A Comparative Analysis of Various Association Rule Mining Algorithms

TRILOKI PRASAD

M. Phil Research Scholar
Dr. C.V. Raman University Bilaspur (C.G.)

***Abstract*: Association Rule Mining (ARM) is one of the major data mining methods used to mine hidden knowledge from databases that can be used by an organization's decision makers to increase overall profit. However, performing ARM needs frequent passes over the entire database. Clearly, for large database, the role of input/output overhead in scanning the database is very important. In this paper, we provide execution time related to association rule mining algorithms and survey the record of existing association rule mining methods and find out the fastest association rule mining algorithm. Obviously, a single article cannot be a entire review of the entire algorithms, yet we wish that the references cited will cover up the major theoretical issues, guiding the researcher in motivating research information that have yet to be explored.**

General Terms: Algorithms, Support, Confidence, Datasets.

Key Words: Association rule, Data mining, Classification, Java Program, Association Rule Mining, Large Dataset, Apriori, Eclate, F-P Growth.

I. INTRODUCTION

Data mining techniques and tools are used to extract useful information from large databases. Association Rule(AR) is one of the most popular data mining techniques and has received lot of attention, since the publication of Apriori and AIS and algorithms [9,10].Generally, data mining (Data or Knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

Association rules are used to establish relationships among a group of items in the database. These relationships aren't supported inherent properties of the in formation themselves, but rather supported co-occurrence of the data items. Its main algorithm Apriori, not solely influenced the association rule mining community, however it affected other data mining fields in addition.

There are two vital basic methods [3] for association rules, support(s) and confidence(c). Since the database is huge and users worry about only those frequently purchased substance, regularly thresholds of support and confidence are determined by users to remove those rules that are not so interesting or needful. The two thresholds are known as minimal support and minimal confidence respectively.

Support(s) of an association rule is defined as the percentage/fraction of tuples that contain A and B to the total number of tuples in the database. Assume the support of an item is 0.1%, it means only 0.1% of the transaction purchased this item. Confidence is defined as the percentage/fraction of the number of transactions that contain A and B to the total number of records that contain A. Various measures have been introduced to explain the strength of the relationship between item sets A and B such as support, confidence and interest.

II. PROBLEM STATEMENT

The problem can be stated as follows: Let I={I_1, I_2……I_m}bc a set of m distinct literals called items, a set of variable D is length transactions over I**.** Each transaction contains a set. of items (i_1,i_2….i_k) **I.** A transaction also has associated unique identifier called T1D. An association rule is an implication- of the form A=>B, where A,B C I, and A∩B=Ø. A is called the antecedent and B is called the consequent of the rule. In general, a set of items (such as the antecedent or the consequent of a rule) is called an itemset. The number of items in an itemset is called the length of an itemset. Itemsets of some length k are referred to as k-itemsets. For an itemset A.B, if B is an m-itemset then B is called R ,n-extension of A.

Each itemset has an associated measure of statistical significance called support. For an itemset A C I, support(A) = s if the fraction of transactions in D, containing A equals s. A rule has a measure of its strength called confidence defined as the ratio support (A U B) / support(A). The problem of mining association rules is to generate all rules that have support and confidence greater than some user specified minimum support and minimum confidence thresholds, respectively. This problem can be

decomposed into the following sub problems:

1. All itemsets that have support above the user specified minimum support are generated. These itemset are called the large itemsets. All others are said to be small.

2. For each large itemset, all the rules that have minimum confidence are generated as follows: for a large itemset A and any B C A, if support (A)/support(A - B) > minimum-confidence, then the rule A-B => B is a valid rule.

For example, let T1 = {P,Q,R}, T2 = {P,Q,S},T3 = {P,S,0} and T4 = {P,Q,S} be the only transactions in the database. Let. the minimum support and minimum confidence be 0.6 and 0.9 respectively. Then the large itemsets are the following :{PMQ}, {S}, {P,Q}, {P,S} and {P,Q,S}. The valid rules are Q-> P AND S => P.

