E-Rules: An Enhanced Approach to Derive Disjunctive and useful Rules from Association Rule Mining without Candidate Item Generation

Kannika Nirai Vaani M
Sr. Faculty,
Tech Mahindra Ltd,
Bangalore 560068, Karnataka, India

E Ramaraj
Professor,
Dept of Computer Science and Engineering,
Alagappa University,Karaikudi, Tamil Nadu.

ABSTRACT

Association rule mining is one of the most important and well-researched techniques of data mining, that aims to induce associations among sets of items in transaction databases or other data repositories. Currently Apriori algorithms play a major role in identifying frequent item set and deriving rule sets out of it. However there are few shortfalls in conventional Apriori Algorithm. They are i) candidate items generation consumes lot of time in the case of large datasets ii) It supports majorly the conjunctive nature of association rules iii) The single minimum support factor to generate the effective rules. However points ii) and iii) have been addressed effectively in the earlier work [10] with reference to Apriori Algorithm. But this paper majorly worked on the above point i). In the proposed algorithm, FP growth algorithm has been taken for a reference in generating frequent item set without candidate generation. There are appreciable amount of modification and enhancements have been worked out in line with the earlier work. Besides this work also taken care the integration issues while fitting it suitably with the earlier worked out algorithm [10]. This packaged Algorithm is named as E-Rules (Effective Rules). After incorporating the modified FP Growth Algorithm, it has been observed that there is a prominent difference in time taken and performance as it does not involve candidate generations. Hence 'E-Rules' addressing the faster generation of frequent item sets, so that to offer interesting/useful rules in an effective and optimized manner with the help of Genetic Algorithm.

Keywords

FP Growth Algorithm, Genetic Algorithm, Lift ratio, Multiple Minimum Support, Disjunctive Rules.

1. INTRODUCTION

1.1. Association rule mining in Data Mining:

Data mining is the task to mining the useful meaningful information from data warehouse. It is the source of inexplicit, purely valid, and potentially useful and important knowledge from large volumes of natural data [8].The selected knowledge must be not only precise but also readable, comprehensible and ease of understanding.

Association rule basically use for finding out the useful patterns, relation between items found in the database of transactions [9].Association rule mining generally experimented to find all rules that satisfy user-specified minimum support and minimum confidence constraints [3].

The important factor that makes association rule mining practical and useful is the minimum support. It is used to limit the number of rules generated. However, using only a single minsup implicitly assumes that all items in the data are of the same nature and/or have similar frequencies in the database. This is often not the case in real-life applications. In many applications, some items appear very frequently in the data, while others rarely appear. If the frequencies of items vary a great deal, we will encounter two problems:

1. If minsup is set too high, we will not find those rules that involve infrequent items or rare items in the data.
2. In order to find rules that involve both frequent and rare items, then minsup to be kept very low. However, this may produce too many rules.

So when one common support is fixed as minimum support for all the items, the rules which are not frequent occur but majorly contributing towards profit may get lost without notice.

For example, in a supermarket transaction data, in order to find rules involving those infrequently purchased items such as food processor and cooking pan (they generate more profits per item) very minimum support needs to be set ; but due to this the unwanted and rare items will not be get pruned. Hence fixing multiple minimum support for each items have become significant.

1.2. Crisis in Apriori:

In the earlier work [10] the Apriori algorithm is customized to include multiple supports, disjunctive and conjunctive rules and Genetic Algorithm was used to generate useful rules effectively. Major shortfall in modified Apriori was 'time taken' to generate frequent itemsets. In general Apriori-like algorithm may still suffer from the following two nontrivial costs, It is costly to handle a huge number of candidate sets [12]. It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for mining long patterns.

If one can avoid generating a huge set of candidates, the mining performance can be substantially improved. Hence the need of introducing an algorithm which takes considerably less time was realized. There comes the modified FP growth Algorithm.

