IJITKM Volume 7 • Number 2 Jan– June 2014 pp. 188-197 (ISSN 0973-4414)

A Prime Number based framework for Frequent Pattern Mining

Sanjay Patel[1] and K. Kotecha[2]

[1]Computer Engineering Department, Vishwakarma Government Engineering College, Ahmedabad, Gujarat, India , Pin -382424.
[2]Director, Institute of Tecgnology, Nirma University, Ahmedabad, Gujarat, India, Pin – 382481,

sanjaypatel.ce@gmail.com, drketankotecha@gmail.com

Abstract: One of the most popular data mining techniques is association rule mining. Frequent Pattern Mining is part of that. Plentiful literature dedicated to this research and remarkable progress has been made, ranging from efficient and scalable algorithms for frequent pattern mining in transaction databases to various research frontiers. Starting from the most common Apriori algorithm to the latest tree based approaches like CanTree and CanTries has mentioned pros and cons in their method. Bottleneck of the Apriori Approach is the Candidate Generation & Test approach and the number of database scan required. Pattern Growth and other tree based approaches concentrated on interactive and incremental mining. Some of the latest survey suggests applying Computational Intelligence techniques to solve NP-Hard Problem for better results. ACO (Ant Colony Optimization) is one of the most common meta-heuristic so solve combinatorial optimization problems. To represent the problem in Graph form is the basic requirement of ACO. Here in this paper prime number mechanism is used to form the graph such that ACO can be easily applied.

Keywords: Data Mining, Frequent pattern mining, Ant Colony Optimization, Artificial Intelligence, Association Rule Mining

Introduction

There are main three techniques of data mining: (1) Classification, (2) Clustering, and (3) Association Rule Mining. Association rule mining process consists of 2 steps: The first step is to generate patterns from the database with user provided minimum support and second step is to use the generated patterns to form the rules with confidence threshold. The second part is straight forward. The first part is very time consuming, so most of the researchers have concentrated on first part. Frequent Pattern Mining is the foundation of Association Rule Mining. The popular are of application is market basket analysis, which studies the buying habits of customers by searching for sets of items that frequently appear together [3] [12] [13].

The first proposed algorithms for frequent itemset mining were AIS and SETM [7][27]. In both the algorithms candidate items are generated on-the-fly during the pass as data is being read. Particularly, after scanning a transaction, it is determined which of the itemsets found large in the previous pass are present in the transaction. However, the problem is that this results in unnecessarily generating and counting too many candidate itemsets that turn out to be small. To overcome drawbacks of the AIS and SETM algorithms, R. Agrawal et al. [26] proposed Apriori algorithm. It works fine when the user provided support threshold is low and database size is small. For the dense database, it is very much costlier approach. It works on the Anti-monotone (Support of an itemset never exceeds the support of its subsets) property. "Candidate-Generation-and-Test" is the major bottleneck of the Apriori approach. There are many variations proposed to improve the efficiency of the Apriori algorithm, they are as [12]:

a) Transaction reduction,
b) Hash-based techniques,
c) Partitioning the data to find candidate itemsets etc.

The second classes of algorithms are based on Pattern-Growth. In the FP-Growth method [3] [4], during first scan of the database, items with minimum support threshold are listed in the Descending order means the highest support to put first in the list. In the second database scan, actual FP-tree is constructed, based on the list prepared during the first database scan [13]. Even though FP-Growth is efficient in mining large dense databases, it may have the problem of building many recursive projected databases and FP-trees and thus may be costly in space. By nature, Association Rule Mining requires trial and errors. Users perform data-mining with specified support and confidence. In most of the cases, the results from the initial data-mining may not be satisfactory. Users have to change the support or confidence and rerun the process until satisfactory results are obtained. FP-Growth method is sensitive to minimum support threshold. To overcome the drawback of the FP-Growth approach, W. Cheung, 2003 [36] came with the new tree based mechanism known as CATS (Compressed Arranged Transaction Sequences) Tree. The data structure allows data mining without referencing the original dataset. At the same time, the data structure is completely insensitive to and unaffected by user parameters. Once the tree is built, data mining with different supports can be performed without having to rebuild the tree structure. CATS tree supports the interactive mining means "build once – mine many" where database remains unchanged and only the minimum support threshold gets changed. Apart from the advantages, it has some limitations. The merging, swapping and deletion takes too much time for the new transaction. Efficiency for incremental mining is still not clear. With this idea in mind CKS Leung et al. [2] [18] proposed the idea of CanTree. The aim of this method is development of a simple but more powerful, novel tree structure that works for incremental mining. It stores the items in the specific order i.e. either in lexicographic or in ascending order depends on the content of the database. The efficiency of the CanTree is dependent on the order in which items are going to be appeared in the transaction. It suffers from the problem of wastage of storage space. CKS Leung et al. [2] proposed a revised version of its which is known as CanTries. The only difference between the CanTree and CanTries is grouping of the nodes which are having very less frequency. The major problem for this approach is to decide the number of groups to form.

