International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 4, April - 2013

A Brief Overview On Frequent Pattern Mining Algorithms

Dr. N.V.E. S Murthy
Professor, CSE Department,
Andhra university, A.P, India,

P. Pushpa Latha
Asst.professor, CSE Department,
GMR Institute of Technology, A.P, India,

Abstract

Frequent pattern mining is one of the most researched areas of data mining and has recently received much attention from the database community. They are proved to be quite useful in the marketing and retail communities as well as other more diverse fields. This survey study aims at giving an overview of the previous researches done in the field of frequent pattern mining algorithms and other related issues available in the literature.

Index Terms*: Frequent pattern minincg, parallel mining, constraint mining, vertical format pattern mining*

1. Introduction:

Frequent pattern mining has been formulated in 1993 as the computational essential step in the process of association rule mining and has been a focused theme in data mining research. Copious literature has been devoted to this study and incredible progress has been made to numerous research frontiers, such as sequential pattern mining, structured pattern mining, correlation mining, associative classification and frequent pattern-based clustering, as well as their broad applications.

Frequent patterns are pattern set of items, subsequences, subgraphs, etc. that occurs frequently in a data set with frequency no less than a user-specified threshold. For example, a set of items, such as milk and bread, which appear frequently together in a transaction data set is a frequent itemset. A subsequence, such as buying first a Computer, then a printer, if it occurs frequently in a database, is a (frequent) sequential pattern. A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently in a graph database, it is called a (frequent) structural pattern. Therefore, frequent pattern mining helps in finding inherent regularities (associations) in data. Moreover, it helps in data indexing, classification, clustering and other data mining tasks as well. Thus, frequent pattern mining plays an essential role in mining associations and has become an important data mining task.

One important task of frequent pattern mining is association rule mining which was first proposed by Agrawal et al. in 1993 for market basket analysis in the form of association rule mining. It analyses customer buying behavior by finding associations between the different items that customers place in their shopping baskets. Many people have proposed several enhanced algorithms for generating frequent itemsets and all these algorithms differ in some or other ways such as traversing the itemset, satisfying the properties of itemset support and confidence, number of times to scan the entire database and how they reduce the size of the processed database in each pass and so on….Based on the above issues, following sections present an overview of the current status of frequent pattern mining algorithms and some challenging research issues.

2. Basic Algorithms:

In this section we provide some basic frequent pattern mining algorithms. Most algorithms used can be classified as either sequential or parallel. In most cases, it is assumed that the itemsets are identified and stored in lexicographic order (based on item name) which provides a logical manner in which itemsets can be generated and counted. This is the normal approach with sequential algorithms. On the other hand, parallel algorithms focus on how to parallelize the task of finding large itemsets. In the following subsections we describe important features of previously proposed algorithms.

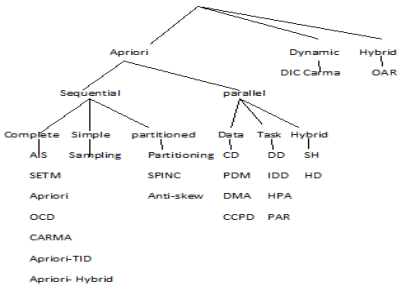

Fig.1 classification of frequent pattern mining algorithms

2.1. Sequential Pattern Mining Algorithms

There are different algorithms used to mine frequent itemsets. All these algorithms vary mainly in terms of properties of itemset support and confidence, number of times to scan the entire database and how they reduce the size of the processed database in each pass

2.1.1 AIS:

The real buzz in 1990's was about how to emulate the biological immune system. The capacity of the immune system to proliferate cells that produce antibodies whenever it detects a high degree of matching with an antigen is. A series of algorithms were invented and new systems called artificial immune systems were designed. [20] AIS was the first algorithm that introduced by agrawal et al. in the year 1993, it was proposed to address the problem of association rule mining. This is a multi-pass algorithm in which candidate itemsets are generated while scanning the database by extending known-frequent itemsets with items from each transaction. The two major drawback of the AIS algorithm is that it generates too many candidates that later turn out to be infrequent and the data structures required for maintaining frequent and candidate itemsets were not specified.