1.3. Conjunctive and Disjunctive rules:

Association rule mining deals conjunctive rules majorly. But using disjunctive rules, extensive rule sets which are very much useful in mining the dataset can be found out effectively. At times disjunctive rule sets are also preferred to infer interesting rules. For example ideal rule set for the below query "If classes B or C or D are committed, what is the chance that A is also committed?" would be B OR C OR D →A, where in the disjunctive nature of rules need to be considered for a better analysis.

1.4. Disjunctive Rules:

Let X = {i1, i2...in} be a set of items. X_C = {i1 AND i2 AND … AND in} or simply {i1i2...in} is a conjunctive term of X, and XD = {i1 OR i2 OR … OR in} is a Disjunctive term of X. The possible rule is of one of the following types, involving conjunctive and disjunctive terms from the set of items X U Y| X ∩ Y = Φ [2][5].

Table I:Disjunctive Rule Set:

Type	Support	Confidence
Xc ⇒Yc	s = P(XC U YC)	c = P(YC | XC)
Xc ⇒YD	s = P(XC U YD)	c = P(YC | XD)
XD ⇒Yc	s = P(XD U YC)	c = P(YD | XC)
XD ⇒YD	s = P(XD U YD)	c = P(YD| XD)

Rule sets List:

XY Z ⇒VW	- type XC ⇒ YC
XY Z ⇒V OR W	- type XC ⇒ YD
X OR Y OR Z ⇒ VW	- type XD ⇒ YC
X OR Y OR Z ⇒ V OR W	- type XD ⇒ YD

The number of rules that are found by the Associations mining function can be reduced by using rule filters if the number of frequent item sets is high. Rule filters are a powerful way to limit the amount of rules to be generated or the content of the rules. This parameter determines the maximum number of items that occur in an association rule. In the example the frequent itemset (E,A,C) has been used for analysis and in this case the maximum rule length is 3 and maximum antecedent allowed is 2; the maximum consequents allowed are 2.

1.5. Lift Ratio

A high value of confidence suggests a strong association rule. However this can be deceptive because if the antecedent and/or the consequent have a high support, we can have a high value for confidence even when they are independent. A better measure to judge the strength of an association rule is to compare the confidence of the rule with the benchmark value where we assume that the occurrence of the consequent item set in a transaction is independent of the occurrence of the antecedent for each rule. This benchmark can be found out from the frequency counts of the frequent item sets. This enables to compute the lift ratio of a rule. The lift ratio is the confidence of the rule divided by the confidence assuming independence of consequent from antecedent. A lift ratio greater than 1.0 suggests that there is some usefulness to the rule. The larger the lift ratio, the greater is the strength of the association. With the lift value, the importance of a rule can be validated in an effective manner.

Confidence and Life factors are calculated as below,

$$Confidence = \frac{No.of\ transactions\ \ contain\ \ all\ the\ items\ \ in\ A\ \&C}{No.of\ transctions\ \ contain\ \ \ the\ items\ \ in\ A}$$

$$= (support\ of\ AUC)/(support\ \ of\ A)$$

$$Expected\ Confidence =$$

$$\frac{No.of\ transactions\ having\ the\ consequent\ items}{Total\ no.of\ transactions}$$

$$Lift = \frac{Confidence}{Expected\ Confidence}$$

The above factor can also be included while validating any rule set.

1.6. Multiple Minimum Supports:

In many data mining applications [4], some items appear very frequently in the data, while others rarely appear. If minsup is set too high, those rules that involve rare items will not be found. To find rules that involve both frequent and rare items, minsup has to be set very low. This may cause combinatorial explosion because those frequent items will be associated with one another in all possible ways. The disadvantage of support is the rare item problem. Items that occur very infrequently in the data set are pruned although they would still produce interesting and potentially valuable rules. The rare item problem is important for transaction data which usually have a very uneven distribution of support for the individual items.

