AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Efficient mining of Frequent Itemsets and Association Rules

[1]P. Asha, [2]Dr.T. Jebarajan, [3]Varun Cyriac Thomas

[1]*Research Scholar, Computer Science and Engineering Department, Sathyabama University, Chennai. India.*
[2]*Head of the Department, Computer Science & Engineering, Rajalakshmi Engineering College, Chennai, India.*
[3]*M.E Student, Computer Science and Engineering Department, Sathyabama University, Chennai, India.*

ARTICLE INFO

Article history:
Received 2 February 2014
Received in revised form
8 April 2014
Accepted 28 April 2014
Available online 25 May 2014

Key words:
Data Mining, Dissimilar Rules, Frequent Items, Hash Function, Rule Generation, Grouping.

ABSTRACT

Background: Association rules are mainly used for finding the frequent item sets in data mining. The results can be used to predict future information or explain current relation. Apriori algorithm is a popular method for association rule mining. **Objective:** The work is based on association rule mining, which uses an efficient Hash based Apriori algorithm and finds dissimilar rules from the generated rules. One of the main problems in developing association rules mining algorithms is that large number of rules are generated, which makes the algorithms inefficient and difficult to implement. The existing mining algorithms cannot perform efficiently due to high and repeated association rules. From the experimental results and discussions we assure that the proposed work can extend a wide support for Decision Support Systems and outperforms other existing algorithms. **Conclusion:** The proposed algorithm suggests an improvement to the Apriori. This method enhances the efficiency of algorithm by generating frequent item set by the number of candidate item sets generated through hash based approach. Proposed algorithm generates more number of frequent item sets and rules in lesser time. Through filtering the infrequent item sets and by retaining the frequent ones strong and best rules are retrieved. Final results confirm that, with frequent item set generated within less time and create relevant rules only after grouping the rules.

To Cite This Article: P.Asha, Dr.T. Jebarajan and Varun Cyriac Thomas., Efficient Mining of Frequent Itemsets and Association Rules. ***Aust. J. Basic & Appl. Sci.,*** *8(7): 510-571, 2014*

INTRODUCTION

Data mining refers to extract useful information from large databases. Association rule mining (Sushma *et al*. 2012), Clustering, Classification, Regression are all different flavors of the data mining techniques. The current work focuses on Association Rule Mining and which again has been done by many different algorithms suggested by many authors. Below are presented few literature surveys on existing Association Rule mining algorithms.

Farah Hanna *et al*. (2011) proposed that the repeated disk overhead can be reduced by reducing candidate itemset (size related). Association rules were generated from relational databases and data warehouses using the basic Apriori. It doesn't tell about the processing speed and the efficiency of the rules generated.

Hranchotchuang *et al*. (2013) declares that Hash function can be used instead of normal joining method. Transaction reduction is done by generating hash function using DHP algorithm. Collision in hash table reduces the effectiveness of hash table. Hash functions are complex to be coded after 2-itemset combination.

Bouker *et al*. (2013) said that dominating rules can be grouped. It has generated rules by means of satisfying user defined itemset and grouped the rules based on user defined filtering methods. The drawback is that, the algorithm doesn't consider all itemsets and rules generation is limited to specific itemsets and no rule filtering.

Jia Ronga *et al*. (2012) said that different filtering methods were used for rule grouping. Rules generation is applied only to smaller database. Only it can be used for tourism dataset and cannot be considered for other dataset.

Shinji Funjiwara *et al*. (2005) found all implications and similarity rules based on confidence pruning without support pruning. Dynamic pruning was done effectively for large data sets by counting the number of rows and they avoided the rows that contained missing values. Only confidence pruning is considered. They focused only on the reduction in transaction and not efficient rule generation.

Corresponding Author: P.Asha, Research Scholar, Computer Science and Engineering Department, Sathyabama University, Chennai. India.
E-mail: ashapandian225@gmail.com.

Huan Wu *et al*. (2009) used count based method. They followed <itemset, Tids> structure for storing the data. The method counts each candidate itemsets only once. The disadvantage of the system is that, it spends more time for building <itemset, Tids> structure, which may never be used for further processing except at the initial phase.

Yang Xiang *et al*. (2012) discovered patterns in very large databases, rather than simply verify that whether the pattern exists. High performance and near-linear scaling on very large (multiple gigabytes) real-life databases was their motto. They have a completeness property which guarantees that all patterns of certain types have been discovered. Rule grouping and performance analysis is done manually.

