IJCSC International Journal of Computer Science & Communication Vol. 1, No. 2, July-December 2010, pp. 357-363

An Association Rule Based Algorithmic Approach to Mine Frequent Pattern in Spatial Database System

Animesh Tripathy[1], Subhalaxmi Das[2] & Prashanta Kumar Patra[3]

[1]School of Computer Engineering, KIIT University, Bhubaneswar, India
[2]CSE Department, Hi-Tech College of Engineering, Bhubaneswar, India
[3]Department of Computer Science & Engineering, CET, Bhubaneswar, India
Email: [1]animesh_tripathy@hotmail.com, [2]subhalaxmi.das@gmail.com, [3]pkpatra@chek.com

ABSTRACT

Emergence of data mining methods in data representation has resulted in discovering knowledge from large database systems. Efficient algorithms to mine frequent patterns are crucial to many tasks in spatial association rule mining. A promising approach for mining such spatial frequent pattern is mining frequent sub-patterns which are closed and maximal patterns. In this paper, we make use of numerical representations and its arithmetic operations to considerably reduce the size of the transaction dataset. The proposed approach generates a TFP-tree that simplifies the generations of frequent patterns. Further analysis shows that this approach generates maximal frequent patterns and performs only minimal generalizations of frequent candidate sets. Experimental results validate the compactness and efficiency for low minimum support against existing methods in spatial domain.

Keywords: Spatial Data sets, Ordered List, Maximal Frequent Pattern, Closed Frequent Pattern

1. INTRODUCTION

Spatial Database Systems (SDBS) are database systems for the management of spatial data. Both, the number and the size of spatial databases are rapidly growing in applications such as geo-marketing, traffic control and environmental studies. This growth by far exceeds human capacities to analyze the databases in order to find implicit regularities, rules or knowledge hidden in the data. Storage and retrieval of spatial data has therefore become a challenge. Knowledge discovery in spatial database is very important to analyze and develop different hypothesis from the spatial database. The discovery process for spatial data is more complex than for relational data. Because in contrast to mining in relational databases, spatial data mining algorithms have to consider the neighbors of objects in order to extract useful knowledge. This is necessary because the attributes of neighboring object of interest may have a significant influence on the object itself.

Frequent pattern mining [4] plays an essential role in many data mining tasks and applications, such as mining association rules [1], correlations [5], sequential patterns [2], classification [7] and clustering [2]. Methods for efficient mining of frequent patterns have been studied extensively by many researchers. The problem is stated as follows. Given a collection of transactions with each transaction consisting of set of items, a frequent pattern is a subset of set of items that occurs in at least user specified percentage (support) of the transactions. Frequent pattern mining algorithms suffers from several drawbacks. The main drawback is the generation of many frequent patterns. For example, if the pattern length is x, then 2^x frequent patterns would be generated. In many applications with long frequent patterns, enumerating all possible subsets is computationally not possible [8,9]. Most of the datasets usually contains long frequent patterns and frequent pattern mining is computationally infeasible. To overcome the "too many patterns" disadvantage, closed pattern concept was developed. The set of closed patterns of the given transactional database is the condensed representation of the set of all frequent patterns without any loss of information i.e. from the given set of Closed Frequent Pattern (CFP) it is possible to generate all the frequent patterns. A frequent pattern is said to be closed if it has no superset with the same frequency (support). To further condense the set of frequent closed patterns, Maximal Frequent Pattern (MFP)[3,6] concept was proposed. The set of Maximal Patterns is the smallest possible representation of all the frequent patterns that can still be used to generate the entire frequent patterns. A frequent pattern is said to be maximal if none of its supersets is frequent. However, unlike in frequent closed pattern, the support information is lost in maximal patterns but can be easily recomputed from the given dataset. This paper proposes Transaction Frequent Pattern (TFP) tree that makes use of numerical representation to generate frequent patterns by intersecting the ordered list of similar type. And finally from the generated frequent patterns we generate

maximal frequent pattern from the set of closed frequent pattern.