Computational Intelligence techniques provide solution to a wide range of problems. Frequent pattern mining is a kind of NP-Hard problem (Guizhen Yang, KDD 2004). Genetic Algorithms, Genetic Programming, Simulated Annealing, Neural Networks, Ant colony Optimization, Artificial Bee Colony are different types of CI Techniques. Literature survey shows results obtained using CI techniques are better. The main issue to apply the CI technique for any kind of problem is to design the specific framework. Ant Colony Optimization is very well known meta-heuristic which has shown its remarkable effect on solution of the wide range of problems. Travelling Salesman Problem, Graph Coloring, and Quadratic Assignment Problem are the most common examples solved by of ACO. Graph based framework is required to apply ACO. Here in this paper Prime number based framework is discussed for frequent pattern mining which will be applicable in future for applying ACO.

This paper is organized as follows. In the next section, the concept of Association Rule Mining is discussed. Related work is described in Section 3.Proposed work is described in Section 4. Results are discussed in section 5. Conclusion and Future Scope is available in Section 6.

1. Association Rule Mining

Association rule mining was proposed by Agrawal et al. in 1993 [27]. It is an important data mining model studied extensively by the database and data mining community. Association rule mining is to find out association rules that satisfy the user defined minimum support and confidence for the given database.

Let $I=I_1, I_2, \dots , I_n$ be a set of n different attributes, T be transaction that contains a set of items such that $T\subseteq I$, D be a database with different transaction records. An association rule is an implication in the form of $P\Rightarrow Q$, where $P, Q\subset I$ are sets of items called itemsets, and $P\cap Q =\emptyset$. P is called antecedent while Q is called consequent, the rule means P implies Q. Considering the example of a store that sells DVDs, Computer, Anti-virus software, Mobile Phones, Videos, CDs and Games, the store owner might want to discover which of these items customers are likely to buy together. With the information above, the store could attempt for finest placement of Computer and Anti-virus software as the sale of one of them may improve the chances of the sale of the other frequently associated item. The mailing campaigns may be fine tuned to reflect the fact that offering discount coupons on Videos may even negatively impact the sales of DVDs offered in the same campaign. A better decision could be not to offer both DVDs and Videos in a campaign. In this case, it is appropriate to use association rule mining to generate the optimum combination of products to increase sales [11]. The computational cost of association rules mining can be reduced in three ways:

- By sampling the database
- By reducing the database passes
- Through parallelization.

In recent years much progress has been made in all these directions. Most commonly areas in which association rules have been used include:

• Credit card transactions: items purchased by credit card give insight into other products the customer is likely to purchase.

• Supermarket purchases: common combinations of products can be used to inform product placement on supermarket shelves.

• Telecommunication product purchases: commonly associated options (call waiting, caller display etc...) help determine how to structure product bundles which maximise revenue.

• Banking services: the patterns of services used by retail customers are used to identify other services they may wish to purchase.
• Insurance claims: unusual combinations of insurance claims can be a sign of fraud.
• Medical patient histories: certain combinations of conditions can indicate increased risk of various complications.

2. Related Work

In this section, existing methods for frequent pattern mining are discussed. Basically three different types of approaches are discussed as mentioned below.

