data from web is a challenging one. Data miningrefers to the process of extracting useful models of data. Sometimes, a model can be a summary of data, or it can be the set of most extreme features of the data. Scanning frequent item is an important topic in data mining and continuously researchers givean new algorithm for solve theseissues [1]. Mining Frequent Pattern (FP) is one of the effective ways to analyze the customer behavior in terms of purchased products. Distributed Parallel algorithm achieves better than apriori, especially in the case of high data volumes and low minimum supports and hence the computation time is high. It will scan the database at multiple times for find the frequent pattern (FP) and it is time consuming process. Next, researcher wants to speed up the apriori process by implementing the distributed environment [2].

The task of discovering all frequent associations in very large sets are quite challenging. Since the search space is exponential in the number of database attributes and with millions of database objects. Thus the problem

of I/O minimization becomes paramount [10].Indistributed environment irregular and imbalance load can be occurred. To avoid those problems use Distributed Parallel Algorithm [7].In paper [31] authorsproposed the load balancing approach parallel algorithm for frequent pattern (FP) mining problem, Load balancing FP-tree (LFP-tree). Moreover, a loading degree function is also developed. The authors use local and global header for load balancing. Our proposed work is frequent patterns are mined using Hashing technique. Manyhashing algorithms are concerned with efficiently determining the set of frequent item sets in a given set of large database. Using this algorithm avoid problems of hash collision, increasing hash table size, double hashing, load imbalance and multiple scans. This paper performs high level overview of frequent pattern mining, Subspace clustering and GSTB tree construction.In frequent item set mining generate frequent item and non-frequent item. But applying hashing non frequent items are eliminated in each level. In our plan of action we integrate hash basedapriori, Sub-space clustering, and generalized self-organized Tree based algorithm.

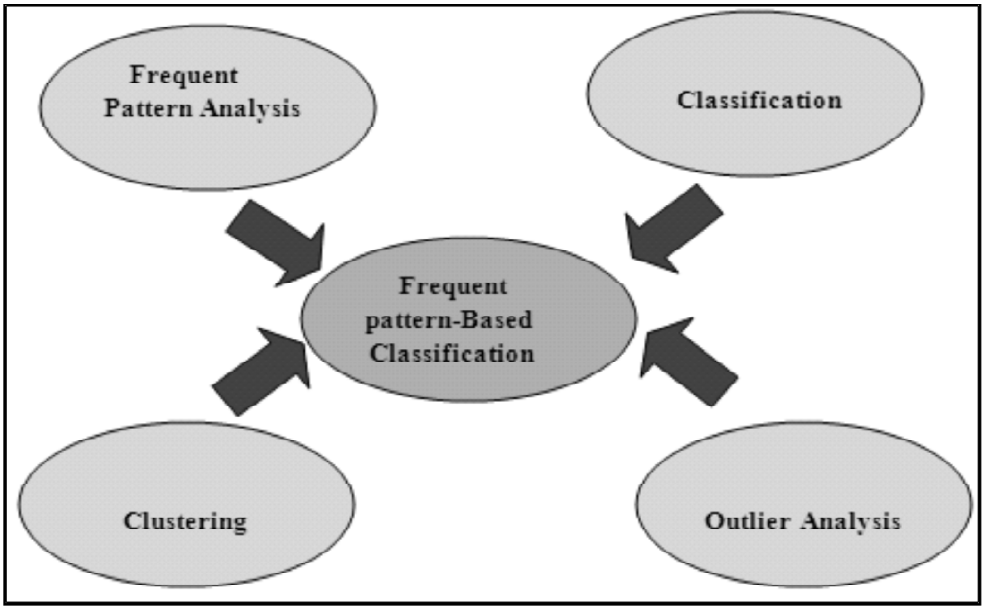

Figure 1: Data mining Themes

We summarize the main contribution of our work as follows:

- In the context of FPM, we perform efficient Rehashing technique, which facilitates an analysis of correlations among transactions to reduce the scanning and computational time. Our technique prevents hash collision and secondary clustering problem across large datasets.
- To implement the above hashing technique by integrating level wise clustering based on subspace clustering
- To validate the effectiveness of our approach, we develop the GSTB tree in which the frequent items applied to construct the tree using adaptively root selection algorithm.

The rest of this paper is systemized as follows: Constituent 2 describes about the previous work under the frequent pattern mining subspace clustering. Constituent3 discussed about problem definition. Constituent 4 chronicles the proposed frequent pattern subspace clustering optimization algorithm. Constituent 5 gives experimental result by comparing existing approach. Constituent 6 concludes the proposed work.Constituent 7 is explained the future work and finally the constituent 8 accommodates what are all the previous authors and their papers are taken for introducing proposed work such as references.

II. RELATED WORK

Finding frequent pattern (FP) is an essential role in mining associations, correlations, regression, time series, sequence and many other interesting relationships among data. Association mining task is introduced and it can be stated as follows: In first step scan the transactional database to make a table of items with item count and their corresponding transactions. Let I = {I1, I2, I3…..Im} be a set of items. Let D be the transactional database where each transaction T is a set of items such that $T \subseteq I$. Each transaction associated with an identifier TID.In transaction T contains X if X ⊆ T, where X is a set of items in I. An association rule form is X=>Y [6].A set of items is referred as an item set. An item set that contains K items i.e. K-itemset. The number of transactions in which a particular item set exists gives the support or frequency count of the item set. If the support of an item set I satisfy the minimum support threshold, then the item set I is a frequent item set i.e. level 1 is generated.Apply hash function to calculate the bucket count. After finds all candidates item sets, calculate bucket count and support value. If there is any collision will occur apply collision resolution function to solve this problem. Thus the frequent pattern mining has become an important data mining task and focused in data mining research. Frequent pattern mining was introduced in [4] [10][12] and [19] for finding frequent patterns using Fp growth and apriori techniques.In paper[4] the performance of data mining in frequent patterns improved, since in each pass set of large k-item sets (L_k) is used to form the set of candidate k+1 item sets (C_{k+1}) by joining L_k with itself on k-1 common items for the next pass. In general the more item sets in C_{k+1}, the higher processing cost of determining L_{k+1}.In apriori algorithm $|C_2| = \left(\frac{|L1|}{2}\right)$, here the step of determining L_2 from C_2 by scanning the whole database and testing each transaction against C_2 is a very expensive. By constructs significantly smaller sized C_2, the DHP algorithm performs well in counting of C_2 much faster than apriori.In this method reduce the size of candidate item set and hence computational time is reduced. The DHP algorithm generates k+1 candidate item sets from large item sets. It uses hashing technique to filter out the unnecessary item sets for the generation of the next set of candidate item sets. It consists of three steps: The first step is to collect a set of large 1-itemsets and constructs a hash table for 2-itemsets. The second step is generates the set of candidate item sets Ck. The third step is same as the second step except it does not use the hash table in determining a particular item set from the candidate item sets.DHP efficiently generating large item sets, reduction in transaction database i.e. datasets size and reducing the number of database scan. Problem of DHP algorithm is increase the size of hash table. In paper [5], authorsproposed H-Bit array hashing function to avoidcollisions. This will give an outcome of the minimal amount of time. It avoids lengthy probing sequence. But the problem is secondary clustering since more than one clustering is not given accurate results. In [6] and [14] PHPalgorithm (Perfect Hashing Pruning) is proposed. In this algorithm, each item set has its own bucket which increases the size of hash table. But it effectively reduces the search space. PHS (Perfect Hashing Scheme) algorithmavoids hash collision and reduces the memory requirements. Hash Based Frequent Items-Double Hashing (HBFI-DH) algorithm uses hashing technology to store items in vertical format and to avoid collision and secondary clustering problem[8] and[15]. In double hashing, less memory space is occupied. Let H (k) be a hash function that maps an item set k to an integer in [0, m-1], where m is the size of hash table. Let it probe position for a value k is given by the function's(kid)=[h_1 (k) +I * h_2 (k)] mod m. where h_2 (k) is obtained from the following equation h_2(k)=[R-(k mod R)] where R is a prime number that should be less than m.$\mathbf{h_1(k).\ h(k)+h_2(k).\ h(k)+2h_2(k).h(k)+\ldots+nh_n(k).h(k)}$. In[9] Transaction hashing and pruning techniques given solution for all previously proposed hashing techniques. THP algorithm overcomes the item set collision problem in DHP algorithm and large hash table problem in PHP algorithm. As a result of THP method, there is some failures occur such as many transactions, huge database, many data and about the datasets. Large sets of frequent item sets describe essentially the same set of transaction. Support counts and bucket counts are arranged in linked list. Papers [18] and [19] explained the concept of Association Rule Mining (ARM). Issues in ARM:

1. Finding minimum support: By calculating minimum support for a frequent pattern, user specified minimum support threshold value. If we place threshold value minimum, infrequentpatterns are evolved. If we place threshold value maximumsome frequent patterns are not retrieved [3].Due to thisoptimization problems occur.
2. Multiple scans in database: Finding frequent item sets in a database, we necessary to scan the database continuously. Themultiple scanning of data leads to:
    1) Wastage of memory: utilizing lot of space to store and retrieve data
    2) Wastage of time: to scan whole database by continuous scanning
3. Performance: performance is varied when number of instructions gets increased. ARM-HDAG [11] is a tree based hashing algorithm. This process terminates when no more item sets exist in the every level of hash table. This process is mainly used for reusability of memory in RAM. When the first level of itemsets is deleted then same memory is allocated to the second level of Candidate keys. By using hash chaining and linked list structure frequent item sets are evolved easily compared to other previous algorithms, hashing is takes place through level by level and equal no of candidate item sets are generated by all levels in a hash table. But this technique is failure for large database.Double hashing technique removes both primary and secondary clustering. In this mechanism new method is required to improve efficiency to insertion, deletion and searching [13] and [15]. Authors proposed an efficient algorithm[22] in which combines three paradigms for improving runtime: A filter-and-refine architecture with a filter step based on weak density monotonicity for pruning the search space, A depth first approach which avoids excess candidate generation on a specialized index structure, Redundancy pruning mine lower dimensional projections in which no redundant higher dimensional cluster was during depth first search. Subspace clustering [16] technique finds interesting subspaces in different subspaces.Traditional methods apply frequent item sets to find dense units. Since these methods are not able to differentiate the density of units in subspaces with different dimensions.Itis not in favor of finding dense units in the sparse subspace or the higher-dimension subspace.In this paper, proposed algorithm is CBNI(Clustering high dimensional data streams Based on N-most interesting Item sets) which finds dense units based on N-most interesting item sets. In [3] Subspace search‘items’ and ‘item sets’ concepts from frequent pattern mining translates to ‘dimension’ and ‘subspace’ respectively. The notion of a ‘frequent item set’ translates in to ‘interesting subspace’. In this technique count index is generated in ascending order.Count index processed in a form of depth-first traversal, the search space is applied for subspace clustering. Paper [17] introduced bottom up algorithm for subspace clustering based on density and grid that computes the dense units in all dimensions and combines these to generate the dense units in higher dimensions.Paper [23] developed an algorithm (termed DFPMT-A Dynamic Approach for Frequent Patterns Mining Using Transposition of Database) for mining frequent patterns which are based on Apriori algorithm and use dynamic function for LCS(Longest Common Subsequence) and also frequent patterns are mined using pattern length.

III. PROBLEM DEFINITIONS

Frequent pattern mining from large datasets is very challenging task in data mining. Lot of methods and techniques were proposed in previously published papers. But still some of the problems arise. Issues are wastage of memory and more time taken for computation. To reduce multiple scanning of database use hash based apriori technique in which reduce the response time, increase accuracy [6]. In case of large dataset, this algorithm is not efficient, Apriori algorithm requires large number of scans and minimum support is provided by user which is uniform or constant for whole transaction.Drawback in this technique is scan database at several times and time consuming process whengenerates candidate item sets. To overcome those problems, hashing technique is introduced. In this technique hash collision may occur due to primary and secondary clustering which means the single hash key assigned two or more values. Main advantage of this algorithm is scans the transaction database once and it

does not produce candidate item sets. Direct Hashing & Pruning (DHP) algorithm is an effective hash based algorithm for candidate item set generation. This algorithm reduces the size of candidate-2 item set so that computational time is reduced. H-Bit Array Hashing algorithm (H-BAH) proposed Quadratic probing to avoid collision problem of DHP algorithm but it increases the size of hash table [15]. Perfect Hashing and Pruning (PHP) algorithm reduces the extra work of DHP algorithm for counting the occurrences of candidate k+1 item set in each bucket. In this algorithm each item set has its own bucket which increases the size of hash table but reduces the search space [14]. Perfect Hashing Scheme (PHS) avoids collision problem. This algorithm uses an encoding scheme to transform large item sets into large-2 item set. The research paper also proposed a variant of PHS algorithm that reduces the memory requirements. Hash Based Frequent Item sets-Double Hashing (HBFI-DH) algorithm uses hashing technology to store database in vertical format which uses for avoid collision and secondary clustering problem [8]. HMFS algorithm takes advantages of DHP and Pincer Search algorithm to reduce database scans.Paper [9] discussed about the transaction hashing and Pruning using minimum support and bucket count. A problem in this algorithm is many transactions and huge database. After completing frequent pattern level 1 apply hash function to find out the bucket count using the following hash function:**H(x, y) = ((order of x)*10+ (order of y)) mod 7** in this hash function 7 is a number of items. So this algorithm is suitable for predefined datasets. To overcome above mentioned problems we proposedrehashing based apriori techniquefor frequent pattern mining. In this algorithm overcomes the following problems: double hashing problem, reduce the hash table size, secondary clustering, insertion, searching, deletion is easy to perform. Using associative array for storing item sets in hash table. In Subspace clustering two approaches are introduced. First one is top down approach and Bottom up approach. Techniques used in clustering are Density based and Grid based. The problem of subspace clustering is given by the fact that there are 2^d different subspaces of a space with ddimensions. If the subspaces are not axis-parallel, an infinite number of subspaces are possible. So we use a density based approach to clustering. A cluster is a region that has a higher density of points than its surrounding region. We propose a count indexing scheme for subspace clustering [31].

IV. PROPOSED WORK

In our proposed frequent pattern Subspace cluster optimization (FPSSCO) translates frequent pattern (FP) mining concept to subspace clustering in which item into dimension, subspace and units. Sub-Space (SS) clustering is a special family of adaptations of clustering approach for high dimensional data[35].Data clustering is an innovative idea in data mining applications. Subspace clustering is the task of extracting group of object in a large dataset that inherits the high similarity object for clustering .For e.g. one organization purchase some goods for their company. File, card reader, scanner, pens, curtain clothes, projectors, notice board, etc. these are the frequent set item 1. From this algorithm calculates frequent item set-2 had some limitation. Frequent items-2 contains both items and itemsetsthat are share common property. Card reader, scanner, projector are in one candidate sets, pen, notice board are the other candidate set. These steps are continued when no further extensions is possible. Frequent item is exponential in large datasets [25]. Apriori algorithm [2] is the best previously proposed algorithm and it uses an efficient candidate item prompting procedure, such that only the frequent item sets at a level are used to construct candidates at the next level. TheDHP (Double Hashing and Pruning) algorithm is to reduce the number of candidate collecting approximate counts in the previous level. PHP (Perfect Hashing and Pruning) algorithmreduces the more number of candidate itemsets. In this algorithm each item set has its own bucket which increases the size of hash table but reduces the search space [9] but the apriorialgorithm required many database passes as the longest item set.In the first pass, it generates the set of all potentially or locally frequent item sets and, in the second pass, it counts their global support. Partition may enumeratestoo many false positives in the first pass i.e., item sets locally frequent in some partition but does not globally frequent.In our proposed work we overcome these problems by FPSSCO algorithm. Let FI be a set of frequent items and T be a database transactions, I be a set of item sets where each transaction has a unique identifier (tid) and contains a set of items. A set of items is also called an item set. An item set with k-items is called a k-item set. A k length subset of an

item set is called a k-subset. An item set is frequent if its support is more than a user-specified minimum support (min-sup)value.

