ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2022.0385

TOP-DOWN AND BOTTOM-UP APPROACH FOR MINING MULTILEVEL ASSOCIATION RULES FROM CONCEPT HIERARCHICAL DATA IN DISTRIBUTED ENVIRONMENT

Dinesh J. Prajapati
Department of Information Technology, A.D. Patel Institute of Technology, India

Abstract

Hierarchical Data mining using distributed environment is an imperative in big data analysis. Multilevel association rules can provide more substantial information than single level rules, and it also determines hierarchical knowledge from the dataset. Nowadays, numerous e-commerce and social networking sites generates vast amount of structural/semi-structural data in the form of sales data, tweets, text mails, web usages and so on. The data generated from such sources is so large that it becomes very difficult to process and analyze it using conventional approaches. This paper overcomes the computing limitation of single node by distributing the task on multi-node cluster. The performance of this system is compared based on minimum support threshold at diverse levels of concept hierarchy and by varying the dataset size. In this paper, the transactional dataset is created from huge sales dataset using Hadoop MapReduce framework. Then, two distributed multilevel frequent pattern mining algorithms MR-MLAB (MapReduce based Multilevel Apriori using Bottom-up approach) and MR-MLAT (MapReduce based Multilevel Apriori using Top-down approach) are implemented to find interesting level-crossing frequent itemset for each level of concept hierarchy. The hierarchical redundancy in multilevel association rules affects the quality of the market basket analysis. Hence, to improve the performance of the system, the hierarchical redundancy has to be removed from it. Finally, the time efficiency of proposed algorithms is compared with existing Traditional Multilevel Apriori (TMLA) Algorithm. The proposed algorithms with MapReduce framework are found efficient compared to the traditional algorithms.

Keywords:

Distributed Frequent Pattern Mining, Multi-Level Association Rule, MapReduce, Level Crossing Rules

1. INTRODUCTION

Association rule mining is one of the data mining techniques to discover the relationships in the given dataset. Related terms used in this paper are mentioned below.

Itemset: Let $I = \{I_1, I_2, \ldots, I_n\}$ be a set of distinct items. A set of items (X) which is subset of I is called itemset. An itemset X with k distinct items is known as k-itemset [1].

Association Rule: An association rule is represented in the form $X \rightarrow Y$, where X and Y are the itemsets. This rule disclosures the connection between the itemset X with the itemset Y [2].

- ***Big Data:*** Big data is a collection of huge data sets that are processed using traditional data processing tools [3]-[5].
- ***Hadoop:*** Hadoop is an open-source MapReduce based programming model in the distributed processing [6].
- ***Hadoop Distributed File System (HDFS):*** The Hadoop runtime system is attached with HDFS that supports parallelism and concurrency to obtain system reliability [6].
- ***MapReduce:*** The MapReduce framework consists of two functions [7]: Mapper and Reducer. The Mapper function takes an input as <*key*, *value*> pair and generates a set of <*key*, *value*> pair as an intermediate result. The Reducer function receives an intermediate key generated and merges the values of the key.
- ***Multi-level Association Rule Mining***: Association rules produced from multiple levels of different concept hierarchy are known as multilevel association rules [8].
- ***Level-crossing Association Rules:*** The set of association rules who's antecedent and/or consequent have different level of hierarchy are called level-crossing association rules [9].

In multiple-level association rule mining, the items exist in an itemset are categorized by conceptual hierarchy. By forming such a conceptual hierarchy, a procedure of determining association rules at multiple concept levels discovers more meaningful and interesting information from the data [10]. A sample concept hierarchy tree of AMUL dairy (the largest dairy of Asia) with taxonomy information is shown in Fig.1.

Actual items are available at last level of concept hierarchy. In the concept-hierarchy tree, each node denotes a single item of an itemset. There are essentially four levels of the concept hierarchy in this dataset. At any level, item *i* is children of item at level *i*-1. Fresh Products and Frozen Products are two items at first level. Moreover, Fresh Products has two child nodes namely Milk Products and Milk. Frozen Products have Ice-cream and Snacks as children. Such hierarchy continues accordingly. Each node is allocated a number that represents its id. The encoding is carried out from left to right in a sequence. For example, the encoding value for Rose Lassi is 1111 where first digit indicates Fresh Products, second digit indicates Milk Products, third digit indicates Fresh.

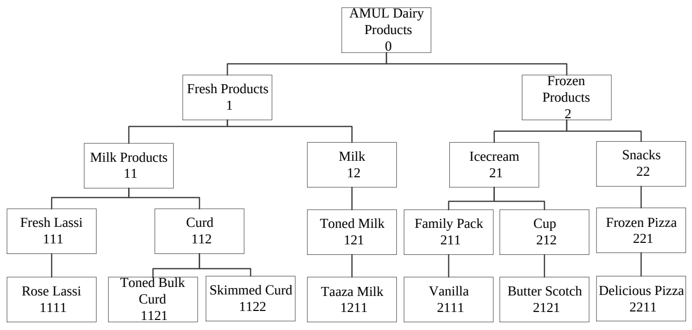

Fig.1. Sample Concept-Hierarchy AMUL Dairy Products with Taxonomy Information

This research paper is summarized in three easy steps in brief: (i) In the first step, the distributed multilevel frequent pattern mining algorithm using top-down and bottom-up approach is implemented to generate multi-level frequent itemsets including level crossing, (ii) In the second step, multi-level frequent itemsets are analyzed based on the retailer, (iii) Finally, the redundant rules are excluded for deriving interesting multilevel association rules. The proposed distributed multi-level frequent pattern mining algorithms are tested on hierarchical sales dataset of AMUL Dairy.

The remaining of this paper is organized as follows. Related work is given in section 2. Section 3 shows the proposed methodology. In section 4, the performance of proposed method is assessed based on sales data. Finally, the conclusions and future scope are drawn in section 5.

2. RELATED WORK

Thakur et al. [9] proposed a top-down approach to reduce support and filter the transaction table, *T* for different levels of concept hierarchy. The processing time is improved and it generates less candidate itemsets. Han and Fu [10] generated more interesting multilevel association rules including level crossing using different interestingness measures. The Authors also suggest the improvement of the methods for mining single level association rules to multiple level association rules. Wan et al. [11] proposed a novel approach for efficiency, integrality and accuracy improvement from primitive concept level of hierarchy. The method proposed in this paper, also considers the dynamic concept hierarchies. By using this approach, different users can generate multilevel association rules based on requirement. The author also declared various issues for support calculation as well as multilevel association rules generation at specific level.

The authors in [12] [13], proposed hierarchical redundancy removal approach using closed itemsets and generators. Author suggests that this approach can also be apply to the approximate basis rule for removal of redundancy and mining of more efficient frequent itemset. Hong et al. [14] proposed an incremental multilevel association rule mining algorithm based on the pre-large concept and efficiently mining of the dataset with taxonomy information. The author proposed algorithm for reduction of mining cost. Gautam and Pardasani [15] proposed a method to discover frequent itemsets using Boolean matrix. The proposed approach scans the transaction database only once without generating itemsets. It also adopts the Boolean vector-based method to discover frequent itemset.

