IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-1 E-ISSN: 2347-2693

Performance Analysis of PFI using FP-Growth Algorithm for various Data-sets

Shital A. Patil[1*], Amol Potgantwar[2]

[1*,2] Department of Computer,
Savitribai Phule Pune University, SITRC, Nashik, India

www.ijcseonline.org

Received: Dec/05/2015 Revised: Dec/14/2015 Accepted: Jan/15/2016 Published: 30/Jan/2016

***Abstract*— The data handled in appearing applications like placement or situation based services, sensor monitoring systems, and data integration, are often not exact in nature. In this article, we study the important problem of extracting frequent item sets[1] from a huge unsure database, illuminated under the Possible World Semantics (PWS)[2].This issue is technically challenging, since an unsure database consist an exponential number of possible worlds. By observing that the mining process can be show as a discrete probability distribution, we develop an FP Growth algorithm [4] which compress a large database into a dense, Frequent-Pattern tree (FP-tree) [4] structure also Develop an efficient, FP-tree-based frequent pattern mining method (FP-growth) and Apriori algorithm for frequent item set mining. We also study the important problem of maintaining the mining result for a database that is developing (e.g. by inserting a tuple). Specifically, we present incremental mining algorithms [13], which enable Probabilistic repeated Item set (PFI) results to be refreshed. This decrease the requirement of re-executing the whole mining algorithm on the new database, which is often more expensive and unnecessary. We observe how an existing algorithm that retrieves exact item sets, as well as our approximate algorithm, can support incremental mining. All our algorithms support both tuple and attribute uncertainty, which are two common uncertain database models. We also perform huge evaluation on real and synthetic data sets to validate our approaches.**

***Keywords*— :** *Frequent Item Sets, Uncertain Data Set, FP Growth Algorithm,Association Rule Mining,Apriori Algorithm*

I. INTRODUCTION

The databases used in much more applications such as location based services, sensor monitoring systems are often uncertain. Frequent patterns are patterns that present in a data set frequently. For example, a set of items such as milk and bread appear frequently together in a transaction data set is frequent data set. A sequence such as buying first a milk, then bread and then butter, if it occurs frequently in a shopping history then it is frequent sequential pattern. Frequent pattern mining [7] searches for relationships which occur repeatedly in a given data set. Frequent item set mining leads to the invention of associations and association among items in large transactional or relational data sets. With massive amounts of data continuously being gathered and stored, more industries are becoming interested in mining such patterns from their databases. The invention of interesting association relationships among huge business transaction records can help in much more business decision-making processes, such as cross-marketing, catalogue design, and customer shopping behavior analysis. An example of frequent item set mining is market basket analysis. In a market basket database rows represents transactions, columns represents product purchases and binary number indicate whether an item is contained within a given transactions. This process analysis customer buying habits by finding associations between the various items that customer locates in their "shopping baskets". Frequent pattern mining is a key technique for the analysis of such data. Fig 1 shows an online market application which carries probabilistic information.

In a probabilistic database each tuple has certain probability of belonging to the database. A Probabilistic database is a probability distribution on all database instances called as possible worlds So, to interpret uncertain database Possible World Semantics(PWS) are often used. An uncertain transaction database generates possible worlds, where each world is defined by a fixed set of transactions .This Possible World Semantics (PWS) are used to interpret uncertain databases. A set of deterministic samples or illustrations called Possible Worlds consisting of set of tuple. Any query estimation algorithm for an unsure database has to be correct under PWS. This means that results produced by the algorithm should be same as if the query is estimated on every possible world. Although PWS is intuitive and useful, querying or mining under this concept is costly. This is because an unsure database has an exponential number of possible worlds. For example, the database in Fig. 1 has 23=8 possible worlds. Performing data mining under PWS can, thus, be technically challenging. In fact, the mining of unsure data has recently attracted research attention. For example, in efficient clustering algorithms were developed for unsure objects; in [naive Bayes and decision tree classifiers designed for unsure data were studied. In scalable algorithms develop for

finding frequent item sets (i.e., sets of attribute values that appear together frequently in tuple) for unsure databases. There are two common unsure database models, which support both tuple uncertainty and attribute uncertainty. In many applications information captured in transactions is unsure since the existence of an item is associated with an existential probability. Given an uncertain transaction database, it is not sure how to identify whether an item or item set is frequent because generally we cannot say for sure whether an item set appears in a transaction. In a certain transaction database, there will be simply scan and count the transaction that include an item set. Dealing with such databases is a difficult but interesting problem.