E.g., Consider four items A, B and C in a database. If the minimum support count is 10 %, then we may lose the rule set AB if it had support 7% .But assume now we have multiple minimum item supports are

MIS (A) = 20% MIS (B) = 3%
MIS(C) = 4%

And their actual supports: {A 18%, B 4%, C 3%}

Then the rule AB is not discarded as it satisfies the rule min (MIS (A, B))>=min (Support (A<B)).

Hence there could be a revision in FP algorithm accordingly.

2. FP-GROWTH ALGORITHM

FP Growth approach is based on divide and conquers strategy for producing the frequent item sets[11]. FP-growth is mainly used for mining frequent item sets without candidate generation. Major steps in FP-growth are,

Step1: It firstly compresses the database showing frequent item set in to FP-tree. FP-tree is built using 2 passes over the dataset.

Step2: It divides the FP-tree in to a set of conditional database and mines each database separately, thus extract frequent item sets from FP-tree directly.

FP-tree is a frequent pattern tree can be defined below.

1. It consists of one root labeled as "null", a set of item prefix sub trees as the children of the root, and a frequent-item header table.

2. Each node in the item prefix sub tree consists of three fields: item-name, count, and node-link, where item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching

this node, and node-link links to the next node in the FP-tree carrying the same item-name, or null if there is none.
3. Each entry in the frequent-item header table consists of two fields, (1) item-name and (2) head of node-link, which points to the first node in the FP-tree.
Based on this definition, we have the following FP-tree construction algorithm. Let I = {a1,a2,….. an} be a set of items, and a transaction database DB = (TID1; TID2,….. TIDn),where each TID(i) refers to transaction from 1...n that contains a set of items in I.

2.1. Algorithm 1: Modified FP-tree construction:

Input: A transaction database DB and a set of minimum support threshold.
Output: Frequent pattern tree

[Note: Appropriate changes for the proposed algorithm has been incorporated in the below algorithm.]

1. Scan the transaction database DB once. Collect the set of frequent items in each TID(i) and their predefined supports(m_{tk}) and actual supports(s_{tk}).

2. Create the root of an FP-tree, and label it as "null". For each transaction TID(i) in DB do the following.
If min (m_{tk}(TID(i))) < = s_{tk}(TID(i)) then select and sort the frequent items in TID(i).Let the sorted frequent item list in TID(i) be [p|P], where p is the first element and P is the remaining list. Call insert tree([p|P]; TID(i)).

3. insert tree([p|P]; TID(i)):
If TID(i) has a child n, where n.item = p then increment n.count by one else create new node N with n.count = 1; Link it up from the header table . If P is nonempty call insert_tree(P, N) recursively.

2.2. How to find all frequent patterns from the FP-Tree:

We can derive the frequent items as follows,

1) Find all frequent patterns containing one of the items

2) Find all frequent patterns containing the next item but NOT containing the previous one

3) Repeat step 2 until we're out of items

2.3. Algorithm 2: FP-Tree frequent patterns generation:

Input:
FP-tree constructed based on Algorithm 1, using DB and a minimum support thresholds.
Output:
The complete set of frequent patterns.

```
Procedure FP-growth (Tree; α)
{
(1) If Tree contains a single path P
(2) then for each combination (denoted as β)
of the nodes in the path P do
(3) generate pattern βUα with support =
    minimum support of nodes in β;
(4) else for each ai in the header of Tree do
{
(5) generate pattern β = ai U α with
        support = ai.support;
(6) construct β 's conditional pattern base and
then β 's conditional FP-tree Tree β;
(7) if Tree $_{β}$ ≠ ø ;
(8) then call FP-growth (Tree $_{β}$; β)
}
}
```

3. PSEUDO CODE FOR E-RULES:

Input:

A set of n transaction data T , a set of p items to be purchased , each item t_i, with a minimum support value m_i, i=1 to p , and a minimum confidence value.

Output:

A set of association rules in the criterion of the maximum values of minimum supports.

