Journal of Web Engineering, Vol. 14, No.3&4 (2015) 286-300
© Rinton Press

COMPARISON OF HYBRID APPROACHES WITH TRADITIONAL ALGORITHMS FOR IMPROVING SCALABILITY OF FREQUENT PATTERN MINING

SURIYA SUNDARAMOORTHY
Department of Computer Science & Engineering
Velammal College of Engineering and Technology, Madurai, Anna University
suriyas84@gmail.com , ssu@vcet.ac.in

S.P. SHANTHARAJAH
Department of Computer Applications
Sona College of Technology, Salem, Anna University
spshantharaj@gmail.com

SURESH KANNAN SUNDARAMOORTHY
Department of Computer Science & Engineering
Velammal College of Engineering and Technology, Madurai, Anna University
sureshsureshkannan@gmail.com

Received April 11, 2014
Revised January 18, 2015

Frequent pattern mining always occupies its space in research activities in spite of various emerging research ideas. This paper focuses on performance of four competitive algorithms namely hybrid k-Direct Count and Intersect algorithm with apriori algorithm, hybrid k-Direct Count and Intersect algorithm with transaction mapping algorithm, Modified Ant Colony algorithm and improved Ant Colony algorithm over frequent pattern mining against traditional algorithms. We focus on a set of well defined parameters such as database layout, scanning of input databases, memory requirement, input / output cost, input / output overhead, computational cost, execution speed, scalability, support for parallelization, complexity of the algorithm to do this comparison. Experimental results support for this effective comparison with help of visualization through graphs. We used ASP.net as front end tool and SQL server 2005 as back end tool for implementing our proposed approaches. As a result of this study, there is good improvement in effectiveness of pattern mining in all aspects while using the improved Ant Colony algorithm against other hybrid approaches and traditional algorithms.

Key words Ant Colony algorithm, Frequent Pattern Mining, k-Direct Count and Intersect algorithm, Selective Item Replication algorithm, Transaction Mapping algorithm.
Communicated by: B. White & E-P. Lim

1 Introduction

Data mining has attracted significant amount of researchers in the past decade. The need of today's generation is data mining tools that assist in extraction of potential information from huge volumes of data. Data mining is a famous knowledge discovery process that involves data analysis through efficient algorithms in order to produce a particular enumeration of patterns over the data. Today size of databases ranges in uncountable manner so that within this mass of data lays hidden data of strategic importance from business point of view. Many organizations worldwide started using data mining techniques to increase their revenues. Data mining adopts a set of data analysis tools to discover patterns and their relationships. Data mining is data analyzing process which ends up with extraction of required data by summarizing its correlation or patterns in different dimensions. The demand for data mining has been increased mainly due to growth of heavy storage of tremendous data.

Data Mining is an effective research tool for extracting potentially useful information from huge repository of data storage. It is one of the important steps of the famous process known as Knowledge Discovery in Databases (KDD). A set of tools that supports all the activities of the KDD are IBM Intelligent Miner, WEKA, DBMiner, and so on. The real motivation of searching frequents items raised because of supermarket transaction data analysis. The main goal was to analyze the behavior of customers in terms of purchased products. Frequent pattern mining discovers patterns among the target dataset and their relationship with one another which is used for valid predictions. There are four types of frequent pattern mining (as shown in figure 1)namely frequent item set mining, frequent sequence mining, frequent tree mining and frequent graph mining. This paper completely focuses on frequent item set mining.

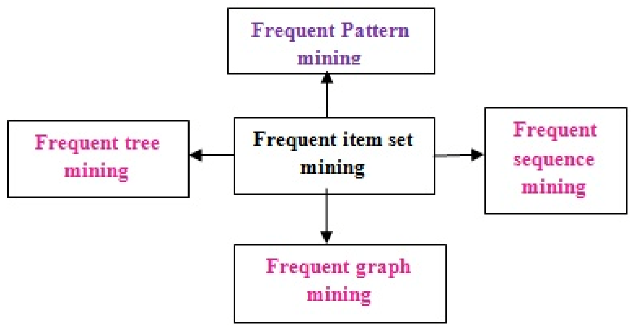


Figure 1: Classification of frequent pattern mining

The goal of frequent sequence mining is finding frequent subsequences from a collection of sequences. The objective of frequent tree mining is finding frequent subtrees from a set of trees. The main task of frequent graph mining is finding frequent subgraphs from a collection of graphs.

Definition: Let DT be transaction database over a set of items IS, with a minimum support of ζ. Frequent item set mining involves estimation of Ffreq(DT, ζ).

$$\text{Ffreq}(DT, \zeta) = \{x \in IS \,/\, \text{Support}(x,DT) \geq \zeta\}$$

The monotonicity property is based on support value. Consider two sets say IX and IY,

$$\text{if } IX \quad IY \quad \text{Support}(IY) \leq \text{Support}(IX).$$

It states that when a set is found to be infrequent then all its supersets are also infrequent. Similarly when a set is found to be frequent then all its supersets are also frequent. The monotonicity property for a set of frequent items is called as downward closure property. The monotonicity property for a set of infrequent items is called as upward closed property.

Motivation: A lot research endeavors are carried out in order to enhance scalability in spite of above issues. Abdullah et al (2010) has proposed a scalable model and addressed the issues related to high computational cost, complicated measurements. Liu & Chang (2011) has employed the strategy of incremental construction of a candidate item-set and Apriori property to reduce the searching space and to improve scalability of the frequent item-set mining algorithm. Wang et al (2012) has addressed issues like I/O cost while mining over uncertain databases. In this framework, it is important to accentuate that the above mentioned issues are very hard and challenging. Borgelt (2012) has discussed about how the search space is structured to avoid the redundant searching and various algorithms for effective frequent item-set mining. Mohanty et al (2013) has addressed the role of association rule mining in breast cancer detection.

