A Survey on Problems and Solutions of Frequent Pattern Mining with the use of Pre-Processing Techniques

Anupriya Babbar
M TECH Scholar, CSE Deptt.
BUIT, Bhopal, India

Anju Singh
Asst. Prof. of IT Deptt.
BUIT, Bhopal, India

Divakar Singh
HOD of CSE Deptt.
BUIT, Bhopal, India

ABSTRACT
Frequent pattern mining is a researched area which is used for extracting interesting associations and correlations among item sets in transactional and relational database. Many algorithms of frequent pattern mining is been devised ranging from efficient and scalable algorithms in transactional database to numerous research frontiers and their wide applications. Many researches been done into FPM [1], but there are still several optimizations are required, so that FPM can be used more efficiently in data mining applications. For optimization purpose in many mining techniques data pre-processing plays an important role in reducing data size and also in lessening the time taken in database scans. This paper is a detailed study of problems and solutions of FPM techniques incorporated with pre-processing techniques. The intent of this paper is to summarize all major problems of FPM and their solutions. From this survey, it concludes that if FPM methods are merged with pre-processing techniques will produce results with better performance.

Keywords
Frequent Pattern Mining, Maximal frequent pattern, Data Pre-Processing

1. INTRODUCTION
Association rule mining is one of the important technique of data mining, it extracts interesting co-relations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories [7]. Association rule mining is to find out association rules that satisfy the user defined minimum support and confidence from a given database. The problem is usually divided into two sub problems. One is to find out those itemsets whose occurrences exceed a user defined threshold in the database; those itemsets are called frequent or large itemsets. And to generate association rules from those large itemsets with the constraints of minimal confidence is a second problem [14]. This is explained with a help of an example.

Example 1.1 In an online shopping website, some tips are always there after purchase of any product, for instance, once you bought the microwave a list of related cook wares such as: micro-cooker 50%, micro plates 15% etc, will be presented to you as recommendation for further purchasing. In the above example, the association rules are: when the microwave is bought, 50% of the time micro cooker is bought together, and 15% of the time micro plates are bought together. Commodities are rearranged in the store according to the rule discovered which can further make purchasing much more convenient [3].

The organization of the paper is as follows. Section 2 introduces the problems and solutions of frequent pattern mining. The result of comparison between the different methods can be found in section 3. Finally, section 4 contains the conclusion and future works.

2. PROBLEMS AND SOLUTIONS OF FPM
The efficiency and scalability of the frequent pattern mining algorithms is well studied. Nevertheless, it is obvious that the runtime is not the single factor that measures the quality of the FPM techniques. The common criticism of the FPM algorithms today is the generation of prohibitively large set of frequent patterns, which limits the avenues of exploring or reusing those [2]. Limiting the number of discovered patterns using high thresholds may reveal only the common knowledge while low thresholds results an explosion of discovered patterns. This and several other factors are the question of concern for existing approaches of FPM.

2.1 Various frequent pattern mining techniques with their problems and proposed solutions

2.1.1 Mining Frequent Itemsets with candidate generation

2.1.1.1 Apriori –based Algorithm

The first published frequent item set mining algorithm is Apriori [16]. Apriori uses breadth first search (BFS). At each level, Apriori uses downward closure property. This method is based on candidate generation. Before scanning any database it generates a candidate whose super set cannot be frequent or which is MFP. During the database scan support of candidate item set is counted Candidate k item sets, Ck are generated with frequent (k –1) item sets. Apriori suffers from large number of candidate generation which requires number of database scans which reduces its efficiency and make it costly. To overcome the disadvantage of Apriori Dynamic Item set Counting; (DIC) is used, which distinguishes between generation and counting of item sets [19]. By using such methods, the number of data scans required by Apriori based algorithms has been reduced up to a great extent.

2.1.1.2 Partition based Algorithms

Partition -based Algorithms [21] solves the problem of high number of database scans, associated with Apriori –based algorithm. Two complete database scans are required to mine frequent item sets. This algorithm breaks the dataset into number of sub sets so that each sub set can be placed into the main memory.