Customer	Purchase Items
Jack	(video:1/2),(food:1)
Mary	(clothing:1),(video:1/3);(book:2/3)

Figure- 1: Illustrating an Uncertain Database

The frequent item sets discovered from unsure data are naturally probabilistic, in order to reflect the confidence located on the mining results. Fig. 2 shows a Probabilistic Frequent Item set (PFI) [14] extracted from Fig. 1. A PFI is a set of attribute values that appear regularly with a sufficiently high probability. In this paper we use Apriori Algorithm for finding frequent item set. As in many cases the Apriori candidate generation and test method significantly decrease the size of candidate sets leading to good result again.

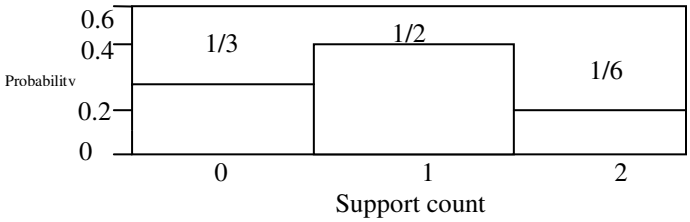


Figure- 2: s-pmf of PMI {Video} from Figure 1

1) It may need to generate huge number of candidate sets. For case, if there are 104 frequent 1-itemsets, the Apriori algorithm will need to generate more than 107 candidate 2-itemsets. Moreover, to invent a frequent pattern of size 100, such as {a1…, a100}, it has to generate at least 2100-1=1030 candidates in total.

2) It may require repeatedly scanning the database and checking a large set of candidates by pattern matching. It is expensive to go over each transaction in the database to determine the support of the candidate item sets. In this article we use the FP-Growth algorithm for finding frequent item set.

Table-1: Our Contributions (Marked [√])

Uncertainty Model	Static Algorithm	Incremental Algorithms
Attribute	Exact[6] Approx.[√]	Exact[√] Approx[√]
Tuple	Exact[30] Approx.(singleton)[35] Approx.[multiple items)[√]	Exact[√] Approx[√]

We can design a method that can mine the complete sets of frequent item sets without candidate generation using Frequent-Pattern growth algorithm. The FP-growth algorithm is now one of the fastest approaches to find frequent item set mining. This algorithm adopts a divide-and-conquer strategy as follows. First, it wraps or summarizes the database representing frequent items into a frequent-pattern tree, or FP-tree, which preserves the item set association information. It then divides the compressed database into a set of limited databases (a special type of projected database), each associated with one frequent item and stores every such database separately.

An Apriori algorithm and FP-Growth Algorithm[4] both support Attribute and Tuple uncertainty models. Experiments on real data set and find reveal that our algorithm significantly improves the performance of PFI analysis, with a high degree of accuracy. In this paper we present a novel array-based technique that greatly decreases the need to traverse FP-trees, thus obtaining significantly improved performance for FP-tree based algorithms. Our technique works well for sparse datasets.

II. LITERATURE SURVEY

The frequent item sets discovered from uncertain data are naturally probabilistic, in order to reflect the confidence placed on mining results. For uncertain databases, Agawam et al. and Chui et al. developed efficient frequent pattern mining algorithms based on the expected support counts of the patterns. However, Bernecker et al. , Sun et al., and Yiu et al. found that the use of expected support may render important patterns missing. Hence, they proposed to compute the probability that a pattern is frequent, and introduced the notion of PFI. In, dynamic-programming based solutions were developed to retrieve PFIs from attribute-uncertain databases. However, their algorithms compute exact probabilities, and verify that an item set is a PFI in O (n^2) time. Our model-based algorithms avoid the use of dynamic programming, and are able to verify a PFI much faster (in O(n) time).Mining frequent item sets is an important problem in data mining, and is also the first step of deriving association rules[5] . Hence, many efficient item set mining algorithms (e.g., Apriori and Incremental Mining