Procedure:

Step 1: Perform Algorithm 1.
Step 2: Perform Algorithm 2.
This will give Frequent Item Set (FIS).
Step 3: Generate bit pattern for FIS for all possible LFH and RHS rule sets.
Step 4: Construct all possible rule sets (Disjunction and conjunction) as per proposed rule guides in Rule sets List. Details are briefed in 3.1.
Step 5: Calculate confident and Lift factors to validate effective rules.
Step 6: Output the rules which are greater than minimum confident and Lift greater than or equal to 1.

3.1. To generate rules:

Each frequent item set in frequent item set derived can be given as one of the input for the below algorithm to derive the respective conjunctive and disjunctive rules.(E,A,C) frequent set has been taken for the demonstration purpose.

Step 1: Create function whose parameters are
Dataset, list_of_antecedents-A
, list_of_consequent –C
List of A and C can be generated using the following conditions using Genetic Algorithm.
Min (A) =Cnt (FIS)-[Cnt (FIS)-1]
Max (A) =Cnt (FIS)-[(Cnt (FIS)-(Cnt (FIS)-1))]
Example, If Frequent Item Set (FIS) =3 then Maximum Antecedes=2 and Minimum Antecedents =1
This holds true for consequents as well, hence
Min(C) = Cnt (FIS)-[Cnt (FIS)-1]

Max(C) = Cnt (FIS)-[(Cnt (FIS)-(Cnt (FIS)-1))]

[A- Antecedent; C-Consequent; Cnt- number of frequent item set
Bit pattern for the frequent item set along with operators are like in Table VI.
And the function returns interesting rules as per Rule sets like in Table VIII]

Step 2: Find out allowed antecedents (A) and Consequent(C)

Step 3:

For each Item set A and for each item set C,

i) Create an offspring when bit pattern (A) is not equal to bit pattern(C).
ii) If Bit pattern (A) and/or Bit pattern(C)>= min_predefined_min_support then return the rule A→C
iii) Else Prune the rule.

Step 4: Return all valid rules

A. Example:

An example is given to demonstrate the proposed algorithm [3]. Table II shows dataset that contains 10 transactions and 7 items and the ordered items as per their frequency.

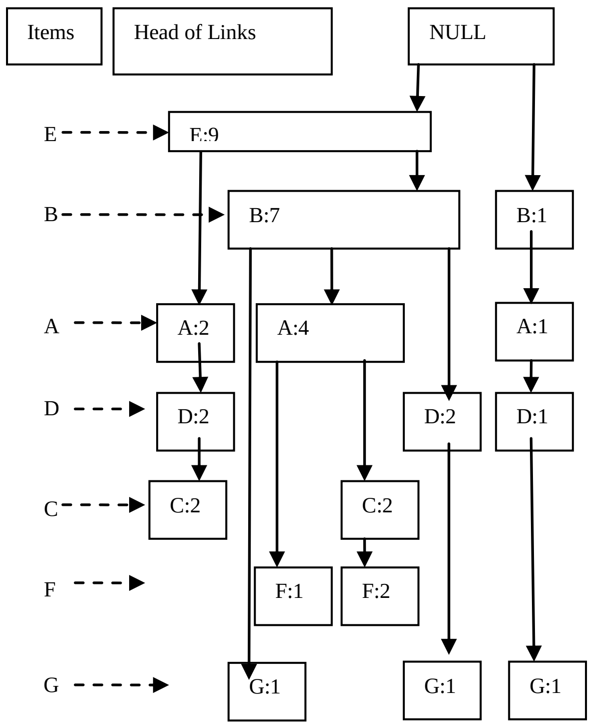
Table II :Dataset for 10 transactions:

TID	**Items**	**Ordered Items**
1	ABDG	BADG
2	BDE	EBD
3	ABCEF	EBACF
4	BDEG	EBDG
5	ABCEF	EBACF
6	BEG	EBG
7	ACDE	EADC
8	ABE	EBA
9	ABEF	EBAF
10	ACDE	EADC

3.2. FP Tree for the dataset in Table II:

A FP tree can be generated as below in Figure :1 for the data set available in Table II. Each node in the tree contains the label of the item along with a counter that shows the number of transactions mapped on to the given path.