2 Traditional Algorithms

2.1 FP Growth Algorithm

Rajendran & Madheswaran (2010) has proposed FP growth algorithm for classifying brain tumor in CT scan brain images. The frequent pattern from CT scan images are mined with the help of FP growth algorithm. The experimental results over prediagnosed database of brain images proved that the proposed methodology exhibits 97% sensitivity and 95% accuracy. Lin et al (2011) has proposed two improvements over basic FP growth algorithm namely implementing an address table to reduce the complexity of FP tree construction and adopting a new structure called FP tree+ to replace conditional FP trees. The performance of the proposed algorithm was compared with basic FP tree algorithm and the experimental results witnessed that memory requirements were lowered and performance was suitable for high performance applications. Rao & Gupta (2012) has proved the efficiency of FP growth algorithm for association rule mining. The proposed algorithm when compared with Apriori algorithm reduces number of candidate generation and database scans. The proposed algorithm makes use of a compact data structure to eliminate repeated database scans. The proposed algorithm was tested in WEKA environment. Bernecker et al (2012) has addressed the issue of inapplicability of traditional algorithms due to the presence of uncertainty of items in the transactions. A probabilistic version of FP growth algorithm is proposed to mine all probabilistic frequent itemsets in uncertain transaction databases without candidate generation. Pei et al (2013) has proposed a solution for sequential pattern mining using FP growth algorithm. The proposed method is popularly known as prefix projected sequential pattern mining. The attractive feature of proposed method is that it reduces the efforts of candidate subsequence generation and the size of the projected databases. These features lead to efficient processing. The pseudo code of basic FP growth algorithm consists of two parts namely construction of FP tree and mining with FP tree (as shown in figure 2 and figure 3 respectively).

FP Growth algorithm takes FP tree as input to start with mining of frequent itemsets. This algorithm mines the FP tree by constructing conditional pattern bases.

```
Algorithm FP_Tree_Construction
// Problem Description: Construction of FP Tree.
//Inputs: Transaction database (TD), Minimum Support ($\hat{\sigma}$).
//Output: FP Tree.
{
Scan the transaction database.
Collect the set of frequent items with their support count.
Sort the items based on descending order of support count.
Create the root of the FP Tree and label it with {}.
For each transaction
{
Reorder the set of items in the transactions based on this priority.
Insert the items in the same order at each level with the count 1.
If any item gets repeated in the same level
{
Increment the count value by 1.
}
}
}
```

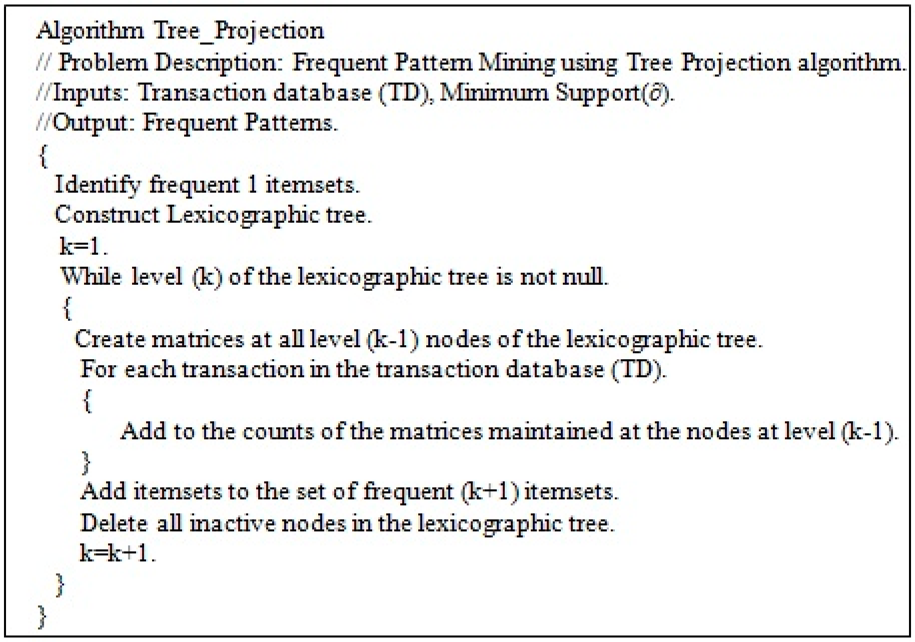
Figure 2 Pseudo code for FP Tree Construction

```
Algorithm FP_Growth
// Problem Description: Frequent Pattern mining using FP Growth
algorithm.
//Inputs: FP Tree, Transaction database (TD), Minimum Support ($\hat{\sigma}$).
//Output: Frequent Patterns.
{
 If FP tree contains single path then
   {
      For each combination of the nodes in the path.
         {
          Generate patterns from the combination that satisfy the minimum
support.
         }
   }
Else
  {
   For each item in the header of the FP Tree.
   {
      Generate patterns from those combinations that satisfy the minimum
support.
     Construct conditional pattern base.
     Construct conditional FP Tree.
    }
  }
}
```

Figure 3 Pseudo code for FP Growth algorithm

2.2 Tree Projection Algorithm

Agarwal et al (2001) has proposed a lexicographic tree based tree projection algorithm in order to mine frequent patterns. The algorithm has a well-structured data access pattern which provides data locality and reuse of data for multiple levels of the cache. Guralnik et al (2001) has addressed the issue regarding discovery of sequential patterns. Tree projection algorithm is used to solve this issue by its effective memory utilization. Han et al (2007) has focussed on current status and future directions of frequent pattern mining. The importance of tree projection algorithm is highlighted like reduction in search time. Li et al (2014) has proposed tree projection algorithm in order to reduce unnecessary storage space and scanning time. This proposed algorithm utilizes Apriori property to delete non frequent items and record the projected position for mining local frequency items and mine each recursively. Han et al (2014) has proposed a novel structure called prefix frequent items graph in order to determine frequent sequential patterns effectively. The proposed technique eliminates redundant data scanning to effectively discover new patterns. The pseudo code of tree projection algorithm is shown in figure 4 respectively. The lexicographic tree is defined in the following way: every vertex exists in the tree corresponding to each frequent itemset and the root of the tree corresponds to the null itemset. The algorithm prunes all those nodes which have been determined to be unnecessary for further counting and tree generation based on the minimum support. Frequent patterns identified through tree projection algorithm along with their support count.



```
Algorithm Tree_Projection
// Problem Description: Frequent Pattern Mining using Tree Projection algorithm.
//Inputs: Transaction database (TD), Minimum Support(σ).
//Output: Frequent Patterns.
{
  Identify frequent 1 itemsets.
  Construct Lexicographic tree.
   k=1.
   While level (k) of the lexicographic tree is not null.
   {
     Create matrices at all level (k-1) nodes of the lexicographic tree.
      For each transaction in the transaction database (TD).
      {
           Add to the counts of the matrices maintained at the nodes at level (k-1).
      }
      Add itemsets to the set of frequent (k+1) itemsets.
      Delete all inactive nodes in the lexicographic tree.
      k=k+1.
   }
}
```

Figure 4 Pseudo code of Tree Projection Algorithm

3 Hybrid Approaches

3.1 Hybrid Direct Count and Intersect algorithm with Apriori algorithm (Hybrid DCIAA)

The motivation behind this hybrid approach is locality requirement for exploiting memory hierarchies to achieve high performance and scalability while mining huge repository of datasets. Due to the huge increase in the number and dimension of available databases, efficient solutions for counting frequent sets are introduced. ParDCI is a parallel version of DCI (Direct Count & Intersect). ParDCI, a distributed and multithreaded algorithm for counting the occurrences of frequent sets within transactional databases. It stores item sets in horizontal format. Non-frequent items are removed using pruning technique. Later the item sets are stored in main memory. Now data is represented in vertical format and mining of frequent items are performed over counting inference strategy. Usage of prefix sharing structure among the lexicographically ordered item sets of the collection. This hybrid algorithm enhances previous proposals by exploiting the highly optimized counting and intersection techniques of DCI, and by relying on a mulli-level parallelization approach as shown in figure 5.