Jayalakshmi *et al*. (2012) proposed that, in addition to introducing the problem of pattern finding, presented three algorithms for solving this problem, but these algorithms do not handle time constraints, sliding windows, or taxonomies. Two of these algorithms were designed only to find maximum sequential patterns, however many applications require all patterns and its supports. Third algorithm is Apriori. It finds all patterns and the performance was better when compared to the first two algorithms.

Phaichayon Kongchai *et al*. (2013) narrated a work in which the rules are generated by Apriori algorithm and the generated rules are grouped by their similarities. Grouping is done based on rule head similarity checking. Grouping increases the performance of mining.

One of the modern approaches for rule reduction technique explained by Ramesh Kumar *et al*. (2013) states that usage and benefits of irrelevant rule elimination technique. This uses cross validation technique, where rules are cross checked during the phase of generation. Selecting the rules after filtrations for grouping is by means of randomized probability. It uses advanced Fisher-Yates algorithm. Complexity of the algorithm is less.

Anand *et al*. (2013) reduced the complexity of algorithm using Correlation Threshold. It uses zeros and ones for its presence and absence while generating rules frequent itemset and rules. Items with minimum correlation are eliminated. It reduces the space complexity.

Xindong Wu *et al*. (2014) explain how to handle big data. They proposed a method HACE which handles the big data issues efficiently. They consider the data as large-volume, heterogeneous, autonomous sources with distributed and decentralized control, and explore complex and evolving relationships among data.

Kannika NiraiVaani *et al*. (2013) proposed a new method for generating rules by lift ratio. Interestingness measures are controlled with minimum level and the generated rules are filtered. Support and confidence are set and lift factor is used for filtering the rules.

Shih-Sheng Chen *et al*. (2011) proposed a method for frequent periodic pattern using multiple minimum supports. This is an efficient approach to find frequent pattern because it is based on multiple minimum threshold support based on real time event. All the items in transaction are arranged according to their minimum item support (MIS), and it does not hold download closure property, instead it uses sorted closure property based on ascending order. Then PFP (periodic frequent pattern) algorithm is applied which is same as that of FP-growth where conditional pattern base is used to discover frequent patterns. This algorithm is more efficient in terms of memory space, thereby reducing the number of database scans.

Agarwal *et al*. (1993) developed an algorithm for mining association rules between sets of items in large databases. Association rule mining is an if/then statement that helps to uncover relationships between seemingly unrelated data in a relational database or other information repository. Apriori Association rule mining technique uses a two step process. The first step is to identify all the frequent itemsets based on the support count value of the itemsets. It uses the download closure property of itemsets to remove the infrequent itemsets. The second step is the generation of association rules from the frequent itemsets using the support and confidence. Few of the efficient algorithms existing were explained below.

Apriori Algorithm:

It finds frequent itemsets based on iterative bottom-up approach by generating the candidates. The Apriori Algorithm (Agrawal *et al*. 1996) is an influential algorithm for mining frequent itemsets for Boolean association rules (Zaki *et al*.1999).

Key Concepts:

- Frequent Itemsets: The sets of item which has minimum support.
- Apriori Property: Any subset of frequent itemset must be frequent.

1. Join step: Generate all possible candidate item sets Ck of length k
2. Prune step: Remove those candidates in Ck that cannot be frequent.

Find the frequent itemsets: the sets of items that have minimum support

- A subset of a frequent itemset must also be a frequent itemset ,i.e., if {AB} is a frequent itemset, both {A} and {B} should be a frequent itemset
- Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset)
- Use the frequent itemsets to generate association rules.

For each frequent item set X, and for each proper nonempty subset A of X, let B = X – A.

Then, $A \rightarrow B$ is termed as association rule, if

Confidence (A → B) ≥ minconf [17],
Support (A → B) = support (A∪B) = support(X)
Confidence (A → B) = support (A ∪ B) / support (A)

Dynamic Hashing and Pruning:

It uses a hash table for precomputing an appropriate support of 2-item sets in the initial iteration. In the next iteration it counts only the candidates present in hash cells with minimum support.

Sear and Spear:

SEAR (Sequential Efficient Association Rules Algorithm) stores the candidates in a prefix tree called Trie. Here each edge is labeled by items and common prefixes are represented by tree branches and unique suffixes are stored at the leaves. It uses pass bundling optimization.