Figure :1 FP Tree generation:



Initially FP tree contains only the root node represented by the null symbol. The FP tree is subsequently extended in the following way,

1. The data set is scanned once to determine the support count of each item, infrequently items discarded while the frequent items are sorted in decreasing support counts. For the data set show in Table 2 E is the most frequent item followed by B,A,D,C,F an G.
2. The algorithm makes second pass over the data to construct the FP tree. After reading the first transaction {B, A, D} the nodes labled as B, A, D are created. Every node along the path has frequency count of 1.
3. The above process continues until every transaction has been mapped onto one of the paths given in the FP tree. The resulting FP tree is as below for the dataset mentioned in Table II.

Below tables show the relevant data needed for further analysis .

Table III: Predefined minimum support for each items (mtk)

Item	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Min_sup	0.4	0.7	0.3	0.7	0.6	0.2	0.4

Table IV: Actual support values of all items(stk=ck/n)

Item	A	B	C	D	E	F	G
Support	0.7	0.8	0.4	0.5	0.9	0.3	0.3

Table V: Minimum Item Support (MIS) for each transaction TID(i)

TID	Items	Ordered Items	MIS
1	ABDG	BAD	0.4
2	BDE	EB	0.6
3	ABCEF	EBACF	0.2
4	BDEG	EBD	0.4
5	ABCEF	EBACF	0.2
6	BEG	EB	0.4
7	ACDE	EADC	0.3
8	ABE	EBA	0.4
9	ABEF	EBAF	0.2
10	ACDE	EADC	0.3

In the above table we can see items in each transaction are ordered in descending order of its MIS values and infrequent items are removed if any. For example item ‘G’ in TID -1 has support count less than its corresponding MIS value. Hence it has been ignored.

3.3. Conditional Pattern Bases:

The conditional pattern bases and condition FP trees are summarized below.

Item	Conditional Path	Conditional Pattern Base	Conditional FP tree
G	{(E:9,B:7), (E:9,B:7,D:2), (B:1,A:1,D:1)	{(E:1,B:1), (E1,B:1), (B1,A1,D1)}	{(E:2,B:3)}|G}
F	{(E:9,B:7,A:4), (E:9,B:7,A:4,C:2) }	{(E:1,B:1,A:1), (E:2,B:2,A:2)}	{(E:3,B:3,A:3)|F }
C	{(E:9,A:2,D:2), (E:9,B:7,A:4)}	{(E:2,A:2,D:2), (E:2,B:2,A:2)}	{(E:4,A:4)|C}
D	{(E:9,A:2), (E:9,B:7), (B:1,A:1)}	{(E:2,A:2), (E:2,B;2), (B:1,A:1)}	{(E:4,A:3,B:3)}| D
A	{(E:9), (E:9,B:7),(B:1)}	{(E2), (E:4,B:4), (B:1)}	{(E:6,B:5)}|A
B	{(E:9)}	{(E:7),(ø)}	{(E:7),(ø:1)}|B
E	ø	ø	ø

3.4. Frequent items generation:

From the FP-tree construction process, we can see that one needs exactly two scans of the transaction database. Let us see how the FP tree is constructed for node C. For node C, it derives a frequent pattern (E:4, A:4) and two paths in the FP-tree (E:9,A:2,D:2) and (E:9,B:7,A:4) .The first path indicates that string (E,A,D and C) appears twice in the database.To study which string appear together with C, only C's prefix path {E:2; A:2; D:2} counts. Similarly, the second path indicates string (E,B and A) appears twice in the set of transactions in DB, or C's prefix path is {E2,B:2,A:2).These two prefix paths of C form C's sub-pattern base, which is called C's conditional pattern base (i.e., the sub-pattern base under the condition of C's existence). Construction of an FP-tree on this conditional pattern base (which is called C's conditional FP-tree) leads to branch (E:4,A:4).