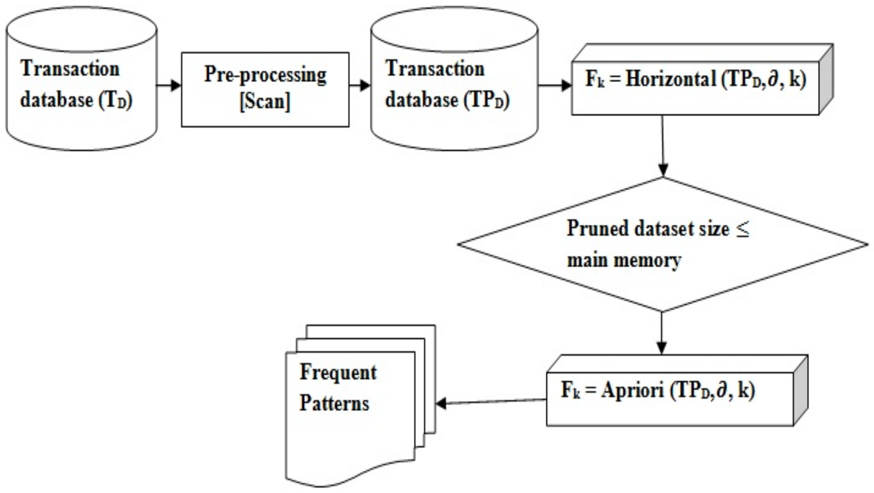
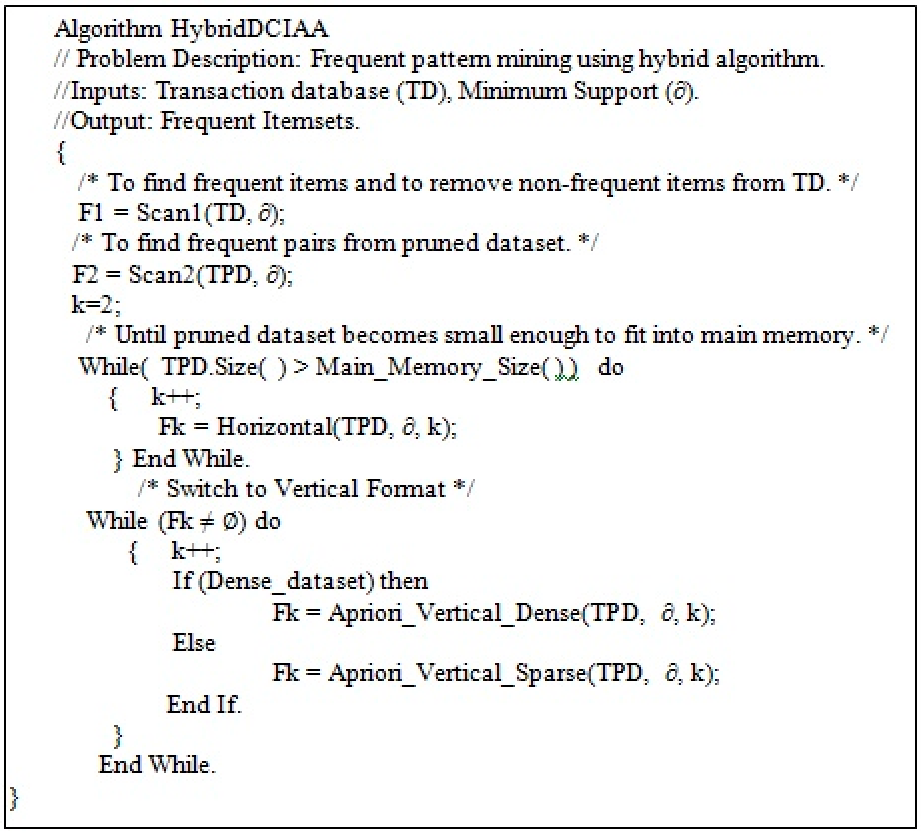

Transaction database (T_D)
Pre-processing [Scan]
Transaction database (TP_D)
F_k = Horizontal (TP_D, ∂, k)
Pruned dataset size ≤ main memory
F_k = Apriori (TP_D, ∂, k)
Frequent Patterns

Figure 5 Working of Hybrid Direct Count and Intersect Algorithm with Apriori Algorithm

DCI uses an Apriori -like technique to generate candidate key. The two phases of this algorithm are Counting based phase - ParDCI directly counts the occurrences of items within all the transactions and Intersection based phase - Transforms the dataset from horizontal into vertical. It exploits an effective counting-based method during the first iterations, and a very fast intersection-based method during the last ones. It relies on optimized data structures for storing and accessing candidate item sets with high locality. The database is partitioned among the processing nodes, and a simple but effective database pruning technique is exploited which allows to trim the transaction database partitions as execution progresses. Later apriori algorithm is hybridized with Direct Count and Intersect algorithm after pruning database comes to processing. This is done to reduce the number of candidate generation both in case of dense and sparse datasets. This algorithm adopts several heuristic strategies to adapt its behaviour to the features of datasets processed. It exploits a horizontal database by building a vertical-layout in-core database. It is an effective candidate distribution is adopted at both the inter and intra node levels. This algorithm provides very good results obtained for sparse and dense datasets. The applications of this algorithm are exhibits excellent scale-up and speedup, market baskets analysis tool and fast sequential algorithm. Pre-processing is done to remove non-frequent items from transaction database (TD). The output of pre-processing step will be pruned dataset (TPD) from which frequent patterns are identified using the hybrid algorithm. Until the memory size of pruned data set becomes fit into main memory, the horizontal layout of pruned dataset is used for processing of Direct Count and Intersect algorithm. When it fits into main memory, vertical representation of the dataset is followed. Now Direct Count and Intersect algorithm calls Apriori algorithm to mine frequent patterns from bit vector representation. The pseudo code of hybrid Direct Count and Intersect algorithm with Apriori algorithm is elaborated in the figure 6. This approach is one of my research works published in an IEEE conference by Suriya & Shantharajah [2013].



```
Algorithm HybridDCIAA
// Problem Description: Frequent pattern mining using hybrid algorithm.
//Inputs: Transaction database (TD), Minimum Support (∂).
//Output: Frequent Itemsets.
{
   /* To find frequent items and to remove non-frequent items from TD. */
   F1 = Scan1(TD, ∂);
   /* To find frequent pairs from pruned dataset. */
   F2 = Scan2(TPD, ∂);
   k=2;
   /* Until pruned dataset becomes small enough to fit into main memory. */
   While( TPD.Size( ) > Main_Memory_Size( ))  do
      {    k++;
           Fk = Horizontal(TPD, ∂, k);
      } End While.
        /* Switch to Vertical Format */
   While (Fk ≠ Ø) do
      {    k++;
           If (Dense_dataset) then
                   Fk = Apriori_Vertical_Dense(TPD,  ∂, k);
           Else
                   Fk = Apriori_Vertical_Sparse(TPD,  ∂, k);
           End If.
      }
   End While.
}
```