Prakash et al. [16] proposed a novel approach for frequent and in-frequent interesting association rules mining by removing the redundant rules. The proposed approach discovers the complete rules based upon propositional logic. Gautam and Pardasani [17] proposed Partition and Boolean based method for frequent itemsets generation at each level of concept hierarchy by reduction of database scans, I/O cost and overhead of CPU. In this paper, a top-down frequent pattern mining approach is used for multi-level rules generation. Srivastava et al. [18] compared multilevel association rule mining algorithms based on number of transactions. In this paper, the performance the ML_TMLA algorithm is compared with ML_T2L1 and ML_T1LA algorithms for various values of minimum support threshold. The authors concluded that ML_TMLA algorithm is, more efficient than ML_T2L1 and ML_T1LA algorithms for lower minimum support and less efficient than ML_T2L1 and ML_T1LA algorithms for higher minimum support. The author also showed that the level crossing association rule mining algorithms ML_T1LA-C and ML_TMLAC are more efficient than the ML_T1LA and ML_TMLA algorithms, respectively, for lower minimum support threshold.

Gautam and Shukla [19] proposed a reduced minimum support threshold at each level to reduces I/O operations and improve the efficiency. Karim et al. [20] proposed an improved MapReduce framework of distributed system to mine the business-related transactional datasets. The proposed model is highly scalable even though the database size is too large. In this paper, authors implemented “Associated-Correlated-Independent” algorithm to mine the purchase rules more effectively. Zhuang and Wang [21] proposed a novel method for mining weighted concise association rules using closed itemsets and weighted support. Here, each item having different importance is assigned different weight to it. The proposed algorithm mines all the weighted association rules and prunes the duplicate weighted itemset. The experimental results show that the present algorithm is scalable in the case of time.

Chandanan and Shukla [22] proposed an algorithm for hierarchical redundant rules removal using upper level closed frequent itemset and generator. The algorithm proposed in this paper decreases the length of the association rules for quality improvement and information loss reduction. Pumjun and Kreesuradej [23] proposed Incremental Multilevel Association Rule Mining of a Dynamic Database under a Change of a Minimum Support Threshold (IML-ARMCS) algorithm that maintains dynamic database with changing minimum support threshold means new transactions can be added dynamically. Authors compared execution time and number of database scans with existing ML-T2 algorithm. The algorithm proposed in this paper, cannot used when few transactions are removed from the actual dataset. Authors have suggested to extend the present algorithm for covering the multilevel reduced dataset as a future work. Muhammad and Usman [24] proposed a conceptual model to mine the multi-level association rules from real world datasets. The authors presented a method to discover multi-level association rules using multidimensional schema. Authors also presented hierarchical clustering at different levels of data abstraction.

In big data analysis, mining huge pattern is more important for the transactional database containing unique itemset. However, some of the above-mentioned work deal with single level association rule mining using MapReduce and other work deals with multilevel association rule mining without use of distributed environment. Hence, the problem of multilevel association rule mining from huge data in distributed environment is novel idea to improve the time efficiency. The distributed multilevel frequent pattern mining algorithm using top-down and bottom-up approach is applied to generate level-crossing frequent itemsets. Existing TMLA algorithm [9] generates large candidate itemset and its execution time is also higher while dealing with big data. The proposed algorithms improve the drawback of existing traditional algorithms and also generate interesting hierarchical patterns. The objective of proposed work is to eliminate the drawbacks of relational database and facilitate MapReduce framework to improve the execution time of the system and it also generates small candidate itemset.

3. PROPOSED METHODOLOGY

The proposed methodology with architecture is shown in Fig.2. The AMUL dairy sales data is given as input to distributed multilevel frequent pattern mining algorithm. The main drawback of existing multilevel frequent pattern mining algorithms is, (i) It generates a huge candidate itemsets; (ii) The execution time is also high while dealing with big data, and (iii) Some of the interesting level-crossing rules are missed completely. The distributed multilevel frequent pattern mining algorithm proposed in this paper removes these drawbacks. In the proposed methodology, once the actual transactional dataset is stored in HDFS, the entire dataset is divided into smaller parts. Then, each part is transformed to the data nodes.

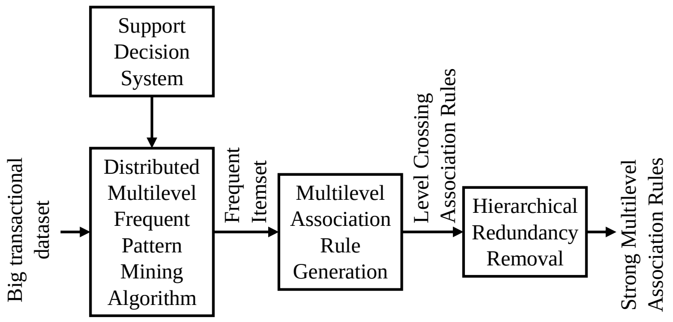
Support Decision System
Big transactional dataset
Distributed Multilevel Frequent Pattern Mining Algorithm
Frequent Itemset
Multilevel Association Rule Generation
Level Crossing Association Rules
Hierarchical Redundancy Removal
Strong Multilevel Association Rules

Fig.2. Proposed Methodology

3.1 DISTRIBUTED MULTILEVEL FREQUENT PATTERN MINING ALGORITHM

The distributed multilevel frequent pattern mining algorithms are categorized as MapReduce based Multilevel Apriori using Top-down Approach (MR-MLAT) and MapReduce based Multilevel Apriori using Bottom-up Approach (MR-MLAB). The proposed algorithms are improved versions of single level Apriori algorithm, where frequent itemsets are generated level wise as described in the following subsections.

3.1.1 MR-MLAT Algorithm:

Mapreduce based top-down approach produces huge frequent itemsets including level-crossing at different concept level. The Map function is executed on each data segment and it generates level crossing <*key*,*value*> pairs for each transaction of dataset. The MapReduce framework makes group of all <*key*,*value*> pairs with same items and also executes the Reducer function by passing the candidate itemsets. Map function generates local candidate itemsets in each database scan. Global counts are generated using Reduce function by adding individual local counts. For each level, the MapReduce function produces a frequent itemset including level-crossing at that specific level. The iteration continues until no further frequent itemsets are found for that level. The Reduce function sums up all the values of Map function and generates a count for the candidate itemset.