Algorithm) have been proposed. Table 1 summarizes the major work done in PFI mining. Here, "Static Algorithms" refer to algorithms that do not handle database changes. Hence, any change in the database necessitates a complete execution of these algorithms. While these algorithms work well for databases with precise values, it is not clear how they can be used to mine probabilistic data. While Zhang et al. only considered the extraction of singletons (i.e., sets of single items), our solution discovers patterns with more than one item. Recently, Sun et al. developed an exact threshold based PFI mining algorithm. However, it does not support attribute-uncertain data considered in this paper. In a preliminary version of this paper, we examined a model-based approach for mining PFIs.

Here, we study how this algorithm can be extended to support the mining of evolving data .In, model based approach use which can efficiently extract threshold and rank based PFIs. Other works on the retrieval of frequent patterns from imprecise data include: approximate frequent patterns [9] on noisy data, association rules on fuzzy sets; and the notion of a "vague association rule." However, none of these solutions are developed on the uncertainty models studied here. A few incremental mining algorithms that work for exact data have been developed. For example, in the Fast Update algorithm (FUP) was proposed to efficiently maintain frequent item sets, for a database to which new tuples are inserted. Our incremental mining framework is inspired by FUP. In the FUP2 algorithm was developed to handle both addition and deletion of tuples. ZIGZAG also examines the efficient maintenance of maximal frequent item sets for databases that are constantly changing. In a data structure, called CATS Tree, was introduced to maintain frequent item sets in evolving databases. Another structure, called CanTree, arranges tree nodes in an order that is not affected by changes in item frequency. The data structure is used to support mining on a changing database. In the density based clustering algorithm DBSCAN enhance to express similarity between two fuzzy objects, which can extract threshold and rank based PFIs. The adaptations of spatial access methods and searching algorithm for probabilistic versions of range queries, nearest neighbors (NNs), spatial skyline and reverse NNs.

To our best knowledge, maintaining frequent item sets in evolving uncertain databases has not been examined before. We propose novel incremental mining algorithms for both exact and approximate PFI discovery [8]. Our algorithms can also support attribute and tuple uncertainty models. Under the Possible World Semantics, D generates a set of possible worlds W. Table 2 lists all possible worlds for Figure. 1. Each world, which consists of a subset of attributes from each transaction, occurs with probability.

Table-2: Possible Words of Figure 1.

W	Tuple in W	Prob.
w1	{food};{clothing}	1/9
w2	{food};{clothing,video}	1/18
w3	{food};{clothing,book}	2/9
w4	{food};{clothing,book,video}	1/9
w5	{food,video};{clothing}	1/9
w6	{food,video};{clothing,video}	1/18
w7	{food,video};{clothing,book}	2/9
w8	{food,video};{clothing,book,video}	1/9

III. IMPLEMENTATION DETAILS

In uncertain dataset the extraction of frequent itemset is tedious work because uncertain database contain an exponential number of possible worlds. To solve this problem we propose PFI (Probabilistic Frequent Itemset) techniques. In our base paper, PFI testing and old PFI set is being used to make the mining but we implement using PFI Approximation method. By using this, large scale of uncertain dataset user like location based service, sensor monitoring system, biometric applications and data integration and so on got beneficial mining result.
To summarize our contributions are:
1. Develop a more efficient method
2. First loading the transaction database as a simple list of integer arrays, sorting it, and building the FP-tree