Figure 6 Pseudo code of Hybrid Direct Count and Intersect Algorithm with Apriori Algorithm

3.2 Hybrid Direct Count and Intersect algorithm with Transaction mapping algorithm (Hybrid DCITMA)

The motivation behind this hybrid approach is to reduce computational cost and I/O overhead. Transaction Mapping algorithm has the ability to reduce the computational costs involved and Direct Count and Intersect algorithm has the ability to reduce I/O overhead. Therefore these two algorithms are hybridized to achieve the research goal. Transaction mapping algorithm adopts vertical database representation. Direct Count and Intersect algorithm adopt adopts vertical database representation when the size of pruned dataset fits main memory. Hence Transaction Mapping algorithm is hybridized with Direct Count and Intersect algorithm when the size of pruned dataset is enough to fit inside the main memory. So DCI algorithm invokes Transaction mapping algorithm. The hybrid algorithm outperforms the previous approaches by exploiting a counting phase and intersection phase of DCI algorithm. The intersection phase involves transaction tree construction and interval list intersection techniques of Transaction Mapping algorithm. The architecture of hybrid approach is shown in Figure7. During the intersection phase of DCI algorithm, the size of pruned dataset is checked. When it is capable of getting fit inside the main memory, there is switch over from horizontal representation of dataset to vertical representation of dataset. This helps in invoking Transaction mapping algorithm. Now Transaction mapping algorithm begins its working with the pruned dataset which does not contain non frequent items. This helps in increasing the execution speed and scalability. Thus the computational cost is lowered and I/O overhead is also reduced. The pseudo code of the hybrid Direct Count and Intersect algorithm with Transaction Mapping algorithm is shown in Figure 8.

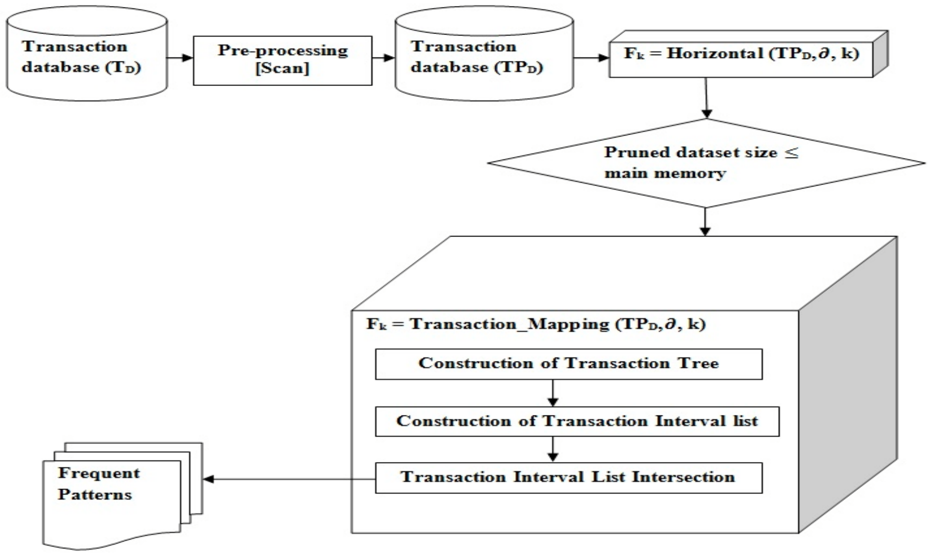
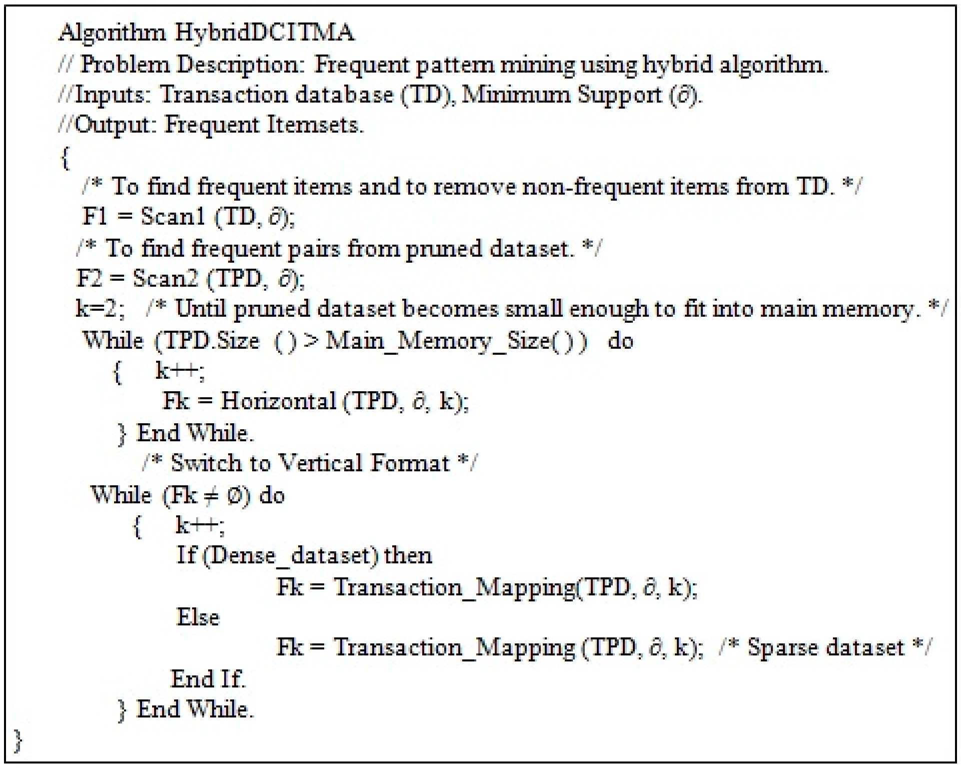