III. RELATED WORK

Several algorithms for mining associations have been proposed in the literature [1, 3, 6, 7, 11, 12, 14, 19, 15]. The Apriori algorithm [3, 2] forms the core of almost all of the current algorithms. The key observation used is that all subsets of a frequent itemset must themselves be frequent. During the initial pass over the database the support for all single items (1-itemsets) is counted. The frequent 1-itemsets are used to generate candidate 2-itemsets. The database is scanned again to obtain their support, and the frequent 2-itemsets are selected for the next pass. This iterative process is repeated for n=3;4_until there are no more frequent n-itemsets to be found. However, if the database is too large to fit in memory, these algorithms incur high I/O overhead or scanning it in each iteration. Paper, we analyzed and studied various existing improved apriori algorithm to mine frequent itemsets. Mainly common drawbacks are found in various existing apriori algorithm which is improved by using different approaches. It can be applied to many different applications like market basket analysis, telecommunication, network analysis, banking services and many others.

A new method for generating frequent itemsets by using frequent itemset tree (FI-tree)[5]. The analysis of total execution time for generating frequent itemsets denoted with standard dataset wine. Method execution time is better compare to SaM method. At 60% support threshold, two methods nearly matches the execution time.

The comparison of 2 new algorithms Apriori and Apriori-hybrid with the previously known algorithms, the AIS and SETM algorithms have done [11].

Eclat is vertical data format algorithm. Basic Features of Eclat have introduced in [13] i.e. Transaction Recoding, Types of Incidence Structures, Incidence Matrix Derivation etc. Survey on three different association rule mining algorithms[16] -AIS, FP-tree and Apriori algorithms have done. It describe their drawbacks which would be helpful to find new solution for the problems found in these algorithms.

[19] Described an algorithm which is not only efficient but also fast for discovering association rules in large databases in the (n-l)th pass are used to generate the candidate item sets Cn, using the Apriori-gen function Next, the database is scanned and the support of candidates in Cn is counted. This process illustrate in fig. 1, which is derived from Table-1. The working of Apriori algorithm is fairly depends upon the Apriori property which states that" All nonempty subsets of a frequent itemsets must be frequent".

IV. METHOD DESCRIPTION

There are various techniques can be used to generate strong association rules among huge number of rules. They are as follow:

1) Apriori Algorithm

It searches for large item sets during its initial database pass and uses its result as the seed for discovering other large datasets during subsequent passes. Rules having a support level above the minimum are called large or frequent item sets and those below are called small item sets. The algorithm is based on the large item set property which states: Any subset of a large item set is large and any subset of frequent item set must be frequent.

The Apriori algorithm performs a breadth-first search in the search space by generating candidate k+1-itemsets from frequent k item sets. The frequency of an item set is computed by counting its occurrence in each transaction. Apriori is an influential algorithm for mining frequent item sets For Boolean association rules. Since the Algorithm uses prior knowledge of frequent item set it has been given the name Apriori.

Working:

1. Find all frequent itemsets:
   - Get frequent items:
     - Items whose occurrence in database is greater than or equal to the min.support threshold.
   - Get frequent itemsets:
     - Generate candidates from frequent items.
     - Prune the results to find the frequent itemsets.
2. Generate strong association rules from frequent itemsets
   - Rules which satisfy the min.support and min.confidence threshold

Pseudo code of Apriori Algorithm

- Ck: Candidate itemset of size k
- Lk : frequent itemset of size k
- L1 = {frequent items};
- **for** (k = 1; Lk !=∅; k++) **do begin**
- Ck+1 = candidates generated from Lk;
- **for each** transaction t in database do
  - increment the count of all candidates in Ck+1 that are contained in t
- Lk+1 = candidates in Ck+1 with min_support
- **end**
- **return**∪kLk;

2) F-P Growth Algorithm

The F-P stands for Frequent Pattern. The FP-Growth Algorithm, proposed by Han in, is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns, e.g. the Apriori Algorithm and the Tree Projection. In some later works it was proved that FP-Growth has better performance than other methods, including Eclat and Apriori. The popularity and efficiency of FP-Growth Algorithm contributes with many studies that propose variations to improve his performance.

