Clustering of Association Rules for Big Datasets using Hadoop MapReduce

Salahadin A. Moahmmed[1], Mohamed A. Alasow[2], El-Sayed M. El-Alfy[3]
Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

***Abstract*—Mining association rules is essential in the discovery of knowledge hidden in datasets. There are many efficient association rule mining algorithms. However, they may suffer from generating large number of rules when applied to big datasets. Large number of rules makes knowledge discovery a daunting task because too many rules are difficult to understand, interpret or visualize. To reduce the number of discovered rules, researchers proposed approaches, such as rules pruning, summarizing, or clustering. For the flourishing field of big data and Internet-of-Things (IoT), more effective solutions are crucial to cope with the rapid evolution of data. In this paper, we are proposing a novel parallel association rule clustering approach which is based on Hadoop MapReduce. We ran many experiments to study the performance of the proposed approach, and promising results have been demonstrated, e.g. the lowest scaleup was 77%.**

***Keywords*—Internet of Things; big data mining; clustering; association rules; Hadoop**

I. Introduction

Big data, business intelligence and Internet of Things (IoT) are among the fastest growing areas shaping the future, that attracted increasing attention of researchers and developers in the recent years. Many use cases have been envisioned to improve life quality and productivity through bridging the gap between physical and digital worlds, e.g. smart cities, smart homes, smart grids, smart meters, smart healthcare devices, wearable devices, smart poultry and animal farming, smart agriculture, and connected cars. As more physical entities are connected, an increasing volume of operational and management data is generated, making data mining and business analytics more crucial to turn this ocean of data into actionable insights generating values [1–5].

A branch of data mining and analytics is extracting association rules to discover patterns and regularities of frequent items in a dataset [6]. It was initially applied for transactional data analysis in point-of-sale systems in supermarkets to find items frequently purchased together (same basket) and predict the purchase of some items based on the observed frequency of others. A typical association rule takes the form $\alpha \rightarrow \beta$, where both α and β are itemsets and $\alpha \cap \beta = \emptyset$. α is known as the rule antecedent and β is the rule consequent. Each rule has a support which is the same as the support of $\alpha \cup \beta$, i.e. the number (or percentage) of transactions containing $\alpha \cup \beta$. Each rule has also a confidence level expressing how likely a person purchasing α will simultaneously purchase β, i.e. the conditional probability P($\beta|\alpha$). For example, an association rule can be {coffee, sugar} $\rightarrow$ {tea} with 75% confidence. Later its application has grown to many other domains. For example in healthcare, a rule can take various forms, such as {Medicine} $\rightarrow$ {Medicine}, {Disease} $\rightarrow$ {Disease}, {Medicine} $\rightarrow$ {Disease}, {Symptoms} $\rightarrow$ {Disease}, etc. [7]. Association rules and frequent-pattern mining have become an attractive area of research to support decision making in a wide spectrum of applications such as information security [8], health informatics [9], airline information systems [10], social networks [11], and several others [12, 13]. Several algorithms have been proposed on different areas in the literature [14]. However, many approaches can suffer from the massive number of discovered rules or spurious relations even for a moderately sized dataset. This limitation can lead to further problems in decision making attempting to visualize or interpret these rules; reducing their utility in decision support [15]. Some efforts have been made in the literature to address this problem in different directions such as rule grouping or clustering, rules pruning, meta-rules and constraint based mining [14, 16–20].

Nowadays, we are in the era of Internet-of-Things (IoT) where huge amount of data is generated by commercial, industrial and consumer IoT devices. However, along with the proliferation of IoT technology, cyberattacks that are exploiting the vulnerabilities of these new systems are becoming very challenging. Also labeling and discovering installed IoT devices is becoming very difficult. Association rules discovered from data generated from IoT devices are essential in securing, labeling, and discovering IoT devices [1, 21–24]. The problem is with the huge number of association rules that are discovered from IoT data. Existing solutions of reducing association rules are limited to traditional datasets.

In this paper, we are proposing a Hadoop MapReduce based algorithm that clusters rules discovered from big datasets. The proposed approach is composed of two phases. In the first phase, it prunes rules based on their structure, and in the second phase, it clusters the rules that were not pruned in the previous phase. The remainder of the paper is structured as follows. Related work is briefly reviewed in Section II. The proposed approach is presented in Section III and experimental evaluation is discussed in Section IV. The paper concludes in Section V with highlights of the paper findings and some future research work.

II. Related Work

Researchers proposed several algorithms in the literature to extract frequent itemsets and association rules in various domains, e.g. [9, 25, 26]. Among the major problems in big data is scalability of existing approaches leading to large number of association rules generated which hinders their interpretations and consuming huge computational resources.

Several studies proposed algorithms to reduce the number of association rules. For instance, in [14] the authors presented two methods to remove redundant rules based on domain knowledge. The first one prunes rules by grouping them based on user-defined semantics, and the second one groups rules based on common items.

In [27], the researchers proposed pruning rules by using the idea of domain ontology, which enables association rules to generalize in the form of is-a hierarchy. They integrated that with user knowledge pertaining to data, as a post-processing step, to select more interesting rules. To identify and then remove redundant rules, Torvonin et al. [18] utilized a rule cover method and then Brijs et al. [28] used integer programming to maximize the redundancy reduction. However, the success of these techniques depends on domain knowledge of users to eliminate uninteresting rules. The algorithm proposed in [29] goes through two phases. In Phase 1, it puts rules with the same consequences in the same group, and in Phase 2 prunes rules from each group that has minimum effect on the group cover.

Another direction uses various subjective and objective measures to identify interesting rules to keep. The study in [30] used chi-square statistical test to evaluate the dependence of rule antecedent and consequent. To make pruning of the rules, a pre-specified threshold value is used. However, this method may fail to prune many rules due to data sparsity. In [31], an idea based on minimum improvement constraint is presented to perform pruning by measuring the confidence difference between a rule and its proper sub-rules. However, the selection of threshold value is critical and lower values can lead to missing many overlapping rules. Contrast sets containing the conjunction of meaningfully different attributes and values was employed in [32]. In 2005, a search algorithm, known as OPUS (Optimized Pruning for Unordered Search-spaces) [33], was used in [34] to anatomically discard insignificant rules.

