TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Cem Şamiloğlu 223607IASM

IMPROVEMENT IN THE APRIORI ALGORITHM TO ENHANCE THE EFFICIENCY OF ASSOCIATION RULE MINING TECHNIQUES

Master's Thesis

Supervisor: Mohammadreza Heidari Iman
Early Stage Researcher

Co-supervisor: Tara Ghasempouri
Senior Researcher

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Cem Şamiloğlu 223607IASM

APRIORI ALGORITMI TÄIUSTAMINE ASSOTSIEERIMISREEGLITE KAEVANDAMISE TEHNIKATE TÕHUSUSE SUURENDAMISEKS

Magistritöö

Juhendaja: Mohammadreza Heidari Iman
Early Stage Researcher

Kaasjuhendaja: Tara Ghasempouri
Senior Researcher

Tallinn 2024

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Cem Şamiloğlu

05.05.2024

Abstract

Association rule mining is one of the core techniques used in data mining that discovers unrevealed interesting relationships and patterns from a given dataset. Classical and widely used association rule mining algorithms in the current literature such as Apriori, come with drawbacks of long execution time and a large number of redundant association rules. This thesis proposes an improved Apriori algorithm called *apriori-D* that overcomes these limitations. Improvements made in frequent itemset generation of the Apriori by an innovative solution of ranking and keeping the most valuable itemsets. Results collected from benchmark datasets have demonstrated that the proposed algorithm surpasses the original Apriori in terms of shorter execution time as well as achieving less redundant association rules as a result of the efficient pruning capability.

The thesis is written in English and is 51 pages long, including 7 chapters, 11 figures and 9 tables.

Annotatsioon

Apriori algoritmi täiustamine assotsieerimisreeglite kaevandamise tehnikate tõhususe suurendamiseks

Assotsiatsioonireeglite kaevandamine on üks peamisi tehnikaid, mida kasutatakse andmete kaevandamisel, mis avastab varjatud huvipakkuvaid seoseid ja mustreid antud andmekogumist. Klassikalised ja laialdaselt kasutatavad assotsiatsioonireeglite kaevandamise algoritmid praeguses kirjanduses, näiteks Apriori omavad puuduseid, milleks on pikk teostusaeg ja suur hulk üleliigseid seoseid ja reegleid. Käesolev lõputöö pakub välja täiustatud Apriori algoritmi nimega *apriori-D*, mis ületab nimetatud piirangud. Apriori liigse sagedusega toimuvat üksuste genereerimist täiustati uuendusliku lahendusega kõige väärtuslikumate üksuste järjestamiseks ja säilitamiseks. Kasete tulemused võrdlusandmekogumitel on näidanud, et pakutud algoritm ületab algse Apriori nii lühema täitmisaja kui ka väiksema hulga üleliigsete assotsiatsioonireeglite saavutamise poolest.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 51 leheküljel, 7 peatükki, 11 joonist, 9 tabelit.

List of Abbreviations and Terms

ARM	Association Rule Mining
CPU	Central Processing Unit
CSV	Comma-Separated Values
FP	Frequent Pattern
GPU	Graphics Processing Unit
KDD	Knowledge Discovery in Databases
OS	Operating System
RAM	Random Access Memory
TID	Transaction Identifier
UCI	University of California, Irvine

Table of Contents

List of Figures

List of Tables

1. Introduction

The advancements in digitized systems and wide usage of information technologies in various fields have resulted in rapid data generation. Nowadays, many different domains such as banking, healthcare, retail, marketing, and telecommunications produce enormous amounts of data [1]. This increased trend of data generation in the last few decades has been associated with the term in literature as big data [2]. Continuous generation of data has introduced many opportunities such as giving important insight for many organizations regarding their decision-making process.

Alongside the advantages of data generation, it also brings many challenges to overcome. One of the major challenges it brings is the task of extracting useful information from the raw data [3]. Manual approaches to finding useful information in smaller amounts of data might be feasible, but once the size increases, manual methodologies would get inefficient excessively. Therefore, the need to analyze the data in an automated manner has paved the way for the foundation of a research field referred to as data mining.

Data mining, also known as KDD (Knowledge discovery in databases) is the process of discovering unrevealed patterns and extracting interesting knowledge from a large amount of data [4]. Starting from the beginning of the 1990s there has been significant growth in data mining and many techniques have been developed over the years [5, 6].