There are 8 main steps in the algorithm

1. Scan the given transactional database to make table of items with transaction id and item sets.
2. I is represented as a set of item sets that denoted as{I1,I2,I3,I4,I5,….In}
3. Generate table L1
4. This is similar to apriori join step. In ith level combined items generates all possible ith level transaction for Ck (candidate itemsets) that using Lk-1itemsets.Then frequency counts are discovered for each combination and generate linked list structure and allocate items in structure.
5. Apply hash function for each item in Ck .If collision occurs apply collision resolution function and then move i -term set to the corresponding Bucket count. If collision not occurs at any one of the item set Go to step 4.
6. End the process when frequent item set is found.
7. Perform Level wise sub-space clusteringfor frequent item set based on their frequency count and then make a table for maximal frequent item sets, closed frequent item sets and frequent item sets.
8. Calculating pattern length of every frequent item sets. Next construct GSTB tree based on the pattern length order avoid full tracery of the database.

Algorithm 1: FPSSCO algorithm

1. **Input:** Database item T, Transaction set.
2. For(k=1;k<=n;k++)
3. FP mining();
4. For each transaction T I.Do begin
5. Apply hashing technique.
6. Obtain all k item set of the transaction and store them in hash table.
7. End for
8. SS clustering();
9. Perform level wise clustering of all frequent item sets. Sort the levels in descending order according to its count value and pattern length.
10. Compute level wise clustering $\forall X_i, Y_i, Z_i$.
11. GSTB();
12. Select the cluster head with maximum pattern length to construct the path.
13. Traverse root node to leaf nodes while comparing the pattern length join leaf node to root.
14. Else
15. Create new set of nodes with pattern length. Insert them into tree according to their pattern length
16. End for
17. End
18. Output: All most frequent pattern with minimum response time and high accuracy.

Algorithm 1 provides working procedure for our proposed system whichreduces the execution time. In this section, we describe several new algorithms for efficient enumeration of frequent item sets. The first step involves computation of frequent items. The next step generates the sub-spaces by applying SS clustering (sub-space clustering), on the frequent item sets. The GSTB tree construction is based on the frequent pattern length in descending order in Main-memory.The algorithms are given below,

1. Frequent pattern mining algorithm using apriority.
2. Sub-space clustering algorithm
3. GSTB algorithm.

Next session describe the overall working procedure of the above algorithm

4.1. Frequent Pattern Mining

In our proposed System FPSSCO algorithm selects the entire frequent M- item set from the high-dimensional database over N items represented by,

$$\Sigma_{M=1}^{N}\binom{N}{M}=2^{N}-1 \tag{1}$$

We use hashing technique for avoid pruning i.e. reduce the non frequent item to simplify the scanning process. The FPSSCO algorithm has multiple transaction of each data set T and their frequent items are FI, and support value (min_sup) is S. Ckis the candidate set for K level. At each step candidate sets are generated from the large data item set of the preceding level. In this phase user given query is scanned from large data set (D1, D2…Dn), and within their transaction (T1,T2..Tn) set also. Each transaction contains transaction id, and item sets in the database .Where the database D is kept normalized and each database record is a <TID, item sets> pair, where TID is the identifier of the corresponding transaction. Each transaction belongs to a transaction set T which is denoted as,

$$\Sigma_{k=0}^{n} tn = T \tag{2}$$

4.1.1. Apriori algorithm using hashing technique

Our hash based apriori technique uses a data structure that directly represents a hash table. This algorithm overcomes some of the problem in the apriori algorithm by reducing the number of candidate k-item sets and reduces multiple numbers of databases scanning. The data structure used here is Associative array. The associative array usually implemented as a hash table. An associative array A_s stores a set of frequentitem sets. Each item set K isassociated with a key i.e. **key(k) ∈ key**. Associative array in hash table supports the following operations:

- A_s.insert (K_s:itemset): $A_s := A_s \cup \{k\}$.
- A_s. remove (e: Key):= \ {K}, where K is the unique element with key (K) = e.
- A_s .find (e : Key): If there is an k ∈ A_s with Key (k) = e, return e; otherwise, return ⊥".

A hash table uses a hash function to compute an index to an array of buckets from which the correct value can be found. In hash table hash function will assigns each item set placed in unique bucket. But in some situation more than one item setshad same bucket. In such situations hash collisions will occur.

4.1.2. Separatechaining

Hashing with separate chaining maintains an array $A_s$in a form of linear linked list. To insert an item set I to the sequence of A_s[h(k)]. To remove an item set with key k which are not frequent ,we scan throughlinked listA_s[h(k)].If

a key k with h (k) = eis encountered, weremoveit from linked list and return ⊥. To find the element with key k, we also scan through A_s[h (k)]. If an element e with h (k) = e is encountered, we return it. Otherwise, we return ⊥.

4.1.3. Open addressing

In open addressing two techniques are introduced: Linear probing and Quadratic probing. All item sets are stored in the bucket array. When a new entry has to be inserted, the bucket count is examined in starting with hashed-to slot to proceeding in some probe sequence.Hash table is inefficient when load factor is increased. In quadratic probing insert may fail if load > ½.

4.1.4. Rehashing: To overcome these problems "Rehashing technique" is introduced. Whenapply hash function to two different item sets, it producethe same key element. Due to this same key value, hash collision will occur in hash table. To avoid such problems collision resolution function is used.

Computing frequent itemset 1: Given the database transaction id and all itemsets generate the database transaction id,itemsets format.Apply hash function to identifyy the frequent item sets ,support value and bucket count .The following equation computes frequent item set 1.

$$\mathbf{H\ (k) = (order\ of\ item\ k)\ mod\ n_s} \quad \mathbf{(3)}$$

The n_s value is determined by using the formula $(2m_s+1)$ wherem$_s$are the number of items in the database. In hashing algorithm items are pruned after generating C_1, pruned item sets are called L1 i.e. level 1 itemsets, Dataset of candidate k+1item set is arranged in vertical format along with thetransaction id and corresponding bucket number. Construct hash table using associative array and linked list which contains bucket count, support value, transaction id in vertical format.

Computing frequent itemset 2: Construct the transactional database for second level frequent itemsets.The following equation computes frequent item set 2.

$$\mathbf{H\ (k) = ((order\ of\ X)*10+order\ of\ Y)\ mod\ n_s} \quad \mathbf{(4)}$$

Where n is a bucket count, when the bucket count is same for more than one itemsets collision will occur.To over come the collision ,collison resolution function is used and hash table size is changed.The following equation describes frequent item set 2 collision resolution function

$$\mathbf{H\ (k)=((order\ of\ X\ )*10+order\ of\ Y)mod\ j_s} \quad (5)$$

$\mathbf{J_s=2*N_s+1}$, wherej is a size of hash table and N_s is initial hash table.

Compute frequent itemset 3: Construct the transactional database for third level baased upon the hash fuction. The following equation describes frequent item set 3.

$$\mathbf{H(k)=((order\ of\ X)*100+(order\ of\ Y)*10+order\ of\ Z)mod\ j} \quad \mathbf{(6)}$$

Where X, Y, Z are the set of item sets. Here, we implement the frequent item,item sets transaction id and bucket count using apriori based hashing technique.

This process is performed until the final frequent item sets were found. Since our major aim is to mining most frequent pattern from the large data base.

Algorithm 2: FP mining using Apriori based hashing

1. Input: D, a database of transactions where all are represented as vertical hash table.
2. Process logic: Finding the frequent item sets.
3. Output: Generating the frequent item sets.

4. begin
5. m_s=0;k=0;
6. Get minimum support, min_sup;
7. //before hash function
8. Generate the database in (Items, Transaction id,) format
9. for all Items I [Dk do
10. //apply hash function
11. Generate new database in(Itemsets,Transaction id,Bucket count, Support count)
12. Increment m;
13. n_s=2*m_s+1;
14. D_k=D;
15. do
16. begin
17. Make a hash table of size n_s.Map items on to the buckets.
18. If collision occurs then use Rehashing technique for increase the size of the hash table j_s=2*N_s+1
19. for all Items I ε [D_k do
20. begin
21. Generate a subset of items.
22. End.
23. Find common transaction between the subsets in the kth level.
24. Eliminate the subset<=min_sup.
25. D_k = Items >=min_sup.
26. Increment k.
27. End until frequent item set is found.
28. End.