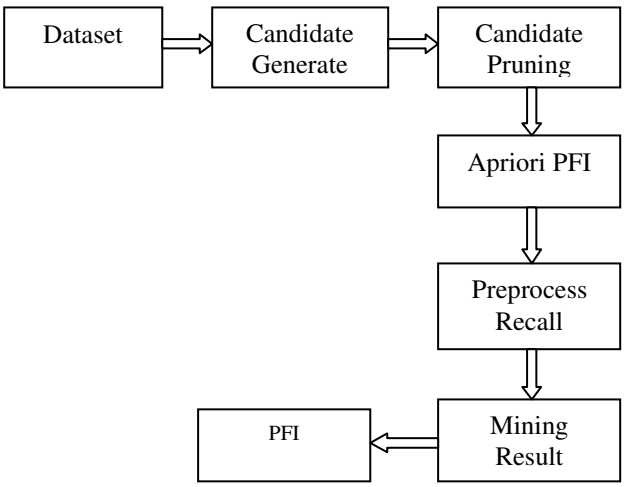


Figure- 3: System Architecture

As figure 3 shows, dataset is given as input to candidate generate phase for the purpose of candidate generation. here we include generate singleton method for generation of candidate. This subroutine simply returns the union of all single items in database also another way is use Apriori-gen method to generate candidate. Next phase is candidate pruning; the main goal of this phase is to remove infrequent

item sets from a set of candidate. Resulted output is passing to Apriori PFI for the purpose of verifying whether these generated candidates are really PFI. Then these candidates are passing to preprocess recall and then generated result is gives to PFI. Here Apriori algorithm arrange item base on probability.

In Apriori algorithm candidate generate [13] and test method, candidate pruning will be used. While using Apriori algorithm this question arise "Can we design a method that mines the complete set of frequent itemset without candidate generation?" So, FP-Growth algorithm is designs, which adopts a divide and conquer strategy as follows. First it compress the database representing frequent items into a frequent pattern tree or also called as FP-Tree, which retains the itemset association information. To ensure that the tree structure is compact and informative, only frequent length-1 items will have nodes in the tree. The tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones. It then divides the compressed database into a set of conditional databases; each associated with one frequent item or "Pattern fragment" and mines each database separately.

The first scan of database is same as Apriori, which derives the set of frequent items (1 item set) and their support counts (frequencies). An FP-tree is constructed as follows. First create the root of the tree as "Null" .Scan database second time. The items in each transaction are processed in *L* order (i.e., sorted according to descending support count), and a branch is created for each transaction.

A. *The Apriori Algorithm:*

Apriori is an algorithm for frequent item set mining and association rule [5] learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules, which highlight general trends in the database.

Algorithm 1: Frequent itemset generation of the Apriori algorithm.

1: k=1.
2: Fk={i|i∈I∧σ({i})≥N×minsup}.
{Find all frequent1-itemsets}
3: repeat
4: k=k+1.
5: Ck= Apriori-gen (Fk−1). {
Generate candidate itemset}
6: foreach transaction t∈T do
7: Ct=subset (Ck,t).
{Identify all candidates that belong to t}
8: for each candidate itemset c∈Ct do
9: σ(c) =σ(c) +1. {Increment support count}
10: end for
11: end for
12: Fk= {c|c∈Ck∧σ(c) ≥N×minsup}.
{Extract the frequent k –itemsets}
13: until Fk=∅
14: Result =Fk

The algorithm initially makes a single pass over the data set to determine the support of each item. Upon completion of this step, the set of all frequent 1-itemsets,F1, will be known (steps 1 and 2).Next, the algorithm will iteratively generate new candidate k-itemset using the frequent (k−1)-itemset found in the previous iteration (step 5). Candidate generation is implemented using a function called Apriori gen.To count the support of the candidates, the algorithm needs to make an additional pass over the data set (steps 6–10). The subset function is used to determine all the candidate itemset in Ck that are contained in each transaction t. After counting their supports, the algorithm eliminates all candidate itemset whose support counts are less than minsup (step 12). The algorithm terminates when there are no new frequent itemset generated, i.e., Fk=∅ (Step 13)

The frequent itemset generation part of the Apriori algorithm has two important characteristics. First, it is a Level-wise algorithm; i.e., it traverses the itemset lattice one level at a time, from frequent 1-itemsets to the maximum size of frequent itemset. Second, it employs a generate-and-test strategy for finding frequent itemset. At each iteration, new candidate itemset are generated from the frequent itemset found in the previous iteration. The support for each candidate is then counted and tested against the minsup threshold. The total number of iterations needed by the algorithm is k max +1, where k max is the maximum size of the frequent itemset.