Proposed Methodology:

The proposed work deals with big data in an efficient way. That is efficiency in terms of time complexity and space complexity. Proposed algorithm has three stages. The hash based Apriori algorithm is first one which makes way for finding the frequent itemset with less complexity. Hash function is used in all level of hash table creation method. Second stage is generating the rules. Third step is that, the mined rules are joined together based on few constraints.

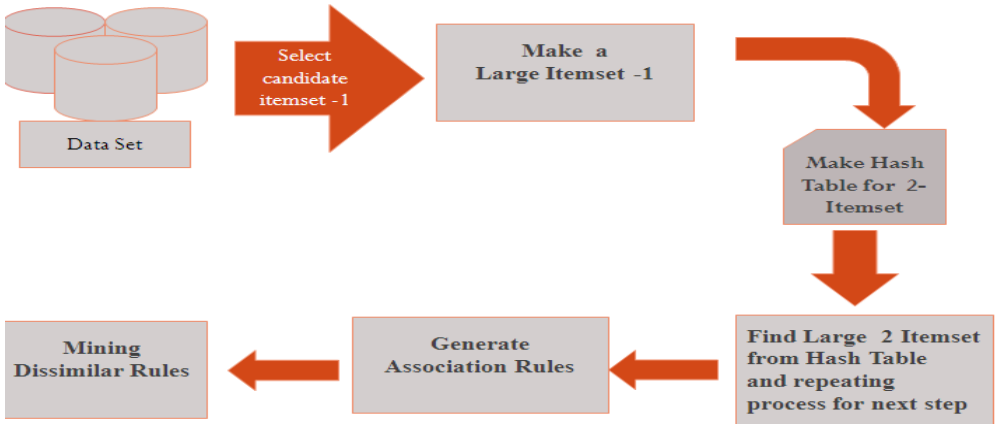

Fig. 1: Architecture Diagram

Hash based Apriori is used and the collision in hash table is reduced by generating separate column for each item set (Figure 1). Additional rule filtering parameters like Confidence, Support, Lift, Leverage, and Coverage is used [15]. Finally we group similar category rules as clusters.

Hash based Frequent Itemset Mining Algorithm:

The algorithm starts with a candidate itemset of one.
C_k: Candidate item set of size k
L_1<- frequent 1-itemsets
Generate candidate for every L_k do begin
C_{k+1} <- candidate(L_k) #New candidates
Join Step: C_k is generated by joining L_{k-1} with itself
For all transactions t ∈*D* do begin
C_k <- subset [C_k, t] #Candidates contained in t
For all candidates c ∈C_t do
c. count++;
Prune Step: Any (k-1) –Subset of infrequent itemset must be infrequent.
L_k {c ∈C | c.count>= minsup}
Final $L_k$as frequent item from dataset.

Sample code for calculating the support count:

The following code explains a portion of R code required:

```
supportcount<-function(itemset,data,sc)
{  n<-dim(data)[1]    m<-dim(itemset)[1]
l<-dim(itemset)[2]    j<-1
```

```
spm <- data.frame(matrix(ncol = 1,
nrow = m),check.rows=FALSE)
#for storing support count
colnames(spm)<-c("x")
while(j<=m)
{    candidateset<-as.data.frame(itemset[j,])
#print(candidateset)
counttot<-0          i<-1
counttot<-sp(itemset,data,sc)
spm[j,]<-counttot     j<-j+1       }
rt<-as.data.frame(cbind(itemset,spm))
sorted<-rt[order(-rt$x),]
limit<-round(sc*m)
#print(limit)    newsort<-sorted[1:limit,]
#return(newsort)    #print(newsort)
return(as.data.frame(newsort[,-(l+1)]))
#print(newsort) }
```

Grouping similar rules:

The following describes the code required for grouping the rules.

Rmerge = merge(RevDup(G1), RevDup(G2))
MG = group_by_RHS(Rmerge)
For each G ∈ MG {agent = find_agent(G)}
DR = sort_by_4condition(agent)
Return (DR)

RESULTS AND DISCUSSIONS

The hash based algorithm has been executed using R Data mining toolkit. The input dataset considered for evaluation is Groceries dataset. Below are few results which narrate the performance of the algorithm designed.

```
> mainpgm(head(data,n=1000),0.06)
[1] "Rule"
[1] 25
[1] "->"
[1] 56
[1] "Rule"
[1] 25
[1] "->"
[1] 30
[1] "Rule"
[1] 25
[1] "->"
[1] 104
[1] "Rule"
[1] 25
[1] "->"
[1] 56
[1] 104
```