In [35], another approach is proposed that goes through two phases. In the first phase, clusters of association rules are created using a version of k-means algorithm called Kmeans-Rules; and in the second phase meta-rules are extracted from each cluster using two algorithms, namely BSO-MR and HBSO-TS-MR. BSO-MR uses bees swarm optimization and HBSO-TS-MR uses tabu search. The meta rules select representative rules and prune the rest. In [36], the authors propose pruning rules using a method called dual scaling to provide semantic contextualization. The method first groups the rules using an algorithm called AKMS and then prunes rules from the groups that have certain number of items to reduce data dimensionality.

An adaptive local pruning graphical method is described by Chawla et al. [37]. The authors defined an association rule network as a weighted B-graph and presented an algorithm to generate it. From a set of association rules with a singleton in the consequent as a goal item. Moreover, they presented an algorithm for rule pruning by removing hypercycles and reverse hyperedges in the B-graph. Visualization based techniques such as parallel coordinate plots [38], matrix-based visualizations [39] are introduced as post-processing techniques to analyze the discovered association rules. These techniques help visualize the interrelations between association rule categories in a great detail. Unfortunately, most visualization techniques cannot display large sets of rules.

Another methodology applies classification or clustering approaches to reduce the number of discovered association rules. For example, Liu et al. [40] proposed a framework integrating classification with association rule mining to focus on a subset of association rules. This approach is known as Classification Based on Associations (CBA) and is composed of two parts: a rule generator (CBA-RG) and a classifier builder (CBA-CB). The first part, CBA-RG, is based on apriori algorithm to discover association rules whereas the second part, CBA-CB, is a heuristic to select the best rule subset. Other approaches used post processing with agglomerative hierarchical clustering to produce more compact set of association rules [41, 42]. Recently, Bui-Thi et al. proposed another approach based on the idea of mining unexpected patterns to automatically detect beliefs and outliers [43].

All the above mentioned solutions are limited to traditional datasets and cannot handle data generated by billions of IoT devices. Pruning or clustering association rules generated from big data is essential for many IoT applications [1, 24]. For example, IoT devices will pose substantial security challenges, some of which are device vulnerabilities, misconfiguration and mismanagement [21]. Also a wide variety of IoT devices are getting connected to residential networks everyday. But most residents lack the knowledge of how to protect their devices from security threats [22, 23]. Another problem is labeling and discovering of IoT devices which is now done manually. This is impractical with the rate at which the IoT devices are getting installed. Association rules can reduce the security issues of IoT related services and they can be used to automate labeling and discovering IoT devices [21–23].

III. Proposed Methodology

The layout for association rule mining is illustrated in Fig. 1. In this study, we extended the work in[20]. After data acquisition and preprocessing, association rule mining is conducted. The proposed framework is composed of four MapReduce algorithms. First, PPrune [20] is applied to reduce the number of ARs based on their structure. Afterwards, Create-ACM, Compute-lift, and Cluster-SAR cluster the association rules that were not pruned.

A. PPrune: Rule Structure based Pruning Algorithm

PPrune reduces the number of association rules based on the structure of the rules. The concept of structural rule cover is presented in [18] and is utilized in PPrune to focus on most general rules of the original set of rules. PPrune is implemented for Hadoop MapReduce.

Algorithm 1 shows the PPrune Mapper which works as follows. For a given set of rules R, the Map method of PPrune reads each $r \in R$ and identifies its antecedent, and consequent, r.antecedent and r.consequent (lines 3 and 4). It then sorts r.antecedent and computes its size, r.antecedent.size, which is the number of items in r.antecedent (lines 5 and 6). Finally, it emits a tuple (key, value) where key = r.consequent and value = r and sends it to the reducer (Line 9).

Algorithm 2 shows the Reduce method of PPrune. This method takes the output of the PPrune Mapper in the form of

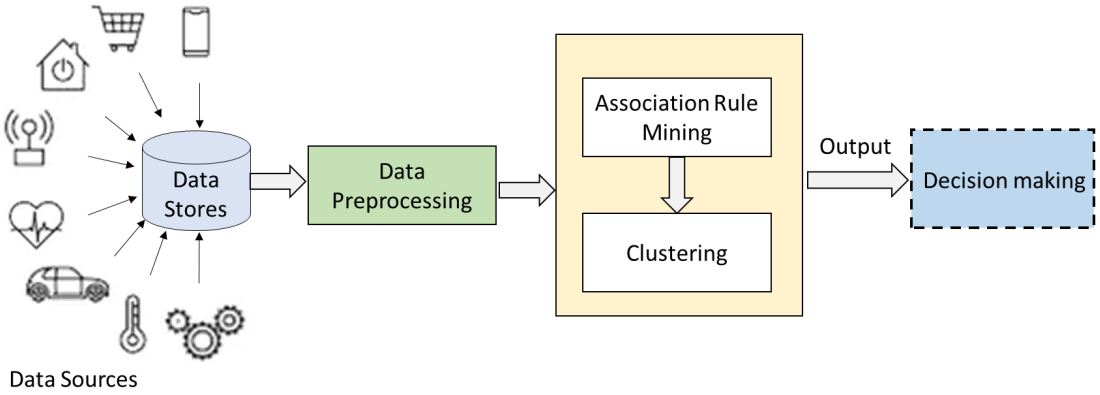

Fig. 1. Layout of Association Rule Mining.