Hence the frequent patterns are generated as per Algorithm 2. They are **{E,A,C,(E,A),(E,D),(E,C),(E,A,D), (E,A,C),(E,A,D,C)}** . The search for frequent patterns associated with C terminates.

Frequent itemset **(E, A, C)** has been taken for the demonstration purpose. The possible association rules can be generated as per below table. First column in the below table denotes whether it is Antecedent or Consequent; 1-Antecedent 0-Consequent.

Table VI :Encoding of Antecedent and Consequent:

Antecedent/Consequent	A	C	E
1	1	0	0
1	0	1	0
1	0	0	1
1	1	1	0
1	0	1	1
1	1	0	1
0	1	0	0
0	0	1	0
0	0	0	1
0	1	1	0
0	0	1	1
0	1	0	1

3.5. Bitwise items storage:

In order to read and scan the item sets for the calculation, each item needs to get encoded with respect to the transaction into bits.Here the encoding style could be 1 for presence of item in transaction and 0 for absence.

Table VII: The Bit pattern for items.

Item	Transaction ID	Bit Pattern	Count
A	{TID1,ITD3,TID5, TID7,TID8,TID9, TID10}	1010101111	7
B	{TID1,TID2,TID3, TID4,TID5,TID6, TID8,TID9}	1111110110	8
C	{TID3,TID5,TID7, TID10}	0010101001	4
D	{TID1,TID2,TID4, TID7,TID10}	1101001001	5
E	{TID2,TID3,TID4, TID5,TID6,TID7, TID8,TID9,TID10}	0111111111	9
F	{TID3,TID5,TID9}	0010100010	3
G	{TID1,TID4,TID6}	1001010000	3

The following possible rules can be generated as per Rule sets List 1,

"If A is bought then C and E also bought"
A⟹CE
"If A is bought then C or E also bought "
A⟹C or E
Similarly the below possible rules can be derived.

Table VIII : Possible Rules for {E,A,C}:

Rule			Confident	Lift
Antecedent	Symbol	Consequent		
E	⟹	AC	0.444444444	1.111111
C+E	⟹	A	0.666666667	0.952381
A	⟹	CE	0.571428571	1.428571
AE	⟹	C	0.666666667	1.666667
A	⟹	C+E	0.857142857	0.952381
E	⟹	A+C	0.666666667	0.952381
A+E	⟹	C	0.4	1
A+C	⟹	E	0.857142857	0.952381
C	⟹	AE	1	1.666667
CE	⟹	A	1	1.428571
AC	⟹	E	1	1.111111
C	⟹	A+E	1	1

Confident and Lift factors can be calculated using the below,

Confident = Support (AUC)/Support (A)
Lift = Support (AUC)/Support(C)

Where A-Antecedent C-Consequent

If predefined minimum confident is 0.75 then the rules whose confident <0.75 can be pruned .Hence the result set is as follows,

Table IX :Pruned Rules:

Rule			Confident	Lift
Antecedent	Symbol	Consequent		
A	⟹	C+E	0.857142857	0.952381
A+C	⟹	E	0.857142857	0.952381
C	⟹	AE	1	1.666667
CE	⟹	A	1	1.428571
AC	⟹	E	1	1.111111
C	⟹	A+E	1	1

From the above table it is inferred that rules having confident greater than minimum confident and lift ratio for the same is almost 1 or more. This confirms that all these rules have high dependency between antecedent and consequents and thus are strong and useful rules.