The basic idea of the Partition based algorithm is that a frequent item set must be frequent in at least 1 subset. Partition-based algorithm, during the first data scan, it generates local frequent item sets for each partitioned sub set. These whole partitions can be accommodated into the main

memory; I/O operations are not required to extract local frequent item sets. The local frequent item sets is added to the global candidate frequent item sets. In the next data scan, candidates who are incorrect removed from the list of global candidate frequent item set. In a rare case where each subset have same local frequent item sets, All frequent item sets can be mined with a single data scan. On contrary, when the data is divided unevenly among the partitions, this algorithm may generate a lot of wrong candidates from a small number of partitions. By implementing the knowledge collected during the mining process, false item sets are pruned when it is found that they cannot be frequent. These types of algorithms reduce the number of scans in the worse case to (2k-1)/k where k is the number of partitions.

2.1.1.3 DFS and Hybrid Algorithms

Eclat and Clique [22] combine both depth first search (DFS) and intersection counting. By using this method, there is no need of using complex data structures. These hybrid algorithms reduces the required memory space, only the TID sets of the path item sets from the root to the leaves have to be kept in the memory simultaneously. TID set intersection stopped as soon as the remaining length of the shortest TID set is shorter than the required support minus the counted support value. The intersection of TID sets of 1item set to generate frequent 2 item sets is very costly. The maximal hyper graph clique clustering is applied to 2 frequent item sets to generate a refined set of maximal item sets. A hybrid approach of BFS and DFS is proposed [6]. It is much cheaper to use item set counting with BFS to determine the supports, when the frequent item sets is small. When the large number of candidate frequent item sets is found, the hybrid algorithm jumps to TID set intersection with DFS, since simple TID set intersections are more efficient than occurrence counting when the number of candidate frequent item sets is large. This results in add on costs on generating TID sets. Then hash tree -like structure [18] is proposed to minimize the cost of transition.

2.1.1.4 Incremental Update with Apriori-based Algorithms

Complete dataset is normally large and the incremental portion is relatively small as compared to the complete dataset. In most of the cases, it is not feasible to perform a complete data mining process while transactions are being added simultaneously. Incremental data mining algorithms implements the idea of reusing the existing information as much as possible, so that computational cost and/or I/O cost can be reduced up to an extent. Fast Update 2(FUP2), an incremental mining algorithm, allows both insertion and deletion of transactions [17]. The major idea of FUP2 is to reduce the cost of generating candidate frequent item sets. Incremental portion of the dataset is scanned; frequent patterns in the incremental data are compared with the existing frequent item sets in the complete dataset. Earlier calculated frequent item sets are removed if they are no longer frequent after the incremental portion of the data is inserted or deleted. The support of earlier calculated frequent item sets that are still frequent, are updated, so that it can reflect the changes occur due to addition or deletion of incremental portion . By using these methods, previously calculated frequent item sets that are still frequent are not needed to be checked for their supports again and again. New (m + 1) candidate frequent item sets are generated from frequent m item sets. The complete updated dataset is scanned to verify that newly added candidate item sets if they are indeed frequent; the process is continue until the set of candidate frequent item set becomes empty. FUP2 offers some advantages over the classic Apriori algorithm. But, it still requires multiple numbers of database scans. Sliding Window Filtering (SWF) [19] is another incremental Apriori based algorithm. SWF uses the main idea of Partition algorithm with Apriori to allow incremental mining. SWF breaks the dataset into number of subsets. During the partitions scan, a filtering threshold is employed in each partition for the generation of candidate frequent 2 item sets. If any item set found to be frequent, its partition number and its frequency are stored. Agglomerated information about candidate frequent 2 item sets is carefully carried for subsequence partition scans. Agglomerated frequencies of previously generated candidate frequent 2 item sets are stored for the scanning of new partitions. False candidates are removed when the cumulative support of the candidate frequent item sets fall below the threshold. Once the scanning of incremental portion is done, scan reduction techniques are incorporated for the generation of all subsequence frequent candidates Item sets. Once more data scanning is done to confirm that the item sets are frequent. Although SWF achieves better performance than pervious algorithms, the performance of SWF still counts on the selection of partition size and data pruning technique used.