The distributed frequent pattern mining algorithm MR-MLAT [25] [26] shown in Fig.3, uses notation CT[l,*k*] as a set of candidate *k*-itemset at level l without frequency count, *C*[l,*k*] as a set of candidate *k*-itemset at level l with frequency count, and *L*[l,*k*] as a set of frequent *k*-itemset at level *l*. A dataset is given to Mapper in line by line manner. Each line of transaction is split into itemsets followed by items. Here, number of digits in itemset is represented by String_length function and first n number of digits is represented by Sub_string function. The map function generates <*itemset*,1> as <*key*,*value*> pairs with level-crossing. The MapReduce framework generates all <key, value> pairs with same items and also executes the reducer function by passing the list of values for candidate itemsets. The output of mapper is combined by the reducer function and finally, frequent itemsets are generated. This computation process will be stopped if the reduce task cannot generate larger candidate itemsets.

3.1.2 MR-MLAB Algorithm:

In the case of top-down approach, for each level of concept hierarchy, the database is repeatedly scanned for each frequent itemset. This is main drawback of top-down approach. As compared to top-down approach, bottom-up approach scans the dataset only once for each level. In the proposed algorithm, candidate and frequent 1-itemset is generated for last level. The candidate 1-itemset at level *l*-1 is derived from level *l* by considering first *l*-1 digits. Then, respective minimum support threshold is applied to find frequent itemset of level *l*-1 to 1. Once, frequent 1-itemset is generated for each level, frequent k-itemset is calculated level wise in bottom-up manner. The distributed frequent pattern mining algorithm MR-MLAB shown in Fig.4, uses notations *C*[l,*k*] as a set of candidate *k*-itemset without level-crossing at level l, CLC[l,*k*] as a set of candidate k-itemset with level-crossing only at level l, CF[l,*k*] as a set of all the candidate *k*-itemset including level-crossing at level l, and *L*[l,*k*] as a set of frequent *k*-itemset at level l.

The Map1 function is executed on each data segment of the last level in concept hierarchy to generate <*key*,*value*> pairs. The transactional data is given as an input to the Map1 function, line by line. Each line is split into itemset and generates the output <*key*,*value*> pair consisting of the candidate itemset and value 1. Here, value is local frequency of the itemset. The Reduce1 function combines the output of Map1 function and it generates the frequent itemset for that level. The Map2 function is used for finding the count of level-crossing itemset. The Map2 function takes input as <candidate itemset in the last level, value> pair and generates the output as a <candidate itemset at level *L*, count> pair. The String_length function returns number of digits in itemset. The Sub_string function gives first n digits from given itemset. The Reduce2 function adds up all the values of list having similar candidate itemset. It generates <*candidate itemset*, *support_count*> pair for the candidate itemset at that level. The illustration of this algorithm is given by following example.

Input: Database in HDFS containing encoded concept hierarchy information (*D*), Maximum level of concept hierarchy (*Max_level*), Minimum Support Threshold for each level *l* (*Min_sup* [*l*]).

Output: *L* [*l*], Level-crossing frequent itemsets for each level *l*.

Method:

```
Step 1: For each level $l$ in concept hierarchy do
Step 2: $L[l,1]$ = find frequent1-itemsets from ($D$).
Step 3: For each frequent k-itemset in level $l$ do
Step 4:  $CT[l, k] = L[l, k-1] \bowtie L[l, k-1]$. //Candidate itemset without level crossing
Step 5:  If ($l > 1$) then
Step 6:  For $j$=1 to $l$-1 do
Step 7:  $CT[l, k]$ + = $L[j, k-1] \bowtie L[l, k-1]$.   // Candidate itemset with level crossing
Step 8:  $C[l, k]$ = Map(); // Apply Map function on $CT[l, k]$ to generate the candidate itemsets
Step 9:  $L[l, k]$ = Reduce(); // Apply Reduce function on $C[l, k]$ to generate the frequent itemsets
Step 10: $L[l] = L[l] \, U_k \, L[l, k]$.
```

Map Function

Input: Transaction T_i

Output: <*candidate itemset*, *value*>

Method:

```
Step 11: For each transaction $T_i \in D$ do
Step 12: For each itemset $S_i$ in Candidate Itemset do
Step 13: For each item $I_i \in S_i$ do
Step 14: $n$ = String_length ($I_i$).  // String_length function returns number of digits
Step 15: If ($Sub\_string(I_i , n) \notin T_i$) then // Sub_string function gives first n digits
Step 16: Terminate the current itemset $S_i$.
Step 17: Generate the output <$S_i$,1> as <key, value> pair.
```

Reduce Function

Input: <*candidate itemset*, *list*>

Output: <*frequent itemset*, *support_count*>

Method:

```
Step 18: count = 0.
Step 19: For each number in list do
Step 20: count + = number.
Step 21: If (count > = $Min\_sup$) then
Step 22: Generate the output <frequent itemset, count> as <key, value> pair.
Step 23: End if
Step 24: End For
```

Fig.3 The MR-MLAT Algorithm

Input: Database in HDFS containing encoded concept hierarchy information (D),

Maximum level of concept hierarchy (*Max_level*),

Minimum Support Threshold for each level l (*Min_sup* [l]).

Output: $L[l]$, Level-crossing frequent itemsets for each level l

Method:

```
Step 1:  // Generation of frequent 1-itemset for each level $l$
Step 2:  For $l = Max\_level$ down to 1 do
Step 3:  If ($l = Max\_level$) then
Step 4:  $T[l,1]$ = Map1 (); // Apply Map1 function on all the items at level $Max\_level$.
Step 5:  Else
Step 6:  $T[l,1]$ = Sub_string ($C[Max\_level,1]$, $l$). // Sub_string function gives first $l$ digits
Step 7:  $C[l,1]$ = Reduce2 (); // Apply Reduce2 function on $T[l,1]$.
Step 8:  $L[l,1]$ = Reduce1 (); // Apply Reduce1 function on $T[l,1]$.
Step 9:  // Generation of frequent $k$-itemset for each level $l$
Step 10: For $l = Max\_level$ down to 1 do
Step 11: For each frequent $k$-itemset in level $l$ do
Step 12: If ($l = = Max\_level$) then
Step 13: $C1[l, k] = C[l, k-1] \bowtie C[l, k-1]$.
Step 14: $T[l, k]$ = Map1 (); // Apply Map1 function on $C1[l, k]$.
Step 15: $C[l, k]$ = Reduce2 (); // Apply Reduce2 function on $T[l, k]$
Step 16: Else
Step 17: $T1[l, k]$ = Sub_string ($C[Max\_level , k]$ , $l$).
Step 18: For each items $x$ in itemset do
Step 19: If any two items are not similar then
Step 20: $T2[l, k] = T1[l, k]$;
Step 21: $C[l, k]$ = Reduce2 (); // Apply Reduce2 function on $T2[l, k]$
Step 22: If ($l > 1$) then
Step 23: If ($k = = 2$) then
Step 24: For $j = l-1$ down to 1
Step 25: For each itemset $x$ in $C[j, k-1]$ do
Step 26: For each itemset $y$ in $C[l, k-1]$ do
Step 27: If $x$ is not an ancestor of $y$ then
Step 28: $C2[l, k]$ + = $C[l, k-1] \bowtie C[j, k-1]$.
Step 29: Else
Step 30: $C2[l, k] = C[l, k-1] \bowtie CLC[l, k-1]$.
Step 31: $CT[l, k]$ = Map2 (); // Apply Map2 function on $C2[l, k]$.
Step 32: $CLC[l, k]$ = Reduce2 (); // Apply Reduce2 function on $CT[l, k]$
Step 33: $CF[l, k] = C[l, k] + CLC[l, k]$.  // Candidate itemset including level crossing
Step 34: $L[l, k]$ = Reduce1 (); // Apply Reduce1 function on $CF[l, k]$.
Step 35: Else
Step 36: $L[l, k]$ = Reduce1 (); // Apply Reduce1 function on $C[l, k]$
Step 37: $L[l] = L[l] \, U_k \, L[l, k]$.
```