Algorithm 1 PPrune Mapper

```
1:  Class MAPPER
2:      Method MAP(Key offset, Rule r)
3:          r.antecedent ← get-antecedent(r)
4:          r.consequent ← get-consequent(r)
5:          r.antecedent.size ← get-size(r.antecedent)
6:          r.antecedent ← sort(r.antecedent)
7:          k ← r.consequent
8:          v ← r
9:          Emit(k, v)
10:     End Method
11: End Class
```

Algorithm 2 PPrune Reducer

```
1:  Class REDUCER
2:      Method REDUCE(Key k, Value v)
3:          for each r ∈ v do
4:              s = r.antecedent.size
5:              ra = r.antecedent
6:              S[s] ← ra
7:          end for
8:          S ← Re-index(S)
9:          G ← ∅
10:         for i = 1; i ≤ |S|; i++ do
11:             for each r ∈ S[i] do
12:                 if i == 1 then
13:                     G ← G ∪ r
14:                 else if ! Cover(G, r) then
15:                     G ← G ∪ r
16:                 end if
17:             end for
18:         end for
19:         Emit(k, G)
20:     End Method
21: End Class
```

(k, v) where the key k is rule consequent and the value v is a set of strong association rules having the same consequent. The Reduce method then generates a set of general rules G as output. It passes through two phases. First, it groups each r in v that has the same r.consequent and r.antecedent.size and forms a sub-partition, S[r.antecedent.size] (lines 3 to 7 in the pseudo code). After it puts each rule in one of the sub-partitions of S, it deletes the empty sub-partitions of S by resizing and re-indexing S (Line 8). In the second phase, reducer prunes covered rules. At Line 9, it initializes the set of general rules G to null. In following lines 10 to 18, the reducer loops through each sub-partition of S. For each rule in a sub-partition. if a rule belongs to the first sub-partition, $S[1]$, or is not a superset of any of the rules in G, then it adds the rule to G. Otherwise, the rule is pruned. Finally, the reducer emits the key k and the value G (Line 19).

B. Lift-based Rules Clustering Approach

This approach consists of the remaining three MapReduce algorithms introduced in the previous section, namely, Create-ACM, Compute lift, and Cluster-SAR. The number of ARs is further reduced by clustering ARs which were not pruned by PPrune. The clustering is based on an interest measure of AR known as lift [44], which is defined for rule $A \rightarrow C$ as,

$$lift(A \rightarrow C) = \frac{support(A \cup C)}{support(A)support(C)} \quad (1)$$

It measures the correlation between the rule antecedent and consequent. A lift of value one indicates no correlation between the antecedent and consequent (i.e. independent); and a value much higher than one shows strong positive correlation. The proposed clustering approach is based on the assumption that rules with antecedents that are highly correlated with the same set of consequents are similar and thus should be clustered together [44]. Unlike the existing clustering approaches, this approach clusters antecedents containing itemsets which are rarely occurring together.

To perform the clustering efficiently, we created a 2-dimensional array (a matrix), M. The size of M is $|A|$ by $|C|$, where $|A|$ is the number of distinct antecedents and $|C|$ is the number of distinct consequents of all the strong association rules that are going to be clustered. The element $m_{i,j}$ of M contains the lift value of the rule $A_i \rightarrow C_j$, where A_i is the antecedent which corresponds to the i^{th} row of M and C_j is consequent which corresponds to the j^{th} column of M. An

element of M which does not belong to a strong association rule is assigned to a lift value of 1.

The distance between antecedents A_i and A_j is defined as:

$$dist(A_i,A_j) = \sqrt{\sum_{k=1}^{|C|} \left| m_{i,k} - m_{j,k} \right|} \quad (2)$$

To cluster antecedents into a set of K groups, namely $G = G_1, G_2, \ldots, G_K$, we use K-means algorithm and minimize the within-cluster sum of squares, $\sum_{i=1}^{K} \sum_{A_j \in G_i} dist(A_j, \mu_i)$, where μ_i is the centroid of G_i.

1) Create-ACM mapreduce algorithm: This algorithm takes as input strong association rules and initializes two 1-dimensional arrays called RHS and LHS, and a 2-dimensional array called ACM. LHS is indexed by the distinct antecedents of the strong association rules whereas RHS is indexed by the distinct consequents of the strong association rules. Let $A = \{A_1, A_2, \ldots A_{|A|}\}$ be the set of all distinct antecedents and $C = \{C_1, C_2, \ldots C_{|C|}\}$ be the set of all distinct consequents in the strong association rules, where $|A|$ is the number of distinct antecedents and $|C|$ the number of distinct consequents. The size of ACM is $|A| \times |C|$. Its rows are indexed by the members of A and its columns are indexed by the members of C. Let $m_{i,j}$ be an element of ACM indexed by A_i and C_j and corresponds to the rule $A_i \rightarrow C_j$.

Algorithm 3 Create-ACM Mapper

```
1:  Class MAPPER
2:      Global: LHS, RHS
3:      Method MAP(Key offset, Rule r)
4:          k ← get-antecedent(r)
5:          v ← get-consequent(r)
6:          LHS ← AddLHS(k)
7:          RHS ← AddRHS(v)
8:          Emit(k, v)
9:      End Method
10: End Class
```

The Create-ACM Mapper extracts the antecedent and the consequent of each rule and emits them to the reducer function. It also initializes two global arrays called LHS and RHS. The size of LHS is $|A|$ and the size of RHS is $|C|$. The elements of LHS and RHS are initially set to 0. The map function of the *Create-ACM* algorithm is depicted in Algorithm 3. At lines 4 and 5, the function extracts the antecedent and consequent of a rule. It then adds the consequent of a rule to RHS and the antecedent to LHS, lines 6 and 7. At Line 8, the function emits the antecedent and consequent of a rule to the reducer.

The Create-ACM Reducer creates ACM array as depicted in Algorithm 4. The reducer takes an antecedent and all its consequents from the Mapper as input. It also uses global lists LHS and RHS initialized by the reducer. At Line 4, the reducer uses a function called ACM-Init to create a row of ACM which contains $|C|$ elements which are all initialized to -1. The row corresponds to one of the antecedents and each of its elements corresponds to one of the $|C|$ consequents. Each element in a row correspond to a rule. At lines 5 to 7, each element of ACM which corresponds to a strong rule is set to 0. At last, the Reducer emits the current antecedent with its corresponding ACM row, Line 8.