2.1.1.5 SQL-based algorithms

DBMS can compliment data mining to become an online, scalable and concurrent process by facilitating the existing querying and inbuilt functions. The SETM algorithm was the first attempt on frequent item set mining [23], expressed as SQL queries working on relational tables. The Apriori algorithm [16] opened up new prospects for FPM. The database -coupled variations of the Apriori algorithm were carefully examined. The SQL-92 based implementations were very slow, but the SQL implementations enhanced with object-relational extensions (SQL-OR) performed acceptable. SQL based frequent mining implemented with FP-tree provides the efficient results than the other SQL based techniques. It is accepted that FP-tree is compact, but it is unrealistic to construct a main memory-based FP-tree when the database is huge. However using RDBMSs provides us the advantages of using their buffer management systems specially developed for freeing the user applications from the size constraints of the data.

2.1.2 Mining frequent itemsets without candidate generation

2.1.2.1 Pattern-Growth Algorithms

Two major expense of Apriori based algorithms are the expense to generate candidate frequent item set t and the expense associated with I/O operations. The issues related to I/O have been resolved, but the issues related to candidate frequent item set t generation remain untouched. If there is n frequent 1 item set t, Apriori algorithms would require to generate t2 /2 candidates approximately in frequent Item set t. Secondly, the memory required to hold the candidate frequent item set t and their supports could be substantial. Han et al. [9] proposed a structured data called frequent pattern tree or FP-Tree. FP-growth is a method to mine frequent item sets from FP-Tree without generating candidate frequent item set t. FP-Tree is an extension of prefix tree structure. Node of a tree is used to store frequent items. Each node contains the item's label along with the number of occurrence of an item in a particular set. Complete path from the zero level to the last level are arranged according to the support value of the items with the descending order of their frequency. That is, each parent node is greater than or equal to the sum of its children's

node frequency. FP-Tree requires two database scans. In the first scan, the support value of each item is found. This computed support values are used in the second scan to sort the items within transactions in higher to lower order. If some transactions have a common prefix, then the frequencies of the nodes are increased by merging the common prefixes. Nodes with the same label are connected with an item link. The item link is used to mine frequent patterns. Each FP -Tree has a header that consists of all frequent items and pointers to the beginning of their respective item links. FP-growth traces the paths of FP-Tree recursively to generate frequent item sets. Pattern fragments are merged to ensure all frequent item sets are properly generated. Thus FP-growth implements the simple methods for generation and testing of candidate item sets. When the data is unevenly distributed, the compaction achieved by the FP-Tree is small and the FP-Tree is bushy. As a result, FP-growth would require a lot of effort to merge fragmented patterns with no frequent item sets being found. A new data structure called H -struct is introduced [20]. In this, transactions are arranged with an arbitrary ordering scheme. Only frequent items are used in the H-struct. H-struct consists of selected transactions and these selected transactions contain item label and a hyper link pointing to the next occurrence of the item. A header table is created for H-struct which contains frequencies of all items, their support values and hyper link to the first transaction containing given item. H-mine uses a method to mine the H-struct iteratively by developing a new header table for each item in the original header with subsequent headers and items that have been mined previously are skipped. For each sub-header, H-mine traverses the H-struct according to the hyper link s and search frequent item sets for the local header. H-Mine also develops a links for items that have not been mined in the local header. The process is iteratively runs until all frequent item sets have been mined completely. In case of a dense dataset, H-struct is not as efficient as FP-Tree because FP-Tree allows compression.

2.2 Basic Problems are as follows:

(a) In many real applications mostly in dense data with long frequent patterns enumerating all possible subsets of a particular length pattern is infeasible.