Map1 Function

Input: Transaction T_i

Output: <*candidate itemset*, *value*>

Method:

```
Step 38: For each transaction $T_i \in D$ do
Step 39: For each itemset $I_i$ in Candidate Itemset do
Step 40: If ($I_i \in T_i$) then
Step 41: Generate the output <$I_i$,1> as <key, value> pair.
```

Map2 Function

Input: <*candidate itemset in the last level* (C_{Max_level}), *value*>

```
Output: <candidate itemset at level L, count>
Method:
Step 42: For each candidate itemset in $C_{Max\_level}$ do
Step 43: For each itemset $S_i$ in Candidate Itemset do
Step 44: For each item $I_i \in S_i$ do
Step 45: n = String_length ($I_i$). // String_length function returns number of digits
Step 46: If (Sub_string ($I_i$,n) $\notin$ $C_{Max\_level}$) // Sub_string function gives first n digits
Step 47: Terminate the current itemset $S_i$.
Step 48: count = $C_{Max\_level}$.value.
Step 49: Generate the output <$S_i$, count> as <key, value> pair.
```

Reduce1 Function

```
Input: <candidate itemset, list>
Output: <frequent itemset, support_count>
Method:
Step 50: count = 0.
Step 51: For each number in list do
Step 52: count + = number.
Step 53: If (count> = Min_sup) then
Step 54: Generate the output <frequent itemset, count> as <key, value> pair.
```

Reduce2 Function

```
Input: <candidate itemset, list>
Output: <frequent itemset, support_count>
Method:
Step 55: count = 0.
Step 56: For each number in list do
Step 57: count + = number.
Step 58: Generate the output <frequent itemset, count> as <key, value> pair.
```

Fig.4. The MR-MLAB Algorithm

3.2 MOTIVATIONAL EXAMPLE

Consider the dataset containing 8 transactions with minimum support threshold at level 3, 2 and 1 is 1, 2 and 3 respectively (Table.1).

Table.1. Transaction Table

Transaction ID	Transactional Dataset
T_1	111,121, 211
T_2	111, 211, 222
T_3	122, 221
T_4	111,121
T_5	111,122,211,221
T_6	311, 411
T_7	113, 221, 231
T_8	112,131, 411

First of all, candidate 1-itemset of level 3, *C*[3,1] is calculated by scanning the database. To prune the infrequent items at this level, minimum support threshold is applied. The frequent 1-itemset, *L*[3,1] is generated for level 3. The candidate 1-itemset at level 2, *C*[2,1] is derived from *C*[3,1] by considering first two digits only. The minimum support threshold at level 2 is filter infrequent itemset and generates *L*[2,1]. Similarly, candidate 1-itemset at level 1, *C*[1,1] is derived from *C*[3,1] by considering first digit only.

The minimum support threshold at level 2 is filter infrequent itemset and generates *L*[1,1]. The result of candidate and frequent 1-itemset along with support count (SC) at all the level is shown in Fig.5.

3.2.1 L[3,2] - Level 3 Frequent 2-Itemset:

*C*1[3,2] is calculated by joining *C*[3,1] with *C*[3,1] which generates 2-itemset pair as {111,112}, {111,113}, {111,121}, {111,122}, {111,131}, {111, 211}, {111, 221}, {111, 222}, {111, 231}, {111, 311}, {111, 411}, and so on. The map1 function is applied on *C*1[3,2] which generates *T*[3,2]. The reduce2 function is applied on *T*[3,2] to generate *C*[3,2], candidate 2-itemset without level crossing only. The *C*2[3,2] is calculated by joining *C*[3,1] with *C*[2,1] and *C*[3,1] with *C*[1,1] after checking ancestor relationship among the items. The map2 function is applied on *C*2[3,2] which generates *CT*[3,2]. The reduce2 function is applied on *CT*[3,2] to generate *CLC*[3,2], candidate 2-itemset with level crossing only. Then, candidate 2-itemset including level crossing is generated by appending *C*[3,2] along with *CLC*[3,2] and result is stored in *CF*[3,2]. Finally, frequent 2-itemset is generated by applying reduce1 function on it.

3.2.2 L[3,3] - Level 3 Frequent 3-Itemset:

*C*1[3,3] is calculated by joining *C*[3,2] with *C*[3,2] which generates 3-itemset pair as {111,112,113}, {111,112,121}, and so on. The map1 function is applied on *C*1[3,3] which generates *T*[3,3]. The reduce2 function is applied on *T*[3,3] to generate *C*[3,3], candidate 2-itemset without level crossing only. The *C*2[3,3] is calculated by joining *C*[3,2] with *CLC*[3,2]. The map2 function is applied on *C*2[3,3] which generates *CT*[3,3]. The reduce2 function is applied on *CT*[3,3] to generate *CLC*[3,3], candidate 3-itemset with level crossing only. Then, candidate 3-itemset including level crossing is generated by appending *C*[3,3] along with *CLC*[3,3] and result is stored in *CF*[3,3]. Finally, frequent 3-itemset is generated by applying reduce1 function on it. Similarly, for level 3 frequent 4-itemset and 5-itemset is generated. Here, frequent 5-itemset is NULL.

3.2.3 L[2,2] - Level 2 Frequent 2-Itemset:

*T*1[2,2] is generated from *C*[3,2] by considering only first two digits of each itemset. The similar items in a itemset is filtered and result is stored in *T*2[2,2]. The 2-itemset pair is generated as {11*,12*}, {11*,13*}, and so on. The reduce2 function is applied on it to generate *C*[2,2], candidate 2-itemset without level crossing only. The *C*2[2,2] is calculated by joining *C*[2,1] with *C*[1,1] after checking ancestor relationship among the items. The map2 function is applied on *C*2[2,2] which generates *CT*[2,2]. The reduce2 function is applied on *CT*[2,2] to generate *CLC*[2,2], candidate 2-itemset with level crossing only. Then, candidate 2-itemset including level crossing is generated by appending *C*[2,2] along with *CLC*[2,2]. Finally, the frequent 2-itemset is generated by applying reduce1 function on it.