- Candidate Generation-and-Test Approach
- Pattern Growth (Tree based) Approach
- Graph based Approach

2.1 Candidate Generation-and-Test

Three algorithms for this approach are discussed herewith. The First ever proposed algorithm is known as AIS [27]. Candidate itemsets are generated and counted on-the-fly as the database is scanned. The next algorithm proposed in this category is known as SETM [7]. It was motivated by the desire to use SQL to compute large itemsets. Like AIS, SETM also generates candidates on-the-fly based on transactions read from the database. It thus generates and counts every candidate itemset that the AIS algorithm generates. However, to use the standard SQL join operation for candidate generation, SETM separates candidate generation from counting. SETM remembers the TIDs of the generating transactions with the candidate itemsets. The problem with this approach is due to the size of candidate sets. Agrawal et al.[26] tried to remove the drawback of AIS and SETM algorithm by Apriori algorithm. Apriori is designed to operate on databases containing transactions. Apriori outperforms AIS on problems of various sizes. It beats by a factor of two for high minimum support and more than an order magnitude for low levels of support. The support counting procedure of the Apriori algorithm has attracted voluminous research owing to the fact that the performance of the algorithm mostly relies on this aspect. There are three versions of the Apriori Algorithm. Apriori (basic version) faster in first Iterations. AprioriTid faster in later iterations and AprioriHybrid can change from Apriori to AprioriTid after first iterations. However, some issues of the Apriori algorithm are discussed below [26].

- Needs several iterations of the data
- Uses a uniform minimum support threshold
- Difficulties to find rarely occurring events

Alternative methods (other than Apriori) can address this by using a non-uniform minimum support threshold. Some competing alternative approaches focus on partition and sampling.

2.2 Pattern-Growth (Tree Based) Approach

FP- Growth, CATS Tree and FELINE Algorithm, CanTree, CanTries are the key approaches which utilizes the Pattern-Growth Approach. Variations and improvements of these algorithms are also available in the literature. After Apriori invention, researchers were in search of the procedure that removes the candidate generation and test procedure. Han et al. [12] came with the solution of the above said problem by the first Pattern growth approach named FP-Growth. Han et al.[12] proved that method outperforms other popular methods for mining frequent patterns like Apriori and the extension of the Apriori methods. Although FP-Growth is more efficient than Apriori algorithm, it may still encounter some problems in some cases as: (1) Hugh space is required to serve the mining, (2) Large applications needs more scalability; (3) Real database contains all the cases. J. Pei et al. [13] proposed a new data structure called H-struct and new mining method, H-mine to overcome the problems of the FP-growth approach. H-struct is unique without considering the order of frequent item projections in queues of the frequent item projections. Only two scans of a transaction database are needed to build an H-struct. H-mine can be scaled-up to very large databases due to its small and precisely predictable run-time memory overhead and its database partitioned mining technique. W. Cheung et al.[36] proposed frequent pattern mining method without candidate generation or support constraint which is known as CATS (Compressed Arranged Transaction Sequences) Tree and

FELINE algorithm. The differences between the CATS Tree and FP-Tree are available in [36] which are presented here.

CATS Tree	**FP-Tree**
Contains all items from every transaction	Contains only frequent items
Single Scan data mining	Two scan data mining
Items within a transaction do not need to be sorted	Items within a transaction are sorted
Sub-trees are locally optimized to improve compression	Sub-trees are not locally optimized

Procedure is divided in two steps. In the first step, CATS Tree is built and in the second step based on CATS Tree, FELINE (FrEquent/Large pattern mINing with CATS Tree) algorithm finds the frequent itemsets. Once CATS Tree is built, it can be mined repeatedly for frequent patterns with different support thresholds without the need to rebuild the tree. It supports the interactive mining. The merge, swap, delete during construction is the most complex procedure in the generation of the CATS tree. Also it does not give guarantee for incremental mining. C.K.S. Leung et al. (2005) [2] [18] designed new method- CanTree (Canonical order Tree), aims for the incremental mining. Attractiveness of the method is when the database updated, algorithm does not need to rescan the entire database. Items in the CanTree can be consistently arranged in lexicographic order or the alphabetic order. Alternatively, items can be arranged according to some specific order depending on the item properties. The number of branches in the tree construction is a major issue in maintaining the data structure. Later, the authors [2] tried to make the tree structure less complex by introducing the next version of CanTree, CanTries. This holds the data in one group of the same frequency. But overall the problem remains the same.