Transaction database (T_D)
Pre-processing [Scan]
Transaction database (TP_D)
F_k = Horizontal (TP_D, ∂, k)
Pruned dataset size ≤ main memory
F_k = Transaction_Mapping (TP_D, ∂, k)
Construction of Transaction Tree
Construction of Transaction Interval list
Transaction Interval List Intersection
Frequent Patterns

Figure 7 Working of Hybrid Direct Count and Intersect Algorithm with Transaction Mapping Algorithm (Hybrid DCITMA)



```
Algorithm HybridDCITMA
// Problem Description: Frequent pattern mining using hybrid algorithm.
//Inputs: Transaction database (TD), Minimum Support ($\partial$).
//Output: Frequent Itemsets.
{
  /* To find frequent items and to remove non-frequent items from TD. */
  F1 = Scan1 (TD, $\partial$);
 /* To find frequent pairs from pruned dataset. */
 F2 = Scan2 (TPD, $\partial$);
 k=2;  /* Until pruned dataset becomes small enough to fit into main memory. */
  While (TPD.Size ( ) > Main_Memory_Size( ) )  do
      {   k++;
          Fk = Horizontal (TPD, $\partial$, k);
      } End While.
         /* Switch to Vertical Format */
   While (Fk $\neq$ Ø) do
        {   k++;
            If (Dense_dataset) then
                        Fk = Transaction_Mapping(TPD, $\partial$, k);
            Else
                        Fk = Transaction_Mapping (TPD, $\partial$, k);  /* Sparse dataset */
           End If.
      } End While.
}
```

Figure 8 Pseudo code of Hybrid Direct Count and Intersect Algorithm with Transaction Mapping Algorithm (Hybrid DCIMA)

3.3 *Modified Ant Colony algorithm (MACA)*

This proposed modified ant colony is adopted to solve the issue of identifying frequent patterns among the datasets. Let n represent the total number of items in the database. So n-stage graph is constructed. Each stage consists of n nodes to represent n items and one more node to represent the end. So totally (n+1) nodes at each stage. Paths are mapped based on the input transactions. Minimum support value is set as the evaporation rate (ρ). Ant agents are now used to identify the frequent patterns from this multistage graph. At the initial stage all the input transactions are mapped to the n-stage graph. The end of every transaction is represented by the path from the last item of the transaction to (n+1)th node in order to indicate the end of the transaction. Ant agent dynamically update their memory based on the pheromone rate. The value for evaporation rate ranges from 0 to1 respectively. So the minimum support value is passed in percentage from 0.1 to 1 accordingly. The working architecture of proposed model for frequent pattern mining is shown in figure 9 followed by the pseudocode of the proposed approach in figure 10. This approach is one of my research works Suriya & Shantharajah [2013].

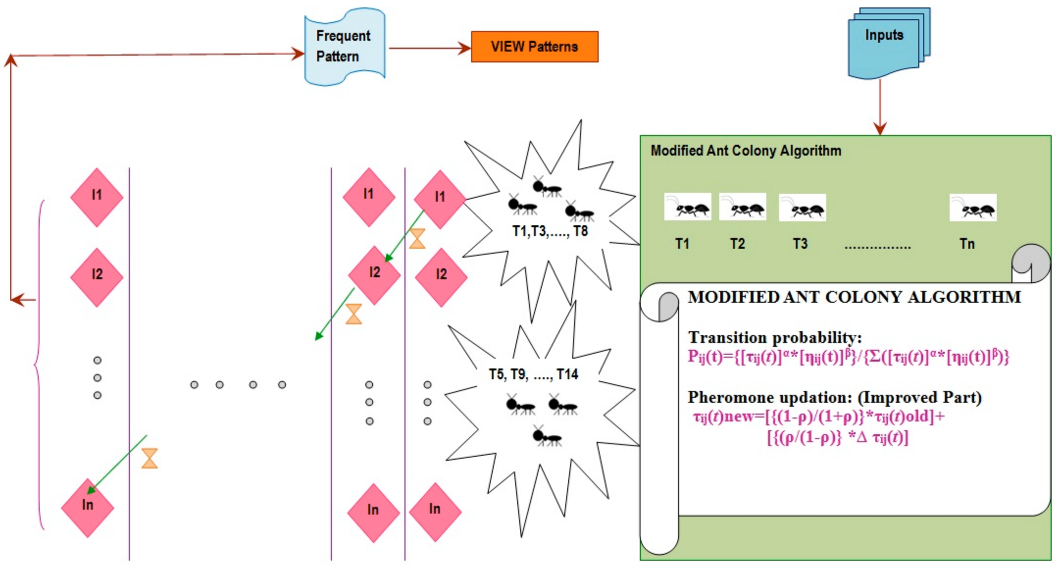
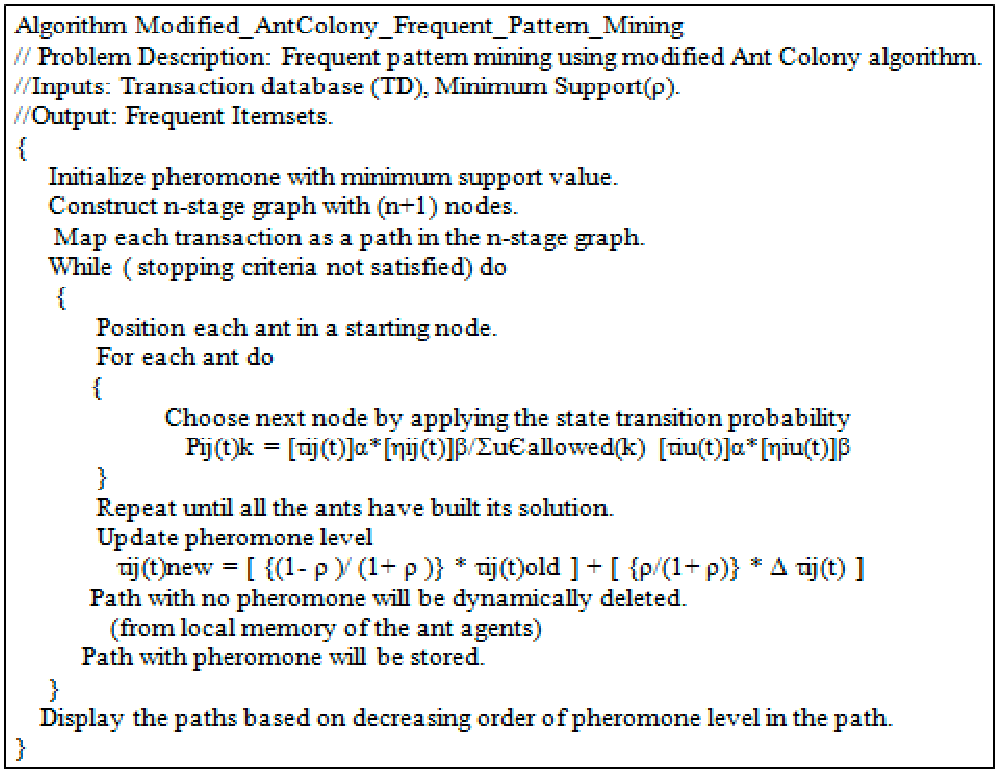