3.2.4 L[2,3] - Level 2 Frequent 3-Itemset:

*T*1[2,3] is generated from *C*[3,3] by considering only first two digits of each itemset. The similar items in an itemset are filtered and result is stored in *T*2[2,2]. The 3-itemset pair is generated as {11*,12*,13*}, and so on. The reduce2 function is applied on it to generate *C*[2,3], candidate 3-itemset without level crossing only. The *C*2[2,3] is calculated by joining *C*[2,2] with *CLC*[2,2]. The map2 function is applied on *C*2[2,3] which generates *CT*[2,3]. The reduce2 function is applied on *CT*[2,3] to generate *CLC*[2,3], candidate 3-itemset with level crossing only. Then, candidate 3-itemset including level crossing is generated by appending *C*[2,3] along with *CLC*[2,3] and result is stored in *CF*[2,3] file. Finally, frequent 3-itemset is generated by applying reduce1 function on it. Similarly, for level 2 frequent 4-itemset is generated. Here, frequent 4-itemset is NULL.

3.2.5 L[1,2] - Level 1 Frequent 2-Itemset:

*T*1[1,2] is generated from *C*[3,2] by considering only first digit of each itemset. The similar items in a itemset is filtered and result is stored in *T*2[1,2]. The 2-itemset pair is generated is {1**, 2**}, {1**, 3**}, and so on. The reduce2 function is applied on it to generate *C*[1,2], candidate 2-itemset without level crossing only. Since there is no ancestor of level 1, there do not exist level crossing itemset pair. Finally, frequent 2-itemset is generated by applying reduce1 function on it.

3.2.6 L[1,3] - Level 1 Frequent 3-Itemset:

*T*1[1,3] is generated from *C*[3,3] by considering only first digit of each itemset. The similar items in a itemset is filtered and result is stored in *T*2[1,3]. The 3-itemset pair is generated i.e. {1**, 2**, 3**}, {1**, 2**, 4**}, and so on. The reduce2 function is applied on it to generate *C*[1,3], candidate 3-itemset without level crossing only. Since there is no ancestor of level 1, there do not exist level crossing itemset pair. Finally, frequent 3-itemset is generated by applying reduce1 function. Here, frequent 3-itemset is NULL. Hence, the algorithm will terminate.

3.3 MULTILEVEL ASSOCIATION RULE GENERATION

The distributed multilevel frequent mining algorithm generates frequent itemsets for each level with level crossing frequent itemset. Multilevel association rules are generated based on following steps [27,28].

1) For all the level of concept hierarchy,
   a) Generate all non-empty subsets of f for each level-crossing frequent k-itemset f.
   b) Generates the multilevel association rule as $s \rightarrow (f - s)$ such that (*Support*(*f*) / *Support* (*s*)) ≥ *min_conf*, where, *min_conf* is the minimum confidence threshold at that level repeat this for every non-empty subset *s* of *f*.

***C*[3,1]**

Item	Support Count
111	4
112	1
113	1
121	2
122	2
131	1
211	3
221	3
222	1
231	1
311	1
411	2

***L*[3,1]**

Item	Support Count
111	4
112	1
113	1
121	2
122	2
131	1
211	3
221	3
222	1
231	1
311	1
411	2

***C*[2,1]**

Item	Support Count
11*	6
12*	4
13*	1
21*	3
22*	4
23*	1
31*	1
41*	2

***L*[2,1]**

Item	Support Count
11*	6
12*	4
21*	3
22*	4
41*	2

***C*[1,1]**

Item	Support Count
1**	11
2**	8
3**	1
4**	2

***L*[1,1]**

Item	Support Count
1**	11
2**	8

Fig.5. Candidate and Frequent 1-itemset for level 3, 2 and 1

3.4 ELIMINATING HIERARCHICAL REDUNDANT RULES

Due to "ancestor" relationships among items, even though multilevel association rules are generated with high confidence threshold, then also, large numbers of redundant rules are generated. The processing of such redundant rules can take a long time, so the performance is degraded. To overcome this problem, hierarchical redundant rules are eliminated. The primary idea for the redundancy elimination is to improve the quality as well as importance of the rules without loss of any information [28]. In the case of top-down approach, candidate itemset of current iteration is calculated based on join operation on frequent itemset of previous iteration; while in the case of bottom-up approach, candidate itemset of current iteration is calculated based on join operation on candidate itemset of previous iteration. Thus, the top-down approach excludes some of the hierarchical interesting association rules while bottom-up approach will not. For example, Suppose the hierarchical dataset have three levels L_1, L_2 and L_3. Minimum support threshold for each level is 5, 4 and 3 respectively i.e. $MS_{L1} = 5$, $MS_{L2} = 4$ and $MS_{L3} = 3$. Suppose the item 1**, 22* and 333 have support count values 4, 3 and 3 respectively. In the case of top-down approach,1** and 22* are not frequent so while calculating candidate 2-itemset; 1** and 22* will never be considered. Hence, for all the rules containing 1** and/or 22* as a subset of consequent and/or antecedent; will never be obtained. While in the case of bottom-up approach, even if 1** and 22* are not frequent 1-itemset then also they are considered while calculating candidate 2-itemset.

4. EXPERIMENTAL SETUP AND RESULTS

For the experimental purpose, a cluster of four computers with i5 processor and 4GB DDR-3 RAM are used. The proposed algorithms are tested on AMUL dairy sales data containing more than 1500 different dairy products. For this experiment, the dataset of last two years having size 5GB is used and it can be extended further. The distributed multilevel frequent pattern mining algorithms are applied only on concept hierarchical dataset. So, the preprocessing is useful to transform sales dataset into hierarchical form.

4.1 GENERATION OF MULTILEVEL FREQUENT PATTERN

The transactional big dataset is given as input to the TMLA, MR-MLAT and MR-MLAB algorithms to generate frequent itemsets without level-crossing and including level-crossing.

4.1.1 Multilevel Frequent Pattern Mining without Level Crossing:

Minimum support threshold needs to be adjusted properly to reduce missing of interesting multilevel association rules. Thus, minimum support threshold is adjusted such that it decreases from higher level to lower level of the hierarchy tree. The execution time for the TMLA, MR-MLAT and MR-MLAB algorithms applied on the 5GB dataset distributed over single node clusters for reducing minimum support threshold level wise is shown in Fig.6. The level wise minimum support threshold 4-3-2-1 indicates min_sup of 4% for level 1, 3% for level 2, 2% for level 3 and 1% for level 4, respectively. For this experiment, 8 mappers and 3 reducers are used. It is observed that the execution time of the proposed MR-MLAB and MR-MLAT algorithms is significantly lower as compared to TMLA algorithm.