4.2. Subspace Clustering

Subspace clustering is the clustering mechanism to detect all clusters present in all subspaces. It is an efficient approach for clustering high dimensional data. In subspace clustering object may be member of more clusters over different subset of the attributes. Sub-Space cluster give a different projection in each dimension. Varies clustering techniques were introduced in the database application. They are CLARANS, BIRCH, DBSCAN, CLIQUE, DENCLUE and OPTICS. These clustering algorithms are designed to provide scalability in high dimensional database.This paper proposes level wise subspace clustering. In this algorithm each attribute represents an item and each subspace cluster consists of item set which contains number of items. Items are representing the attributes of the subspace. It uses level wise algorithm, starting from the clusters in one dimensional subspacetowards joins (k-1) dimensional clusters. In this paper, frequent patterns consider in to a various sub-spaces clusters. We retrieve the most frequent item from various space clusters.

Table 1
Elucidated frequent patterns (FP) mining concept into Sub-Space Search

Frequent pattern mining	*Subspace clustering*
Item	Dimension (attribute)
Item set	Subspace (set of attribute)
Interesting	subspacefrequent item set

4.2.1. Level wise subspace clustering

In this section, we describe the synthesis of candidate clusters from the cluster-projections on individual attributes (computed as described in Section 4.2). The central idea is that a cluster on a set of attributes induces a sub-cluster on any subset of the attributes (monotonicity property). The monotonicity property follows directly from the definition of a cluster. We also exploit the fact that we want to compute clusters over the set of all attributes $\{A_1 \ldots\ldots A_n\}$. Informally, we start with cluster-projections on and then extend them to clusters over $(A_{1,}$ A2) then to clusters over (A_1, A2, A3) and soon. In level wise subspace clustering inherits the frequent item sets from the above process.

Algorithm 4: level wise subspace clustering algorithm

1. Input :{(L_1, H_1), (L_1, H_2)……(L_k,H_k)}
2. Output: Interesting Subspaces.
3. Algorithm:
4. Look up Hash table hTable ← { }
5. //An entry h_k in htable is {sum, U,S}
6. For j=1 to k do
7. Du → finds dense units (j)
8. // Get dense units in dimension j.
9. u^1= (L_j^1, H_j^1) (or) (L_k^1, H_k^1) in same as
10. u^R= (L_j^R, H_j^R) (or) (L_k^R, H_k^R).
11. For all Du = {Sum,U} Du do
12. //Frequent item set find based on descending order.
13. If (Selectivity of U e" min-selectivity)then
14. Append dimension k to subspace S.
15. else
16. Add new entry {U} to the hTable.
17. end if
18. end for
19. end for
20. For all set of entries {H_k,……H_1) ∈ hTable do
21. until all of C_k has been examined.
22. End for.

Let A= {A1, A2, A3, An) be a set of attributes i.e. Subspaces, total ordered domain is S = (A1×A2×…×An).We partition database in to dimensional units u. The units (group of items) are obtained from transaction set. Each unit's u is the intersection of one interval from each attributes. Dense units du is an item sets and j is a dimension which represents the number of item sets present in dense units. It has the form {*u1, u2… ud}* where ui= [li, hi] is a portioning element of *Ai*. We say the point *V = {V1, V2, V3..Vd}* is contained in a unit u= {u1, u2…ud} if li < Vi<hi for all u. The selectivity of the unit is defined to be the fraction of total data point contained in the unit. We similarly define unit in all sub-spaces of the original d-dimensional space. Consider the projection of the dataset V into At1 × *At2* × … ×*Atk*, where *K< d and ti<tj if i<j.* A unit in the sub-space is the

intersection of an interval from each of the k-attributes. A zone in q- dimension is an axis-parallel k-dimensional set. We are only interested in those zones that can be expressed as unions of units: henceforth all references to a zone mean such unions. A zone Z is contained in a cluster C if $Z \cap C = Z$. A zone Z contained a cluster which is said to be a maximal zone and then there is no proper subset of Z contained in C. A minimal description of a cluster is a non-frequent item does not cover in the cluster within the in our plan of action, Sub-Space clustering translating all frequent pattern mining parameter into Sub-Space parameter. At this point frequent items are said to be units, item set are called as sub space and their item in the data set are said to be dimension.

4.2.2. Algorithm Description

This algorithm identifying the subspace in the cluster, identifying the cluster, and find out the maximal frequent item in the cluster. The algorithm proceeds level by level. It first determines maximal frequent item sets by making a pass over the data. The maximal frequent item sets are descending order of the units with their count and pattern length. Cluster node is selected based on their maximum pattern length. This CH selects their neighbor node by comparing next level of pattern length. Two item are contain same pattern length means they placed left and right side of the cluster head. In this process sort the entire frequent sets based on their frequent pattern length. Maximum pattern length count will be placed in the top of the tree, so the computation time is reduced for large datasets. Subspace clustering only considers frequent item sets andnon-frequentitem sets are eliminated in hashing technique.In subspace-Search inherits the input unit from the above process. Input unit U is in D.The result of the above algorithm is used to find the pattern length.In Subspace (SS) Clustering Xi, Yi, Ziis a set of units and its represents Maximal dense units, closed frequent dense units, frequent dense units. The given below formulas are described the pattern length: $X_i = \{x_1, x_2, x_3 \ldots . x_n\}, Y_i = \{y_1, y_2, \ldots\ldots\ldots y_n\}, Z_i = \{z_1, z_2, z_3, \ldots . z_n\}$

where $x_1 \geq x_2 \geq \ldots . \geq x_n, y_1 \geq y_2 \geq y_3 \geq \ldots \geq y_n, z_1 \geq z_2 \geq, \ldots\ldots . \geq z_n$.

$$X_i = \left\{\Sigma_{j=1}^{k} \times 1, \Sigma_{j=1}^{k} \times 2, \ldots \ldots \ldots \ldots \ldots \Sigma_{j=1}^{k} \times n\right\} \tag{7}$$

$$Y_i = \left\{\Sigma_{j=1}^{k} y1, \Sigma_{j=1}^{k} y2, \ldots \ldots \ldots \ldots \ldots \Sigma_{j=1}^{k} yn\right\} \tag{8}$$

$$Z_i = \left\{\Sigma_{j=1}^{k} z1, \Sigma_{j=1}^{k} z2, \ldots \ldots \ldots \ldots \ldots \Sigma_{j=1}^{k} zn\right\} \tag{9}$$

4.2.3. Count Indexing

Count indexing structure only maintains the frequent item sets. Starting from the most frequent pattern and followed by the closed frequent item and frequent item sets are generated. Using this count index structure cluster head (i.e.) root node is predicted. In this technique levels are indexed in descending order for selecting maximal frequent item sets.

4.3. GSTB Tree construction phase

In GSTB, each individual frequent pattern length are stored in the count index table. This algorithm selects the cluster head with respect to the most frequent item set from the count index table. Then we construct the route from parent to child based on their frequent count. Sub-Space clustering with GSTB avoid full tracery of all nodes found in the clustering. Instead of use entire value ranges within their approximation value. Priority queues were maintained in order to generate the most promising candidates in the lattice first. As a result, it becomes possible to avoid the prompting of many relatively low-dimensional subspace clusters and to steer the search towards high dimensional subspace clusters directly. Tree construction is used for connecting interesting pattern in the frequent pattern mining. It is same as the FP-Growth method. Subspace cluster are perceived based

on pattern length in the count index table which is perform in the depth first manner. In frequent item set mining, neighboring nodes are merged if they contain cells that are potentially part of the cluster.

4.3.1. GSTB Tree construction phase

A GSTB-tree has one root node and set of leaf nodes. It maintained a transaction list of each item with their pattern length.GSTB-tree can define path from root node to the end of the tree. Transactions of the items are sorted in a descending order. The main goal of GSTB algorithm is to construct the tree which is based on their pattern length.GSTB perform in the form of iterations. In first iteration, parent node is connected to their child for avoiding multiple scanning. In second iteration GSTB protocol can dynamically changing their root and reconstructing the tree in self-adaptive manner.GSTB is a self-organizing protocol which builds a routing tree from root node towards child node. Assign a root node and then buildGSTB tree based on the transaction list which contains the pattern length. GSTB is a dynamic and parallel protocol, which can change the root and reconstruct routing tree to reduce thescanning process. Therefore a better balanced load is achieved, especially for dense nodes deployed. As a result of hashing technique, the frequent items set among all the items are available in the database. Then Sub-Space clustering output is providing as an input for the GSTB tree construction phase. They give only frequent item sets by comparing with their support value (min_sup).In Sub-Space Clustering cluster this frequent item with respect to their count value and frequent pattern length. Maximum number of pattern length as a cluster head of GSTB tree and their child nodes are joined to the cluster head.User pass a query to the database then processing query and searching relevant datasets based on their user query. Assume that when the user query will be matched in the root node of GSTB tree, fetch the relevant data from the root node and send it to the user; otherwise the cluster head transfer the user query to their neighbor node. This process is performed until to find the relevant data from the large database. GSTB protocol assign the cluster head which is maximum frequency pattern length that means it is asked more and more times. So, we satisfy the maximum number of user's query.