Fig. 2: Rules with support=0.06

Fig. 2 shows few set of rules generated when the support count is 0.06. Out of 74 rules generated only the head, i.e. only the initial few rules were listed here.

```
> mainpgm(head(data,n=1000),0.07)
[1] "Rule"
[1] 25
[1] "->"
[1] 56
[1] "Rule"
[1] 25
[1] "->"
[1] 30
[1] "Rule"
[1] 25
[1] "->"
[1] 104
[1] "Rule"
[1] 25
[1] "->"
[1] 103
[1] "Rule"
[1] 25
[1] "->"
[1] 59
[1] "Rule"
[1] 25
[1] "->"
[1] 56
[1] 104
```

Fig. 3: Rules with support=0.07

Fig. 3 shows few set of rules generated when the support count is 0.07. The total number of rules generated for such a support range is 96.

```
> mainpgm(head(data,n=1000),0.09)
[1] "Rule"
[1] 25
[1] "->"
[1] 56
[1] "Rule"
[1] 25
[1] "->"
[1] 30
[1] "Rule"
[1] 25
[1] "->"
[1] 104
[1] "Rule"
[1] 25
[1] "->"
[1] 103
[1] "Rule"
[1] 25
[1] "->"
[1] 59
[1] "Rule"
[1] 25
[1] "->"
[1] 106
[1] "Rule"
[1] 25
[1] "->"
[1] 163
```

Fig. 4: Rules with support=0.09

Fig. 3 shows few set of rules generated when the support count is 0.07. The total numbers of rules generated are 109.

As the rules generated were more in number, we use various rule interestingness measures to filter and fine tune the rules as rules may be redundant in nature.

Finding the Best N Rules:

To find the best rule various quality measures are used. The following explains the various Rule Interesting measures like lift, leverage, J-measure, Chi-Square, Strength etc used in this paper.

Strength = N_{both} / $N_{total.}$

Lift (L-> R) = count (L U R) / count(L) * support(R)

Leverage (L-> R) = support (L U R) - support (L) * Support (R)

J-measure is used to measure the information content of a rule. Given a rule of the form, if Y=y then X=x, the information content of the rule measured in bits of information, is denoted by J(X; Y=y), called the J-measure (or Cross Entropy) for the rule.

Let, N_{left} be instances matching left

N_{right} be instances matching right

N_{both} be instances matching both left and right

N_{total} be the total number of instances

J-measure is defined by,

$J(X: Y=y) = p(x|y) . \log_2 (p(x|y) / p(x)) + (1- p(x|y)). \log_2 (1- p(x|y) / 1-p(x))$

p(x) is the probability that RHS of the rule will be satisfied if we have no other information.

p(x|y) is the probability that RHS of the rule will be satisfied if we know that the LHS is satisfied.

$p(x) = N_{right} / N_{total}$

$p(y) = N_{left} / N_{total}$

$p(x|y) = N_{both} / N_{left}$

$J_{max} = p(y). Max\{ p(x|y) . \log_2(1/ p(x)) , (1- p(x|y)) . \log_2 (1/(1-p(x))) \}$

Therefore if a given value is known to have a J value of say 0.352 bits and J_{max} of say 0.352, there is no benefit to be gained by adding further terms to the LHS as far as information content is concerned.

The next measure is *Chi-Squared statistic* can be calculated in the following manner. Suppose the set of items is $I=\{I_1,I_2,I_3 \ldots\ldots I_m\}$.Because we are interested in both the occurrence and non-occurrence of an item, a transaction t_j can be viewed as ,$t_j \in \{ I_1 ,\bar{I}_1 \} * \{ I_2 ,\bar{I}_2 \} * \ldots\ldots\ldots * \{ I_m ,\bar{I}_m \}$. Given any possible itemset X, it also is viewed as a subset of the Cartesian product. The chi squared statistic is then calculated for X as

$$X^2 = \sum_{x \in I} (O(x) - E[X])^2 / E[X].$$

Here O(X) is the count of the number of transactions that contain the items in X. For one item I_i , the expected value is $E[I_i]=O(I_i)$, the count of the number of transactions that contain I_i.

$E[\bar{I}_i] = n\text{-}O(I_i)$,

where n represents the number of transactions. The expected value,

$$E[X] = n^* \prod_{i=1}^{m} E[I_i] / n$$

If all values were independent then the chi squared statistic should be0. And a chi squared value < 3.84 indicates that we should not reject the independent assumption. This is done with 95% confidence. Thus, even though there appears to be a negative correlation between A and B, it is not statistically significant.