2.3 Graph Based Approach

In most of the tree based approaches, the same item needs to be stored at different places in the tree structure due to the individual property of the algorithms and methods. This seems to be even complex in the designing approach and mining procedure. Based on this observation, P. Deepa Shenoy et al. (ADCOM -2003) [19] presented *Compress and Mine*: Graph based algorithm to generate frequent itemsets. The algorithm scans the database once to find the frequent one items and scan it again to construct an association graph to derive the path matrix and finally it traverse the path matrix to generate frequent itemsets. The details regarding the same are available in [19]. R.S. Thakur et al. (2008) [23] worked for graph theoretic based frequent pattern mining algorithm. In that, focus was on reducing database scan and avoiding candidate generation. It reduces I/O time as well as CPU overhead.

V.Tiwari et al. (2010) [34] projected graph based approach, known as FP_Graph. This algorithm mainly contains three parts: First is Generation of a graph from the transaction database, The second part is for pruning the graph for the items having value lower than minimum support, and the last section is for generating the results from the pruned graph. Each edge in graph contains three values marked on it. Out of the three values on the edge, the first value is the direct support of the two items, the second value in form of ‘Character’ indicates the staring node and the third value justifies the total number of nodes visited up to that node. Under small minimum support condition it gives the good results. The method creates the problems for the large minimum supports. Some others concepts related concepts are also available in [1],[6], [8], [9][10][14][24][25][30].

All the concepts related to graph based frequent pattern mining, have the same characteristics of directed graph mechanism. As far as the database and the frequent pattern mining are concerned, it is not related to the direction. For example, if in one transaction the items are in the sequence ‘a’ and ‘b’ and in another transaction items are in the sequence ‘b’ and ‘a’ then both having the same importance. But, in the case of directed graph based mining, both are treated separately. To solve this problem, non directed graph based method is required. Henceforth, the method is discussed which is based on the non directed graph. K. Kotecha et al. (2011) [15] proposed non directed graph based approach, in which the item of the transaction database acts as a node in the graph. While the transaction number stored as a string between two items. For example, transaction 2 and 3 contain items D and F than edge between D and F will represent 23, means the relation of D and F is because of transaction 2 and 3. Normally the counter between any two items is not enough because it is difficult to distinguish the items sequence. Also K. Kotecha et al. (2013) [16] extended the same approach for the incremental mining.

3. Proposed Work

As mentioned earlier, frequent pattern mining is a NP Hard problem[Guizhen Yang, KDD'04,]. As the number of transactions grow, storage and retrieval of data becomes difficult. Computational intelligence has shown great impact to solve the NP and P Problems. The representation of the database in non directed form has the problem related to number of parameters on the edge. All the computational intelligence techniques work sound if the numbers of parameters are fixed. In the work try has been made to fix the number of parameters on the edge.

3.1 Present Investigation

Probably, there are two ways to build an intelligent algorithm. Deciding the final solutions from past experience (learning) or create a model of the system. Here, the system changes with input and there can be many new inputs in data. System cannot be predicted beforehand. So, here try is to make a model of the present system from the available transactions. Next question arose like what type of model to create: Tree, Graph, Plot functions? With tree, as with previous approaches like FP tree, as the transactions and items increased, the tree height increases and it is not a dynamic approach. For Graph, can dynamically expand and get the solution starting from any node. Also can get the solution by traversal techniques while building the graph and do not need to wait till the whole graph is prepared. Other issues faced were:

- Deciding nodes and edges in graph – whether to take items as nodes and links between them denoting they occurred in the same transaction.
- If nodes and edges are final then what will be the attributes for each.
- If assign an **id (identity)** to each transaction or just keep a tab on the number of occurrences of item sets.