Frequent Pattern
VIEW Patterns
Inputs
Modified Ant Colony Algorithm
T1
T2
T3
....
Tn
I1
I2
In
T1,T3,...., T8
T5, T9, ..., T14
MODIFIED ANT COLONY ALGORITHM
Transition probability:
$P_{ij}(t)=\{[\tau_{ij}(t)]^{\alpha}*[\eta_{ij}(t)]^{\beta}\}/\{\Sigma([\tau_{ij}(t)]^{\alpha}*[\eta_{ij}(t)]^{\beta})\}$
Pheromone updation: (Improved Part)
$\tau_{ij}(t)new=[\{(1-\rho)/(1+\rho)\}*\tau_{ij}(t)old]+[\{(\rho/(1-\rho)\} *\Delta\ \tau_{ij}(t)]$

Figure 9 Working of modified Ant Colony algorithm for Frequent Pattern Mining

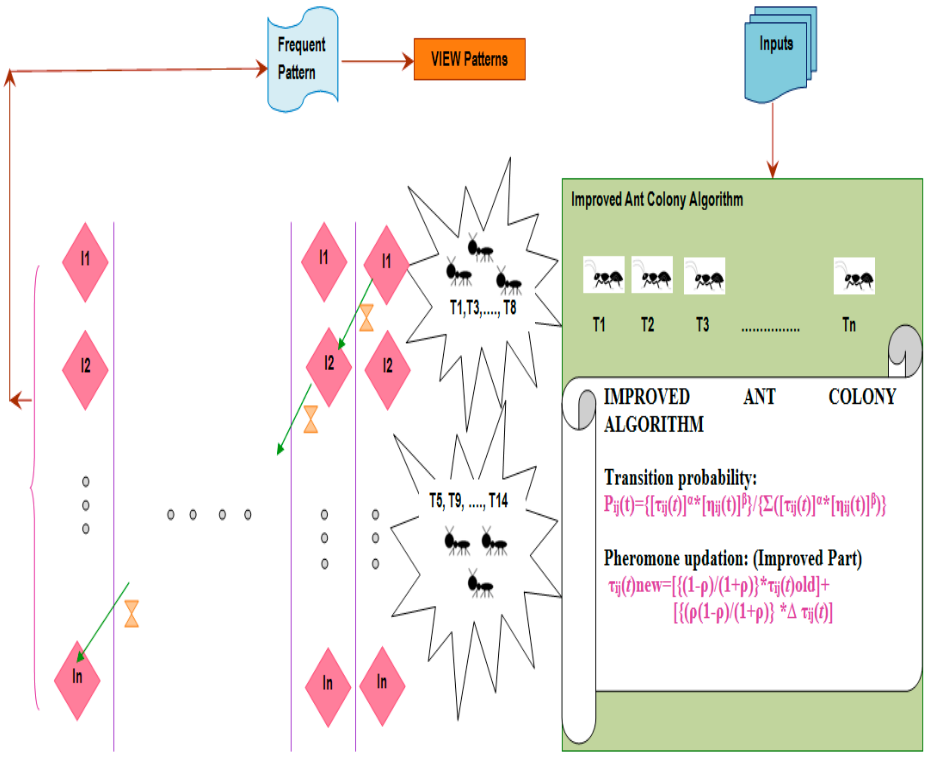


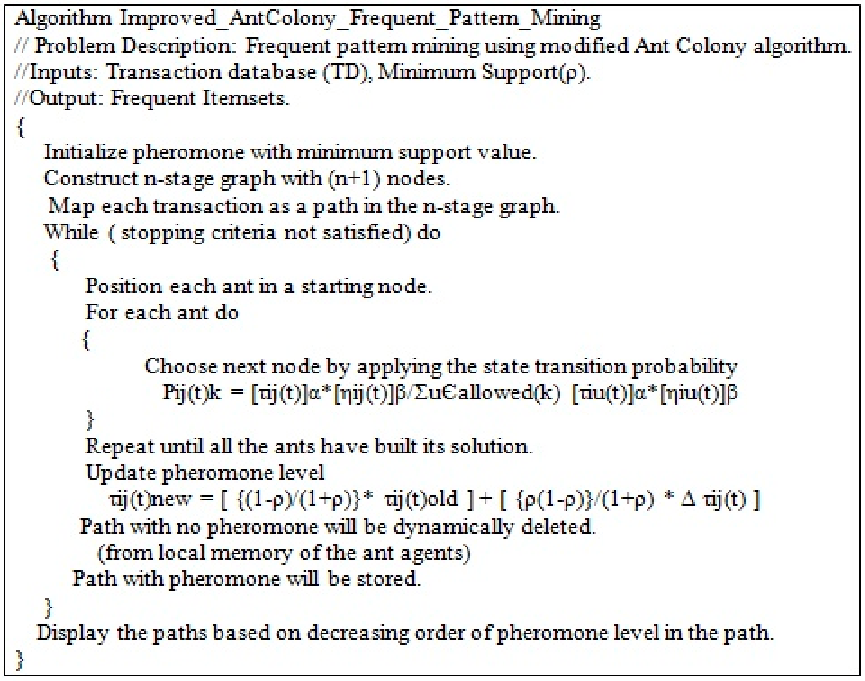
```
Algorithm Modified_AntColony_Frequent_Pattern_Mining
// Problem Description: Frequent pattern mining using modified Ant Colony algorithm.
//Inputs: Transaction database (TD), Minimum Support(ρ).
//Output: Frequent Itemsets.
{
   Initialize pheromone with minimum support value.
   Construct n-stage graph with (n+1) nodes.
   Map each transaction as a path in the n-stage graph.
   While ( stopping criteria not satisfied) do
    {
       Position each ant in a starting node.
       For each ant do
       {
            Choose next node by applying the state transition probability
              Pij(t)k = [τij(t)]α*[ηij(t)]β/ΣuЄallowed(k) [τiu(t)]α*[ηiu(t)]β
       }
       Repeat until all the ants have built its solution.
       Update pheromone level
         τij(t)new = [ {(1- ρ )/ (1+ ρ )} * τij(t)old ] + [ {ρ/(1+ ρ)} * Δ τij(t) ]
      Path with no pheromone will be dynamically deleted.
        (from local memory of the ant agents)
      Path with pheromone will be stored.
   }
  Display the paths based on decreasing order of pheromone level in the path.
}
```