2.1.2 SETM:

[17] The desire to use SQL to generate frequent itemsets results in the introduction of another new algorithm known as SETM. This algorithm was actually created by Houtsma and Swami in October 1993 and included in a research report while they were working in the IBM Almaden Research Center but for some reason it was officially released only in 1995. The algorithm also generates candidates on the fly based on the transaction read from the database, just like AIS algorithm. But SETM was more created for SQL computing and uses relational operations. To use standard SQL join operation for candidate generation, SETM separates candidate generation from counting. It first generates the candidates using equi-joins and then it sorts them all and removes the ones that don't meet the minimum support. Like AIS, SETM also makes multiple passes over the database and generates many candidate itemsets that in the end turn out not be frequent.

2.1.3APRIORI:

The AIS and SETM algorithm was followed by the Apriori algorithm that was shown to perform better than AIS and SETM by an order of magnitude. Agrawal and srikanth in 1994 observed that the main problem that arises in SETM is due to the number of candidates itemsets. Since, for each candidate itemset there is a TID associated with it, it requires more space to store a large number of TIDs. Furthermore, Sarawagi et al. in 1998 has also mentioned that SETM is inefficient. [21] Apriori algorithm was proposed by Agrawal and srikanth in 1994 that was shown to perform better than AIS and SETM. The authors became the legends in the data mining area. They both received masters and PhDs from University of Wisconsin, Madison and both worked for IBM. The IBM's Intelligent Miner was created mainly by them. The most important property of Apriori is that it does not process any itemset whose subset is known to be infrequent (downward closure property). This implies that frequent itemsets can be mined by first scanning the database to find the frequent 1-itemsets, then using the frequent 1-itemsets to generate candidate frequent 2-itemsets and check against the database to obtain the frequent 2-itemsets. This process iterates until no more frequent k-itemsets can be generated for some k. It utilizes a data structure called hash tree to store the counters of candidate itemsets and alternatives to this algorithm is Mannila et al., 1994.

The two major drawbacks with this algorithm is that it performs n passes over the database, where n is the length of the longest frequent itemset and it follows a tuple-by-tuple approach that is, it updates counters of candidate itemsets after reading in each transaction from the database. Hence it suffers from the drawback that much redundant is performed after each and every transaction.

2.1.3.1 Improving the efficiency of Apriori

Since, the Apriori algorithm was proposed there have been many studies on the improvements or extensions of Apriori. Following section describes some important variations of Apriori algorithm.

DHP(Direct Hashing and Pruning): [11] As its name suggests, DHP uses a hash technique that makes it very efficient for the generation of candidate itemsets, in particular for the large two-itemsets, thus greatly improving the performance bottleneck of the whole process. In addition, DHP employs effective pruning techniques to progressively reduce the transaction database size.

Partitioning technique: [1] The algorithm is fundamentally different from all the previous algorithms in that it reads the database at most two times to generate all significant association rules. In the first scan of the database, it generates a set of all potentially large itemsets by scanning the database once and dividing it in a number of non-overlapping partitions. This set is a superset of all frequent itemsets so it may contain itemsets that are not

frequent. During the second scan, counters for each of these itemsets are set up and their actual support is measured.

Sampling approach: [8] This algorithm was proposed by Toivonen in the year 1996 which reduces the database activity. The idea is it first mines a random sample of the database to obtain itemsets that are frequent within the sample. The algorithm thus produce exact association rules, not approximations based on a sample. The approach is, however, probabilistic, and in those rare cases where our sampling method does not produce all association rules, the missing rules can be found in second pass over the entire database. The sampling algorithm too follows a tuple-by-tuple approach and hence, like Apriori, suffers from the above mentioned drawback. Parthasarathy [19] presented an efficient method to progressively sample for association rules. Chuang et al. [3] explore another progressive sampling algorithm, called Sampling Error Estimation (SEE), which aims to identify an appropriate sample size for mining association rules.