Fig. 5 shows the set of filtered and refined rules in which the maximum number of the rules allowed is taken as 150 but based on other constraints specified, the algorithm has generated only 31 best rules. We have chosen various constraints, ie the min leverage > -1.0, min coverage >0.1, min support >0.09, mini lift >1.0 and mini strength >0.6.

```
Data file: asha.txt
Sun Jan 12 09:08:02 2014
------------------------------
Search for rules
Filter out rules that are unsound.
Maximum number of rules allowed = 150
Minimum leverage = -1.0
Minimum coverage = 0.1
Minimum support = 0.09
Minimum lift = 1.0
Minimum strength = 0.6

Only 31 rules satisfy the specified constraints.

104 -> 163
[Coverage=0.263 ; Support=0.263 ; Strength=1.000; Lift=3.80; Leverage=0.1939 ;
25 -> 12
[Coverage=0.632 ; Support=0.500 ; Strength=0.792; Lift=1.58; Leverage=0.1842 ;
58 -> 105
[Coverage=0.237 ; Support=0.237 ; Strength=1.000; Lift=4.22; Leverage=0.1807 ;
24 -> 36
[Coverage=0.237 ; Support=0.237 ; Strength=1.000; Lift=4.22; Leverage=0.1807 ;
59 -> 103
[Coverage=0.158 ; Support=0.158 ; Strength=1.000; Lift=6.33; Leverage=0.1330 ;
21 -> 103
[Coverage=0.158 ; Support=0.158 ; Strength=1.000; Lift=6.33; Leverage=0.1330 ;
56 -> 59
[Coverage=0.158 ; Support=0.158 ; Strength=1.000; Lift=6.33; Leverage=0.1330 ;
```

Fig. 5: Filtered Rules

Table I: Rules after removal of Redundancy

Total no of Rules when Support = 0.09	
Before Filtering	After Filtering
109	31

Table I shows the total number of rules generated when the support=0.09. Comparing Fig. 4 and Fig. 5, we can observe that the number of rules has been reduced from 109 to 31 after redundancy removal and refinement by the various rule interestingness measures.

Fig. 6 displays the total number of rules generated after rule refinement, filtration and grouping. That is when the support count goes on increasing the number of rules generated are also more in number. But after some range the number of rules generated can also be nil. In this graph structure, for the support value > 4/6/8 no rules are generated. Graphs are plotted using various commands like,

- plot (rules)
- plot (rules, method="grouped")
- plot (rules, method="graph")

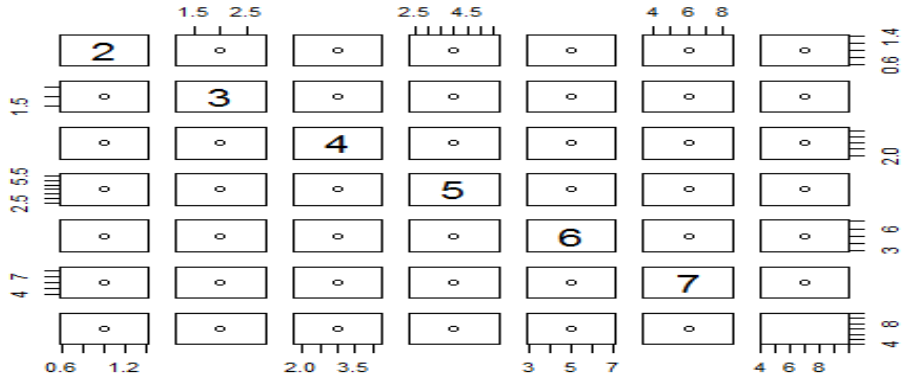

Fig. 6: Graph representing Rules

Conclusion:

The proposed algorithm suggests an improvement to the Apriori. This method enhances the efficiency of algorithm by generating frequent itemset by the number of candidate itemsets generated through hash based approach. Proposed algorithm generates more number of frequent itemsets and rules in lesser time. Through filtering the infrequent itemsets and by retaining the frequent ones strong and best rules are retrieved. Final results confirm that, with frequent itemset generated within less time and create relevant rules only after grouping the rules.

REFERENCES

Farah Hanna AL-Zawaidah, YosefHasanJbara, 2011."An Improved Algorithm for Mining Association Rules in Large Databases," World of Computer Science and Information Technology Journal (WCSIT), pp: 487-499.