1.1. Motivation

Our work deals with efficient discovery of spatial frequent patterns from the existing spatial objects. When we talk about spatial frequent patterns we mean the patterns (such as spatial object set) that appear in a spatial data set frequently. First we represent the set of spatial objects as a set of transactions. Each transaction basically represents the existence of spatial objects considered within the geographic boundary limits of a city. Finding such frequent patterns plays an essential role in mining of interesting relationship using spatial associations and correlations among spatial objects. The major difference between knowledge discovery in relational databases and in spatial databases is that attributes of the neighbors of some object of interest may have an influence on the object itself. Therefore, spatial knowledge discovery algorithms heavily depend on the efficient processing of neighborhood relations. This tendency is termed spatial autocorrelation or spatial dependence [1]. It's natural that most spatial data in GIS are not independent, they have high autocorrelation. For example, temperatures of nearby locations are often related. Most of the spatial association rule mining algorithms derived from the attribute association rule mining algorithms which assume that spatial data is independent. In these situations, the rules or knowledge derived from spatial mining proves to be incorrect. It is, therefore, important that mining spatial association rules take into consideration of spatial autocorrelation. Spatial autocorrelation is when the value at any one point in space is dependent on values at the surrounding points. In this paper the major contributions relate to removing the difficulty of analyzing huge and large spatial dataset transactions by using numerical representation that compresses and prune the data set. To perform mining of spatial objects we have used an extension of the frequent pattern mining technique for mining spatial frequent patterns.

2. RELATIVE WORK

Several algorithms have been proposed for efficient mining of association rules. The Apriori algorithm [2] is the best known traditional candidate generation algorithm. It has been followed by several variations for improving efficiency and scalability. They almost suffer inherently from two problems, multiple database scans that are costly and generating lots of candidates. Many variants of the Apriori algorithm have been developed, such as AprioriTid[3], ArioriHybrid[4], direct hashing and pruning (DHP)[5], dynamic item sets counting (DIC)[6], Partition algorithm[7]. FP-growth [8] is another well-known algorithm that uses the FP-tree data structure to achieve a condensed representation of the database transactions and employs a divide-and-conquer approach to decompose the mining problem into a set of smaller problems. The FP-Tree stores only the frequent items in a frequency-descending order. The highly compact nature of FP-tree enhances the performance of the frequent pattern mining. FP-Tree and almost all its extensions have a high compactness rate for dense data set. However, they need a large amount of memory for sparse data set where probability for sharing common paths is low. A variant of FP-growth is the H-mine algorithm [5]. It uses array-based data structures to deal with sparse and dense datasets respectively. In FP-growth based algorithms, recursive construction of the FP-tree affects the algorithm's performance. Eclat [10] is the first algorithm to find frequent patterns by a depth-first search and it has been shown to perform well. It uses a vertical database representation and counts the item sets supports using the intersection of transactions. However, because of the depth-first search, pruning used in the Apriori algorithm is not applicable during the candidate item sets generation. VIPER and Mafia [10] also use the vertical database layout and the intersection to achieve a good performance. The only difference is that they use the compressed bitmaps to represent the transaction list of each item set. However, their compression scheme has limitations especially when datasets are uniformly distributed. This algorithm has been shown to gain significant performance improvements over Eclat. However, when the database is sparse, difference in transactions will lose its advantage. In this paper, we present a novel approach that maps and compresses the both dense and sparse database by constructing a Transaction Frequent Pattern tree using numerical representation and its arithmetic operations. This approach performs only on generalizations of frequent candidate sets. Due to these properties, the number of valid frequent patterns generated and tested is minimized as compare to existing algorithms. The generation of maximal patterns is done by intersecting the ordered list of similar type.