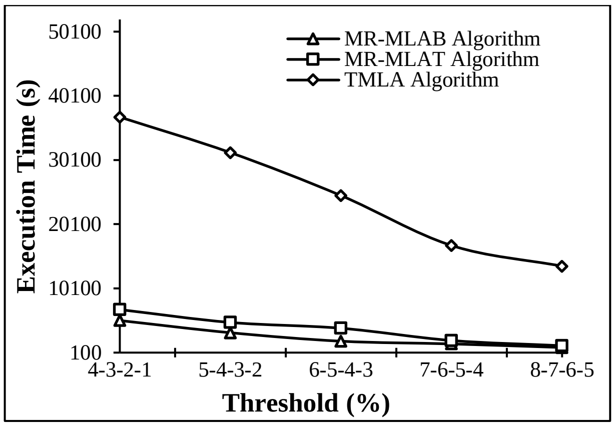

Fig.6. Level-wise Minimum Support Threshold vs. Execution Time

4.1.2 Multilevel Frequent Pattern Mining including Level Crossing:

The minimum support threshold is considered similar for all the levels of concept hierarchy for generating multilevel frequent itemsets. For this experiment, 8 mappers and 3 reducers are used. The results of TMLA, MR-MLAT and MR-MLAB algorithms on AMUL datasets for the varying database size 256MB, 512MB,1GB, 2GB and 5GB is applied on single node cluster with minimum support threshold of 1% is shown in Fig.7. For a data set of size 5GB that was distributed on single node, the execution time for the TMLA, MR-MLAT and MR-MLAB algorithms are 68000 seconds, 4800 seconds and 3120 seconds respectively. The experiment shows that the execution time of proposed algorithms is less as compared to the traditional approach. The performance of proposed algorithms is improved compared to TMLA algorithm for large dataset.

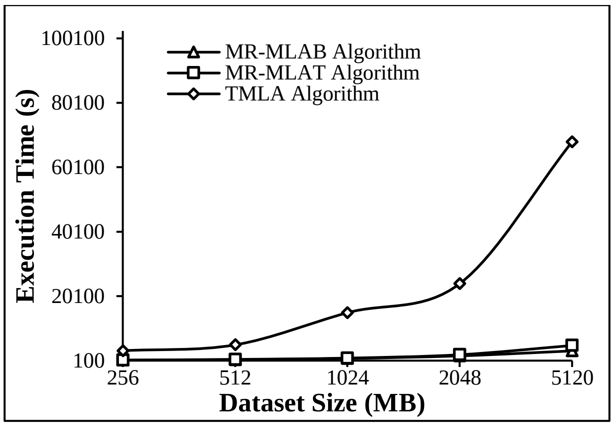

Fig.7. Dataset Size vs. Execution Time for Single Node Cluster

Similarly, the results of MR-MLAT and MR-MLAB algorithms on AMUL datasets for the varying database size 256MB, 512MB,1GB, 2GB and 5GB is applied on multi-node clusters of two nodes and three nodes with minimum support threshold of 1% are shown in Fig.8 and Fig.9, respectively. For the small dataset size, the execution time of both the algorithms is almost closer but when the size of dataset increases, the execution time of bottom-up approach is considerably less as compared to the top-down approach. It can be observed from the experimental results that the performance of the algorithm depends on the number of nodes and the size of dataset. So, in order to improve the time efficiency, the number of nodes must be increased with an increased database size. When the number of processors is increases in private hadoop cluster then the privacy is not reduced necessarily since we would be controlling all of the nodes. The privacy may be loss up to some extend if a MapReduce job is deployed on a public cloud.

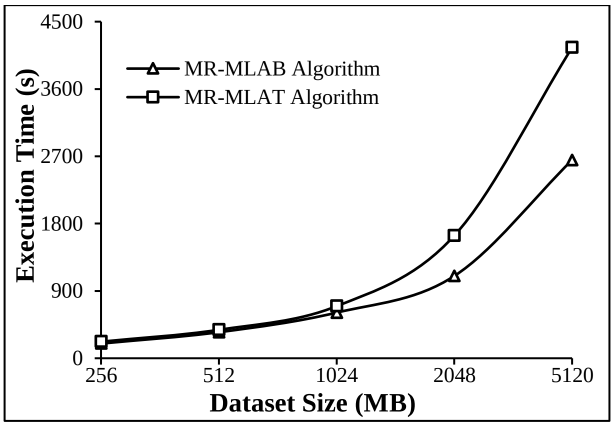

Fig.8 Dataset Size vs. Execution Time for Two Node Cluster

The results of MR-MLAT and MR-MLAB algorithms on 5GB AMUL datasets is applied on single node, two node and three node cluster with minimum support threshold of 1% is shown in

Fig.10. The experimental results shows that the execution time of MR-MLAT algorithm on single node, two node and three node cluster is 4800 seconds, 3891 seconds and 2418 seconds, respectively. Thus, the speedup of two node cluster with respect to single node is 1.234 times and three node cluster with respect to single node is 1.985 times. Similarly, the execution time of MR-MLAB algorithm on single node, two node and three node cluster are 3120 seconds, 2489 seconds and 1565 seconds, respectively. Thus, the speedup of two node cluster with respect to single node is 1.254 times and three node cluster with respect to single node is 1.994 times.

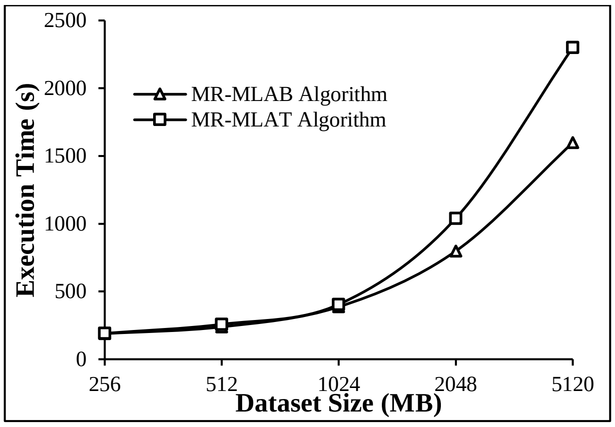

Fig.9. Dataset Size vs. Execution Time for Three Node Cluster

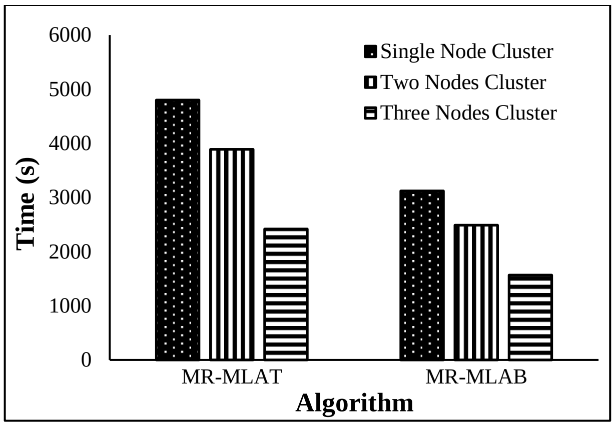

Fig.10. Time Efficiency Comparison based on Number of Nodes in a Cluster

The results of TMLA, MR-MLAT and MR-MLAB algorithms on 5GB AMUL dataset for the varying minimum support threshold of 1% to 5% is applied on single node cluster as shown in Fig.11. The experiment shows that the execution time of proposed algorithms is significantly improved compared to traditional approach.