B. *FP-Growth:*

The FP-Growth Algorithm is an alternative way to find frequent itemset without using candidate generations, thus improving performance. For so much it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the itemset association information. In simple words, this algorithm works as follows: first it compresses the input database creating an FP-tree instance to represent frequent items. After this first step it divides the compressed database into a set of conditional databases, each one associated with one frequent pattern. Finally, each such database is mined separately. Using this strategy, the FP-Growth reduces the search costs looking for short patterns recursively and then concatenating them in the long frequent patterns, offering good selectivity. In large databases, it's not possible to hold the FP-tree in the main memory. A strategy to cope with this problem is to firstly partition the

database into a set of smaller databases (called projected databases), and then construct an FP-tree from each of these smaller databases. The next subsections describe the FP-tree structure and FP-Growth Algorithm[17]

1) FP-Tree structure

The frequent-pattern tree (FP-tree) [16] is a compact structure that stores quantitative information about frequent patterns in a database.

FP-tree as the tree structure defined below:

1. One root labelled as "null" with a set of item-prefix sub trees as children, and a frequent-item-header table

2. .Each nodes in the item-prefix sub tree consist of three fields:

1. Item-name: registers which item is represented by the node;
2. Count: the number of transactions represented by the portion of the path reaching the node;
3. Node-link: links to the next node in the FP-tree carrying the same item-name, or null if there is none.

3. Each entry in the frequent-item-header table consists of two fields:

1. Item-name: as the same to the node;
2. Head of node-link: a pointer to the first node in the FP-tree carrying the item-name.

Additionally the frequent-item-header table can have the count support for an item. The Figure below show an example of a FP-tree.

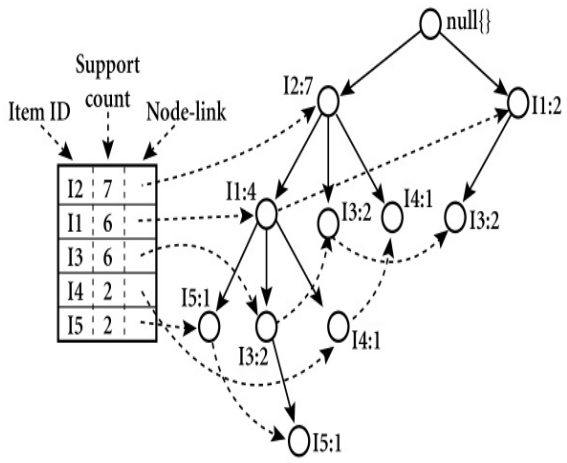


Figure- 4: An Example of an FP-Tree From

Algorithm 2: FP-tree construction

Input: A transaction database DB and a minimum support threshold?

Output: FP-tree, the frequent-pattern tree of DB.

Method: The FP-tree is constructed as follows.

1. Scan the transaction database DB once. Collect F, the set of frequent items, and the support of each frequent item. Sort F in support-descending order as F List, the list of frequent items.

2) Create the root of an FP-tree, T, and label it as "null". For each transaction Trans in DB do the following:

a. Select the frequent items in Trans and sort them according to the order of F List. Let the sorted frequent-item list in Trans be [p | P], where p is the first element and P is the remaining list. Call insert tree ([p | P], T).

b. The function insert tree ([p | P], T) is performed as follows. If T has a child N such that N .item-name = p. item-name, then increment N 's count by 1; else create a new node N , with its count initialized to 1, its parent link linked to T , and its node-link linked to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree (P, N) recursively.

By using this algorithm, the FP-tree is constructed in two scans of the database. The first scan collects and sort the set of frequent items, and the second constructs the FP-Tree.

2) 3.3 FP-Growth Algorithm

After constructing the FP-Tree it's possible to mine it to find the complete set of frequent patterns. To accomplish this job, a group of lemmas and properties, and there after describes the FP-Growth Algorithm as presented below in Algorithm 3.

Algorithm 3: FP-Growth

Input: A database DB, represented by FP-tree constructed according to Algorithm 1, and a minimum support threshold?.

Output: The complete set of frequent patterns.

Method: call FP-growth (FP-tree, null).