Algorithm 4 Create-ACM Reducer

```
1:  Class REDUCER
2:      Global: LHS, RHS, ACM
3:      Method REDUCE(Key k, Value v)
4:          ACM-Init(k, RHS, ACM)
5:          for each x ∈ v do
6:              ACM[k,x] = 0
7:          end for
8:          Emit(k, ACM[k])
9:      End Method
10: End Class
```

Algorithm 5 Compute-Lift Mapper

```
1:  Class MAPPER
2:      Global: TXN-count, LHS, RHS, ACM
3:      Method MAP(Key k, Value v,)
4:          TXN-count++
5:          R ← generate-rules(v)
6:          for each r ∈ R do
7:              c ← r.consequent
8:              a ← r.antecedent
9:              update(LHS, a)
10:             update(RHS, c)
11:             if Exists(ACM, a, c) then
12:                 Emit(a, c)
13:             end if
14:         end for
15:     End Method
16: End Class
```

2) Compute-Lift mapreduce algorithm: This algorithm takes as input transactions and gives as output the lift values of the strong association rules. It uses the three global arrays (TXN-count, LHS, RHS) and ACM to compute the lift values. The Compute-Lift Mapper is depicted in Algorithm 5. It counts the number of input transactions at Line 4 then generates all the possible rules from a transaction at Line 5. At lines 7 and 8, it extracts the antecedent and consequent of each rule generated at Line 5. If the antecedent is in LHS, then the corresponding element in LHS is incremented by 1, Line 9. Also if the consequent is in RHS, then the corresponding element in RHS is incremented by 1, Line 10. At last, the map function emits the current antecedent and consequent if they have a corresponding element in ACM, Lines 11 to 13.

The Compute-Lift reducer is shown in Algorithm 6. It takes an antecedent and all its consequents. It also uses the global variables TXN-count, ACM, LHS and RHS. This function receives from the mapper, an antecedent and all its consequents and checks the corresponding ACM element if it belongs to a strong association rule, Line 6. If it belongs to a strong association rule, then the count of that element is incremented by 1, Line 7. At last, the reducer computes the lift values using the Compute-lifts function and emits the antecedent and the corresponding lift values, Line 12.

3) AR Clustering algorithm: Cluster-SAR: This algorithm takes as input the rows of the 2-dimensional array ACM and

Algorithm 6 Compute-Lift Reducer

```
1: Class REDUCER
2:     Global: TXN-count, LHS, RHS, ACM
3:     Method REDUCE(Key k, Value v )
4:         a ← k
5:         for each c ∈ v do
6:             if ACM[a,c] ≥ 0 then
7:                 ACM[a,c]++
8:             end if
9:         end for
10:        k ← a
11:        v ← Compute-lifts(ACM, count, LHS, RHS,a,c)
12:        Emit(k, v);
13:    End Method
14: End Class
```

returns as output the cluster of each strong association rule. Let us refer to each row of ACM as a sample. As explained above, each sample corresponds to a distinct antecedent and each element of a sample corresponds to a distinct consequent. Each element $m_{i,j}$ of ACM contains the lift value of the rule $A_i \rightarrow C_j$.

Algorithm 7 Cluster-SAR Mapper

```
1: Class MAPPER
2:     Global: centroid
3:     Method MAP(Key k, Value v)
4:         Init(index, minDistance)
5:         for i = 0; i < k; i++ do
6:             distance ← EuclideanDistance(v, centroid[i])
7:             if dis ≤ MinDistance then
8:                 minDistance ← distance
9:                 index = i
10:            end if
11:        end for
12:        k ← index
13:        v ← to_string(v)
14:        Emit(k, v)
15:    End Method
16: End Class
```

Cluster-SAR uses K (a pre-specified value) and global variable called centroid, which is initialized to random values. The algorithm is a slight modification of the one proposed in [44]. The Mapper of this algorithm is depicted in Algorithm 7. At Line 4, the function initializes the local variable index to -1 and the MinDistance to the highest real number. For each sample it reads, the mapper computes the distance of the sample from each of the K centroids. It associates each sample with the index of the closest centroid, Lines 5 to 11. At last, the mapper emits each centroid with its corresponding samples at Line 14.

The Cluster-SAR reducer is shown in Algorithm 8. It computes the new centroids. It takes as input each centroid and associated samples, then counts and computes the sum of the corresponding elements in its corresponding samples, lines 5 and 6. It then computes the average of the samples to generate the new centroids, Line 8.

Algorithm 8 Cluster-SAR Reducer

```
1: Class REDUCER
2:     Method REDUCE(Key k, Value v)
3:         Init(SumV2, count)
4:         for each x ∈ v do
5:             count++
6:             ComputeSum(SumV2, x)
7:         end for
8:         centroids ← ComputeCentroids(SumV2, count);
9:         v ← to_string(centroids);
10:        Emit(k, v);
11:    End Method
12: End Class
```

IV. EVALUATION

We conducted a number of experiments to evaluate the performance of the proposed algorithms. In this section, we begin with a description of the experimental settings, datasets and evaluation metrics. We then describe the work conducted and discuss the obtained results and their analysis.

A. Workspace Settings and Datasets

Hadoop 2.81 was used for the experiments. We used a hadoop cluster of three nodes; one was configured as a master node and the other two as slaves. We created four data nodes, each two in a machine. Python and Java were used to implement the proposed algorithms. The datasets used in the experiments were Chess, Mushroom, T10I4D100K, AllElectronics and Webdocs. We chose these datasets because they are publicly available benchmark datasets with different characteristics and frequently used in related work. A summary description of these datasets is shown in Table I including dataset name, notation, number of items, number of transactions, average number of items per transaction, and number of association rules for each dataset. In order to have larger datasets, we replicated each dataset to have four sizes: 1GB, 2GB, 3GB and 4GB; we will refer to each one of them as Di-j, where $i \in 1,2,3,4,5$ denotes the dataset and $j \in 1,2,3,4$ denotes the sizes in GB.

B. Evaluation Metrics

The proposed algorithms were evaluated using four performance measures, namely elapsed time, speedup, scaleup, and sizeup. Elapsed time is the difference between the completion time of job and its submission time. In short, it measures the duration of time a job took to be processed. Speedup compares the elapsed time of a single node to that of n nodes to complete the same job. It is defined as T_1/T_n, where T_1 and T_n are the elapsed times of one and n nodes to complete the same job, respectively. Scaleup compares the elapsed time of a single node to complete a workload to that of n nodes to complete n times the original workload. It is defined as $T_1/T_{n,n}$ where T_1 is the elapsed time of one node and $T_{n,n}$ is that of n nodes. Sizeup is defined as T_n/T_1 and measures the scalability of a system. It compares the elapsed time to complete a single workload (T_1) to the elapsed time of completing n times the original workload (T_n).