(b) The complexity of frequent pattern mining from a large amount of data is generating a huge number of patterns satisfying minimum support threshold, especially when threshold value is low.

(c) Generation of candidate item sets is expensive (Huge candidate sets).
•104frequent 1-itemset will generate 107candidate 2-itemsets
• To discover a frequent pattern of size 100, e.g., {a1, a2 ..., a100}, one needs to generate 2100≈103 candidates.

(d) It is tedious to repeatedly scan the database and check a large set of candidates by matching the patterns, especially in the case of long pattern mining [6].

2.3 Basic Proposed solutions

A major challenge in mining frequent patterns from a large data set is the fact that such mining often generates a large number of patterns satisfying the min_sup threshold, especially when it is set low. If a pattern is frequent, it is much obvious that each of its sub patterns is also frequent. A large pattern will contain an n number of smaller sub-patterns, which are frequent. To overcome this problem, closed frequent pattern mining and maximal frequent pattern mining was proposed [9], where an Apriori-based algorithm called A-Close for such mining was proposed. Other closed pattern mining algorithms include CLOSET [9], CHARM [8], CLOSET+ [10], FPClose [11] and AFOPT [12]. The main challenge in closed (maximal) frequent pattern mining is to check whether a pattern is closed (maximal). There are two strategies to approach this issue: (1) to keep track of the transaction ID (TID) list of a pattern and index the pattern by using hash functions with TID values. This method is used by CHARM, a diffset named compact list is maintained by this algorithm which maintains TID values; and (2) to maintain the discovered patterns in a pattern-tree similar to FP-tree. Advantage of this method is taken by CLOSET+, AFOPT and FPClose. Carpenter is a method for finding closed patterns in multi-dimensional biological datasets, which integrates the benefits of vertical data formats and pattern growth methods. By changing the data into vertical data format {item: TID_set}, the TID_set can be viewed as row set and the FP-tree is so build that it can be viewed as a row enumeration tree. Carpenter uses DFS traversal of the row enumeration tree and checks weather each node in a row set is frequent or closed [9]. Another method is COBBLER, to find frequent closed itemsets by combining row enumeration with column enumeration [11].TD-Close method is used for multi dimensional data to find out the complete set of frequent closed patterns. It take advantages of new search strategy, top-down traversal, by starting from the maximal row set, integrated with a novel row enumeration tree, which makes complete use of the pruning power of the min_sup threshold to shorten the search space. Furthermore, another effective closeness-checking method is also developed that avoids scanning the dataset multiple times. Even after various enhancements, the above algorithms still encounter problems in mining large (called colossal) patterns, since the process will require to produce an explosive number of smaller frequent patterns.

Data pre-processing can be used as an efficient tool in the field of FPM. It can be used to filter the data before input it into the FPM algorithms. Data reduction, Data transformation is an important step of data pre-processing, Data reduction are the methods to produce compact representation of the data set which is much smaller in size but yet produces the almost same results. Data transformation is used to remove noise from the data and transform the data into variable forms.

2.4 Various methods of data reduction and data transformation are as follows:

2.4.1 Data reduction techniques

2.4.1.1 Histograms

It is a popular data reduction technique, it Divides data into buckets and store average (sum) for each bucket, it can be Can be constructed optimally in single dimension using dynamic programming Related to quantization problems [25]. This technique may used with Apriori at the time of candidate generation for better results.

2.4.1.2 Clustering

Partition data set into clusters, it can be very effective if data is grouped but it is not if data is “smeared”, it can have hierarchical clustering and be stored in multi-dimensional index tree structures there are many options for clustering definitions and clustering algorithm [25].

2.4.1.3 Sampling

It allows a mining algorithm to execute in complexity that is potentially sub-linear to the amount of the data. It chooses a representative subset of the data. Random sampling may have very poor performance in the presence of skew [26].

This reduction technique can be used with Partition based algorithms, clustering or sampling may used with already done partitions, it may generate frequent patterns more efficiently.