Procedure FP-growth (Tree, a)

{

1. If Tree contains a single prefix path then // Mining single prefix-path FP-tree {
2. let P be the single prefix-path part of Tree;
3. Let Q be the multipath part with the top branching node replaced by a null root;
4. for each combination (denoted as ß) of the nodes in the path P do
5. Generate pattern ß ∪ a with support = minimum support of nodes in ß;
6. Let freq pattern set (P) be the set of patterns so generated;

}

7. Else let Q is Tree;
8. for each item ai in Q do

{

// Mining multipath FP-tree

9. Generate pattern ß = ai ∪ a with support = ai .support;

10. Construct ß's conditional pattern-base and then ß's conditional FP-tree Tree ß;
11. If Tree ß ≠ Ø then
12. Call FP-growth (Tree ß, ß);
13. Let freq pattern set (Q) be the set of patterns so generated ;}
14. Return (freq pattern set(P) ∪ freq pattern set(Q) ∪ (freq pattern set(P) × freq pattern set(Q)))
}

When the FP-tree contains a single prefix-path, the complete set of frequent patterns can be generated in three parts: the single prefix-path P, the multipath Q, and their combinations (lines 01 to 03 and 14). The resulting patterns for a single prefix path are the enumerations of its subpaths that have the minimum support (lines 04 to 06). Thereafter, the multipath Q is defined (line 03 or 07) and the resulting patterns from it are processed (lines 08 to 13). Finally, in line 14 the combined results are returned as the frequent patterns found.

Advantages of FP-Growth

1] Only 2 pass over data-set
2] No candidate generation
3] Much faster than Apriori

Disadvantages of FP-Growth

1] FP-Tree may not fit in memory!
2] FP-Tree is expensive to build
3] Support can only be calculated once the entire data-set is added to the FP-Tree.

IV. EXPERIMENAL RESULT

A. Comparison of FP-Growth and Apriori Algorithm:

The main purpose of this test is to verify whether the tree based FP Growth algorithm is time efficient or not as compared with the Apriori based algorithm. In the experiment the same dataset accident is used for the algorithms for mining of the frequent itemsets. Here, the experimentation is done with the minsup value as 0.4 so the same constant is used in these experiments. The execution time for all the algorithms is noted by varying the minsup value from 0.1 to 0.9.The process is repeated number of times and the average value for the time required is computed. The performance of the algorithms is as shown in the graph.

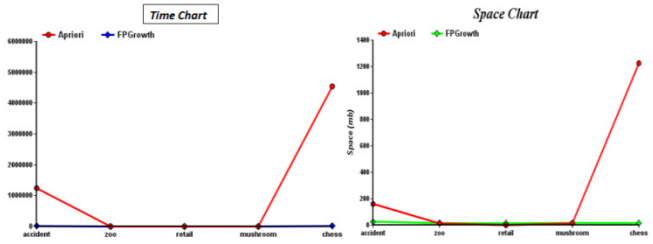
Result Comparison

Time

Filename	Apriori	FPGrowth
Accident	1239235.1333333	11386.47
200	23.375	19.5555555556
Retail	371	530
Mushroom	605.3	124
chess	4532826.667	12100.16667

Space

Filename	Apriori	FPGrowth
Accident	161.2454062	26.7385482788086
200	12.892643929	15.712817382812
Retail	0	13.5192947387695
Mushroom	13.6092506408691	16.9984924316406
chess	1222.67089109969	15.7512817382812



A) Time Chart B) Space Chart

Fig- 5: Comparision of Apriori Algorithm and FP- Growth Algorithm

So,from Fig.a) above we can say that as the time required for FP-Growth is very less than Apriori algorithm.Also from fig b)shows that as the space required for Apriori and FP-Growth.

B Min_Sup vs Space

The main purpose of this test is to verify whether the tree based FP Growth algorithm is space efficient as compared with the Apriori based algorithm. In the experiment the same dataset accident is used for the algorithms for mining of the frequent item sets. Here, the experimentation is done with the minsup value as 0.4 so the same constant is used in these experiments. The execution time for all the algorithms is noted by varying the minsup value from 0.1 to 0.9.The process is repeated number of times and the average value for the space required is computed. The performance of the algorithms is as shown in the graph fig. 6 and fig.7.