Figure 10 Pseudo code of Modified Ant Colony Algorithm for Frequent Pattern Mining

3.4 Improved Ant Colony algorithm (IACA)

This proposed improved ant colony is adopted to solve the issue of identifying frequent patterns among the datasets. Let n represent the total number of items in the database. So n-stage graph is constructed. Each stage consists of n nodes to represent n items and one more node to represent the end. So totally (n+1) nodes at each stage. Paths are mapped based on the input transactions. Minimum support value is set as the evaporation rate (ρ). Ant agents are now used to identify the frequent patterns from this multistage graph. At the initial stage all the input transactions are mapped to the n-stage graph. The end of every transaction is represented by the path from the last item of the transaction to (n+1)th node in order to indicate the end of the transaction. Ant agents dynamically update their memory based on the pheromone rate. The value for evaporation rate ranges from 0 to1 respectively. So the minimum support value is passed in percentage from 0.1 to 1 accordingly. The working architecture of proposed model for frequent pattern mining is shown in figure 11 followed by the pseudo code of the proposed approach in figure 12. This approach is one of my research works , Suriya & Shantharajah [2014].


Frequent Pattern
VIEW Patterns
Inputs
Improved Ant Colony Algorithm
T1
T2
T3
....
Tn
IMPROVED ANT COLONY ALGORITHM
Transition probability:
Pij(t)={[τij(t)]^α*[ηij(t)]^β}/{Σ([τij(t)]^α*[ηij(t)]^β)}
Pheromone updation: (Improved Part)
τij(t)new=[{(1-ρ)/(1+ρ)}*τij(t)old]+
[{(ρ(1-ρ)/(1+ρ)} *Δ τij(t)]
T1,T3,...., T8
T5, T9, ..., T14
I1
I2
In
Figure 11 Working of Improved Ant Colony algorithm for Frequent Pattern Mining



```
Algorithm Improved_AntColony_Frequent_Pattern_Mining
// Problem Description: Frequent pattern mining using modified Ant Colony algorithm.
//Inputs: Transaction database (TD), Minimum Support(ρ).
//Output: Frequent Itemsets.
{
   Initialize pheromone with minimum support value.
   Construct n-stage graph with (n+1) nodes.
   Map each transaction as a path in the n-stage graph.
   While ( stopping criteria not satisfied) do
   {
      Position each ant in a starting node.
      For each ant do
      {
          Choose next node by applying the state transition probability
            $P_{ij}(t)^k = [\tau_{ij}(t)]^\alpha * [\eta_{ij}(t)]^\beta / \Sigma_{u \in allowed(k)} [\tau_{iu}(t)]^\alpha * [\eta_{iu}(t)]^\beta$
      }
      Repeat until all the ants have built its solution.
      Update pheromone level
         $\tau_{ij}(t)_{new} = [ \{(1-\rho)/(1+\rho)\} * \tau_{ij}(t)_{old} ] + [ \{\rho(1-\rho)\}/(1+\rho) * \Delta \tau_{ij}(t) ]$
      Path with no pheromone will be dynamically deleted.
        (from local memory of the ant agents)
      Path with pheromone will be stored.
   }
   Display the paths based on decreasing order of pheromone level in the path.
}
```

Figure12 Pseudo code of Improved Ant Colony Algorithm for Frequent Pattern Mining

4 Experimental Results

The experimental setup consists of ASP.net as front end tool and SQL Server 2005 as back end tool. All the above approaches are compared against their memory requirements and execution efficiency.

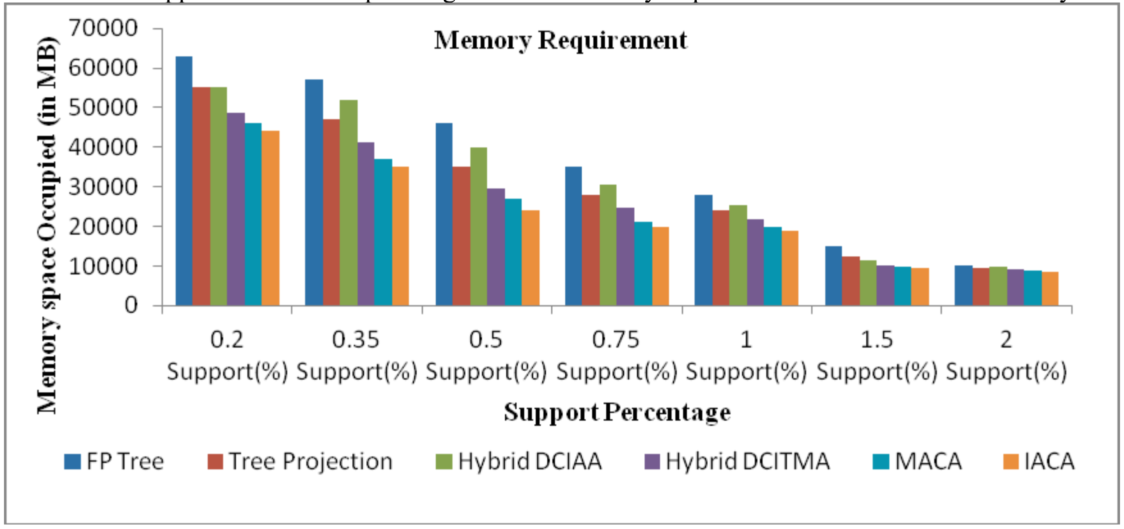


Figure13 Memory Requirement

The memory requirements of all the approaches are compared to identify the effective one. The memory ranges in MB. The memory requirement of hybrid k-DCI and apriori algorithm is less when compared to basic apriori algorithm and traditional DCI algorithm. The memory requirement of hybrid k-DCI and transaction mapping algorithm is less when compared to basic transaction mapping algorithm. The memory requirement of modified ant colony algorithm is less when compared to basic ant colony algorithm. It is improved ant colony algorithm whose memory requirement is less when compared to remaining approaches.