DIC (Dynamic Itemset Counting): [23] The DIC algorithm was proposed by Brin et al., in the year 1997 also known as non-level-wise algorithm. In DIC, candidates are generated and removed after every M transaction, where M is a parameter to the algorithm. Although, it is a multi-pass algorithm, it was shown to complete within two passes typically. It however, suffers from the drawbacks of tuple-by-tuple approaches. This algorithm is considered as a closer to the sampling approach proposed by Toivonen, 1996.

Mining with RDBMS: [24] Data mining on large data warehouses is becoming increasingly important. In this regard the association rule algorithms were studied with the twin goals of finding the trade-offs between architectural options and the extensions needed in a DBMS to efficiently support mining. Many experiments were conducted in different ways for implementing the association rules mining algorithm in SQL to find if it is at all possible to get competitive performance out of SQL implementations.

CARMA(Continuous Association Rule Mining Algorithm): [10] This is a novel algorithm which is proposed to compute large itemsets online. Being online, the user is free to change the support threshold parameters such as minimum support and minimum confidence, at any time during the first scan of the transaction sequence. After at most 2 scans the algorithm terminates with the precise support for each large itemset. Although this algorithm did not perform consistently better than Apriori, but by order of magnitude memory utilization is more efficient than Apriori or DIC.

2.1.4 FP-GROWTH:

The Apriori algorithm significantly reduces the size of candidate sets using the Apriori property. However, it can suffer from two-nontrivial costs: (1) generating a huge number of candidate sets and (2) repeatedly scanning the database and checking the candidates by pattern matching. So to overcome these two drawbacks Han et al., devised an FP-growth algorithm [12] based on divide and conquer principle that mines the complete set of frequent itemsets without candidate generation.

It adopts a divide-and-conquer strategy and a FP-Tree[9]. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. In the first pass, the algorithm counts occurrence of items in the dataset, and stores them to 'header table'. In the second pass, it builds the FP-tree structure by inserting instances. Items in each instance that do not meet minimum coverage threshold are simply discarded.

2.1.4.1 Improvements in FP growth algorithm

There are many alternatives and extensions to the FP-growth approach such as:

[28] pascal, by Bastide et al.,(2000) which is named after the French mathematician Blaise Pascal it is basically an optimization of the Apriori algorithm. The authors introduce the notion of key patterns and show that other frequent patterns can be inferred from the key patterns without access to the database. The algorithm finds both frequent and closed sets and it is twice as fast as Close and 10 times as fast as Apriori but is only practical when the pattern length is short.

[14] H-Mine(Hyper-Structure Mining of Frequent Patterns in Large Databases), by Pei et al. (2001) which introduces the concept of hyperlinked data structure (H-struct) and uses it to dynamically adjust links in the mining process and explores a hyper-structure mining of frequent patterns. MAFIA(Maximal frequent itemset algorithm) [6] for transactional databases, by DougBurdick et al., is an algorithm where search strategy integrates a depth-first traversal of the itemset lattice.

An array-based implementation of prefix-tree-structure for efficient pattern growth mining by Grahne and Zhu in 2003 is an efficient array-based algorithm for mining frequent itemsets that greatly reduces the need to traverse FP-trees, thus obtaining significantly improved performance for FP tree based algorithms. This algorithm suits well for sparse datasets. This method outperforms not only

the existing methods that use the FP-tree structure, but also all existing available algorithms in all the common data mining problems. [2] RELIM (Recursive Elimination), by Christian Borgelt in 2005 algorithm is strongly inspired by FP-growth and very similar to H-mine. It doesn't use prefix trees and any other complicated structures. The work is done in one simple recursive function, which can be written with relatively few lines of code.