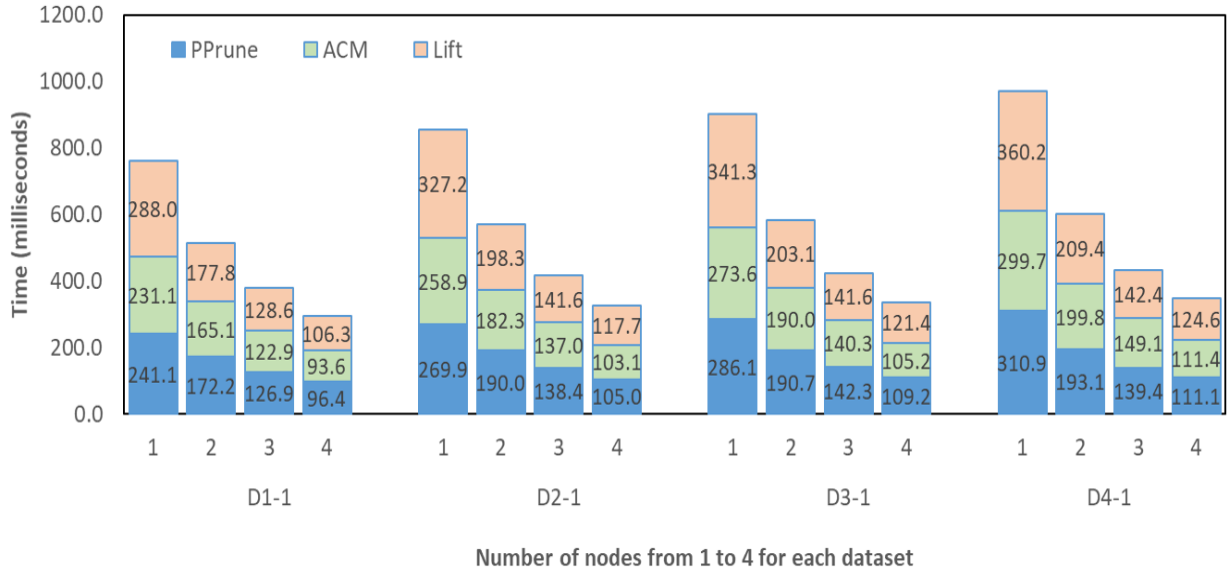

Fig. 2. Elapsed Time for each Algorithm (PPrune, Create-ACM, and Compute-Lift) for various Datasets and Number of Nodes.

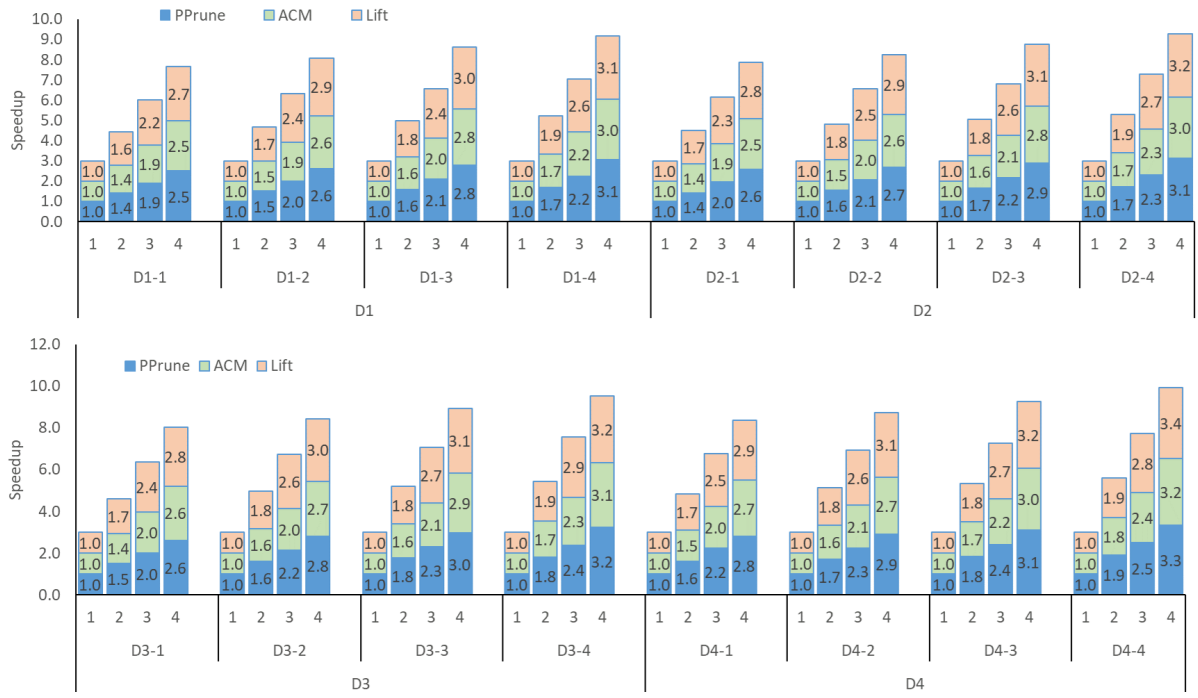

Fig. 3. Speedup for each Algorithm (PPrune, Create-ACM, Lift) for various Datasets and Number of Nodes.

TABLE I. EXPERIMENTAL DATASETS (EACH IS REPLICATED TO GENERATE LARGER DATASETS OF SIZE 1GB, 2GB, 3GB AND 4GB)

Dataset	Notation	Size (KB)	Items	Trans	Avg. Items/Trans	Rules
AllElectronics	D1	1	5	9	2.6	52
Chess	D2	335	75	3196	37	108061
Mushroom	D3	558	119	8124	23	111790
T10I4D100K	D4	3928	870	100000	10	5608
Webdocs	D5	1480	5,267,656	1,692,082	61	1,231,984