Time complexity for GSTB tree

Number of steps=f (n) where f(n)=f(data. Length)

Length of a tree is 2^n-1.So the time complexity is O (log n) and space complexity is O(1).

Algorithm 4: GSTB Tree construction

Select cluster head (n, S) in level wise Subspace clustering.

Begin

Step:1 Let we have a set S of n nodes in a cluster $S=\{S_1,S_2,S_3,\ldots.S_n\}$

Step:2compute a pattern length for required subspace clusters.

Step:3 for j=1to k do length from X_i to Z_i.

Step:4 Pickup one by one pattern lengths sequentially compare with the corresponding pattern-length.

End for

Step: 5select the cluster head based on all lwsc values.

Step: 6 if (CH >=threshold) then consider the corresponding node as a CH

Step: 8 else eliminate it from tree constructionElse do until final leaf node is examined.

4. EXPERIMENTAL RESULTS

In this section we presented the detailed execution of our FPSSCO algorithm. It amalgamates frequent pattern (FP) and Sub-Space (SS) clustering concept for effective mining large datasets. To reduce the overall scanning and efficient load balancing we uses network protocol GSTB. This algorithm constructs the tree from parent to child node and contains the prefix path to avoid full tracery of dataset item. Fig 2 describes the working of our proposed technique. It carries three phases, first one is frequent pattern (FP) mining phase, and it extracts all frequent items from the overall dataset. Second phase is Sub-Space (SS) clustering it perform the second level dimension data by comparing with support value (min_sup). Pruning application is not used at this phase for eliminating the non-frequent item. The output of this phase is entering into the tree construction phase.

In Frequent pattern mining approach, the frequent item sets are arranged depending up on the occurrences of item sets. By using this hashing technique mining frequent item sets in data mining in an efficient manner. It will reduce the multiple time of scanning the database. The following tables and figures are described about our FPSSCO algorithm

Table 2
Initial Transaction database

Item sets	*Transaction ID*
I1	T1,T3,T4,T5,T7,T9
I2	T1,T2,T4,T6,T8,T9
I3	T1,T4,T6,T7,T10
I4	T1,T2,T5,T8,T10
I5	T3

Table 2 represents the example dataset for our proposed framework Let us consider the above datasets B, D, I, N, V as an item sets which represents in a form of I1,I2,I3,I4,I5 then hashing function is applied in the following manner:

Table 3
Transaction database

Transaction ID	*Item Sets*
T1	B,D,I,N
T2	D,N
T3	B,V
T4	B,D,I
T5	B,N
T6	D,I
T7	B,I
T8	D,N
T9	B,D
T10	I,N

A transaction may contain one or more frequent items and few of them may contain single itemsets. Next we consider frequent item in each transaction in below process.

Table 4
Vertical format of transaction database

Transaction id	*Item sets*
T1	I1,I2,I3,I4
T2	I2,I4
T3	I1,I5
T4	I1,I2,I3
T5	I1.I4
T6	I2,I3
T7	I1,I3
T8	I2,I4
T9	I1,I2
T10	I3,I4

Each and every item set in table 4 represents candidate-1 item set of the transaction database. After generating the candidate-1 item set hash table is constructed using the hash function. The items in the transaction are hashed based on the hashfunction:

$$H(k) = (\text{order of item } k) \bmod n_s$$

Where n_s value is determined by using the formula of $(2m_s + 1)$ where m_s is the number of items in the database. The transaction in which I1 is connected in the form of linked list and the first node denotes the number of occurrences of the item in the transactions. It can be observed from Figure 2, I1 is hashed to 1st location and it is determined using the hash function. Similarly all items are hashed into the hash table. The cross symbol indicates the end of the items in the list. Here, the linked list is created based on the item set and not on the transactions because the transactions are more so that it occupies more memory and it is very difficult to access the items sine there is a link between a transactions and item sets. The linked list is created for all levels of frequent item set generation. In the next higher level, the item subsets become low and it is easy to find frequent item sets of that level. The process continues until the exact frequent item set is found.

Table 4.1
Bucket no after hash function

Item sets	*Transaction id*	*Bucket count*	*Support count*
I1	T1,T3,T4,T5,T7,T9	1	6
I2	T1,T2,T4,T6,T8,T9	2	6
I3	T1,T4,T6,T7,T10	3	5
I4	T1,T2,T5,T8,T10	4	5
I5	T3	5	1

Table 5
Vertical format for second level transactional database

Item set	*Transaction id*
I1,I2	T1,T4,T9
I1,I3	T1,T4,T7
I1,I4	T1,T5
I2,I3	T1,T4,T6
I2,I4	T1,T2,T8
I3,I4	T1,T10

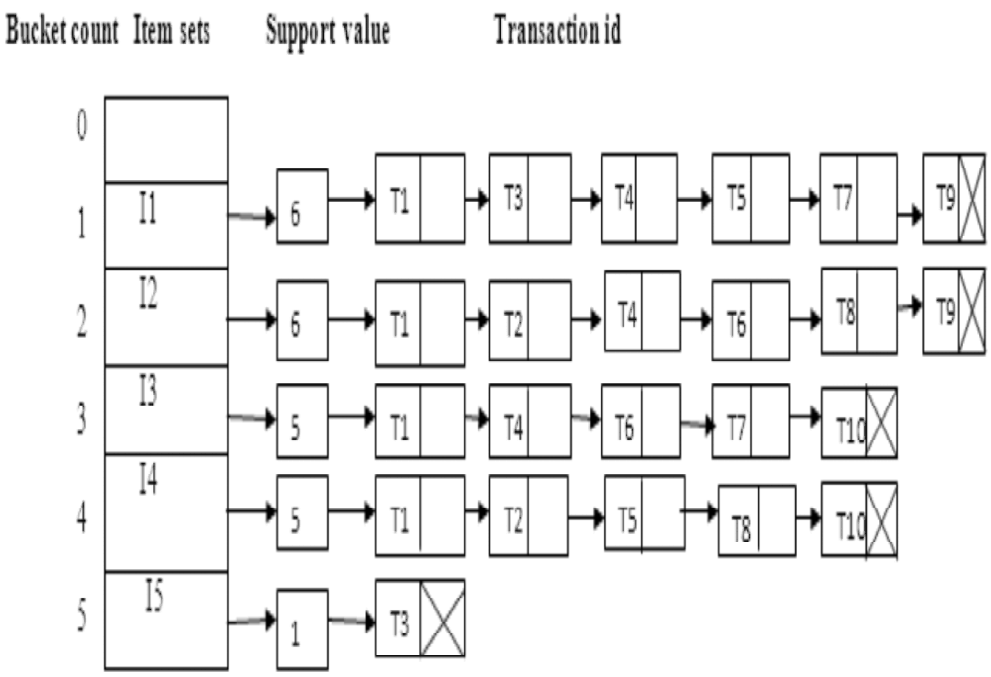

Figure 2: Hash table including links for the transactional database in the first level

The item set in the second level from Table.4 are hashed based on the hash function,

H (k) = ((order of X)*10+order of Y) mod n_s

Here , the item sets are mapped to 1,2,3,1,2,1.Here , there is a collision for {I1,I2}{I2,I3};they are mapped to 1 and {I1,I2}{I3,I4} are mapped to 1 and {I1,I3}{I2,I4} are mapped to 2.Rehashing technique is used to overcome this collision. Here, we increase the size of the hash table by doubling the actual size, so that the resulting hash table size is also a prime number. Thus the size of the hash table after increasing is $j_s = (2*m_s+1)$, where m=11 (initial hash table size) .Therefore, *j=23.*Now, we apply the hash function.