3. PROPOSED FRAMEWORK

We propose a framework to discover knowledge in terms of maximal frequent patterns of spatial database objects for both dense and sparse datasets. The detail of proposed framework is given in fig.1. The Proposed Framework can be described as a six step process. In the first step we find dense and sparse datasets from Spatial Transaction Datasets on the basis of occurrence of higher ranked objects [9] in spatial database. Second step finds frequent ordered list for dense and sparse dataset. In the third step we represent the frequent order list in form of numerical representation where each object in the transaction is represented as a prime value and then each transaction can be viewed by the product value of the

prime dataset which compresses the transaction. The fourth step constructs a TFP tree using the numerical representation of each transaction dataset. Fifth step finds frequent patterns with their respective support count by intersecting its numerical ordered list. And finally we generate the set of maximal frequent dense and sparse patterns. Our method uses a prime-based data transformation technique to reduce the size of transaction database to enhance the performance of mining algorithms.

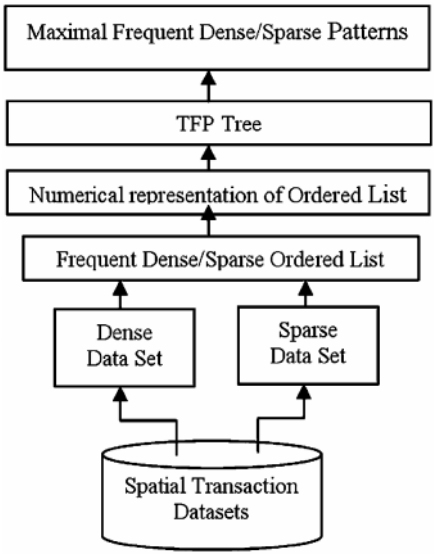

Fig. 1: Proposed Framework

3.1. Building of Sample Dataset

In this section, we build a sample spatial dataset for discovering frequent pattern. The analysis is based on frequent spatial objects from Geographic Map Database and the correlation between those objects. Each of these object sets will occur at least as frequently as a predetermined minimum support count as only frequent objects can play roles in frequent patterns. We have considered 100 Indian cities. In order to avoid wasting a lot of computing resources when verifying a few frequent objects in database we have considered only 7 spatial objects. The spatial objects considered are Museum (A), Lake (B), Monument(C), Zoo (D), Forest (E), River (F), Hill (G). For each city we find the occurrence of these spatial objects to form a real time dataset of spatial transactions. A city together with its spatial objects forms a transaction within an aerial distance of 20kms.

From the geo spatial dataset of 100 Indian cities we have considered a sample of 28 cities to analyse proposed framework. While building each tuple that corresponds to a city we have assumed lexicographic ordering of spatial objects. For example consider the first transaction (TID: 1) as shown in Table 1. The Agra city has the spatial objects such as Museum (A), Monument (C), Forest (E) and River (F). Therefore in Table 1, {A,C,E,F} is denoted corresponding to Agra. Similarly for each city the occurrence of spatial objects in taken into consideration and represented in Table 1. Assuming a minimum threshold support of 40% on 28 Indian cities a sample of 12 cities are considered to be dense and other 16 are considered to be sparse. A dataset is said to be *dense* when most of the transactions is said to be similar among them and they have about the same length and contain mostly the same objects. A dataset is said to be *sparse* datasets where the length of the transaction differs a lot from another and contains mostly different objects.

Table 1
Sample Spatial Dataset

TID	*Reference City*	*Objects*
1	Agra	A,C,E,F
2	Darjeeling	A,D,G
3	Ranchi	A,B,G
4	Chennai	A,B,R
5	Ernakulum	A,C,E,F
6	Coimbatore	A,C,F
7	Kanpur	A,C,E
8	Amritsar	B,C,E
9	Bhubaneswar	A,B,C,D,F,G
10	Bangalore	A,B,C,D,E
11	Ajmer	A,B,C,F
12	Mumbai	A,B,C,D,F,G
13	Chandigarh	A,B,D,E
14	Trivandrum	A,B,D
15	Delhi	A,C,DE,F
16	Ahmadabad	A,B,C,D,F
17	Pune	A,B,C,D,F,G
18	Mysore	B,C,D,E,G
19	Nagpur	A,B,C,D
20	Patna	A,C,D,F
21	Nashik	A,C,F
22	Mussorrie	A,B,E,G
23	Varanasi	A,C,F
24	Ooty	A,B,G
25	Kodaikanal	A,B,E,G
26	Port blair	A,C,F
27	Thekaddy	B,E,D,G
28	Chamba	A,B,E,G