2.4.2 Data transformation techniques

Different trees are used such as FP-tree, Prefix tree etc so that data can be compressed and the size needed to store it into the memory can be reduced up to a certain extent which in turn shortens the time required and number of database scans and can produce frequent patterns efficiently [26]. Some of these trees use data pre-processing techniques for efficient working and other trees may use or can be implemented with data pre-processing techniques, and could produce frequent patterns more efficiently in less time and with reduced database scans [27].

2.4.2.1 Prefix-tree

The itemsets are checked in the order of increasing size (breadth-_first/level wise traversal of the prefix tree).This tree uses the "Attribute" technique of data transformation and convert item sets into canonical form. This canonical form of item sets are merged with the developed prefix tree and are used to check weather that each candidate item set is produced once or not. The previously generated levels are used to execute a priori pruning of the candidate item sets (using the Apriori property). Transactions are represented as simple arrays of items (horizontal transaction representation) [5]. The support of a candidate item set is computed by checking whether they are subsets of a transaction or producing or searching subsets of a transaction.

2.4.2.2 FP-Tree

FP-Growth means Frequent Pattern Growth. The item sets are traverse in lexicographic order (Depth-_first traversal of the prefix tree). Step by step removal of items from the transaction database. The transaction database is represented as an FP-tree. An FP-tree is extended structure of prefix tree: nodes of this tree that correspond to the same item are linked together. By this, it combines horizontal and vertical database representation. Conditional databases can be calculated by this data structures much more efficiently. All transactions containing a given item can easily be traced by the links between the nodes corresponding to this item [8].

Algorithm: FP-Tree

Input: *T; ¾; I µ I* (initially called with *I = fg*)

Output: *N*[*I*](*T; ¾*)

1: *N*[*I*] := *fg*;

2: for all *i 2 I* occurring in *T* do

3: Add *I [fig* to *N*[*I*];

4: *Ti* := *fg*;*H* := *fg*;

5: for all *j 2 I* occurring in *T* such that *j > i* do

6: if *support*(*I [fi; jg*) ¸ *¾* then

7: Add *j* to *H*;

8: for all transaction *X 2 T* with *i 2 X* do

9: Add *X \ H* to *Ti*;

10: Compute *N*[*I [fig*](*Ti; ¾*)] recursively;

11: Add *N*[*I [fig*] to *N*[*I*]];.

2.4.2.3 PC_Tree

The data transformation is an essential process in data preprocessing step which can reduce the size of database. This method uses a prime-based data transformation technique to compress the size of transaction database. It converts each transaction into a positive number called Transaction Value (TV) during of the PC_Tree construction.

PC_Tree (Prime-based encoded and Compressed Tree) is based on prime number characteristics which can make use of both data compressing and pruning techniques to enhance efficiency. A PC_Tree includes of a root and some nodes that formed sub trees as children of the root or descendants. The node structure consists of several different fields: local-count, status, value global-count and link. The Value field keeps track of transaction value, it records which transaction is represented by which node. The local-count field set by 1 during inserting Current TV and it is increased by 1 if its TV and current TV are equal. The global-count field registers support the pattern P which contained by its TV. In fact during of insertion procedure the support of all frequent and infrequent patterns is registered in the global-count field [13].

2.4.2.4 CP-Tree

FP-growth technique, which produces frequent patterns without candidate generation, requires two Database scans to achieve a highly compact frequency-descending tree structure (FP-tree). At the time of first scan, it produces a list of frequent items in which items are arranged in decreasing order with subject to frequency. According to the frequency-descending list, the second Database scan produces a frequent-pattern tree (FP-tree), which keeps the information on transactions having frequent patterns. Two Database scans are required, which is a problem for incremental, interactive, and stream data mining. The CP-tree (compact pattern tree), constructs an FP-tree like frequency-descending tree which is a compressed structure with a single-pass of a transaction database. CP-tree construction mainly consists of two phases:

(i)Insertion phase: scans transaction(s), inserts them in the tree according to the current item order of I-list and revise the frequency count of respective items in the I-list.