  Here, keeping only a count on number of times the item sets occur there arose ambiguity while deciding the solution from a traversal method. In traversal, could not differentiate if the item sets occurred in the same transaction or in different transactions.
- If associate an **id** for each transaction OR per each item OR per each item sets.
- Method to store and retrieve the id in implementation point of view.
- If all above is set, then how to traverse in graph for solutions.

3.2 Design

Method is based on undirected graph. Each item is a node and items in a single transaction are linked with the edges means every item node is linked i.e. edge is made; with every other item node in a single transaction. If number of unique items (nodes) in a transaction is **n** then it has $\mathbf{nC_2}$ edges in graph. Edge has two weights (attributes) namely ***fid*** (**F**requent item set **Id**entifier) and ***c*** (**C**ounter). Each transaction is associated with a unique prime number starting from the first prime number 2. Table 1 shows the transaction database and Figure 1 shows the representation of database of Table 1.

For instance:

Transaction No.	Prime id	Items in each transaction
1	2	A, B, C
2	3	B, C
3	5	B, C D
4	7	A,B, C, D
5	11	A, B, C, D, E

Table 1: Dataset

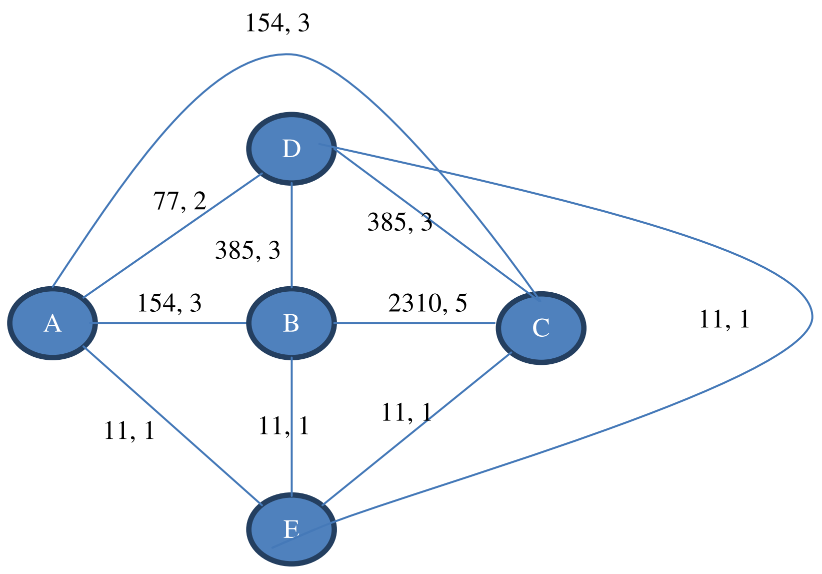

Figure 1: The representation of Dataset of Table 1

4.3. Proposed FUID (Frequent Item set Unique Identifier) Algorithm

Algorithm works in two phases:

1) Transaction storage procedure
2) Data extraction procedure:

Each transaction is appended with its prime number.

1) Transaction storage procedure

```
start
create graph.
If(edge is present)
    update attributes of the edges
    c = c +1;
    fid= fid* primeid;
else
    create edge;
    c=1;
    fid = primeid;
end;
```

2) Data extraction procedure:

Traversal of graph so to visit all the nodes.

```
START:
select next unvisited node;
present-node = selected node;
```

```
in-traversal-node = present-node;
TRAVERSE:
if(in-traversal-node has unvisited edge)
        tempedge = next_edge;
        in-traversal-node = src(tempedge);
        d_node = destination(tempedge);

        if(checkCondition(tempedge))
                append nodes in answer;
                in-traversal-node = d_node;
                mark tempedge visited;
                goto TRAVERSE;
        else
                mark tempedge visited;
                goto TRAVERSE;
else
        in-traversal-node = previously traversed node;
        if(in-traversal-node is not equal to present-node);
                goto TRAVERSE;
        else
                if(in-traversal-node has unvisited edge)
                        goto TRAVERSE;
                else
                        mark present-node visited;
                        goto START;
END;
Method: checkCondition(edge)
if(measured_support(edge)>= minimum support)
        return true;
else
        false;
```

Final answer after traversal from all nodes is.