Huan Wu, Zhigang Lu, Lin Pan, Rongsheng Xu, Wenbao Jiang, 2009. "An Improved Apriori-based Algorithm for Association Rules Mining," fskd, 2: 51-55. Sixth International Conference On Fuzzy Systems And Knowledge Discovery

Hranchotchuang, W., T. Rakthanmanon, K. Waiyamai, 2009."Using Hash Based Apriori Algorithm to Reduce Candidate Itemset-2 for Mining Association Rule," IJCSI International Journal of Computer Science Issues, 6: 35-4.

Bouker, R., S.B. Saidi, M.E. Yahia, Nguifo, 2013."Ranking and selecting association rules based on dominance relationship," Proceedings of the International MultiConference of Engineers And Computer Scientists 2013 Vol.1, IMECS 2013, March 13-15. Hong Kong .

Shinji Funjiwara, Jeffrey D. Ullman, Rajeev Motwani, 2005. "Dynamic Miss Counting Algorithm Finding Implication and Similarity Rules with Confidence Pruning,"In the Proceedings of the 1st National Conference on Computing And Information Technology, pp: 24-25.

Jia Ronga, Huy Quan Vua, Rob Lawb, GangLia, 2012. "A behavioral analysis of web sharers and browsers in Hong Kong using targeted association rule mining," Tourism Management, 33: 731-740.

Phaichayon, Nittaya Kerdprasop, and Kittisak Kerdprasop, 2013."Dissimilar Rule Mining and Ranking Technique for Associative Classification,"InProceedings of the International MultiConference of Engineers And Computer Scientists, 1: 15, Hong Kong.

Yang Xiang Philip, PayneKun Huang, 2012."Transactional Database Transformation and Its Application in Prioritizing Human Disease Genes by Yang Xiang, Philip R.O. Payne, and Kun Huang," IEEE/ACM Transactions On Computational Biology And Bioinformatics, 9(1).

Sushma Bhasgi, Dr. Parag Kulkarni, 2012. "Multilevel Association rule based data mining," International Journal of Advances in Computing and Information ResearchesISSN:2277-4068, 1(2).

Jayalakshmi, V. Vidhya, Krishnamurthy, Kannan, 2012. "Frequent Itemset Generation using Double Hashing Technique," International Conference On Modelling Optimization And Computing, 38: 1467-1478.

Ramesh kumar, K., M. Sambath, S. Ravi, 2013. "Relevant association rule mining from medical dataset using new irrelevant rule elimination technique, " Information Communication and Embedded Systems (ICICES), 2013 International Conference on.

Anand, H.S., S.S. Vinod Chandra, 2013. "Applying Correlation Threshold on Apriori Algorithm, "IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology.

Xindong Wu, Xing Quan Zhu, Gong-Qing Wu, and Wei Ding, 2014. "Data Mining with Big Data" IEEE Transactions On Knowledge And Data Engineering, 26: 1.

Kannika NiraiVaani, M. Ramaraj, 2013. "An integrated approach to derive effective rules from association

rule mining using genetic algorithm," Pattern Recognition, Informatics And Mobile Engineering (Prime), 2013 International Conference on.

Asha, P., Dr.T. Jebarajan, 2013. "Analyzing the Sequential and Parallel ARM Algorithms and its Impact in Grid Computing Environments", in the International Journal of Advanced Computing, Recent Science Publications, pp: 1109-1114, 36(1): 20151-0845.

Zaki, M.J., 1999. " Parallel and distributed association mining: A survey," IEEE Concurrency, Special Issue on Parallel Mechanisms for Data Mining, 7(4): 14-25.

Avi Silberschatz and Alexander Tuzhilin, 1996. "What makes patterns interesting in knowledge Discovering Systems," IEEE Trans. on Knowledge and Data Engineering, 8(6).

Agrawal, R. *et al*., 1996. "Fast Discovery of Association Rules," Advances in Knowledge Discovery and Data Mining, (U. Fayyad *et al*.,eds., AAAI Press, Menlo Park, Calif., pp: 307-328.

Shih-Sheng Chen, Tony Cheng-Kui Huang, Zhe-Min Lin, 2011. "New and efficient knowledge discovery of partial periodic patterns with multiple minimum supports", The Journal of Systems and Software, 84: 1638-1651.

Agrawal, R., T. Mielinski, A. Swami, 1993. "Mining association rule between sets of items in large databases", in: proceedings of the ACM SIGMOD international Conference on Management Of Data. pp: 207-216.