Among the transactions considered in Table 1, TIDs {9,10,11,12,13,14,15,16,17,18,19,20} are considered to be dense while TIDs {1,2,3,4,5,6,7,8,21,22,23,24,25,26,27,28} is the set of sparse data sets. Based on dense and sparse data sets we apply the proposed framework and generate the set of frequent patterns. To generate the set of frequent patterns we propose an algorithm in section 4 and further in section 5 perform the analysis of the framework using the sample spatial data sets.

4. PROPOSED ALGORITHM

This section outline the details of the algorithm based the proposed framework to mine frequent patterns from spatial datasets. The central concept is based on the numerical representation of ordered list to generate the TFP Tree for dense and sparse spatial objects. The proposed algorithm is used to obtain valid frequent patterns. The algorithm is divided into three phases:

Step 1: Call Frequent Order List procedure to find out all frequent objects.

Step 2: Call TFP Tree procedure to build a tree.

Step 3: Call TFP procedure to find out all frequent patterns.

Step 1: Procedure to Find Frequent Ordered List:

Input : Spatial Dataset (SD), minimum support(ms)
Output: Order list (OL)
Procedure: Frequent Order List generates as follows
Step 1: Scan SD and consider object[i] from object_list
Step 2: Generate count for each object[i]
For each transaction T
If object[i] ==found in object_list
Insert into temp_list
Count[object[i]]++
Step 3: For each object[i] in temp_list
Determine the support count for each object
Step 4:Obtain Order List(OL)
If (support count (object[i]) <min_support)
Discard (object[i])
Else
Sort (object[i].list) in descending order.
Generate Order_list(OL)

Step 2: Procedure for TFP Tree:

Input : Ordered List (OL), Set of Prime Numbers (PN)

Output : TFP Tree

Procedure TFP Tree creates as follows

Step 1: Scan each transaction T and its Order_List(OL)

Step 2: Assign PN to OL

Scan each object[k] in Order_list[j]

Insert PN to each object[i] in Order_list[j] in decreasing order

Step 3: Obtain the Product Sum(PS)

Initialize PS=1

For each Order_list[j]

PS+=MULTIPLY(PN(object[i]))

Step 4: Build TFP Tree

For PS in each transaction

Temp=Compare(PS.transaction[t],PS.transaction[t+1])

If(temp==0)

Generate a leaf node and increment count[t]++

Elseif

PS.transaction[t+1])MOD(PS.transaction[t]=0

Generate new leaf as child of transaction[t]

Assign count[t]=1

Else

Generate new leaf and assign count[t]=1

Step 3: Procedure for TFP Frequent Pattern:

Input : TFP Tree, minimum support(ms), tree(vertex, edge]

Output: Transaction Frequent Patterns(TFP)

Procedure: TFP generation as follows

Step 1: Scan TFP Tree and consider each branch of the tree[v,e]

Step 2: For each set of nodes in tree[v,e]

List[i].tree[v,e]= prime_nos

Object_pattern[i]=list[i].tree[v,e]

Step 3: Initialize Frequent_Pattern=={}

For each branch in tree[v,e]

TFP[j]=object_pattern[i])" object_pattern[i+1]

Frequent_Pattern=TFP[j]∪ Frequent_Pattern

Increment object_pattern[i]

Increment list[i].tree[v,e]

Generate TFP

5. ANALYSIS FOR PROPOSED FRAMEWORK

To validate the proposed framework we consider a data set of spatial objects. For our analysis we have considered the transaction list given in Table 1. The spatial objects are correlated to each other in space and there is a ranking relationship between a set of objects such that, for any two objects, the first is either 'ranked higher than', 'ranked lower than' or 'ranked equal to' the second as per their frequency. The ranking relationship can be obtained from total number of frequency of an object occurred in dataset. Next we consider two different types of objects representation i.e. dense and sparse objects on the basis of occurrence of higher ranked objects in the spatial database. We separately consider the dense and sparse spatial data set for analysis.