An important method of data mining is ARM (Association rule mining), used to find frequently occurring patterns and correlations in a given database [7]. Possible applications of association rule mining include market basket analysis, mining web usage, healthcare analytics, fraud detection, finance, marketing, sales, and many more in numerous fields [8, 9]. The result of ARM methods is the association rules, which are defined as if-then statements that show the relations of items alongside interestingness measures [10]. These association rules are later used by data analysts as input to process and provide insight for making business decisions.

1.1 Problem Statement

Several algorithms have been proposed in the literature throughout the years for association rule mining implementation. One of the earliest and most researched is the Apriori algorithm. The studies proved that Apriori performs well overall when the database is

sparse and discovered patterns have short lengths [11].

However, the efficiency of Apriori degrades steeply, especially when being applied in larger databases. It has two major downsides that affect the performance, a very large number of candidate itemset generation and scanning the database multiple times during the execution. As the dataset gets larger, both of these drawbacks contribute to the total execution time generated association rules as well as memory usage significantly [12].

1.2 Research Motivation

The main objective of this research work is to propose an improved version of the Apriori algorithm by analyzing the gaps that exist in the current literature. The first aspect of improvement is to shorten the total execution time as it grows significantly alongside the database size.

Another aspect to be optimized is the generation of more meaningful association rules. The number of association rules can get huge and often the output of ARM algorithms contains many redundant rules that provide the same or less information of usefulness and relevance [13]. Therefore, the number of association rules is also aimed to be reduced by eliminating the redundant rules, thus enabling a more convenient analysis of the association rules.

1.3 Thesis Structure

This thesis consists of seven main chapters and the content of each chapter is briefly described below.

- Chapter 2 explains the data mining and association rule mining process in more detail and provides the necessary preliminary concepts to build a foundation for ARM algorithms. Alongside these, the well-known Apriori algorithm is described in this chapter.
- Chapter 3 conducts a literature review where the previously proposed improved Apriori algorithms are investigated. The most notable algorithms that develop an optimized Apriori algorithm in terms of execution time and more relevant association rules are examined together with their benefits and drawbacks.
- Chapter 4 explains the methodology of this research work and the proposed algorithm in this thesis is presented to improve the Apriori algorithm by utilizing a novel approach called dominance relation.
- Chapter 5 contains the experimental results of the original Apriori and the proposed

algorithm based on their execution time, number of generated rules, and redundant rules.

- Chapter 5 provides a conclusion of this research work, current limitations, and also possible improvements for future work.
- Chapter 6 summarizes briefly the initial problem, proposed solution, and achievements of the thesis.

Additionally, Appendix 2 includes the sources of all the datasets used for experiments together with the implementations of all algorithms and scripts.

2. Background

The procedure of data mining consists of several steps until there is a piece of meaningful information extracted as the output. A general overview of data mining is depicted in Figure 1. The first step is selection where appropriate data is selected based on the content of the research. Next, the preprocessing step involves removing incomplete or inconsistent data and integrating the data in case of using multiple sources. The preprocessed data is converted into a suitable format for mining in the transformation step. Then, the mining phase takes place where interesting patterns are discovered and extracted for further evaluation. Finally, these patterns can be interpreted to acquire knowledge [14].

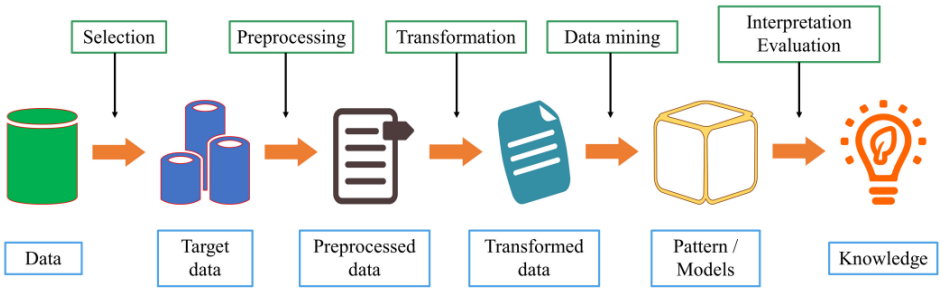

Figure 1. Overview of the data mining process [14]

The data mining process itself can be classified into two categories as *supervised* and *unsupervised* methods as shown in Figure 2.