H (k) = ((order of X)*10+order of Y)mod j_s

It better avoids primary, secondary clustering problems and some collisions that may still occur using Double hashing technique also. After rehashing the collision is resolved and the 2-itemsets {I1,I2}, {I1,I3},{I2,I3},{I2,I4},{I3,I4} ,{I1,I4} are mapped to 12 ,13,0,1,11,14 buckets respectively as shown in figure.2. In the second level, item sets {I1,I2} ,{I1,I3},{I2,I3},{I2,I4},{I3,I4} whose support counts are greater than or equal to 3 are said to be frequent item sets.

Table 6
Transactional database after applying hash function.

Item sets	*Transaction id*	*Bucket count*	*Support value*
I1,I2	T1,T4,T9	12	3
I1,I3	T1,T4,T7	13	3
I1,I4	T1,T5	14	2
I2,I3	T1,T4,T6	0	3
I2,I4	T1,T2,T8	1	3
I3,I4	T1,T10	11	2

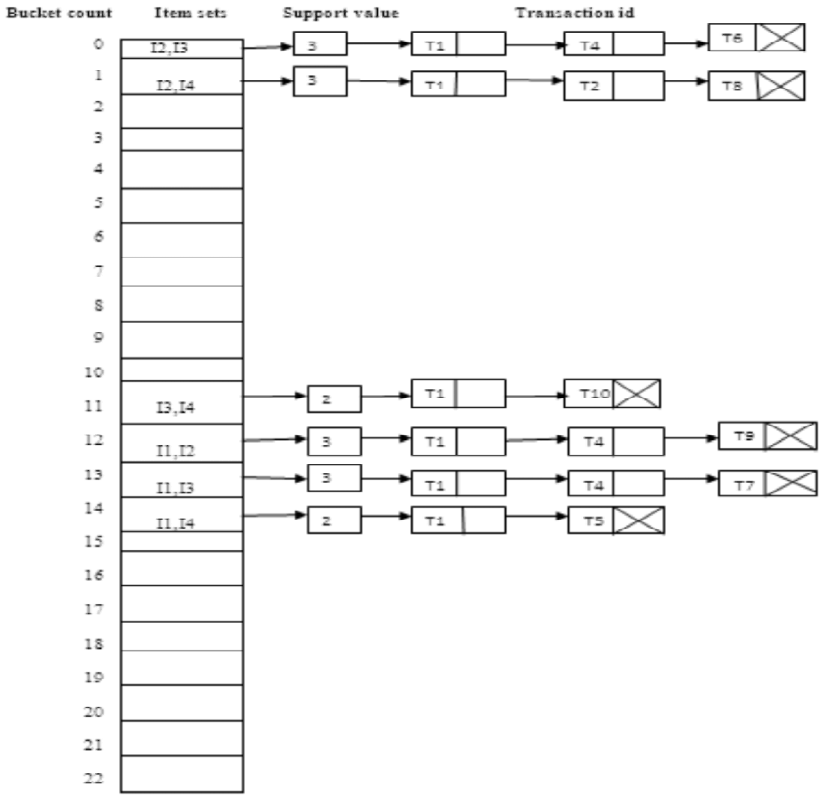

Figure 3: Hash table including links for the transactional database in the second level

Table 7
Vertical formation of the transactional database in the third level

Item set	*Transaction id*
{I1,I2,I3}	T1,T4
{I1,I2,I4}	T1
{I2,I3,I4}	T1

The 3-item sets are generated from frequent item sets of second level as shown in the Table .7 the item sets in the third level are hashed based upon the hash function

Table 8
Vertical formation of transactional database after Applying hash function

Item sets	*Transaction id*	*Bucket count*	*Support value*
{I1,I2,I3}	T1,T4	8	2
{I1,I2,I4}	T1	9	1
{I2,I3,I4}	T1	4	1

H(k)=((order of X)*100+(order of Y)*10+order of Z)mod j. Using this hash function the itemsets {I1,I2,I3},{I1,I2,I4},{I2,I3,I4} are mapped to location 8,9 and 4 respectively as shown in the Figure.3.

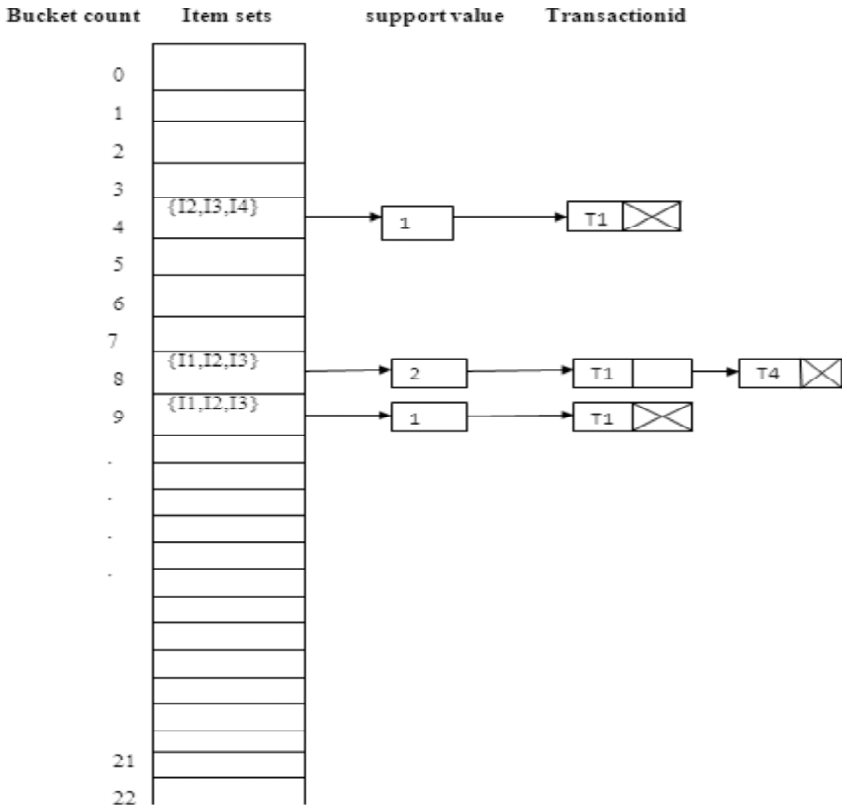

Figure 4: Hash table including links for the transactional database in the third level

Table 9
Levelwise subspace clustering

Level wise interesting subspaces	Subspaces(set of attributes)	Transaction id	Sum(support)
Maximal frequent item sets(level 3)	{I1,I2,I3}	T1,T4	2
Closed frequent item sets(level 2)	{I1,I2}	T1,T4,T9	3
	{I1,I3}	T1,T4,T7	3
	{I1,I4}	T1,T5	2
	{I2,I3}	T1,T4,T6	3
	{I2,I4}	T1,T2,T8	3
	{I3,I4}	T1,T10	2
Frequent item sets (level 1)	{I1}	T1,T3,T4,T5,T7,T9	6
	{I2}	T1,T2,T4,T6,T8,T9	6
	{I3}	T1,T4,T6,T7,T10	5
	{I4}	T1,T2,T5,T8,T10	5

Table 10
Pattern length after applying length function for n subspaces

Subspaces	*Pattern length*
{I1,I2,I3}	3
{I1,I2}	2
{I1,I3}	2
{I1,I4}	2
{I2,I3}	2
{I2,I4}	2
{I3,I4}	2
{I1}	1
{I2}	1
{I3}	1
{I4}	1