2.2 Parallel Frequent pattern mining algorithms

The algorithms which adopt the parallelism paradigm can be classified into two categories such as data and task. The algorithms relating to data parallelism include:

2.2.1 CD:

[22] In CD, the database D is partitioned into $\{D^1, D^2, \ldots, D^p\}$ and distributed across n processors. In this algorithm it involves three steps. In step 1, local support counts of the candidates Ck in the local database partition D^i are found. In step 2, each processor exchanges the local support counts of all candidates to get the global support counts of all candidates. In step 3, the globally large itemsets Lk are identified and the candidates of size k+1 are generated by applying an algorithm on each processor independently.

2.2.2 PDM (parallel Data Mining):

[13] PDM is a modification of CD with inclusion of the direct hashing technique proposed in [Park1995]. The hash technique is used to prune some candidates in the next pass. It is especially useful for the second pass, as Apriori doesn't have any pruning in generating C2 from L1.

2.2.3 DMA (Distributed Mining Algorithm) :

[5] DMA is also based on the data parallelism paradigm with the addition of candidate pruning techniques and communication message reduction techniques introduced.

2.2.4 CCPD (Common Candidate Partitioned Database):

[18] CCPD implements CD on a shared-memory SGI Power Challenge with some improvements. It proposes techniques for efficiently generating and counting the candidates in a shared-memory environment. It groups the large itemsets into equivalence classes based on the common prefixes and generates the candidates from each equivalence class.

The algorithms relating to task parallelism include:

2.2.5 DD (Data Distribution):

[22], in DD the candidates are partitioned and distributed over all the processors in a round-robin fashion. There are three steps. In step one, each processor scans the local database partition to get the local counts of the candidates distributed to it. In step two, every processor broadcasts its database partition to the other processors and receives the other database partitions from the other processors, then scans the received database partitions to get global support counts in the whole database. In the last step, each processor computes the large itemsets in its candidate partition, exchanges with all others to get all the large itemsets, and then generates the candidates, partitions and distributes the candidates over all processors. These steps continue until there are no more candidates generated.

2.2.6 IDD (Intelligent Data Distribution):

IDD is an improvement over DD [7]. It partitions the candidates across the processors based on the first item of the candidates, that is, the candidates with the same first item will be partitioned into the same partition. Therefore, each processor needs to check only the subsets which begin with one of the items assigned to the processor.

2.2.7HPA(Hash-based Parallel mining of Association rules):

[25] HPA uses a hashing technique to distribute the candidates to different processors [Shintani1996], i.e., each processor uses the same hash function to compute the candidates distributed to it.

2.2.8 PAR (Parallel Association Rules):

[16] PAR consists of a set of algorithms, which use different candidate partitioning and counting. They all assume a vertical database partition (tid lists for each item), contrast to the natural horizontal database partition (transaction lists). By using the vertical organization for the database, the counting of an itemset can simply be done by the intersection of the tid lists of the items in the itemset.

There are some other parallel algorithms which cannot be classified into the two paradigms if strictly speaking. Although they share similar ideas with the two paradigms, they have distinct features. These parallel algorithms include Candidate Distribution [22], SH(Skew Handling) [15] and HD(Hybrid Distribution) [7].

2.3. Constraint based frequent pattern mining

In order to improve the efficiency of existing mining algorithms, constraints were applied during the

mining process to generate only those patterns that are interesting to users instead of all the patterns. Very often users want to restrict the set of patterns to be discovered by adding extra constraints on the structure of patterns. Data mining systems should be able to exploit such constraints to speedup the mining process. Wojciechowski and Zakrzewicz [27] focus on improving the efficiency of constraint-based frequent pattern mining by using dataset filtering techniques. Dataset filtering conceptually transforms a given data mining task into an equivalent one operating on a smaller dataset. Tien Dung Do et al [26] proposed a specific type of constraints called category-based as well as the associated algorithm for constrained rule mining based on Apriori. The Category-based Apriori algorithm reduces the computational complexity of the mining process by bypassing most of the subsets of the final itemsets. An algorithm, ExAnte, was proposed by Bonchi *et al*. (2003) to further prune the data search space with the imposed monotone constraints. Gade *etal*. (2004) proposed a block constraint which determines the significance of an itemset by considering the dense block formed by the pattern's items and transactions. An efficient algorithm is developed to mine the closed itemsets that satisfy the block constraints. Bonchi and Lucchese (2004) proposed an algorithm for mining closed constrained patterns by pushing deep monotonic constraints as well. Yun and Leggett (2005) proposed a weighted frequent itemset mining algorithm with the aim of pushing the weight constraint into the mining while maintaining the downward closure property.