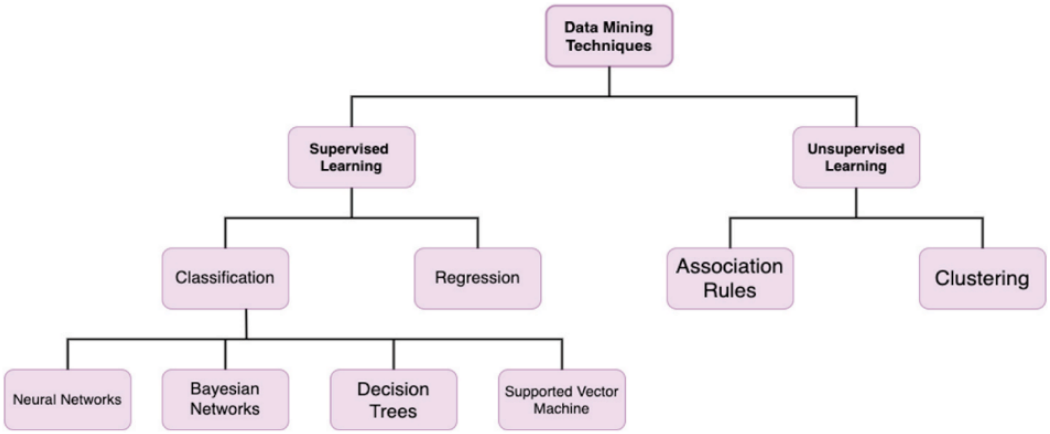

Figure 2. Techniques used in data mining [15]

Supervised techniques, which consist of classification and regression use labeled input and output data to predict future occurrences. Whereas in unsupervised techniques, such as association rule mining and clustering unlabeled data is being used and explores hidden relations during real-time execution [16].

Association rule mining is a fundamental method that discovers relationships of variables from a given dataset [8]. It identifies the patterns and correlations of variables that occur frequently. The goal is to extract meaningful information that provides insights into the data [17].

2.1 Preliminary Concepts

In this section, the preliminary concepts about data mining and ARM will be presented to develop a foundation for the algorithms proposed in the literature.

Definition 1. A ***dataset*** is a collection of data that contains transactions. An example of *dataset* is $D = \{T_1, T_2, ..., T_n\}$, where each transaction is denoted as *T*.

Definition 2. ***Itemsets*** are collection of data that contains items. An example of *itemset* is $I = \{i_1, i_2, ..., i_n\}$, where each item is denoted as *i*.

Definition 3. ***Frequent itemsets*** are a collection of itemsets that appear frequently in a dataset. Indicating the notable relations between the itemsets [18]. The *frequent itemsets* with a length *k*, is denoted as $\{L_k\}$.

Definition 4. ***Candidate itemsets*** are a collection of any valid itemsets which are potentially frequent itemsets [18]. The *candidate itemsets* with a length *k*, is denoted as $\{C_k\}$.

Definition 5. An ***association rule*** is an if-then statement that uncovers relationships of items in a dataset [10]. *Association rule* is defined as an implication presented in Equation 2.1, meaning that if *X* is satisfied, it is highly probable that *Y* is also satisfied [19]. *I* is a *set of items*, *X* and *Y* are *frequent itemsets*.

$$X \rightarrow Y; \text{where } X \subseteq I, Y \subseteq I, X \cap Y = \emptyset \tag{2.1}$$

Definition 6. ***Support*** is a metric used in association rule mining to measure how frequently an itemset occurs in a dataset [20]. For an association rule $X \rightarrow Y$, it states the ratio of transactions in a dataset that contains the union of itemsets *X* and *Y*, and calculated by the formula presented in Equation 2.2. It has a range of [0,1] where a larger support value yields a higher statistical significance.

$$support(X \rightarrow Y) = P(X \cup Y) = \frac{\text{Count of transactions containing itemsets } X \text{ and } Y}{\text{Total number of transactions in } T} \tag{2.2}$$

Definition 7. ***Minimum support*** is a threshold used to identify if an itemset is frequent in a dataset or not. If an itemset has a higher support value than *minimum support*, then it is considered as a *frequent itemset* [18].