The FP-Growth Algorithm is an alternative way to find frequent itemsets without using candidate generations, thus improving performance. For so much it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the itemset association information.

Working:

Algorithm 1: FP-tree construction

Input: A transaction database DB and a minimum support threshold ?.
Output: FP-tree, the frequent-pattern tree of DB.
Method: The FP-tree is constructed as follows.

1. Scan the transaction database DB once. Collect F, the set of frequent items, and the support of each frequent item. Sort F in support-descending order as FList, the list of frequent items.
2. Create the root of an FP-tree, T, and label it as "null". For each transaction Trans in DB do the following:

- Select the frequent items in Trans and sort them according to the order of FList. Let the sorted frequent-item list in Trans be [p | P], where p is the first element and P is the remaining list. Call insert tree([p | P], T).

The function insert tree([p | P], T) is performed as follows. If T has a child N such that N.item-name = p.item-name, then increment N 's count by 1; else create a new node N , with its count initialized to 1, its parent link linked to T , and its node-link linked to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree(P, N) recursively.

Algorithm 2: FP-Growth

Input: A database DB, represented by FP-tree constructed according to Algorithm 1, and a minimum support threshold ?.
Output: The complete set of frequent patterns.
Method: call FP-growth(FP-tree, null).

Procedure FP-growth(Tree, a)

{
(01) if Tree contains a single prefix path then { // Mining single prefix-path FP-tree
(02) let P be the single prefix-path part of Tree;
(03) let Q be the multipath part with the top branching node replaced by a null root;
(04) for each combination (denoted as ß) of the nodes in the path P do
(05) generate pattern ß ∪ a with support = minimum support of nodes in ß;
(06) let freq pattern set(P) be the set of patterns so generated;
}
(07) else let Q be Tree;
(08) for each item ai in Q do { // Mining multipath FP-tree
(09) generate pattern ß = ai ∪ a with support = ai .support;
(10) construct ß's conditional pattern-base and then ß's conditional FP-tree Tree ß;
(11) if Tree ß ≠ Ø then
(12) call FP-growth(Tree ß , ß);
(13) let freq pattern set(Q) be the set of patterns so generated;
}

(14) return(freq pattern set(P) ∪ freq pattern set(Q) ∪ (freq pattern set(P) × freq pattern set(Q)))
}

Pseudo Code of F-P Growth Algorithm
Input: constructed FP-tree
Output: complete set of frequent patterns
Method: Call FP-growth (FP-tree, null). procedure FP-growth (Tree, a)
{
1) if Tree contains a single path P then
2) for each combination do generate pattern p a with support = minimum support of nodes in p.
3) Else For each header ai in the header of Tree
do {
4) Generate pattern p = ai a with support = ai.support;
5) Construct p.s conditional pattern base and then p.s conditional FP-tree Tree p
6) If Tree p = null
7) Then call FP-growth (Tree p, P)}
}

3) Eclate Algorithm

Eclat is a program for frequent item set mining, a data mining method that was originally developed for market basket analysis. Frequent item set mining aims at finding regularities in the shopping behavior of the customers of supermarkets, mail-order companies and online shops. In particular, it tries to identify sets of products that are frequently bought together. Once identified, such sets of associated products may be used to optimize the organization of the offered products on the shelves of a supermarket or the pages of a mail-order catalog or web shop, may give hints which products may conveniently be bundled, or may allow to suggest other products to customers. However, frequent item set mining may be used for a much wider variety of tasks, which share that one is interested in finding regularities between (nominal) variables in a given data set. For an overview of frequent item set mining in general and several specific algorithms (including Eclat).
The Eclat algorithm is defined recursively. The initial call uses all the single items with their tidsets. In each recursive call, the function IntersectTidsets verifies each itemset-tidset pair [(x),t(x)] with all the others pairs [(y),t(y)] to generate new candidates N_{XY}. If the new candidate is frequent, it is added to the set P_X. Then, recursively, it finds all the frequent itemsets in the X branch. The algorithm searches in a DFS manner to find all the frequent sets.