We use a six step analysis process as per the framework described in the previous section:

Step 1: Obtain Dense & Sparse Transaction List.

Step 2: Build Ordered list of objects in descending order of their frequencies.

Step3: Mapping Ordered List in form on numerical representation.

Step 4: Build a TFP Tree using numerical representation.

Step 5: Find frequent patterns and validate it against their respective support count.

Step 6: Generate the set of Maximal Frequent Patterns.

5.1. Transaction Mapping Technique

In this section we have considered the TIDs {9,10,11,12,13,14,15,16,17,18,19,20} to form the dense data set. The spatial objects are arranged in the ordered list in decreasing order of their occurrence in the complete dense data set. The lower occurring objects having frequency less than 40% are pruned. The dense and sparse object list generated is shown in Table 2 & 3. The spatial objects of dense and sparse datasets are arranged in descending order of their frequencies. The ordered list of each transaction is mapped using prime-based data transformation technique to reduce the size of transaction database. The details of which is shown in Table 2 & 3. We also store the product value for each order list. Let's examine this through an example. Suppose we take sample dense database of #1 where Bhubaneswar is a reference city and its corresponding spatial objects are {A,D,B,C,F }. Using the prime numbers in decreasing order for representation such as [(A:11),(D:7),(B:5),(C:3),(F:2)]. Therefore the ordered list can be mapped as {11,7,5,3,2} for {A,D,B,C,F}. Further we find the product of the numerical order list as product value of prime numbers i.e. (2310=11*7*5*3*2) and finally the both dense and sparse transactions are arranged in descending order of their product value. The product values generated here will be used for constructing the TFP-tree as shown in Fig.2 & 3.

Table 2
Numerical Representation of Dense Dataset

TID #	*References City*	*Ordered Dense Object*	*Prime No. Representation*	*Product Value*
1	Bhubaneswar	A,D,B,C,F	11,7,5,3,2	2310
2	Mumbai	A,D,B,C,F	11,7,5,3,2	2310
3	Ahmadabad	A,D,B,C,F	11,7,5,3,2	2310
4	Pune	A,D,B,C,F	11,7,5,3,2	2310
5	Bangalore	A,D,B,C	11,7,5,3	1155
6	Nagpur	A,D,B,C	11,7,5,3	1155
7	Delhi	A,D,C,F	11,7,3,2	462
8	Patna	A,D,C,F	11,7,3,2	462
9	Ajmer	A,B,C,F	11,5,3,2	330
10	Chandigarh	A,D,B	11,7,5	385
11	Trivandrum	A,D,B	11,7,5	385
12	Mysore	D,B,C	7,5,3	105

Table 3
Numerical Representation of Sparse Dataset

TID#	*References City*	*Ordered Dense Object*	*Prime No. Represen-tation*	*Product Value*
1	Mussorrie	A,B,E,G	13,11,5,2	1430
2	Kodaikanal	A,B,E,G	13,11,5,2	1430
3	Chamba	A,B,E,G	13,11,5,2	1430
4	Agra	A,C,E,F	13,7,5,3	1365
5	Ernakulam	A,C,E,F	13,7,5,3	1365
6	Kanpur	A,C,E	13,7,5	455
7	Chennai	A,B,F	13,11,3	429
8	Amritsar	B,C,E	11,7,5	385
9	Ranchi	A,B,G	13,11,2	286
10	Ooty	A,B,G	13,11,2	286
11	Coimbatore	A,C,F	13,7,3	273
12	Nashik	A,C,F	13,7,3	273
13	Varanasi	A,C,F	13,7,3	273
14	Port blair	A,C,F	13,7,3	273
15	Thekaddy	B,E,G	11,5,2	110
16	Darjeeling	A,G	13,2	26