Frequent Itemsets	**Measured_support**
➢ D, G	4
➢ D, F, G	4
➢ D, F	4
➢ F, E	3
➢ F, G, E	3
➢ F, G	6
➢ G, E	3

If comparison between the exact solution and explicitly prune list is made, optimal answer getting, i.e. all possible frequent item sets before applying traversal algorithm.

4. Results

Following Charts shows the comparison and results obtained by proposed method. Initially the results are not convincing as compared to other available methods. Improvement is required for the better results. As non directed graph based mechanism and implementation are not found in the available literature, not possible to compare with the same kind of method. So, try has been made to compare with available methods. At the initial phase, there may be some time and space limitations but can be improved later on. Figure 2 Shows the Comparison between FP-Growth-Graph and Prime_Based_Graph (Proposed Method). The comparison is made for Time (Y-Axis) and Minimum Support (X-Axis). In case of Low Minimum support the FP-Growth-Graph performs better. As the Minimum support increases the Prime_Based_Graph also performs better.

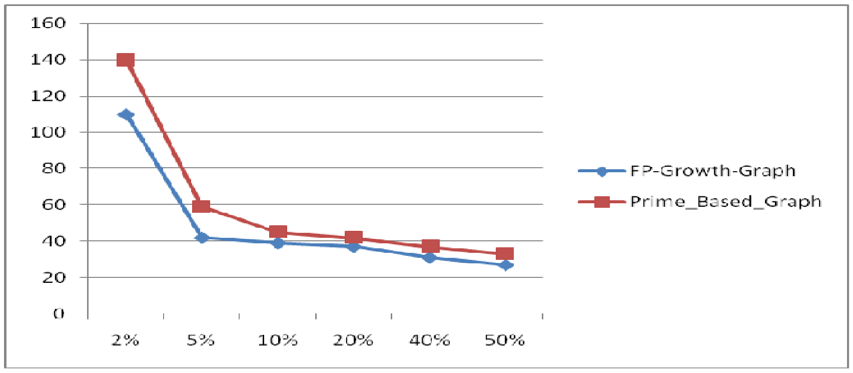

Figure 2: Minimum Support vs. Time

Figure 3 shows the comparison between Apriori Algorithm and Prime_Based_Graph for the Time (Y-Axis) and Number of Transactions (X- Axis). Apriori Algorithm performs better for more transactions because the prime based calculation is complex more increasing number of transactions.

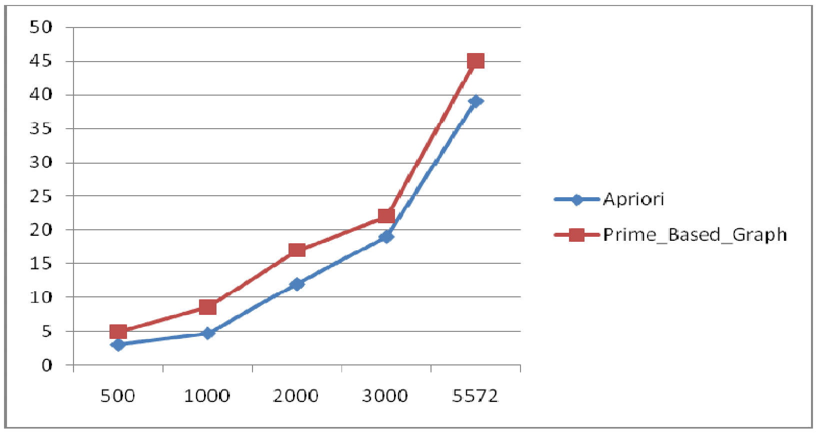

Figure 3: Number of Transactions Vs. Time

5. Conclusion and Future work

As the literature review is concerned, here first try has been made to represent the database in the form of undirected graph. In comparison of tree, the traversal of the graph is difficult, but if want to apply any of the computational intelligence technique than graph representation is better. Directed graph is not suitable for this kind of problem. This paper introduces prime number based framework to mine the frequent itemsets. The key advantage of this approach is the number of parameters remains same on every edge. Although using GCD concept, it is typical to mine frequent itemsets using limited memory. Here it is assumed that there is an unlimited amount of main memory, but there is possibility that system may run out of memory while it is running. The results obtained using this method is accurate.
As a future work, the same approach can be solved by any of the computational intelligence technique like ACO, Genetic Algorithm, Neural Networks to reduce the GCD overhead and to solve the memory issues.