2.4 Incremental Frequent Pattern Mining

[4] An incremental updating technique is developed for maintenance of the association rules discovered by database mining. There have been many studies on efficient discovery of association rules in large databases. However, it is nontrivial to maintain such discovered rules in large databases because a database may allow frequent or occasional updates and such updates may not only invalidate some existing strong association rules but also turn some weak rules into strong ones. The various algorithms include:

2.4.1 FUP algorithm(Fast Update algorithm) (Cheung *et al*., 1996) FUP is the first algorithm in the field of incremental mining. It operates on an iterative basis and makes a complete scan of the current database. In each scan, the increment is processed first and the results obtained are used to guide the mining of the original database. An important point to note about the FUP algorithm is that it requires k passes over the entire databse, where k is the cardinality of the longest frequent itemset. Further, it does not generate the mining results for solely the increment.

2.5 Frequent Pattern Mining with Vertical Data Format

Most of the algorithms discussed earlier generate frequent itemsets from a set of transactions in horizontal data format (i.e., {TID: itemset}), where TID is a transaction- id and itemset is the set of items contained in transaction TID. Alternatively, mining can also be performed with data presented in vertical data format (i.e., {item: TID_set}). Algorithms which use vertical data format are MAXECLAT, CLIQUE, MAXCLIQUE, TOP-DOWN but ECLAT remained the best known. Few algorithms that supports vertical data format are:

2.5.1 MaxClique:

While the above mentioned algorithms were primarily horizontal (tuple) based approaches, the MaxClique (Zaki *et al*., 1997) algorithm is designed to efficiently mine databases that are available in a vertical layout.

2.5.2Eclat(Echivalence Class Clustering and Bottom-up Lattice Traversal):

[16] It is the first algorithm that uses a vertical data (inverted) layout. ECLAT is very efficient for large itemsets but less efficient for small ones. The frequent itemsets are determined using simple tid-list intersections in a depth-first graph. Zaki (2000) proposed Equivalence CLASS Transformation (Eclat) algorithm by exploring the vertical data format. The first scan of the database builds the TID_set of each single item. Starting with a single item (k = 1), the frequent (k+1)-itemsets grown from a previous k-itemset are generated by Apriori property, with a depth-first computation order similar to FP-growth (Han *et al*., 2004). The computation is done by intersection of the TID_sets of the frequent k-itemsets to compute the TID_sets of the corresponding (k+1)-itemsets. This process repeats, until no frequent itemsets or no candidate itemsets can be found.

2.5.3VIPER:

Unlike earlier vertical mining algorithms which were subject to various restrictions on the underlying database size, shape, contents or the mining process, the viper (Shenoy *et al*., 2000) algorithm does not have any such restrictions. It includes many optimizations to enable efficient processing and was shown to outperform earlier vertical mining algorithms.

3. Comparison of algorithms

The comparison of various algorithms is based upon several metrics. Space requirements can be estimated by looking at the maximum number of candidates being counted during any scan of the database. We can estimate the time requirements by counting the maximum number of database scans needed and the maximum number of comparison operations. Comparison of various algorithms is:

Algorithm	Scan	Data Structure	comments
AIS	m+1	Not Specified	Suitable for low cardinality sparse transaction database; Single consequent
SETM	M+1	Not Specified	SQL Compatible
Apriori	m+1	Lk-1 : Hash table Ck: Hash tree	Transaction database with moderate cardinality; Outperforms both AIS and SETM; Base algorithm for parallel algorithms
Apriori-TID	m+1	Lk-1 : Hash table Ck: array indexed by TID *Ck* : Sequential structure ID: bitmap	Very slow with larger number of *Ck* ; Outperforms Apriori with smaller number of *Ck*
Apriori-Hybrid	m+1	Lk-1 : Hash table 1st Phase: Ck: Hash tree 2nd phase: Ck: array indexed by IDs *Ck* : Sequential structure ID: bitmap	Better than Apriori. However, switching from Apriori to Apriori-TID is expensive; Very crucial to figure out the transition point.
OCD	2	Not specified	Applicable in large DB with lower support threshold.
Partition	2	Hash Table	Suitable for large DB with high cardinality of data; Favors homogenous data distribution
Sampling	2	Not Specified	Applicable in very large DB with lower support.
DIC	Depends On inter Val size	Trie	Database viewed as intervals of transactions; Candidates of increased size are generated at the end of an interval
CARMA	2	Hash Table	Applicable where transaction sequences are read from a Network; Online, users get continuous feedback and change support and/or confidence any time during process.
CD	m+1	Hash table and tree	Data Parallelism.
PDM	m+1	Hash table and tree	Data Parallelism; with early candidate pruning
DMA	m+1	Hash table and tree	Data Parallelism; with candidate pruning
CCPD	m+1	Hash table and tree	Data Parallelism; on shared-memory machine
DD	m+1	Hash table and tree	Task Parallelism; round- robin partition
IDD	m+1	Hash table and tree	Task Parallelism; partition by the first items
HPA	m+1	Hash table and tree	Task Parallelism; partition by hash function

4. Challenging Issues in FP-Mining:

The main research issues with regard to frequent pattern mining are:

- A lot of attention was focused on the performance and scalability of the algorithms, but not enough attention was given to the issues related to ease, flexibility and reusability for generating frequent patterns.
- Most of the algorithms for discovering frequent patterns available in the literature require multiple passes over the database resulting in a large number of disk reads and placing a huge burden on the I/O subsystem. This calls for the introduction of mining algorithms that offers single database scan.
- Various algorithms are available that can help reveal patterns and relationships, but they does not tell the user the value or significance of these patterns.
- Most of the algorithms available in the literature do not offer flexibility for testing the validity of Meta rules.
- There is a requirement for the development of parallel and/or distributed algorithms in order to speed up the computation activity
- Most of the algorithms available in the literature for mining frequent itemsets do not offer flexibility for reusing the computation during mining process.
- Much research is still needed to substantially reduce the size of derived patterns and enhance the quality of retained patterns (compact high quality pattern set).

Conclusion

Data mining has importance regarding finding the patterns, forecasting, discovery of knowledge etc., in different business domains.In this paper, we presented a brief overview of the status of frequent pattern mining algorithms. Over a decade there have been a extensive research, many publications, development and application activities in this domain. It is impossible to give a overall developments on this topic with limited space. Hopefully, this short overview may provide a rough outline to the people a general view of the field.

References

[1] Ashok Savasere, Edward Omiecinski, Shamkant Navathe," An efficient algorithm for mining association rules in large databases" proc. The21stVLDBConference,1995.

[2] ChristianBorgelt, "Keeping Things Simple: Finding Frequent ItemSets by Recursive Elimination ", Workshop Open Source Data Mining Software (OSDM'05, Chicago,IL), in 2005.

[3] Chuang, K., Chen, M., Yang, W., "Progressive Sampling for Association Rules Based on Sampling Error Estimation", Lecture Notes in Computer Science, Volume 3518, Jun 2005, Pages 505 – 515.

[4] David W. Cheung , Jiawei Han , Vincent T. Ng , C. Y. Wong "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique ", 1996.

[5] David Wai-Lok Cheung, Jiawei Han, Vincent Ng, Ada Wai-Chee Fu, and Yongjian Fu, A Fast Distributed Algorithm for Mining Association Rules, Proceedings of PDIS, 1996.