5.2. Construction of TFP-Tree

The TFP-Tree is based on prime number characteristics for compressing data and enhancing the efficiency of pattern generation. A TFP-Tree includes a root node and a child node that forms a sub tree as children of the root or descendants where each child stores product value of each transaction represented by this node. While insertion of a new node takes place, the first and second transaction product values are compared and if the product value of two transactions is not divisible then it creates new descendants. In case, it is divisible then it is inserted as a child of the existing node and count of node increases by 1. If product value is equal to the current node only the count of the current node is increased by 1. The final TFP tree for dense and sparse datasets is shown in Fig. 2 and Fig. 3 respectively. Suppose we take #1,#2 and #3 of sparse dataset (Fig. 3) where the product values of transaction are equal so count value of node is increased by 3.

Further when we take #4 of sparse dataset, it is not divisible by 1430 so 1365 create a new descendant. Similarly for every successive insertion of a new node its product value is examined against existing nodes

product values. The construction of TFP tree stops after considering every transaction present in the data set. The node under Root node in Fig. 2 denotes {2310:4} where {2310} is the product value and {4} represents the frequency of occurrence of the product in similar transaction lists. Thus one can observe and verify that the tree generated in TFP is much simple with respect to level and complexity as compared to similar tree generated using FP-Growth or other successive algorithms.

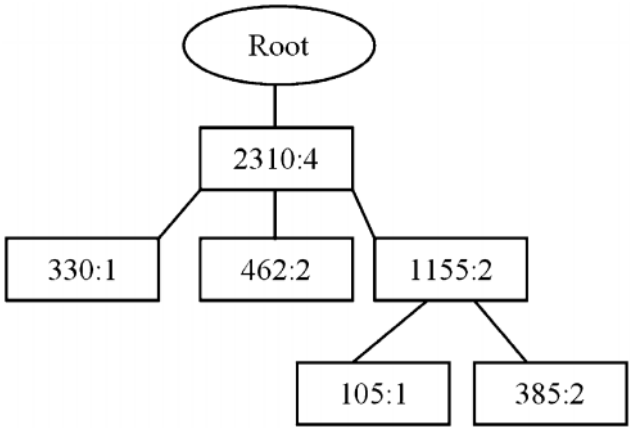

Fig 2: TFP Tree for Dense Dataset

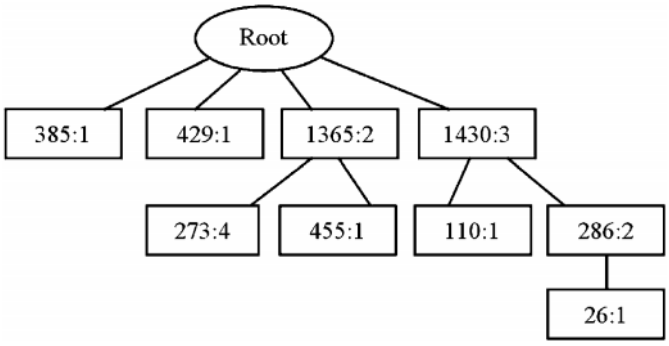

Fig. 3: TFP Tree for Sparse Dataset

5.3. Generation of Frequent Patterns

To mine the frequent pattern from TFP-tree we traverse from root to each branch of tree. Taking each branch as a new entry to an array list we find the set of prime numbers for each product value for a given node. Then by intersecting of prime number set of each node's product value, we find the frequent patterns. Suppose we consider Fig.2 and take (2310, 330) branch as a new entry in an array list (say TFP-1) which is shown in Fig 4. Then by intersecting its prime-based representation of {11,7,5,3,2} with {11,5,3,2, we finally get {11,5,3,2} as its prime factor. Corresponding to the prime factor {11,5,3,2}, the possible frequent patterns are (11,5),(11,5,3) and (11,5,3,2). Each prime number set represents a spatial object pattern like {Museum, Lake},{Museum, Lake, Monument}, {Museum, Lake, Monument, River} which are correlated with each other. To generate frequent pattern we assume the lexicographic ordering of spatial objects. Similarly for every successive insertion of a new array its pattern is examined against existing branch values to generate the consolidated list of frequent patterns.