4.2 HIERARCHICAL INTERESTING ASSOCIATION RULE GENERATION

After finding level-crossing frequent k-itemset for each level, hierarchical redundant association rules are removed from it. For this experiment, the number of hierarchical redundant association rules is calculated for minimum confidence threshold, which varies from 40% to 90% and minimum support threshold varies from 1% to 5%, as shown in Fig.12. For this experiment,12 mappers and 3 reducers are used. It can be observed from the graph that a smaller number of hierarchically redundant rules is generated for the minimum confidence threshold $\geq$ 70% and minimum support threshold $\geq$3%. The hierarchical association rules which are generated using MR-MLAB algorithm, MR-MLAT algorithm and TMLA algorithm for the minimum support threshold of 1% and minimum confidence threshold varying from 40% to 90% are shown in Fig.13. It can be observed that MR-MLAB algorithm generates additional hierarchical interesting association rules as compare to other two algorithms.

4.3 COMPARATIVE ANALYSIS OF MULTILEVEL FREQUENT PATTERN MINING ALGORITHMS

The proposed MR-MLAT and MR-MLAB algorithms are compared with the TMLA algorithm [9] and also with the algorithm proposed by Chandanan and Shukla [22], based on various parameters. For this experiment, multilevel frequent pattern mining algorithms are compared based on approach, efficiency, number of database scans, effectiveness of algorithm, dependent on levels of concept hierarchy, database scalability and removal of hierarchical redundancy as shown in Table.2.

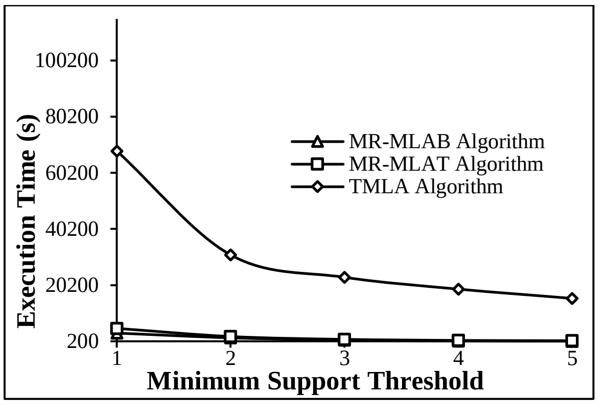

Fig.11. Minimum Support Threshold vs. Execution Time (s)

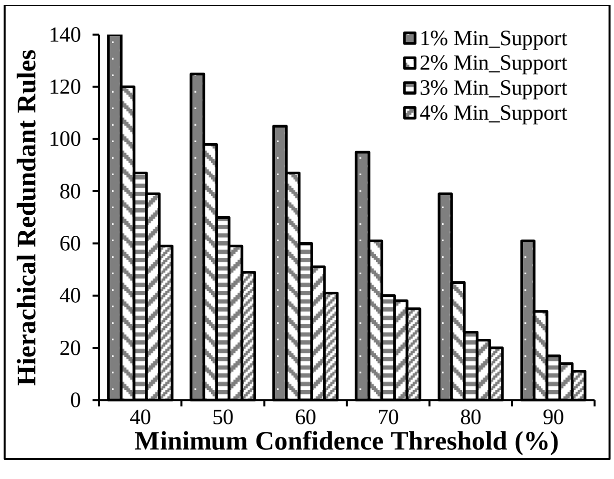

Fig.12. Minimum Confidence Threshold vs. Hierarchical Redundant Rules

The experimental results shows that multilevel distributed frequent pattern mining using top-down and bottom-up approach is more efficient as compare to other two algorithms.

Table.2. Comparative Analysis of Apriori based Multilevel Frequent Pattern Mining Algorithms

Parameters	**TMLA [9]**	**Chandanan and Shukla [22]**	**MR-MLAT**	**MR-MLAB**
Approach	Top-down	Top-down	Top-down	Bottom-up
Time Complexity	O(l^2*k)	NA	O (l^2*k)	O(l*k)
Database Scans	$\sum_{i=1}^{l}\sum_{j=1}^{k} L[j,i]$	NA	$\sum_{i=1}^{l}\sum_{j=1}^{k} L[j,i]$	$\sum_{j=1}^{k} L_{\max\_level}[j]$
Exclusion of Hierarchical Interesting Association Rules	Yes	Yes	Yes	No
Dependency on Levels of Concept Hierarchy	High	High	High	Low
Database Scalability	No	No	Yes	Yes
Removal of Hierarchical Redundancy	No	Yes	Yes	Yes

The experimental results also show that top-down approach excludes some of the hierarchical interesting association rules as compared to bottom-up approach. Hence, the MapReduce based bottom-up approach is much better compare to other three algorithms. In Table.2, $L[j,i]$ indicates frequent j-itemset at level i, L[j] indicates frequent j-itemset at last level and NA indicates not available.

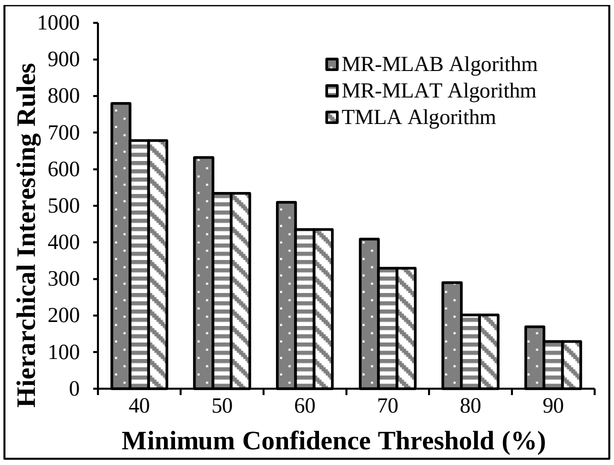

Fig.13. Minimum Confidence Threshold vs. Hierarchical Interesting Rules

5. CONCLUSIONS AND FUTURE SCOPE

Traditional multilevel association rule mining algorithms are not effectively used for analysis of huge data. HDFS and MapReduce based algorithm plays a vital role for analysis of such data. In this paper, MapReduce based distributed frequent pattern mining approach is presented to process the transactional dataset. MR-MLAB and MR-MLAT based on top-down and bottom-up approaches respectively are proposed to mine multilevel association rules for the same level as well as on different level of concept hierarchy. The experimental results show that the distributed frequent pattern mining algorithms are linearly scalable with respect to number of nodes in cluster and size of the dataset. The experimental results also show that even though size of dataset increases, the proposed algorithms generate less candidate itemset and uses less message passing. Hence, the execution time of the proposed algorithms is relatively lower. It was observed that in order to reduce execution time, the number of nodes must increase with an increase in database size. For higher value of minimum confidence threshold and minimum support threshold, numbers of hierarchically redundant rules are relatively less. The experimental results also show that TMLA and MR-MLAT algorithm excludes some of the hierarchical interesting association rules as compare to MR-MLAB algorithm. The proposed algorithms are more flexible, scalable and efficient distributed multilevel frequent pattern mining algorithm for mining big data. The given approach is much more helpful in analysis of big data and support decision system. The time efficiency of the proposed algorithms can be yet improved by using FP-tree based data structures for the candidate itemset generation.