References

1. Alva Erwin, Raj P.Gopalan and N.R.Achuthan(2007), ”CTU-Mine: An efficient High Utility Itemset Mining Algorithm using Pattern growth approach” Seventh International conference on Computer and Information Technology,2007.
2. Carson Kai-Sang Leung et al. (2007), “CanTree: a canonical- order tree for incremental frequent-pattern mining”, Knowledge and Information Systems, 2007, 11(3), Page 287-311.
3. Cheung W.(2002), ”Frequent Pattern mining without candidate generation or support constraint.” Master’s thesis, University of Alberta, 2002, SPRING ’03, doi.ieeecomputersociety.org/10.1109/IDEAS.2003.1214917.
4. Christian Borgelt(2005), “ An Implementation of the FP-growth Algorithm” OSDM’05, August 21, 2005, Chicago, Ilinois, USA.
5. Elena Baralis et al. (2013), “ P- Mine: Parallel itemset mining on large datasets”, Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on, Date 8 – 12 April, 2013.
6. En Tzu Wang et al. (2011) , “ Mining Frequent Itemsets over distributed data streams by continuously maintaining a global synopsis” Data Mining and Knowledge Discovery, Springer, September 2011, Volume 23, Issue 2, pp 252-299
7. G. Piatetsky-Shapiro et al. (1991), “ Knowledge Discovery in Databases” , MIT Press, 1991.
8. Geoffrey I. Webb(2010), “ Self-Sufficient Itemsets: An Approach to Screening Potentially Interesting Association Between Items” , ACM Transactions on Knowledge Discovery from Data, Volume 4, No.1, Article 3, January 2010.
9. Guimei Liu et al. (2013), “ A Flexible Approach to finding Representative Pattern Sets”, *IEEE Transactions on Knowledge and Data Engineering*, 04 Feb. 2013. IEEE computer Society Digital Library. IEEE Computer Society.
10. H.D.K. Moonesinghe, S. Fodeh, P.N. Tan(2006), “Frequent Closed Itemset Mining using Prefix Graphs with an efficient Flow-Based Pruning Strategy” , Proceedings of the sixth International Conference on Data Mining (ICDM ’06) IEEE Computer Society.
11. Irina Tudor(2008), “Association Rule Mining as a Data Mining Technique” , Buletinul Universitatii Petrol Gaze din Ploiesti, Seria Matematica-Informatica-Fizica , Vol. LX, No. 1/2008, page 49 – 56.
12. Jiawei Han and Micheline Kamber, Book :”Data Mining, Concept and Techniques”, 578 pages, books.google.co.in.
13. Jien Pei, ”Pattern-Growth methods for frequent pattern mining “ Ph. D. Thesis, Simon Fraser University, 2002.
14. Jilles Vreeken et al.(2011), “ KRIMP : mining itemsets that compress” Data Mining and Knowledge Discovery, Springer, July 2011, Volume 23, Issue 1, pp 169-214.
15. K. Kotecha et al. (2011), “Frequent Pattern Mining Using Graph Based Approach” , International Journal of Computer Science Research and Application *2011, Vol. 01, Issue. 02*
16. K. Kotecha et al. (2013), “Incremental Frequent Pattern Mining using Graph based approach” , International Journal of Computers & Technology, Volume 4 No. 2, March-April, 2013, ISSN 2277-3061
17. Liang Wang et al.(2012),”Efficient Mining of Frequent Itemsets on Large Uncertain Databases”, IEEE Transactions on Knowledge and Data Engineering, Vol. 24, No. 12, December 2012.