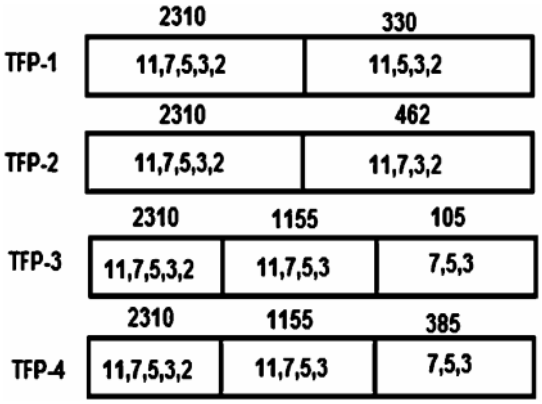

Fig. 4: Dense Pattern in TFP Tree

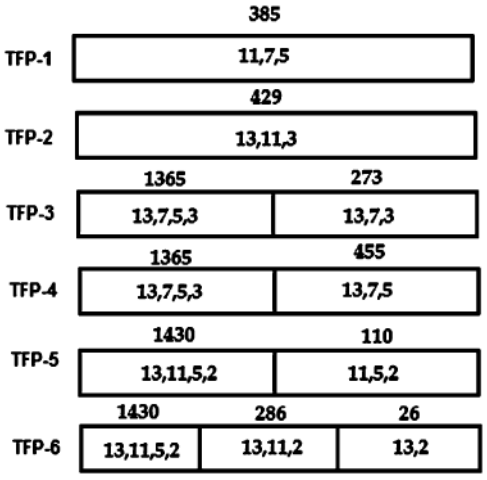

Fig. 5: Sparse Patterns in TFP Tree

5.4. Computing MFP from CFP

Further from the set of generated frequent patterns we find the set of maximal frequent pattern (MFP) from a set of close frequent pattern (CFP). As discussed earlier CFP can be defined as closed if it has no superset with the same frequency (support). In the previous section we find the intersecting patterns as {(11,5,3,2), (11,7,3,2), (7,5,3)}. From this set of intersecting patterns we first find the set of frequent patterns as {(11,5), (11,5,3), (11,5,3,2), (11,7), (11,7,3), (11,7,3,2), (7,5), (7,5,3)} with support counts as (9,7,5,10,8,6,9,7). Each of these patterns is considered to be a set of closed frequent pattern (CFP). A MFP is defined as the smallest possible representation of all frequent patterns that can still be used to generate the entire frequent patterns. From this set of CFP we compute the set of maximal frequent pattern (MFP) as {(11,5,3,2), (11,7,3,2), (7,5,3)}. Similarly the MFP for sparse is computed as ({11,7,5}, {13,11,3}, {13,7,3}, {13,7,5}, {11,5,2}, {13,2}). Moreover, it has been proven all the

maximal patterns are unique to each other, whereas all the frequent closed patterns can be related by subset or superset relationship with respect to each other. Each of this MFP is capable of generating the full set of frequent patters which is verified from analytical results.

6. EXPERIMENTAL RESULTS

In this section, we report our experimental results on the performance of TFP with respect to efficiency and scalability in comparison with *Apriori* and *FP-growth*. It shows that TFP outperforms *Apriori* and *FP-growth* and is efficient and highly scalable for mining very large databases. The proposed approach was tested with the famous UCI Machine Learning Repository: Mushroom Dataset. All the experiments were performed on a 2.00 GHz Pentium PC machine with 984 MB of RAM, running Microsoft Windows XP. TFP and *FP-growth* were implemented by us using Visual C++6.0, while the version of *Apriori* that we used is a well-known version, available at http://fuzzy.cs.uni-magdeburg.de/~borgelt/. All reports of the iterations and number of rules generated as shown in Fig. 6. The results denote that TFP outperforms the existing algorithms as the number of frequent pattern generated for finding MFP is less over a series of iterations. Limited by space, only some typical results is reported here.