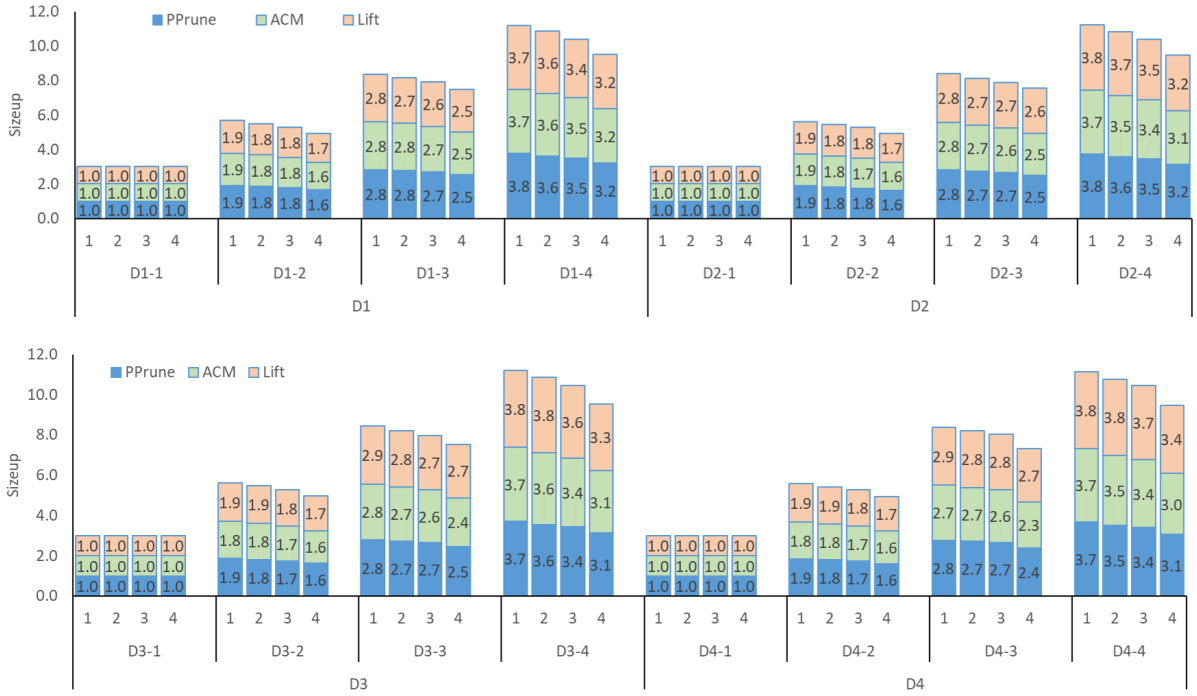

Fig. 4. Sizeup for each Algorithm (PPrune, Create-ACM, Lift) for various Datasets and Number of Nodes.

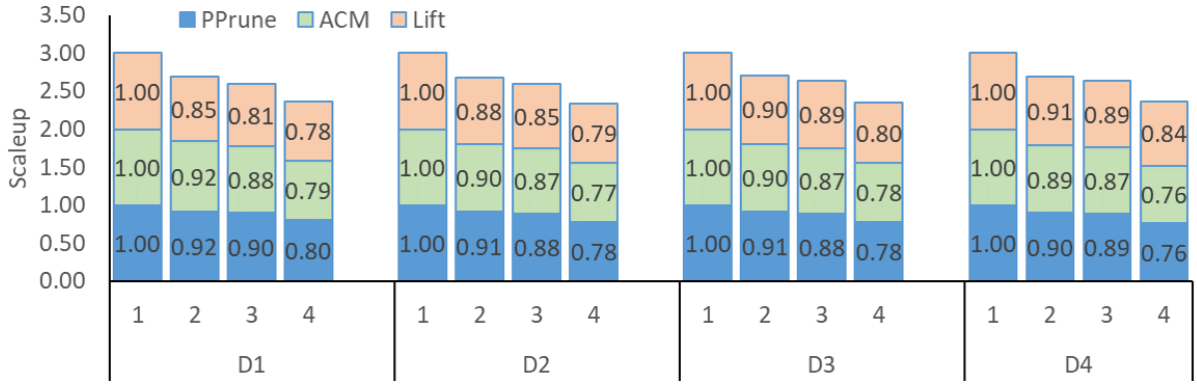

Fig. 5. Scaleup for each Algorithm (PPrune, Create-ACM, Lift) for various Datasets and Number of Nodes.

C. Results

The performance of a MapReduce algorithm is significantly affected by the percentage of communication cost to that of I/O and CPU costs. The higher is the percentage of communication cost in comparison to the I/O and CPU costs the less efficient is the MapReduce algorithm. To study the performance of the proposed MapReduce algorithms, we experimented using the datasets D1-1, D2-1, D3-1, and D4-1 with different number of data nodes (from one to four). Fig. 2 shows the results

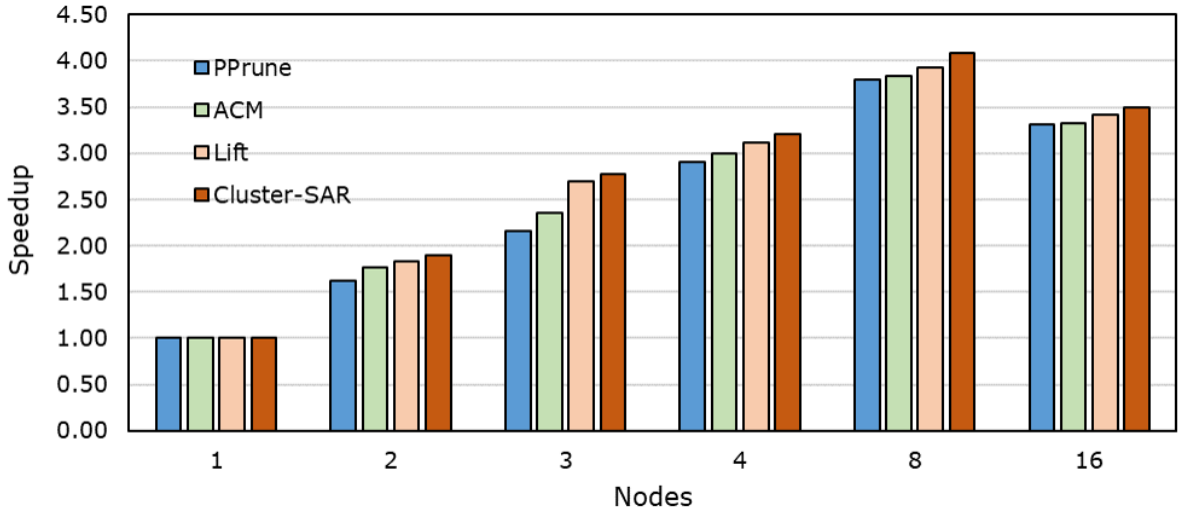

Fig. 6. Speedup of the Proposed Algorithms. Dataset used: Webdoc.

for the elapsed time for each algorithm (PPrune, Create-ACM, and Compute-Lift). As the number of nodes decreases, the elapsed time increases because more items in a node requires more elapsed time. As the number of nodes increases, less number of items is assigned per node, hence the elapsed time decreases. Moreover, the elapsed time increases with the number of items in a dataset; that is why D4-1 and D3-1 required more processing time than D1-1 and D2-1.