Definition 8. ***Confidence*** is another metric used in association rule mining to measure the proportion of transactions in a dataset that contains itemset Y in the same transactions that contain itemset X [21]. For an association rule $X \rightarrow Y$, *confidence* is calculated by the formula presented in Equation 2.3. It has a range of [0,1].

$$confidence(X \rightarrow Y) = P(Y|X) = \frac{support(X \rightarrow Y)}{support(X)} \tag{2.3}$$

Definition 9. ***Minimum confidence*** is a threshold used to determine the strength of the association between itemsets. If an association rule has a higher confidence value than *minimum confidence*, then it is considered as a valid association rule [18].

Definition 10. ***Lift*** is also a key metric used in association rule mining to measure how often itemsets X and Y occur together than expected if they happen to be independent statistically [22]. For an association rule $X \rightarrow Y$, *lift* is calculated by the formula presented in Equation 2.4. It has a range of [0,∞] where a lift value less than 1 indicates a negative correlation, equal to 1 indicates an independent correlation, and higher than 1 indicates a positive correlation.

$$lift(X \rightarrow Y) = \frac{P(Y|X)}{P(Y)} = \frac{confidence(X \rightarrow Y)}{support(Y)} \tag{2.4}$$

The interestingness metrics, also referred to as the measures such as *support*, *confidence* and *lift*, are used for assessing the relevance and value of discovered patterns in association rule mining.

The main high-level process of association rule mining methods can be viewed as a two-step approach [20]:

1. Finding all the *frequent itemsets* that satisfy a pre-defined *minimum support* threshold from a given dataset,
2. Generating association rules using the frequent itemsets that satisfy a pre-defined *minimum support* and *minimum confidence* thresholds using the frequent itemsets found in the first step.

Finding the frequent itemsets is considered to be the most computationally expensive step. When the size of the dataset is larger, the frequent itemsets found in this step get bigger in size, therefore it contributes most to the total execution time and use of resources [23].

Therefore, the majority of the previous research in literature has been done to optimize this step in different approaches. Also worth mentioning is that the second step, which is generating association rules is usually a generic task and does not vary widely between different ARM algorithms.

In terms of the algorithms used in association rule mining, the Apriori algorithm is the most known and widely used one. This is due to its convenient implementation, scalability, and flexibility in terms of various types and sizes of datasets, and interpretability of generated association rules.

2.2 Apriori Algorithm

Apriori is one of the first algorithms proposed for ARM applications by Agrawal and Srikant [24]. It is a level-wise method that discovers the frequent itemsets (Definition 3) from a dataset in an iterative way. These frequent itemsets found at each iteration are extended to find longer frequent itemsets. Next, the association rules (Definition 5) are generated based on all of the found frequent itemsets to discover the patterns and relations of items in the dataset.

TID	Items
$T100$	I1, I2, I5
$T200$	I2, I4
$T300$	I2, I3
$T400$	I1, I2, I4
$T500$	I1, I3
$T600$	I2, I3
$T700$	I1, I3
$T800$	I1, I2, I3, I5
$T900$	I1, I2, I3

Table 1. Sample dataset having a horizontal data format [18]

The frequent itemset discovery process is presented in Figure 3, based on a sample dataset given in Table 1. The sample dataset shown in Table 1 has a horizontal data format containing two columns. The first column represents the TID (Transaction Identifier), and the second column represents the list of items that belong to a specific TID.

The first table $\{C_1\}$, in Figure 3 represents the candidate 1-itemsets (Definition 4) and

their support count. In this example, support count is used instead of the ratio stated in Definition 6. The next table $\{L_1\}$, shows the frequent 1-itemsets, where it is constructed by the itemsets in $\{C_1\}$ that have a higher support than minimum support count. The minimum support count of 2 is used. Since all of the itemsets have higher support than 2, the itemsets in $\{C_1\}$ and $\{L_1\}$ are the same.

After that, the table $\{C_2\}$ in Figure 3, is generated by joining the itemsets present in $\{L_1\}$. The process continues iteratively until there are no frequent itemset found that exceed the minimum support count. The last table $\{L_3\}$ shows the frequent 3-itemsets found at the end of this process.