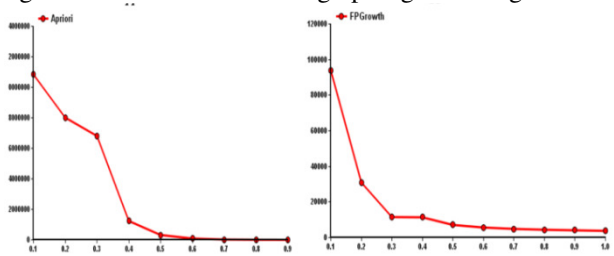


Fig 6: Time Chart for Apriori and FP-Growth Algorithm for Accident Dataset with Different Support

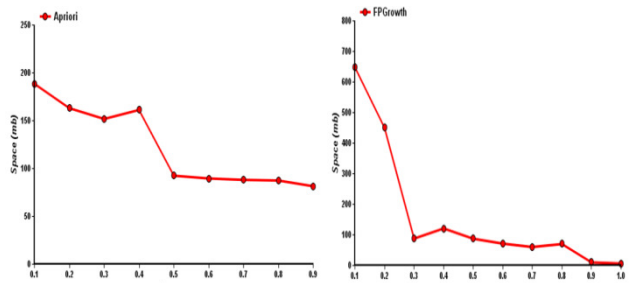


Fig 7: Space Chart for Apriori and FP Growth Algorithm for Accident Dataset with Different Support

V. CONCLUSION

We have proposed a novel data structure, frequent pattern tree (FP-tree), for storing compressed, crucial information about frequent patterns, and developed a pattern growth method, FP-growth, for efficient mining of frequent patterns in large databases. There are several advantages of FP-growth over other approaches:

(1) It constructs a highly compact FP-tree, which is usually substantially smaller than the original database, and thus saves the costly database scans in the subsequent mining processes.

(2) It applies a pattern growth method which avoids costly candidate generation and test by successively concatenating frequent 1-itemset found in the (conditional) FP-trees: In this context, mining is not Apriori-like (restricted) generation-and-test but frequent pattern (fragment) growth only. The major operations of mining are count accumulation and prefix
Path count adjustment, which are usually much less costly than candidate generation and pattern matching operations performed in most Apriori like algorithms.

(3) It applies a partitioning-based divide- and-conquer method which dramatically reduces the size of the subsequent conditional pattern bases and conditional FP-trees.

ACKNOWLEDGMENT

We thank anonymous references whose comments improved this paper. Inspiration and guidance are invaluable in every aspect of life, especially in the field of education, which I have received from our respected principal, H. O. D., guide, teachers, colleagues and family to accomplishment of the project.

REFERENCES

[1] Liang Wang, David Wai-Lok Cheung, Reynold Cheng, S Lee, X Yung,"Efficient Mining of Frequent Item Sets onLarge Uncertain Databases",IEEE Trans Knowledge and Data Eng.,**2012** pp**.110-115.**

[2] A. Veloso, W. Meira Jr., M. de Carvalho,S.Parthasarathy, and M.J. Zaki, "Mining Frequent Itemsets inEvolving Databases",Proc. Second SIAM Int'l Conf. Data Mining(SDM), **2002** pp**.210-215**.

[3] C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent PatternMining with Uncertain Data",Proc. 15th ACM SIGKDD Int'l Conf.Knowledge Discovery and Data Mining (KDD), **2009** pp**.160-170**.

[4] C. Aggarwal and P. Yu, "A Survey of Uncertain Data Algorithms and Applications",IEEE Trans Knowledge and Data Eng., vol. 21,no. 5, **May 2009,**pp. **609-623**.