4. GENETIC ALGORITHM:

Genetic Algorithm (GA)[1] incorporates Darwinian evolutionary theory with sexual reproduction. GA is stochastic search algorithm modelled on the process of natural selection, which underlines biological evolution. GA has been successfully applied in many search, optimization, and machine learning problems.GA works in an iteration manner by generating new populations of strings from old ones. Every string is the encoded binary, real etc., version of a candidate solution .An evaluation function associates a fitness measure to every string indicating its fitness for the problem. This type of representation is relative to position. Presence of 1 at i^{th} position indicates occurrence of the item in transaction[i]. Similarly presence of 0 at j^{th} position indicates absence of item in transaction[j].

For example,

Bit pattern for A can be given as 1010101111 based on its availability in the transactions in table V. Hence conjunction and disjunction of few rules can be evaluated and its relative count could be calculated as follows,

4.1. Boolean Operation on bit pattern for few rules:

Table X : Boolean Operation on bit pattern

Items	Operation	Result	Count
A and C	{1010101111} and {0010101001}	{0010101001}	4
A or C	{1010101111} or {0010101001}	{1010101111}	5
(A or C) and E	({1010101111} or {0010101001}) and {0111111111}	{0010101111}	6

4.2. Genetic operators:

Genetic Algorithm uses genetic operators [7] to generate the offspring of the existing population. This section describes three operators of Genetic Algorithms that were used in GA algorithm: selection, crossover and mutation.
1) Selection: The selection operator chooses a chromosome in the current population according to the fitness function and copies it without changes into the new population.GA algorithm used route wheel selection where the fittest members of each generation are more chance to select.
2) Crossover: The crossover operator, according to a certain probability, produces two new chromosomes from two selected chromosomes by swapping segments of genes.

3) Mutation: The mutation operator is used for maintaining diversity. During the mutation phase and according to mutation probability, 0.005 in GA algorithm, value of each gene in each selected chromosome is changed.

5. RESULT AND DISCUSSION:

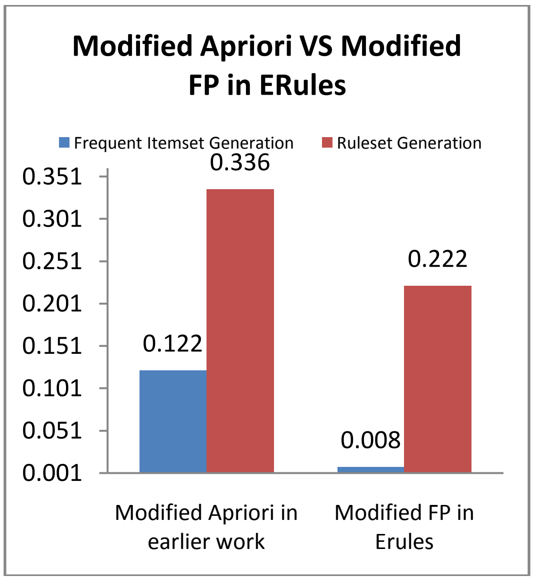
From the result of E-Rules, the following comparison has been made. Time taken for generating rule sets using Modified Apriori (earlier work) and the proposed modified FP algorithm for all the item sets generated in the above example was observed. Though the volume of itemset taken for is not huge enough, the trivial difference has been felt. However when E-Rules is applied to large dataset like any medical, pharmaceutical data etc., definitely the difference will be felt tremendously.

Table X1: Time taken to generate useful rules by E-Rules using Modified Apriori and Modified FP

Algorithm	Time Taken to generate frequent itemset (in Sec)	Time Taken to generate useful rules (in Sec)
Modified Apriori	0.122	0.336
Modified FP in ERules	0.008	0.222

The difference can be shown as below in Figure 2.The proposed Algorithm in the earlier work took overall 0.336 sec to generate the final useful rule sets. Amongst it took initially 0.122 sec for generating frequent item set. Whereas 'ERules' captured only 0.222 sec for generating final useful rules because it needed only 0.008 sec for generating frequent item sets with the proposed algorithm. Hence it drastically reduces the time taken to generate the ultimate useful rules. This difference is very obvious data for comparison to justify our claim effectively.