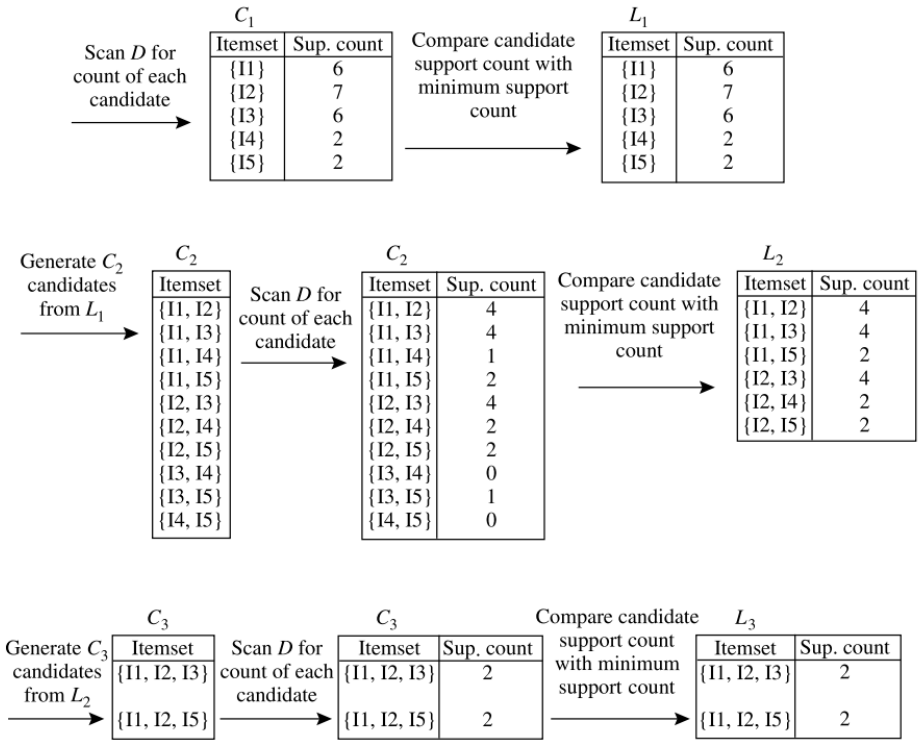

Figure 3. Itemset generation process of Apriori algorithm [18]

Discovering the frequent itemsets procedure of Apriori is outlined in Algorithm 1. The algorithm has two inputs, the minimum support (Definition 7) and frequent 1-itemsets. The frequent 1-itemsets are found by scanning the dataset and keeping the itemsets that have higher support values than the minimum support threshold.

The lines between 1 and 10 of Algorithm 1, find frequent itemsets iteratively starting from frequent 2-itemsets. Candidate generation is executed in line 2 that returns a super-set of all *k*-itemsets from frequent (*k*-1)-itemsets. Candidate generation consists of two steps, join and prune which will be explained further separately.

In lines 3 to 8, all of the candidate *k*-itemsets are scanned in the transactions of the dataset

Algorithm 1 Finding frequent itemsets in Apriori algorithm

Input: L_1 = {frequent 1-itemsets}, $minimum\_support$

1: **for** (k = 2; $L_{k-1} \neq \varnothing$; k++) **do**
2: C_k = *apriori-gen*(L_{k-1}) ▷ candidate generation
3: **for all** transactions $t \in D$ **do**
4: C_t = subset(C_k, t)
5: **for all** candidates $c \in C_t$ **do**
6: **Calculate** support of c
7: **end for**
8: **end for**
9: L_k = {$c \in C_k$ | c.support $\geq minimum\_support$}
10: **end for**

Output: $\cup_k L_k$

to determine their support value. Finally, the support values of these candidate itemsets are compared with minimum support in line 9. If they exceed the minimum support, they are added to the frequent k-itemsets. This process continues iteratively until there is no itemset that has a higher support value than minimum support. The output of the algorithm is all of the discovered frequent itemsets.

The join step of candidate generation is presented in Algorithm 2. Frequent (k-1)-itemsets are joined with each other to generate the candidate k-itemsets.