Support Percentage	FP Tree	Tree Projection	Hybrid DCIAA	Hybrid DCITMA	MACA	IACA
0.2	63000	55000	55000	48700	46000	44000
0.35	57000	47000	51820	41280	37000	35000
0.5	46000	35000	39840	29470	27000	24000
0.75	35000	28000	30540	24560	21000	19850
1	28000	24000	25470	21890	20000	18970
1.5	15000	12540	11450	10150	9950	9430
2	10000	9560	9750	9120	8970	8520

Table 1 Memory Requirement

The scalability of all the approaches are tabulated in Table 2 and visualized in figure 14.

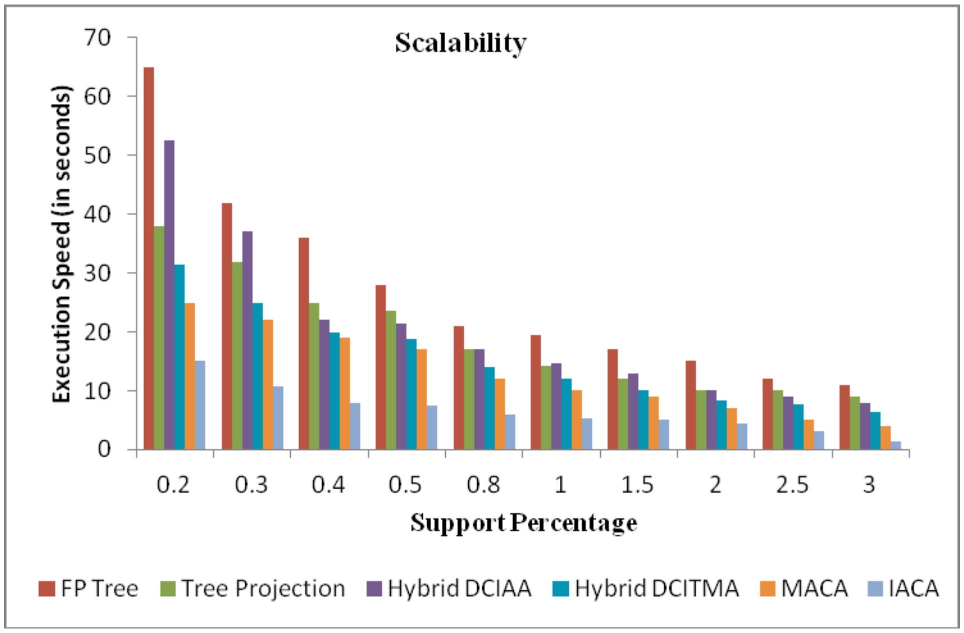


Figure14 Scalability

Support(%)	FP Tree	Tree Projection	Hybrid DCIAA	Hybrid DCITMA	MACA	IACA
0.2	65	38	52.5	31.4	25	15
0.3	42	31.9	37	25	22	10.8
0.4	36	25	22	20	19	8
0.5	28	23.6	21.5	18.8	17	7.5
0.8	21	17	17	14	12	6
1	19.5	14.3	14.6	12	10	5.4
1.5	17	12	13	10.2	9	5
2	15	10.2	10	8.3	7	4.5
2.5	12	10	9	7.6	5	3.1
3	11	9	7.81	6.4	4	1.4

Table 2 Execution Speed (in seconds)

5 Conclusions and Future Work

Thus from the above experimental results, it is very obvious that out of all four hybrid methods and traditional algorithms , improved ant colony algorithm is found to be the suitable one for improving the scalability of frequent pattern mining.

References

1. Abdullah, Z, Herawan, T, & Deris, MM 2010,'Scalable model for mining critical least association rules', Lecture Notes in Computer Science, Information Computing and Applications, Springer, vol. 6377, pp. 509-516.
2. Bernecker, T, Kriegel, H P, Renz, M, Verhein, F, & Züfle, A 2012, 'Probabilistic frequent pattern growth for itemset mining in uncertain databases', Scientific and Statistical Database Management, Springer Berlin Heidelberg , vol. 7338, pp. 38-55.
3. Borgelt, C 2012, 'Frequent item set mining', Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 2, no. 6, pp. 437-456.
4. Lin, K C, Liao, I E, & Chen, Z S 2011, 'An improved frequent pattern growth method for mining association rules', Expert Systems with Applications, vol.38, no. 5, pp. 5154-5161.
5. Liu, Z, & Chang, R 2011, 'Study on efficient algorithm of frequent item-set mining', IEEE International Conference on Electronics and Optoelectronics, vol. 1, pp. 222 – 225.
6. Mohanty, A K, Senapati, M R, & Lenka, S K 2013, 'An improved data mining technique for classification and detection of breast cancer from mammograms', Neural Computing and Applications, Springer, vol. 22, no. 1, pp. 303-310.
7. Pei, J, Han, J, Mortazavi-Asl, B, Pinto, H, Chen, Q, Dayal, U, & Hsu, M C 2013, 'Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth', 29th IEEE International Conference on Data Engineering (ICDE).

8. Rajendran, P, & Madheswaran, M 2010, 'Hybrid medical image classification using association rule mining with decision tree algorithm', Journal of Computing, vol. 2, no.1, pp.127-136.
9. Rao, S, & Gupta, P 2012, 'Implementing Improved Algorithm Over APRIORI Data Mining Association Rule Algorithm', International Journal of computer Science and Technology, vol. 3, no. 1, pp. 489-493.
10. Suriya, S & Shantharajah, S P 2013, 'A Hybrid k-DCI Algorithm and Apriori Algorithm for Mining Frequent Itemsets', IEEE International Conference on Circuit, Power and Computing Technologies (ICCPCT'13), ISBN 978-1-4673-4921-5, pp. 1059-1064.
11. Suriya, S & Shantharajah, S P 2013, 'Selective Marketing for Retailers to promote Stock using improved Ant Colony Algorithm', International Journal of Engineering and Technology, vol.5, no.5, pp. 45-58
12. Suriya, S & Shantharajah, S P 2014, 'An Improved Ant colony Algorithm for Effective Mining of Frequent items', Journal of Web Engineering, vol.13, no.3&4, pp. 263-276
13. Wang, L, Cheung, DL, Cheng, R, Lee, SD, & Yang, XS 2012, 'Efficient mining of frequent item sets on large uncertain databases', IEEE Transactions on Knowledge and Data Engineering, vol. 24, no.12, pp. 2170-2183.