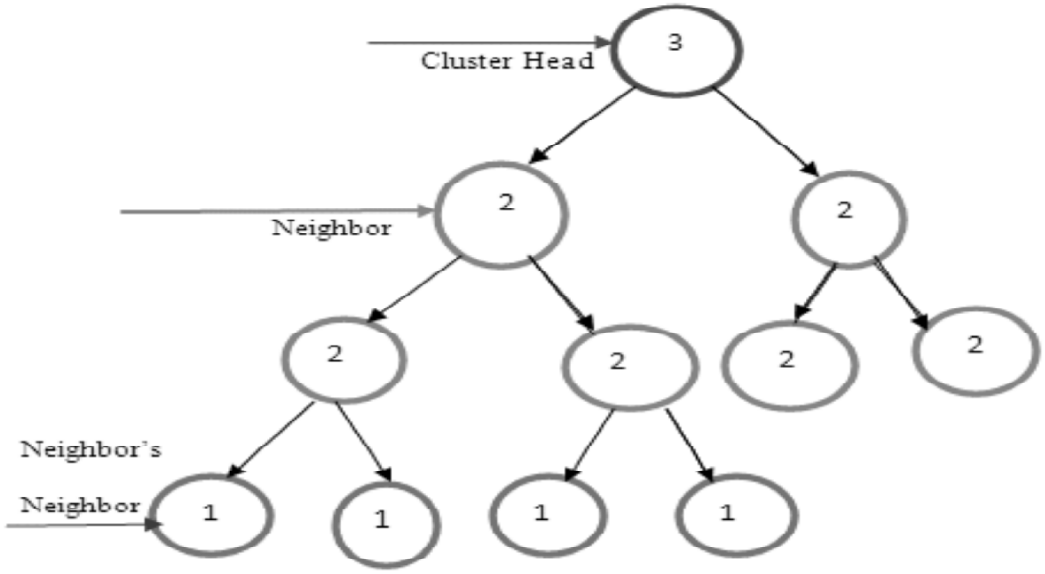

Figure 5: Selecting a cluster head and using sub-space clustering algorithm select the neighbors of the root node

Table 11
Load balancing among all the processor.

units	*Transaction id*	*processor*
{I1,I2,I3}	T1,T4	P0
{I1,I2}	T1,T4,T9	P1
{I1,I3}	T1,T4,T9	P2
{I1,I4}	T1,T5	P3
{I2,I3}	T1,T4,T6	P4
{I2,I4}	T1,T2,T8	P5
{I3,I4}	T1,T10	P6

Comparative analysis

We compare our new algorithm with existing hashing techniques such as PHP (Perfect Hashing and Pruning) and Apriori in which our rehashing based apriori produce accurate result. The datasets are generated randomly depending on the number of distinct items, the maximum number of items in each transaction and the number of transactions. The performance of the FPSSCO implementation is dependent on the number of large item sets found and the frequency of the units.

We evaluated most frequent pattern mining algorithm performance by means of a large set of experiments addressing the following issues: (i) Time for mining frequent item from the large dataset, (ii) Existing algorithm performance with our proposed algorithm (iii) response time of the each processor.

Dataset description

The characteristics of the evaluated dataset are summarized in the following. To validate the usefulness of the FPSSCO algorithm we analyzed 18 collections, each one composed of various real time frequent item set. An initial analysis of the frequent pattern extracted from the large dataset with support their minimum value. In our FPSSCO algorithm reduce frequent pattern mining time all sampled dataset to existing algorithm.

The bench mark of dataset has been taken from [35].Doctors, wages,iqitems, verbalaggression and males'dataset are taken for analysis.The bench mark of dataset considers more items. We have taken the above mentioned datasets for experimentation. Table 12 shows name of the datasets and their corresponding number of items.FPSSCO algorithm is finding frequent pattern in a quick manner whenever there are a large number of frequent item sets present in large database.

7.2.3. Frequent pattern mining

Table 12
Details of smaller datasets

TID	*Dataset*	*Numberof items*
1	Doctors	5190
2	Wages	3294
3	Economics	479
4	Housing	452
5	Iris	151
6	Ice cream	518
7	Males	4390
8	Ovary cancer	789
9	Primary school	568
10	Animals	657

This chart is comparing the DPA algorithm with our proposed algorithm FPSSCO with different type of transactional datasets. FPSSCO algorithm finds the maximum number of frequent items in this datasets. FPSSCO can take minimum computational time to extract the frequent item when compared to the Distributed Parallel Algorithm.

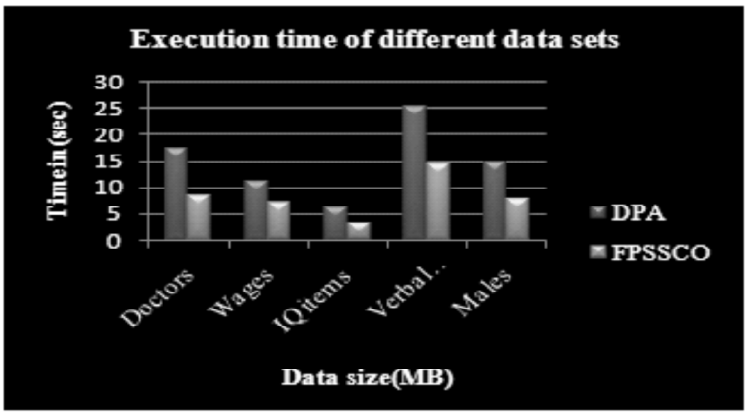

Figure 6: Execution time of different datasets

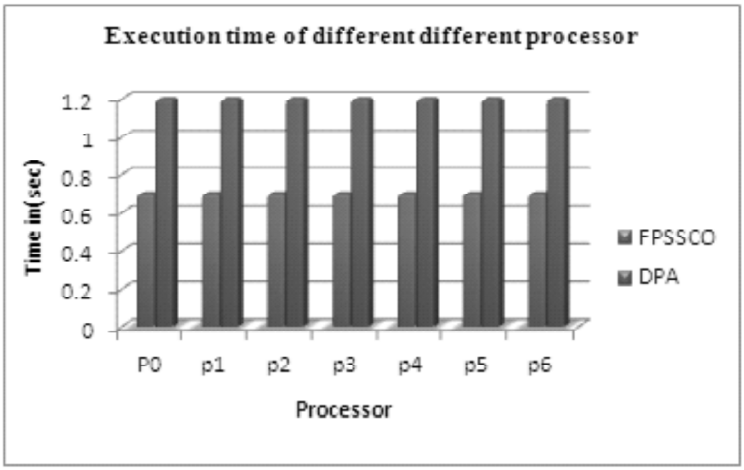

Figure7: Execution time of different processor

Fig. 7 shows the execution time of different processor by FPSSCO on the above dataset. It can be observed that FPSSCO achieved a good balance of workload [7].

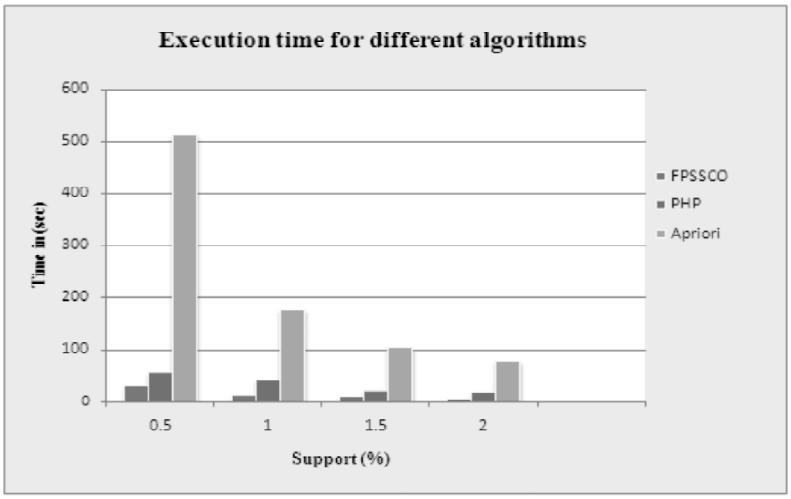

Figure 8: Execution time for different algorithms

Fig. 8 shows the execution time of Apriori, PHP algorithms and FPSSCO algorithm on the real life dataset with different support values. The FPSSCO algorithm outperforms the above mentioned algorithms. It can be observed the FPSSCO can efficiently balancing the load and reduce execution time with low minimum support values [6].

6. CONCLUSION

In this paper, weproposed a new algorithm for mining frequent patterns i.e. frequent pattern subspace clustering optimization algorithm. This algorithm is quickly extracting the frequent pattern from the large database. The process of finalized them into a frequent item and it is taken more time in DPA algorithm. FPSSCO algorithm takes the frequency count of the items and generating the cluster head for sub-space clustering. Then we introduce the novel network application of GSTB is self organized tree based adaptively root selecting algorithm. This approach is adaptively selecting the root from parent to child node which is work with the combination of the frequent pattern and sub-space clustering concept. We presented sub-space clustering to determine the most frequent item among all the transaction in the database. GSTB tree is used for efficiently balancing the load for all processor. On the entire process FPSSCO take only few scans. Experimental result shows that our FPSSCO algorithm is reduce the number of processors and execution time.