Figure 2: Comparison between the modified Apriori and the modified FP Algorithms:



6. CONCLUSION:

E-rules is an integrated algorithm for useful and effective association rule mining to capture even useful rare itemsusing multiple minimum support; Conjunctive& Disjunctive rules sets using Genetic Algorithm; Lift Factor to analyse the strength of derived rules. One of the major challenges quoted in the previous work was 'time taken' in frequent item generation. That has been drastically reduced using modified FP algorithm in the present work. It is claimed that this approach is novel as the modified Apriori Algorithm has been replaced with modified FP algorithm in Erules to reduce the time in generating frequent item generation.

7. FUTURE WORKS:

In future it is planned to integrate UML model or multi level data mining to analyse the dataset based on their hierarchy before they get into the algorithm. There is also a plan of identifying the redundant item sets in the early stage to avoid the time taken during rule set generation. And there is also a plan to compare the proposed algorithm with other existing algorithms [6] to validate the robustness and effectiveness. Also planning to simulate a tool for incorporating the proposed algorithm. And the scope of including negative rules into the proposed algorithm is also lined up in one among the future plans.

8. REFERENCES:

[1] Anandhavalli M, Suraj Kumar Sudhanshu, Ayush Kumar and Ghose M.K. *"Optimized association rule mining using genetic algorithm"*, Advances in Information Mining, ISSN: 0975–3265, Volume 1, Issue 2, 2009

[2] Marcus C. Sampaio, Fernando H. B. Cardoso, Gilson P. dos Santos Jr.,Lile Hattori *"Mining Disjunctive Association Rules"* 15 aug. 2008

[3] Bing Liu, Wynne Hsu and Yiming Ma *"Mining Association Rules with Multiple Minimum Supports";* ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-99), August 15-18, 1999, San Diego, CA, USA.

[4] Yeong-Chyi Lee a, Tzung-Pei Hong b,_, Wen-Yang Lin , Mining *"Association Rules with Multiple Minimum Supports Using MaximumConstraints";* Elsevier Science, November `22, 2004.

[5] Michelle Lyman, *"Mining Disjunctive Association Rules Using Genetic Programming"* The National Conference On Undergraduate Research (NCUR); April 21-23, 2005

[6] Farah Hanna AL-Zawaidah, Yosef Hasan Jbara, Marwan AL-Abed Abu-Zanona, *"An Improved Algorithm for Mining Association Rules in Large Databases"* ; Vol. 1, No. 7, 311-316, 2011

[7] Rupesh Dewang, Jitendra Agarwal, *"A New Method for Generating All Positive and Negative Association Rules";* International Journal on Computer Science and Engineering, vol.3,pp. 1649-1657,2011

[8] Olafsson Sigurdur, Li Xiaonan, and Wu Shuning.*"Operations research and data mining,in":* European Journal of Operational Research 187 (2008) pp:1429–1448.

[9] Agrawal R., Imielinksi T. and Swami A. *"Database mining: a performance perspective"*, IEEETransactions on Knowledge and DataEngineering 5 (6), (1993), pp: 914–925.

[10] Kannika Nirai Vaani.M, Ramaraj E *"An integrated approach to derive effective rules from association rule mining using genetic algorithm"*, Pattern Recognition, Informatics and Medical Engineering (PRIME), 2013 International Conference, (2013), pp: 90–95.

[11] Jiawei Han, Jian Pei, and Yiwen Yin. "Mining Frequent Patterns without Candidate Generation", Data Mining and Knowledge Discovery (8), (2004), pp: 53-87.

[12] Oskar Kohonen ,Popular Algorithms in Data Mining and Machine Learning (http://www.cis.hut.fi/Opinnot/T-61.6020/2008/fptree.pdf ;),2008.