Function Eclat_growth-Algorithm (Database: D, Min_sup: MS)
// the basic function of Eclat_growth
Define: VM : Vertical Matrix, PT: Two-dimensional Pattern Tree,
FIs: Frequent Itemsets
//Input: D, MS; Output: All FIs
L1. VM = CreateVMfromDatabase(Database: D)
L2. PT= CreateNullPatternTree
L3. for i = 1 to length(VM) do
L4. if length(VM[i].TID_sets) >= MS then
L5. AddItemsettoPatternTree(VM[i], PT, MS);
L6. end // end for i = 1 to length(VM)
L7. FIs=GetAllFrequentItemsetsfromPatternTree(Two-dimensional
Pattern Tree: PT)
End

V. RESULT AND DISCUSSION

This paper is based on available algorithm and the data set, hence the data collection is as follows:
I have develop java program based on the association rule mining algorithm and the execution time is consider for the data. Based on the data of various algorithm the comparison has complete.
The data set with different number of transaction will be consider for the execution time of the program. Here we take the data set of diabetic patients from the UCI Data repository.
Then the paper contains the result of the proposed approach followed by comparison of the various algorithms
Execution time of various table according to the transaction of data set

Data Set	No. Of Transaction	Apriori(s)	Eclate(s)	F-P Growth(s)
Diabetic Patient1	2562	35	0.42	0.13
Diabetic Patient2	6287	065	0.47	0.16
Diabetic Patient3	1728	32	0.4	0.12
Diabetic Patient4	8184	75	0.5	0.19
Diabetic Patient5	1219	30	0.39	0.1
Diabetic Patient6	16632	95	0.6	0.3
Diabetic Patient7	22940	102	0.63	0.35
Diabetic Patient8	5736	60	0.44	0.16
Diabetic Patient9	26887	108	0.65	0.32
Diabetic Patient10	6641	68	0.48	0.16
Diabetic Patient11	6785	68.5	0.48	0.16
Diabetic Patient12	31703	120	0.71	0.45
Diabetic Patient13	18441	97	0.61	0.3
Diabetic Patient14	7128	72	0.49	0.18
Diabetic Patient15	12984	91	0.58	0.25
Diabetic Patient16	11856	88	0.56	0.22
Diabetic Patient17	6802	69	0.48	0.16
Diabetic Patient18	16080	93	0.59	0.27
Diabetic Patient19	28030	112	0.68	0.38
Diabetic Patient20	11592	85	0.55	0.2