[6] DougBurdick,ManuelCalimlim,JohannesGehrke, "MAFIA: A maximal frequent itemset algorithm for transactional databases", Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, in 2001.
[7] Eui-Hong Han, George Karypis, and Vipin Kumar, Scalable Parallel Data Mining For Association Rules, Proceedings of the ACM SIGMOD Conference, pp. 277-288, 1997.
[8] Hannu Toivonen , "Sampling Large databases for association rules approach"proc. of the 22nd VLDB conference Mumbai, India, 1996.
[9] Han, J. and Pei, J., "Mining frequent patterns by pattern-growth: methodology and implications", ACM SIGKDD Explorations Newsletter 2, 2, 14-20, in 2000.
[10] Hidber , "online association rule mining", 1999.
[11] J.S.Park,M. Chen, P.S. Yu., "An effective hash based algorithm for mining association rules" proc. ACM SIGMOD International Conference on Management of Data, 1995.
[12]JiaweiHan,JianPei,YiwenYin,"MiningFrequentPatterns without Candidate Generation", Proc. of ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, May 16-18, 2000.
[13] Jong Soo Park, Ming-Syan Chen, and PhilipS. Yu, Efficient Parallel Data Mining for Association Rules, Proceedings of the International Conference on Information and Knowledge Management, pp. 31-36, Baltimore, Maryland, 22-25 May 1995.
[14] Jian Pei, Jiawei Han Lu, Shojiro Nishio, Shiwei Tang, Dongqing Yang, "H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases", IEEE International Conference on Data Mining, 2001.
[15] Lilian Harada, Naoki Akaboshi, Kazutaka Ogihara, and Riichiro Take, Dynamic Skew Handling in Parallel Mining of Association Rules, Proceedings of the 7th International Conference on Information and Knowledge Management, pp.76-85, Bethesda, Maryland, USA, 1998.
[16]Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, "New Algorithms for Fast Discovery of Assoctiation Rules", proc.of the 3rd International Conference on Knowledge Discovery and Data Mining, 1997.
[17] Maurice Houtsma, Arun Swami, "Set-oriented mining for association rules in relational databases", proc. Of the11th International Conference on Data Engineering, 1995.
[18] Mohammed Javeed Zaki, Mitsunori Ogihara, Srinivasan Parthasarathy, and Wei Li, "Parallel Data Mining for Association Rules on Shared-Memory Multiprocessors", Technical Report TR 618, University of Rochester, Computer Science Department, May 1996.
[19]Parthasarathy, S., "Efficient Progressive Sampling for Association Rules", ICDM 2002:354-361
[20] Rakesh Agrawal, Tomasz Imielinski, Arun Swami , "Mining association rules between sets of items in large databases", Proceedings of the ACM SIGMOD Conference on Management of Data, Washington, D.C.
[21] RakeshAgrawal and RamakrishnanSrikant, " Fast algorithms for mining association rules", Proceedings of 20th International Conference on Very Large Data Bases, Santiago deChile,Chile, September 12-15, 1994.
[22] [Agrawal1996] Rakesh Agrawal and John C. Shafer, Parallel Mining of Association Rules, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 962-969, December 1996.
[23] S.Brin, R. Motwani, S. Tsur, "Dynamic itemset counting and implication rules for market basket data",1997.
[24] Sunita Sarawagi , Shiby Thomas , Rakesh Agrawal , "integrating mining with relational database systems", in sigmoid,1998.
[25] Takahiko Shintani and Masaru Kitsuregawa, Hash Based Parallel Algorithms for Mining Association Rules, Proceedings of PDIS, 1996.
[26] Tien Dung Do, Siu Cheung Hui, Alvis Fong, "Mining Frequent Itemsets with Category-Based Constraints", Lecture Notes in Computer Science, Volume 2843, 2003, pp. 76 – 86.
[27] Wojciechowski, M., Zakrzewicz, M., "Dataset Filtering Techniques in Constraint-Based Frequent Pattern Mining", Lecture Notes in Computer Science, Volume 2447, 2002, pp.77-83
[28] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, L. Lakhal , "Mining Frequent Patterns with Counting Inference" ACM SIGKDD Explorations Newsletter, in 2000.