18. Q. I. Khan, T. Hoque and C.K. Leung, " CanTree : A Tree structure for Efficient Incremental mining of Frequent Patterns". Proceeding of the Fifth International conference on Data Mining (ICDM'05), csdl2.computer.org/.../&toc=comp/proceedings/icdm/2005/2278/00/2278toc.xml&DOI=10.1109/ICDM 2005.
19. P. Deepa Shenoy et al.(2003) , " Compress and Mine: An Efficient Graph Based Algorithm to Generate Frequent Itemsets" , Advance Computing and Communicating Society , 2003.
20. P. M .Kanade, et al. (2007), " Fuzzy Ants and Clustering", IEEE Transactions on Systems, Man and Cybernatics, Volume 37, No. 5, September 2007.
21. R. J. Kuo et al. (2007) , " Association Rule mining through the ant colony system for National Health Insurance Research Database in Taiwan" , An International Journal of Computers and Mathematics with applications 54, Science Direct, pp 1303-1318.
22. R. S. Parpinelli et al. (2002), " Data Mining with an Ant Colony Optimization Algorithm", IEEE Transactions on Evolutionary Computing, Volume 6, No. 4, August 2002.
23. R. S. Thakur, R.C. Jain and K.R. Pardasani(2008), " Graph Theoretic Based Algorithm for Mining Frequent Patterns" Neural Networks, IJCNN (IEEE world Congress on Computational Intelligence). 1-8 June, 2008.
24. Rajnish Dass and Ambuj Mahanti, " An efficient heuristic search for Real- Time frequent pattern mining".International Conference on System Sciences – 2006, ieeexplore.ieee.org/iel5/10548/33362/01579371.pdf
25. Rajanish Dass et al. (2006) ," An Efficient Algorithm for Real-Time Frequent Pattern Mining for Real-Time Business Intelligence Analytics", Proceedings of the 39th International Conference on System Sciences- IEEE, 2006
26. Rakesh Agrawal et al.(1994) , " Fast Algorithms for mining Association Rules" , Proceedings of the 20th VLDB Conference Santiago, Chile , 1994.
27. Rakesh Agrawal et al. (1993), " Mining Association Rules between set of items in large databases", In proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD ' 93) pages 207-216, May 1993.
28. S. Shankar et al.(2009),"Utility Sentient Frequent Itemset Mining and Association Rule Mining : A Literature Survey and Comparative Study" , International Journal of Soft Computing, ISSN: 1453 – 2277 Issue 4 (2009), pp 81-95.
29. S. Shankar et al. (2009), " A Novel Utility Sentient Approach for Mining Interesting Association Rules" , IACSIT International Journal of Engineering and Technology, Volume 1, No. 5, December 2009.
30. Shailesh Kumar, et al.(2012) , " Logical Itemset Mining " IEEE 12th International Conference on Data Mining Workshops, December 2012.
31. Tianyi Wu et al.(2010), " Re-examination of interestingness measures in pattern mining : a unified framework" Data Mining and Knowledge Discovery, Springer, November 2010, Volume 21, Issue 3, pp 371-397
32. Tias Guns et al. (2013), " k- Pattern Set Mining under Constraints" , IEEE Transactions on Knowledge and Data Engineering, Vol. 25, No. 2, February 2013.
33. Vincent S. Tseng et al.(2013), " Efficient Algorithms for mining high utility Itemsets from Transactional Databases" , IEEE Transactions on Knowledge and Data Engineering, Vol. 25, No. 8, August 2013.
34. Vivek Tiwari et al., (2010) , Association rule mining: A Graph Based Approach for mining Frequent Itemsets, 2010 International Conference on Networking and Information Technology, 978-1-4244-7578-0 © 2010 IEEE
35. W.J. Jiang et al.(2005), " A Novel Data Mining Algorithm based on Ant Colony System" , Proceedings of the fourth International Conference on Machine Learning and Cybernatics, Gaungzhou, 18-21 August 2005.
36. William Cheung et al. (2003), " Incremental Mining of Frequent Patterns without candidate Generation or Support Constraint", IDEAS'03, doi.ieeecomputersociety.org/10.1109/IDEAS.2003.1214917.