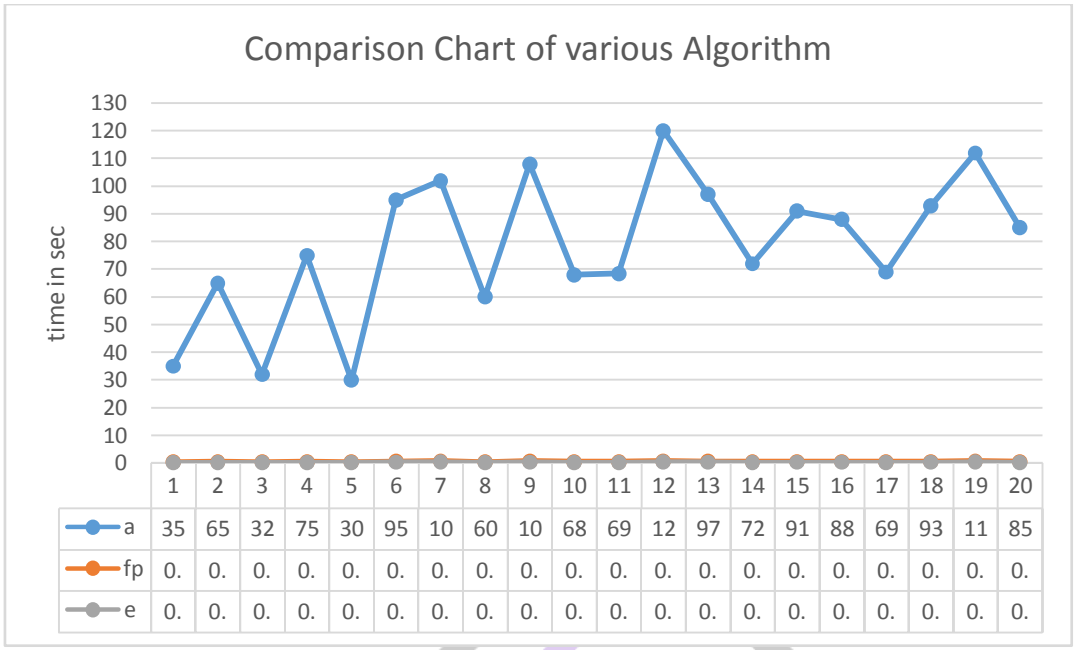
For evaluation of Algorithms, we used 3 scenarios for best, average and worst cases. It should be noted that in all of the instances of our simulation, algorithm predicted either FP-Growth or Eclat, which had running times much shorter than Apriori. This showed that Algorithm Programs was able to predict the better algorithm, but was limited by the overhead in feature exaction and classification.

Best Case:
For the best case in the simulation, i.e if the algorithm guessed is one with the longest running time (Apriori), Algorithm program guarantees that it will predict a better algorithm. The overhead is significantly low compared to the running time of the slower algorithm. The running time of the guessing scenario will be the same as the slowest algorithm.

Average Case:
In the average case, the probability of guessing any algorithm is the same. The running time of the guessing scenario will be the mean of the algorithms. Algorithm still outperforms guessing in this scenario.

Worst Case:
Obviously in the worst case, if the fastest algorithm is guessed, then APRIORI is beaten because of the overheads. This is because the running time of this guessing scenario will be the same as the fastest algorithm. From the simulation, PROGRAM can predict the fastest algorithm with 80% accuracy. More generally, if the difference
between the slowest and fastest algorithms is significantly high, and more than other algorithm overhead (around 1s in the simulation), then other algorithm(F-P, ECLATE) beats the guessing strategy in the best and average scenarios. Even though Apriori was faster than FP-Growth and Eclat for some datasets during training, in our model testing simulation, Apriori was thes lowest on all data set. However, there may be special cases where an algorithm is fast on some nodes, and slow on other nodes due to the way the dataset is split, and the characteristics of the sub-datasets. In other words, different algorithms are optimal for different dataset partitions. Consequently,
a different algorithm will be predicted for each sub-dataset, with a different overhead value for the sub-dataset. In this special case as well, if the difference between the running times of the fastest and slowest algorithm on each node is more than the largest overhead, Algorithm program will out perform arbitrary guessing, based on our high accuracy on predict-ing the best algorithm. This would be investigated in future research.

Chart for the table



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
a	35	65	32	75	30	95	10	60	10	68	69	12	97	72	91	88	69	93	11	85
fp	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
e	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.

It is clear from the Figure that at the comparison of execution time of APRIORI, F-P GROWTH and ECLATE algorithm and result is that the ECLATE algorithm is more faster than less than APRIORIalgorithm and F-P Growth algorithm. If support is very less, overhead of APRIORI increases because more frequent item sets will be generated thatwill give rise to more combinations of those frequent item sets which in turn increases execution time. If support is very high,no frequent Item set will be generated easily, which increasesand execution time.