7. FUTURE WORK

In our proposed work we introducedFPSSCO mechanism in order to find out the frequent items in an effective manner. FPSSCO is the combination of clustering and routing optimization technique. In future, we use some new clustering algorithm instead of subspace clusteringand also furthermore to reduce the time consumption and also increase the accuracy. Initially discover frequent items from the given item sets in the database. After that apply effective clustering mechanism like k-means++ for clustering the frequent items and apply routing mechanism to find out the optimal route from the root to child nodes in order to offer effective and accurate results.

REFERENCES

[1] Zhao zhu 2016 Research on the Structure of Computer Interaction Hash Algorithm for Mining, Hunan Communication Polytechnic, Changsha, Hunan, 410004. (ICCSAE 2015).

[2] Farah Hanna AL-Zawaidah, YosefHasanJbara, Marwan AL-Abed Abu-Zanona 2011 An Improved Algorithm for Mining Association Rules in Large Databases, World of Computer Science and Information Technology Journal (WCSIT). Vol. 1, No. 7, 311-316, 2011.

[3] Arthur Zimek, Ira Assent and JillesVreeken 2014 Frequent Pattern Mining Algorithms for Data Clustering IEEE Trans. Springer International Publishing Switzerland. pp 403-423.

[4] J.S. Park 1997 Using a Hash-Based Method with Transaction Trimming for Mining Association Rules, IEEE Trans. Knowledge and Data Engineering. vol. 9, pp. 813 – 825.

[5] Padmavathy L, UmaraniVAn Efficient Association Rule Mining Using the H-BIT Array Hashing Algorithm International Journal of Advanced Research in Computer Science and Software Engineering. vol.3, pp. 410-419.

[6] Ozel S.A and Guvenir H.A 1998 An Algorithm for Mining Association Rules Using Perfect Hashing and Database Pruning", In 10th Turkish Symposium on Artificial Intelligence and Neural Networks, Gazimagusa, T.R.N.C., A. Acan, I. Aybay, and M. Salamah, Eds. Springer,Berlin, Germany, pp.257–264.

[7] Kun-Ming Yu, Jiayi Zhou, Tzung-Pei Hong, Jia-Ling Zhou 2010 A load-balanced distributed parallel mining algorithm, Elsevier publications, Vol.37, PP.2459-2464.

[8] Rao N.G and AguruS 2012 A Hash based Mining Algorithm for Maximal Frequent Item Sets using Double Hashing, Journal of Advances in Computational Research. Vol. 1 No. 1-2.

[9] JyotiAgarwalArchana Singh 2014 Frequent Item set generation based on Transaction Hashing, IEEE Trans.pp-182-187.

[10] Prof. PareshTanna, Dr. YogeshGhodasara 2013 Foundation for Frequent Pattern Mining AlgorithmsImplementation, International Journal of Computer Trends and Technology (IJCTT). Volume 4 Issue 7

[11] Vijay kumar P.V, Prof. Vijayalakshmi 2014 A Association Rule Mining Using Hash-BasedDecision Tree as Directed a-Cyclic Graph(IJERT). Vol. 3 Issue 4.

[12] Charu C. Aggarwal, Mansurul A. Bhuiyan and Mohammad Al Hasan 2014 Frequent Pattern Mining Algorithms: A Survey, Springer International Publishing Switzerland.

[13] MsPriyanka Mali, Prof. ShrutiYagnik 2014 Survey of Various Frequent Pattern Mining Techniques, JETIR. Volume 1 Issue 7.

[14] Hassan Najadat, AmaniShatnawiand GhadeerObiedat 2011 A New Perfect Hashing and PruningAlgorithm for Mining Association Rule, IBIMA Publishing Communications of the IBIMA.

[15] JayalakshmiN,VidhyaV,KrishnamurthyKannan M 2012 A Frequent Item set Generation using Double Hashing Technique,Procedia Engineering 38 pp:1467 – 1478.

[16] FujiangAo, Jing Du, Jingyi Yu, Fuzhi Wang, Qiong Wang 2013 Clustering High Dimensional Data Streams Based on N-most Interesting Item sets Mining, IEEE Trans.

[17] SanjayGoil,HarshaNagesh,AlokChoudhary1999MAFIA:Efficient and Scalable Subspace clustering for very large data sets. Center for Parallel and Distributed Computing.June-1999.

[18] MeeraNarvekara ,ShafaqueFatmaSyedb," An optimized algorithm for association rule mining using FP tree", Procedia Computer Science 45 (2015) 101 – 110.

[19] Ke-chung Lin, I-En Liao Zhi-sheng chen "An improved frequent pattern growth method for mining association rules" Expert Systems with Applications: An International Journal Volume 38 Issue 5, May, 2011

[20] NitinAgarwalEhteshamHaqueHuan Liu Lance Parsons 2004 A Subspace Clustering Framework for Research Group Collaboration, Prop 301 No. ECR A601 and CEINT.

[21] Ira Assent, Ralph Krieger Emmanuel Muller Thomas Seidl 2008 INSCY: Indexing Subspace Clusters with In-Process-Removal of Redundancy Data management and exploration group. PP: 719-724.

[22] Ira Assent Ralph Krieger Emmanuel Muller Thomas Seidl 2007 DUSC: Dimensionality Unbiased Subspace Clustering proc.IEEE international Conference on Data mining (ICDM 2007), Omaha, Nebraska, USA.

[23] Sunil Joshi, Dr. R. S. Jadon, Dr. R. C. Jain 2010 An Implementation of Frequent Pattern Mining Algorithm using Dynamic Function",International Journal of Computer Applications (0975 – 8887) Volume 9– No.9.

[24] SaibabaCH.M.H. Dr. RekhaRedamalla 2011 Mining Frequent Item sets by using Binary Search Tree Approach International Journal of Computer Applications.Volume 27– No.5.

[25] Jiawei Han · Hong Cheng · Dong Xin ·Xifeng Yan 2007 Frequent pattern mining: current status and future directions, Springer.

[26] Ian Sandler, Alex Thomo," Large-scale Mining of Co-occurrences: Challenges and Solutions",IEEE,2012

[27] BasheerMohamad Al-Maqaleh, Saleem Khalid Shaab," An Efficient Algorithm for Mining Association Rules using Confident Frequent Item sets" IEEE2013.

[28] FujiangAo, Jing Du , Jingyi Yu, Fuzhi Wang, Qiong Wang 2013 Clustering High Dimensional Data Streams Based on N-most Interesting Item sets Mining, IEEE Trans.

[29] KaziMahbubMutakabbir, Shah S Mahin, Md. AbidHasan 2014 Mining Frequent Pattern within a Genetic Sequence Using Unique Pattern Indexing and Mapping Techniques ,IEEE transactions. pp1-5.

[30] RahmatWidiaSembiring, JasniMohamadZain 2010 Cluster Evaluation of Density Based Subspace Clustering, Journal of Computing. Vol 2, PP. 2151-9617.

[31] Kun-Ming Yu, JiayiZhouand, Wei Chen Hsiao 2007 Load Balancing Approach Parallel Algorithm for Frequent Pattern Mining, The 9th international conference on Parallel Computing Technologies. PP. 623–631.

[32] Takashi Washio, Yuki Mitsunaga and Hiroshi Motoda2005 Mining Quantitative Frequent Item sets using Adaptive Density-based Subspace Clustering, Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05).

[33] Jieqing Xing and Chuanyi Fu 2013 LTPI: A Spectral Clustering Method Based on Local Topology Preserving Indexing and Its Application for Document Clustering, International Journal of Hybrid Information Technology Vol. 6, No. 1.

[34] RakeshAgrawal Johannes Gehrke, DimitriosGunopulosPrabhakarRaghavan 1998 Automatic Subspace Clustering of High Dimensional Data for Data Mining ApplicationsACM SIGMOD international conference on Management of data . PP.94-105.

[35] *http://vincentarelbundock.github.io/Rdatasets/datasets.html.*