(ii)Reconstruction phase: rearranges the I-list according to frequency-descending order of items and rebuild the tree nodes according to this revised I-list [15].

Many researches is been done, many is been proposed there is a lot to do in the field of frequent pattern mining to increase its efficiency.

3. COMPARATIVE ANALYSIS OF VARIOUS FPM TECHNIQUES

Every method of FPM has its own advantages and disadvantages Comparison of different FPM techniques is given in Table 1, where M represents the length of maximal frequent item set and N represents the number of partitions. As described in the table, various algorithms are compared on the basis of some parameters, like

I. Number of database scans required for best case,

II. Number of database scans required for worst case,

III. Candidate generation technique required or not,

IV. Incremental mining is possible or not,

V. Accepts the change in user parameter [24].

Other approaches like Prefix tree checking whether a given code word is canonically transformed can be faster than building canonical code word from a scratch. FP growth is an efficient technique for mining frequent pattern in frequent patterns can be efficiently produced by implementing trees with some other methods.

TABLE 1. Comparative Analysis Of Various FPM Techniques

	Apriori	Partition	Incremental Apriori	FP Growth	SQL Based
I	2	1	2	2	1
II	M+1	(2N-1)/N	M+1	2	1
III	Yes	Yes	Yes	No	No
IV	No	No	Yes	No	No
V	Yes	Yes	Yes	Yes	Yes

Data reduction and transformation techniques reduces the size of data which in turn reduces memory space utilization and the time taken for search a pattern from the bulk of data, it also reduces number of database scans. FP growth improves Apriori, as with this algorithm calculating frequent item set is possible without candidate generation, with two Database scan.

4. CONCLUSION

Frequent pattern mining is one of the most intensely investigated and challenging work domains in contemporary work in the data mining discipline as a whole [3], In this study we have tried to find some clues to the question of whether the frequent patterns can be calculated more efficiently, what are the problems and what will be the solutions using some publicly available methods that are proven successful by several studies. As a further study frequent patterns can be calculated by using different empirical approaches with pre-processing techniques, for the upcoming researcher to work in the field of data mining.

5. REFRENCES

[1] Thashmee Karunaratne, "Is Frequent Pattern Mining Useful In Building Predictive Models?" Stockholm University, Forum 100, Se-164 40 Kista, Sweden.

[2] Jiawei Han , Hong Cheng , Dong Xin Xifeng Yan, "Frequent Pattern Mining: Current Status And Future Directions" Springer Science+Business Media, Llc 2007.

[3] Norwati Mustapha, Mohammad-Hossein Nadimi-Shahraki, Ali B Mamat, Md. Nasir B Sulaiman "A Numerical Method for Frequent Patterns Mining Journal of Theoretical And Applied Information Technology". Journal of Theoretical and Applied Information Technology, 2009.

[4] Renáta Iváncsy, István Vajk, "Frequent Pattern Mining In Web Log Data" Acta Polytechnica Hungarica Vol. 3, No. 1, 2006.

[5] Bart Goethals, "Survey on Frequent Pattern Mining" Journal On Computer Science And Engineering 2010.

[6] Jiawei Han ,Jian Pei , Iwen Yin , "Mining Frequent Patterns Without Candidate Generation: A Frequent-Pattern Tree Approach", Received May 21, 2000; Revised April 21, 2001.

[7] Qiankun Zhao,Sourav S. Bhowmick, "Association Rule Mining: A Survey", Technical Report, Cais, Nanyang Technological University, Singapore, No. 2003116 , 2003.

[8] Bart Goethals, "Memory issues in frequent item set mining" SAC'04, March 14–17, 2004.

[9] Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M-C, " Prefixspan: Mining Sequential Patterns Efficiently By Prefix-Projected Pattern Growth". In: Proceeding Of the 2001International Conference on Data Engineering (ICDE'01), Heidelberg, Germany, 2011.