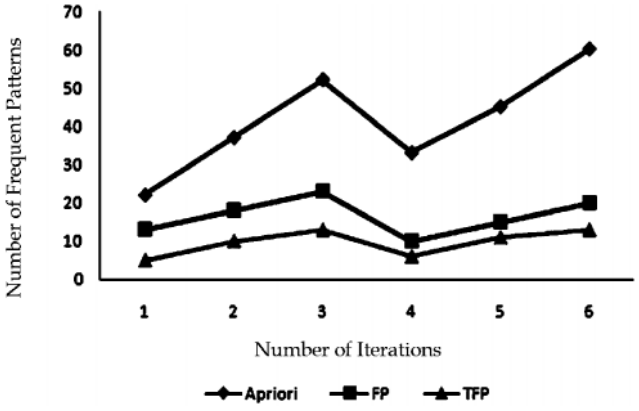

Fig. 6: No. of Iterations Vs No. of Rules Generated

7. CONCLUSION

In this paper we proposed a approach to mine frequent patterns using TFP Tree. TFP-tree compresses both dense and sparse datasets by using numerical representation. Our method introduces a new method based on prime number characteristics to find CFP and then the MFP. Some existing algorithms can only be applied on sparse or dense data sets and do not work efficiently on both. Our proposed approach is efficient on both dense and sparse datasets, and has and therefore overcomes limitations of the previous methods. The algorithm will definitely enhance the efficiency of finding the correlation between spatial objects and further can be used in correlation analysis. The experiments verified the compactness of the data transformation technique. The generation of maximal frequent patterns is done by intersecting the ordered list of similar type which reduces the search space.

REFERENCES

[1] Y. Huang and P. Zhang, (2006), "On the Relationships between Clustering and Spatial Co-location Pattern Mining", *In ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, pp.513-522.

[2] Agrawal R. and R. Srikant, (1994), "Algorithms for Mining Association Rules", *In Proc. 20th Int. Conf. Vert Large Data Base*, **1215**, pp.487-499.

[3] Burdick D., M. Calimlim and J. Gehrke, (2001), "Mafia: A Maximal Frequent Itemset Algorithm for Transactional Databases", *17th International Conference on Data Engineering*, pp. 443-452, 2001.

[4] Han J., H. Cheng, D. Xin, and X. Yan,Frequent,(2007), "Pattern Mining: Current Status and Future Directions", *Data Mining and Knowledge Discovery*, **15**, pp. 55-86.

[5] Han J., J. Pei, Y. Yin, and R. Mao, (2004), "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach", *Data Mining and Knowledge Discovery*, **8**, pp. 53-87, 2004.

[6] Norwati Mustapha, Mohammad-Hossein Nadimi-Shahraki, Ali B Mamat, Md. Nasir B Sulaiman, (2009), "A Numerical Method For Frequent Patterns Mining", *Journal of Theoretical and Applied Information Technology*, pp92-98.

[7] Mustapha N., M. N. Sulaiman, M. Othman, and M. H. Selamat, (2003), "Fast Discovery of Long Patterns for Association Rules", *International Journal of Computer Mathematics*, **80**, pp. 967-976, 2003.

[8] Nadimi-Shahraki M.H., N. Mustapha, M. N. Sulaiman, and A. Mamat, (2008), "Incremental Updating of Frequent Pattern: Basic Algorithms", *Proceedings of the second International Conference on Information Systems Technology and Management (ICISTM 08),* pp. 145-148.

[9] Nadimi-Shahraki M.H., N. Mustapha, M. N. B. Sulaiman, and A. B. Mamat, (2008), "A New Method for Mining Maximal Frequent Itemsets", *Presented at International IEEE Symposium on Information Technology*, Malaysia, pp. 309- 312.

[10] P. Shenoy, J. R. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah, (2000) "Turbo-charging Vertical Mining of Large Databases ", *Procedings of ACM SIGMOD International Conference on Management of Data* ACM Press, Dallas, Texas, pp. 22-23.