[5] R. Aggrawal, T. Imieli_nski, and A. Swami, "Mining AssociationRules between Sets of Items in Large Databases",Proc. ACMSIGMOD Int'l Conf. Management of Data, **1993** pp**.348-255.**

[6] O. Benjelloun, A.D. Sarma, A. Halevy, and J. Widom, "ULDBs:Databases with Uncertainty and Lineage",Proc. 32nd Int'l Conf.Very Large Data Bases (VLDB), **2006** pp.**440-445.**

[7] T. Bernecker, H. Kriegel, M. Renz, F. Verhein, and A. Zuefle,"Probabilistic Frequent Itemset Mining in Uncertain Databases",Proc. 15th ACM SIGKDD Int'l Conf. Knowledge Discovery and DataMining (KDD), **2009** pp**.523-530**

[8] L.L. Cam, "An Approximation Theorem for the Poisson Binomial Distribution",Pacific J. Math., vol. 10, **,1990,** pp. **1181-1197.**

[9] H. Cheng, P. Yu, and J. Han, "Approximate Frequent ItemsetMining in the Presence of Random Noise",Proc. Soft Computing forKnowledge Discovery and Data Mining, **2008,**pp. **363-389.**

[10] R. Cheng, D. Kalashnikov, and S. Prabhakar, "EvaluatingProbabilistic Queries over Imprecise Data",Proc. ACM SIGMODInt'l Conf. Management of Data, **2003** pp**.789-792**.

[11] D. Cheung, J. Han, V. Ng, and C. Wong, "Maintenance ofDiscovered Association Rules in Large Databases: An IncrementalUpdating Technique",Proc. 12th Int'l Conf. Data Eng. (ICDE), **1996** pp**.890-892**.

[12] D. Cheung, S.D. Lee, and B. Kao, "A General IncrementalTechnique for Maintaining Discovered Association Rules",Proc.Fifth Int'l Conf. Database Systems for Advanced Applications(DASFAA), **1997** pp.**267-270**.

[13] W. Cheung and O.R. Zaiane, "Incremental Mining of FrequentPatterns with Candidate Generation or Support Constraint",Proc. Seventh Int'l Database Eng. and Applications Symp. (IDEAS),**2003** pp**.300-345**.

[14] C.K. Chui, B. Kao, and E. Hung, "Mining Frequent Itemsets fromUncertain Data",Proc. 11th Pacific-Asia Conf. Advances in KnowledgeDiscovery and Data Mining (PAKDD), **2007** pp**.867-870**.

[15] G. Cormode and M. Garofalakis, "Sketching Probabilistic DataStreams",Proc. ACM SIGMOD Int'l Conf. Management of Data,**2007**.

[16] M Adnan, R Alhajj, K Barker - Advances in Artificial Intelligence "Costructing complete FP tree for incremental mining of frequent patterns in dynamic database", **2006 – Springer**

[17] J. Han, M. Kamber, "Data Mining Concepts and Techniques", 3rd edition, Morgan Kaufmann Publishers, San Francisco, USA, ISBN 9780123814791, **2012,**pp**. 243-262.**

[18] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz,FlorianVerhein,AndreasZuefle,"Probabilisttc Frequent Itemset Mining in Uncertain Databases"
In Proc.11th Int. Conf. on Knowledge Discovery and Data Mining (KDD'09),Paris, France,pp.300-365,**2009**.

[19] Leung, C.K.-S., Carmichael, C.L., Hao, B.: Efficient mining of frequent patterns from uncer-tain data. In: Proc. IEEE ICDM Workshops, pp. 489–494 ,**2007**

[20] C.-K. Chui, B. Kao, E. Hung. Mining Frequent Itemsets from Uncertain Data.PAKDD.pp.500-512, **2007**

[21] H. Cheng, P. Yu, and J. Han."Approximate frequent itemset mining in the presence of random noise".Soft Computing for Knowledge Discovery and Data Mining, pp.612-615,**2008**

[22] Carson Kai-Sang Leung,Quamrul I. Khan ,Tariqul Hoque." CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns"pp.700-712,**2005**

[23] Charu C. Aggarwal,Senior Member, and Philip S. Yu,Fellow "A Survey of Uncertain Data Algorithms and Applications"conf on ieee transactions on knowledge and data engineering, vol. 21,may **2005** pp.523-540.

[24] Jianxiong Luo, Susan M.Bridges"Mining fuzzy Association rules and fuzzy frequency episodes for intrusion detection"pp.812-816.August,**2000**.

[25] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In ACM SIGMOD, pp.435-440,**2003**