[10] Wang J, Han J, Pei J , " CLOSET+: Searching For The Best Strategies For Mining Frequent Closed Itemsets". In: Proceeding Of the 2003 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03), Washington, DC, Pp 236–245, 2003.

[11] Han J, Pei J, Yin Y , " Mining Frequent Patterns without Candidate Generation". In: Proceeding Of the 2000 ACM-SIGMOD International Conference on Management of Data (SIGMOD'00), Dallas, TX, Pp 1–12, 2000.

[12] Liu J, Paulsen S, Sun X, Wang W, Nobel A, Prins J , " Mining Approximate Frequent Itemsets In The Presence Of Noise: Algorithm and Analysis". In: Proceeding Of the 2006 SIAM International Conference on Data Mining (SDM'06), Bethesda 2006.

[13] Nadimi-Shahraki M.H., N. Mustapha, M. NSulaiman, and A. Mamat, "Incremental Updating of Frequent Pattern: Basic Algorithms", Proceedings of the secondInternational Conference on InformationSystems Technology and Management (ICISTM 08), pp. 145-148, 2008.D, Pp 405–416, 2006.

[14] Sotiris-Kotsiantis,Dimitris, "Association Rules Mining: A Recent Overview", GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006

[15] Gosta Grahne and Jianfei Zhu," Efficiently sing Prefix-trees in Mining Frequent Itemsets", on ordia University Montreal,Canada,2002.

[16] Agrawal Rakesh, Imilienski T., and Swami Arun. "Mining association rules between sets of items in large datasets", SIGMOD, 207-216, 1993.

[17] Cheung David W., Lee S. D., and Kao Benjamin. "A General Incremental Technique for Maintaining Discovered Association Rules", Proc.International Conference On Database Systems For Advanced Applications, April 1997

[18] Hipp Jochen, Güntzer Ulrich, and Nakhaeizadeh Gholamreza, " Mining Association Rules: Deriving a Superior Algorithm by Analyzing Today's Approaches", 159-168, Lyon, France, September 2000.

[19] Lee Chang Hung, Lin Cheng Ru, and Chen Ming Syan, " Sliding Window Filtering: An Efficient Method for incremental Mining on a Time-Variant Database", Proceedings of 10th International Conference on

Information and Knowledge Management, 263-270, November 2001.

[20] Pei Jian, Han Jiawei, Nishio Shojiro, Tang Shiwei, and Yang Dongqing, "H-Mine: Hyper- Structure Mining of Frequent Patterns in Large Databases", Proc.2001 Int.Conf.on Data Mining, San Jose, CA, November 2001.

[21] Savasere Ashok, Omiecinski Edward, and Navathe Shamkant. "An Efficient Algorithm for Mining Association Rules in Large Databases", Proceedings of the Very Large Data Base Conference, September 1995.

[22] Zaïane Osmar R. and Oliveira Stanley R. M. "Privacy preserving frequent itemset mining", Workshop on Privacy, Security, and Data Mining, in conjunction with the IEEE International Conference on Data Mining, Japan, December 2002.

[23] M. Houtsma and A. Swami, "Set-oriented data mining in relational databases", Data Knowl. Eng.,245–262, 1995.

[24] Han & Kamber ,"Data Pre-processing & Mining Algorithm", 3 edition ISCE,2001.

[25] Agrawal, Rakesh and Ramakrishnan Srikant, "Fast Algorithms for Mining & Preprocessing Assosiation Rules", Proceedings of the 20th VLDB Conference,Santiago, Chile (1994).

[26] Salleb, Ansaf and Christel Vrain, "An Application of Assosiation Knowledge Discovery and Data Mining" (PKDD) 2000 , LNAI 1910, pp. 613-618, Springer Verlag (2000).

[27] Agrawal, R., and Psaila, G. " Active Data Mining" In Proceedings on Knowledge Discovery and Data Mining (KDD -95), 3–8. Menlo Park, 1995.