VI. CONCLUSION AND FUTURE WORK

Due to the size of database has increased rapidly. Therefore require a system to handle such huge amount of data. In this paper, algorithmic aspects of association rule mining are dealt with. From a various variety of efficient algorithms the most important ones are compared. The algorithms are systemized and their performance is analyzed based on execution time considerations. Despite the identified fundamental differences concerning employed strategies, execution time shown by algorithms is different. The comparison table shows that the Apriori algorithm out performs other algorithms in cases of closed item sets whereas FP growth displayed better performance in all the cases. The overall goal of the frequent item set mining process helps to form the association rules for further use. This paper gives a brief survey on association rule mining algorithm Apriori and FP Growth algorithm and Eclate algorithm.

The result of the various algorithm are satisfactory on single processor. The future work may include the implementation of the various improved association rule mining algorithm on the data distributed on parallel processors. The result are expected to be different in the case. We also wish to see whether the database scans reduces on each processor using this approach.

REFERENCES

[1] S. Srivastava et al, 2011 “On Performance Evaluation of Mining Algorithm for Multiple-Level Association Rules based on Scale-up Characteristics”, Journal of Advances in Information Technology, VOL. 2, NO. 4.

[2] [Nuntawut et al., 2014] “A Technique to Association Rule Mining on Multiple Datasets”, Journal of Advances in Information Technology, vol. 5, no. 2, may 2014.

[3] S. Kotsiantis, D. Kanellopoulos “Association Rules Mining: A Recent Overview”, GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp. 71-82

[4] P. Kandpal, “ Association Rule Mining In Partitioned Databases: Performance Evaluation and Analysis”,(Master Thesis) IIIT-Allahabad,India

[5] T. Siddiqui, M Afshar Aalam, and Sapna Jain, 2012 “Discovery of Scalable Association Rules from Large Set of Multidimensional Quantitative Datasets” journal of advances in information technology, vol. 3, no. 1

[6] R. S. Thakur et al., 2006 “Mining Level-Crossing Association Rules from Large Databases” Journal of Computer Science 2 (1): 76-81, 2006 ISSN 1549-3636

[7] N. Kaoungku et al, 2014 “ A Technique to Association Rule Mining on Multiple Datasets” Journal of Advances in Information Technology, vol. 5, no. 2,

[8] S. Dehuri, et al. 2006] “Multi-objective Genetic Algorithm for Association Rule Mining Using a Homogeneous Dedicated Cluster of Workstations” American Journal of Applied Sciences 3 (11): 20862095, 2006 ISSN 1546

[9] R. Agrawal, T. Imilienski and A. Swami, "Mining association rules between sets of items in large databases," Proceeding of the ACM SIGMOD Int"l Conference on Management of Data, Washington DC, **(1993)**, pp. 207-216
[10] Agrawal R, Srikant R., "Fast Algorithms for Mining Association Rules", VLDB. Sep 12-15 1994, Chile, 487-99, pdf, ISBN 1-55860-153-8.
[11] Paresh Tanna, Dr. Yogesh Ghodasara**,** Using Apriori with WEKA for Frequent Pattern Mining, (IJETT) -Volume 12 Number 3 -Jun 2014.
[12] Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207-216
[13] C.Borgelt. ―Efficient Implementations of Apriori and Eclat‖. In Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations, CEUR Workshop Proceedings 90, Aachen, Germany 2003.
[14] D.N. Goswami, Anshu Chaturvedi, C.S. Raghuvanshi. Frequent Pattern Mining Using Record Filter Approach. International Journal of Computer Science Issues. 2010; 7(4): 38–43. Available from: ijcsi.org/papers/7-4-7-38-43.pdf
[15] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. IBM Research Report RJ9839, IBM Almaden Research Center, San Jose, California, June 1994.