To study the remaining three measures, namely, scaleup, sizeup, and speedup, we performed three other sets of experiments. The result are shown in Fig. 3 for the speedup of PPrune, Create-ACM, and Compute-Lift. For p number of nodes, the ideal speed up is p. In our experiments, the worst speedup 1.6 for 2 nodes, 2.2 for three nodes and 2.7 for four nodes. The lowest speedup gained was when experimented with D1 and the best speedup was gained when experimented with D4. This is because D1 has few items compared to D4. Fewer elements results in shorter and fewer association rules, hence less CPU time. That is why the communication cost compared to the CPU cost was more when processing D1 than D4.

The second set of experiments was done to study the sizeup of PPrune, Create-ACM, and Compute-Lift. The results are shown in Fig. 4. The ideal sizeup is n when the size of the workload increases n times. The worst speedup was 3.1 for 4 nodes and the best was 3.4. The lowest speedup was attained with D1 and the best speedup was attained with D4. This is because D1 has few items compared to D4. Fewer elements result in shorter and fewer association rules, hence less CPU time. That is why the communication cost, compared to the CPU cost, was more when processing D1 than D4.

The third set of experiments was done to study the scaleup of PPrune, Create-ACM, and Compute-Lift. The results are shown in Fig. 5. The ideal scaleup is 1 when the size of the workload increases n times. In our experiments, the scaleups ranged between 0.78 and 0.84 when the number of nodes was 4. Again, the lowest scaleup was with D1 and the best scaleup was with D4. This is because of the same reason that we discussed before, which is the number of items in D1 and D4.

Though we have replicated each dataset many times, the resulting number of rules was the same as the original dataset. Therefore, the number of rules was small. To experiment with a huge number of rules, we used the Webdoc dataset. We minimized the minimum support so that we can generate a huge number of rules from the dataset. Some attributes of the dataset are shown in Table I. We added another physical machine (with Intel i7-8750H Processor) and created up to 16 data nodes and then tested the proposed MapReduce algorithms. The performance of each algorithm is shown in Fig. 6. As expected, as the number of data nodes increased, the efficiency decreased since it is the ratio $Speedup/p$, where p is the number of processors. Also, when the number of data nodes used exceeded 8, the speedup decreased. This is because the percentage of the communication cost was too high compared to the CPU and I/O costs. The main reason for the high communication cost is the size of the Webdoc dataset, 1.48 GB, which is very small for a Hadoop machine with more than four data nodes. In general, the speedup of a Hadoop cluster with many nodes improves with bigger datasets.

V. Conclusions

With the increasing size of datasets, the number of association rules mined by traditional approaches is growing exponential making them difficult to visualize or interpret. As a solution for this problem, researchers proposed pruning, grouping and clustering algorithms. The advent of big data technology motivates more research to be conducted in this field. This paper presented a novel approach for clustering huge number of association rules. The proposed MapReduce-based algorithms reduce the number of association rules by first pruning them based on rule structure and then clustering them based on lift value. To study the performance of the proposed algorithms, we used four measures, namely, elapsed

time, speedup, sizeup, and scaleup. We experimented using five benchmark datasets of which two are synthetic. We did all the experiments in a hadoop cluster and the results showed that the proposed algorithms are efficient. For example, the lowest scaleup achieved was 77%.

For future work, further experiments with more nodes and bigger datasets need to be conducted. The proposed algorithms can also be extended to relax the number of items in the consequent of a rule. Also different clustering algorithms and visualization tools can be employed to improve the efficiency of the proposed algorithms.

ACKNOWLEDGMENT

The authors would like to thank King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, for the support during this work.

REFERENCES

[1] C. Tsai, C. Lai, M. Chiang, and L. T. Yang, "Data mining for internet of things: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 77–97, 2014.

[2] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[3] T.-M. Choi, H. K. Chan, and X. Yue, "Recent development in big data analytics for business operations and risk management," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 81–92, 2016.

[4] E. Siow, T. Tiropanis, and W. Hall, "Analytics for the internet of things: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 74, 2018.

[5] Y. Cheng, K. Chen, H. Sun, Y. Zhang, and F. Tao, "Data and knowledge mining with big data towards smart production," *Journal of Industrial Information Integration*, vol. 9, pp. 1–13, 2018.

[6] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[7] K. Lakshmi and G. Vadivu, "Extracting association rules from medical health records using multi-criteria decision analysis," *Procedia computer science*, vol. 115, pp. 290–295, 2017.

[8] L. Zhang, W. Wang, and Y. Zhang, "Privacy preserving association rule mining: Taxonomy, techniques, and metrics," *IEEE Access*, vol. 7, pp. 45 032–45 047, 2019.

[9] W. Altaf, M. Shahbaz, and A. Guergachi, "Applications of association rule mining in health informatics: a survey," *Artificial Intelligence Review*, vol. 47, no. 3, pp. 313–340, 2017.

[10] F. J. V. Martín, J. L. C. Sequera, and M. A. N. Huerga, "Using data mining techniques to discover patterns in an airline's flight hours assignments," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 13, no. 2, pp. 45–62, 2017.

[11] H. Si, J. Zhou, Z. Chen, J. Wan, N. N. Xiong, W. Zhang, and A. V. Vasilakos, "Association rules mining among interests and applications for users on social networks," *IEEE Access*, vol. 7, pp. 116 014–116 026, 2019.

[12] P. Fournier-Viger, J. C.-W. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le, "A survey of itemset mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 4, p. e1207, 2017.

[13] V. S. Kireev, A. I. Guseva, P. V. Bochkaryov, I. A. Kuznetsov, and S. A. Filippov, "Association rules mining for predictive analytics in iot cloud system," in *Biologically Inspired Cognitive Architectures Meeting*. Springer, 2018, pp. 107–112.