REFERENCES

[1] K. Srikumar and B. Bhasker, "Metamorphosis: Mining Maximal Frequent Sets in Dense Domains", *International Journal of Artificial Intelligence Tools*, Vol. 14, No. 3, pp. 491-506, 2005.

[2] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules between Sets of Items in Large Databases", *Proceedings of International Conference on ACM-SIGMOD on Management of Data*, pp. 207-216, 1993.

[3] J. Woo, S. Basopia and S.H. Kim, "Market Basket Analysis Algorithm with NoSQL DB HBase and Hadoop", *Proceedings of International Conference on Emerging Databases*, pp. 56-62, 2011.

[4] J. Woo, S. Basopia and S.H. Kim, "Market Basket Analysis Algorithm with MapReduce of Cloud Computing", *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, pp. 1-13, 2011.

[5] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows and R.E. Gruber, "Bigtable: A Distributed Storage System for Structured Data", *ACM Transactions on Computer Systems*, Vol. 26, No. 2, pp. 1-14, 2008.

[6] Apache Hadoop, Available at http://hadoop.apache.org/, Accessed at 2015.

[7] J.H.C. Yeung, C.C. Tsang, K.H. Tsoi, B. Kwan, C. Cheung, A.P.C. Chan and P.H.W. Leong, "Map-Reduce as a Programming Model for Custom Computing Machines",

Proceedings of International Conference on Field-Programmable Custom Computing Machines, pp. 1-13, 2008.

[8] R.A. Angryk and F.E. Petry, “Mining Multi-Level Associations with Fuzzy Hierarchies”, *Proceedings of International Conference on Fuzzy System*, pp. 785-790, 2005.

[9] R.S. Thakur, R.C. Jain and K.R. Pardasani, “Mining Level-Crossing Association Rules from Large Databases”, *Journal of Computer Science*, Vol. 2, No. 1, pp. 76-81, 2006.

[10] J. Han and Y. Fu, “Mining Multiple-Level Association Rules in Large Databases”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 5, pp. 1-8, 1999.

[11] G. Shaw, Y. Xu and S. Geva, “Eliminating Redundant Association Rules in Multilevel Datasets”, *Proceedings of International Conference on Data Mining*, pp. 14-17, 2008.

[12] Y. Xu, G. Shaw and Y. Li, “Concise Representations for Association Rules in Multilevel Datasets”, *Journal of Systems Science and Systems Engineering*, Vol. 23, No. 1, pp. 53-70, 2009.

[13] T. Hong, T. Huang and C. Chang, “Mining Multiple-level Association Rules Based on Pre-large Concepts”, *Data Mining and Knowledge Discovery in Real Life Applications*, pp. 187-200, 2009.

[14] P. Gautam and K. R. Pardasani, “A Fast Algorithm for Mining Multilevel Association Rule Based on Boolean Matrix”, *International Journal on Computer Science and Engineering*, Vol. 2, No. 3, pp. 746-752, 2010.

[15] S. Prakash, M. Vijayakumar, R.M.S. Parvathi, “A Novel Method of Mining Association Rule with Multilevel Concept Hierarchy”, *International Journal of Computer Applications*, Vol. 12, No. 1, pp. 26-29, 2011.

[16] P. Gautam and K.R. Pardasani, “Efficient Method for Multiple-Level Association Rules in Large Databases”, *Journal of Emerging Trends in Computing and Information Sciences*, Vol. 2, No. 12, pp. 722-732, 2011.

[17] S. Srivastava, H.K. Verma and D. Gupta, “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, pp. 234-238, 2011.

[18] P. Gautam and R. Shukla, “An Efficient Algorithm for Mining Multilevel Association Rule Based on Pincer Search”, *International Journal of Computer Science Issues*, Vol. 9, No. 4, pp. 235-241, 2012.

[19] M.R. Karim, C.F. Ahmed, B. Jeong and H. Choi, “An Efficient Distributed Programming Model for Mining Useful Patterns in Big Datasets”, *IETE Technical Review*, Vol. 30, No. 1, pp. 53-63, 2013.

[20] H. Zhuang and G. Wang, “Mining Multiple Level Association Rules under Weighted Concise Support Framework”, *Computer Modelling and New Technologies*, Vol. 18, No. 11, pp. 394-400, 2014.

[21] A.K. Chandanan and M.K. Shukla, “Removal of Duplicate Rules for Association Rule Mining from Multilevel Dataset”, *Proceedings of International Conference on Advanced Computing Technologies and Applications*, pp. 143-149, 2015.

[22] N. Pumjun and W. Kreesuradej, “Incremental Multilevel Association Rule Mining of a Dynamic Database Under a Change of a Minimum Support Threshold”, *Advanced Multimedia and Ubiquitous Engineering*, Vol. 34, pp. 87-94, 2016.

[23] U. Muhammad and M. Usman, “Multi-Level Mining and Visualization of Informative Association Rules”, *Journal of Information Sciences and Engineering*, Vol. 32, pp. 1061-1078, 2016.

[24] D.J. Prajapati and S. Garg, “MapReduce Based Multilevel Association Rule Mining from Concept Hierarchical Sales Data”, *Proceedings of International Conference on Advances in Computing and Data Sciences*, pp. 624-636, 2017.

[25] D.J. Prajapati, S. Garg and N.C. Chauhan, “MapReduce based Multilevel Consistent and Inconsistent Association Rule Detection from Big Data Using Interestingness Measures”, *Big Data Research*, Vol. 9, pp. 18-27, 2017.

[26] T. Ban, M. Eto, S. Guo, D. Inoue, K. Nakao and R. Huang, “A Study on Association Rule Mining of Darknet Big Data”, *Proceedings of International Conference on Neural Network*, pp. 1-7, 2015.

[27] J. Han and M. Kamber, “*Data Mining Concepts and Techniques*”, Morgan Kaufmann Publishers, 2004.