[14] A. An, S. M. Khan, and X. Huang, "Objective and subjective algorithms for grouping association rules," in *Proc. 3rd IEEE International Conference on Data Mining (ICDM'03)*, vol. 3, 2003, p. 477.

[15] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Record*, vol. 22, no. 2, pp. 207–216, 1993.

[16] S. Chawla, J. G. Davis, and G. Pandey, "On local pruning of association rules using directed hypergraphs." in *ICDE*, vol. 4, 2004, pp. 832–841.

[17] A. Berrado and G. C. Runger, "Using metarules to organize and group discovered association rules," *Data Mining and Knowledge Discovery*, vol. 14, no. 3, pp. 409–431, 2007.

[18] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila, "Pruning and grouping discovered association rules," in *ECML'95 MLnet Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, April 1995, pp. 47–52.

[19] B. Lent, A. Swami, and J. Widom, "Clustering association rules," in *Proceedings 13th International Conference on Data Engineering*, 1997, pp. 220–231.

[20] M. A. Alasow, S. A. Mohammed, and E.-S. M. El-Alfy, "Parallel association rules pruning algorithm on hadoop mapreduce," in *International Conference on Advanced Communication and Networking*. Springer, 2019, pp. 117–130.

[21] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering internet-of-thing devices," in *Proceedings of the 27th USENIX Conference on Security Symposium*, ser. SEC'18, 2018, pp. 327–341.

[22] N. Hashimoto, S. Ozawa, T. Ban, J. Nakazato, and J. Shimamura, "A darknet traffic analysis for iot malwares using association rule learning," *Procedia Computer Science*, vol. 144, pp. 118 – 123, 2018.

[23] R. Kumar, X. Zhang, R. Khan, and A. Sharif, "Research on data mining of permission-induced risk for android iot devices," *Applied Sciences*, vol. 9, p. 277, 01 2019.

[24] Z. Wang, W. Liang, Y. Zhang, J. Wang, J. Tao, C. Chen, H. Yan, and T. Men, "Data mining in iot era:a method based on improved frequent items mining algorithm," in *Proc. 5th International Conference on Big Data and Information Analytics (BigDIA)*, 2019.

[25] P. Sunhare, R. R. Chowdhary, and M. K. Chattopadhyay, "Internet of things and data mining: An applications oriented survey," *Journal of King Saud University-Computer and Information Sciences*, 2020.

[26] W. Xiao and J. Hu, "Sweclat: a frequent itemset mining algorithm over streaming data using spark streaming," *The Journal of Supercomputing*, vol. 76, no. 10, pp. 7619–7634, 2020.

[27] C. Marinica, F. Guillet, and H. Briand, "Post-processing of discovered association rules using ontologies," in *Data*

Mining Workshops, 2008. ICDMW'08. IEEE International Conference on. IEEE, 2008, pp. 126–133.

[28] T. Brijs, K. Vanhoof, and G. Wets, "Reducing redundancy in characteristic rule discovery by using ip-techniques," *Intelligent Data Analysis Journal*, vol. 4, pp. 200–0, 2000.

[29] S. Kannan and R. Bhaskaran, "Association rule pruning based on interestingness measures with clustering," *International Journal of Computer Science Issues*, vol. 6, 12 2009.

[30] B. Liu, W. Hsu, and Y. Ma, "Pruning and summarizing the discovered associations," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 125–134.

[31] R. J. Bayardo, R. Agrawal, and D. Gunopulos, "Constraint-based rule mining in large, dense databases," *Data Mining and Knowledge Discovery*, vol. 4, no. 2-3, pp. 217–240, 2000.

[32] S. D. Bay and M. J. Pazzani, "Detecting group differences: Mining contrast sets," *Data mining and knowledge discovery*, vol. 5, no. 3, pp. 213–246, 2001.

[33] G. I. Webb, "Opus: An efficient admissible algorithm for unordered search," *Journal of Artificial Intelligence Research*, vol. 3, pp. 431–465, 1995.

[34] S. Huang and G. I. Webb, "Discarding insignificant rules during impact rule discovery in large, dense databases," in *Proc. SIAM International Conference on Data Mining*, 2005, pp. 541–545.

[35] Y. Djenouri, H. Drias, and A. Bendjoudi, "Pruning irrelevant association rules using knowledge mining," *International Journal of Business Intelligence and Data Mining*, vol. 9, p. 112, 01 2014.

[36] L. A. Fernandes and A. C. B. García, "Association rule visualization and pruning through response-style data organization and clustering," in *Lecture Notes in Artificial Intelligence (LNAI)*, vol. 7637, 2012, pp. 71–80.

[37] S. Chawla and J. Davis, "On local pruning of association rules using directed hypergraphs," in *Proc. 20th IEEE International Conference on Data Engeneering (ICDE'04)*, 2003.

[38] L. Yang, "Visualizing frequent itemsets, association rules, and sequential patterns in parallel coordinates," in *International Conference on Computational Science and Its Applications*. Springer, 2003, pp. 21–30.

[39] M. Hahsler and R. Karpienko, "Visualizing association rules in hierarchical groups," *Journal of Business Economics*, vol. 87, no. 3, pp. 317–335, 2017.

[40] B. L. W. H. Y. Ma and B. Liu, "Integrating classification and association rule mining," in *Proc 4th International Conference on Knowledge Discovery and Data Mining*, 1998.

[41] A. Strehl, G. K. Gupta, and J. Ghosh, "Distance based clustering of association rules," in *Proceedings ANNIE*, vol. 9, 1999, pp. 759–764.

[42] J. Mattiev and B. Kavšek, "Cmac: Clustering class association rules to form a compact and meaningful associative classifier," in *Proc. International Conference on Machine Learning, Optimization, and Data Science*. Springer, 2020, pp. 372–384.

[43] D. Bui-Thi, P. Meysman, and K. Laukens, "Clustering association rules to build beliefs and discover unexpected patterns," *Applied Intelligence*, pp. 1–12, 2020.

[44] M. Hahsler, "Grouping association rules using lift," in *Proc 11th INFORMS Workshop on Data Mining and Decision Analytics (DMDA'16)*, 2016.