Algorithm 2 Join step of *apriori-gen*

1: **insert into** C_k
2: **select** $p.item_1, p.item_2, ...p.item_{k-1}, q.item_{k-1}$
3: **from** L_{k-1} p, L_{k-1} q
4: **where** $p.item_1 = q.item_1, ...p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

The prune step of candidate generation is presented in Algorithm 3. Where all itemsets c that are present in candidate itemsets $\{C_k\}$, are removed if (k-1)-subset of c is not present in $\{L_{k-1}\}$. By doing so, the final output of $\{C_k\}$ will be smaller, and thus while generating $\{C_{k+1}\}$ from $\{L_k\}$ in the next step, the total number of itemsets will be less. Since it is an iterative process, this attribute will apply to all itemsets with length k.

Algorithm 3 Prune step of *apriori-gen*

1: **for all** itemsets $c \in C_k$ **do**
2: **for all** (k-1)-subsets s of c **do**
3: **if** $s \notin L_{k-1}$ **then**
4: **delete** c from C_k
5: **end if**
6: **end for**
7: **end for**

After all the frequent itemsets are found, they are used to generate association rules. For each of the frequent itemset, all of its non-empty subsets are generated. Then for all of these subsets, antecedent and consequent parts are used to construct association rules.

If an itemset has length *k*, then there are $2^k - 2$ candidate association rules that will be generated from this itemset. For instance, if the itemset of length 3 is $[X, Y, Z]$, then there are 6 candidate association rules such as $(X, Y) \rightarrow Z$, $(X, Z) \rightarrow Y$, $(Y, Z) \rightarrow X$, $X \rightarrow (Y, Z)$, $Y \rightarrow (X, Z)$ and $Z \rightarrow (X, Y)$.

Despite being a straightforward algorithm to implement, Apriori also has drawbacks that are discovered as the datasets get larger and contain longer patterns. The candidate generation step has a major impact especially on the execution time due to generating a huge number of candidates. Since it is an iterative approach, the dataset needs to be scanned repeatedly which adds up to execution time. Also, Apriori generates many redundant rules that do not provide any extra insight in the presence of another more meaningful rule. Therefore, in the literature, most of the improvements have been targeted to improve the Apriori algorithm in these aspects, which will be examined in the next chapter.

3. Related Works

There are several algorithms developed in the literature with the aim of improving the original Apriori algorithm in different aspects. The literature review conducted in this chapter will focus on the proposed Apriori algorithms that mainly improve execution time and reduce redundant association rules in recent years.

A traditional algorithm used in association rule mining named FP-growth (Frequent pattern growth) [25], was proposed to mitigate the negative effect of the huge candidate generation of Apriori. It was a divide-and-conquer method that introduced a novel approach by compressing the dataset into an FP-tree structure. Then this compressed structure is divided into conditional datasets that only one frequent item is associated with and mines each dataset separately. However, it also brought some disadvantages such as having a complex data structure that is cumbersome to implement. Also, due to its recursive nature, when the dataset gets larger and more sparse the memory requirement becomes an issue for the FP-tree structure [19].

An innovative algorithm named Eclat [26], proposed to utilize a different data format where each itemset is associated with a TID. Because of using this data format, Eclat is considered to be a depth-first search algorithm. The main benefit of this is to be able to calculate the support count of an itemset directly from the length of TIDs that it is associated with. This means that there is no need to scan the dataset continuously to calculate the support measure.

Eclat performs better compared to Apriori regarding execution time when the number of transactions and the length of transactions are relatively small, whereas in higher numbers and longer transactions, Apriori was still able to run faster than Eclat [27]. This is mainly due to the fact that TID lists can get very long when dealing with a high number of frequent patterns, having a negative effect on computation time for intersecting the long lists [19].

Zaki and Gouda later proposed another algorithm called dEclat [28] to address the high cost of intersecting TIDs. It is based on keeping track of only the differences of TIDs of a k-itemset and a correspondent (k-1)-itemset. When the dataset contains dense and long patterns it shows favorable results compared to Eclat while mining frequent itemsets, but still has downsides such as consuming a significant amount of memory for its data structure [18].

A fundamental method used to improve Apriori is to establish parallel execution. The proposals made in [29] and [30] employ the parallel processing of the Apriori algorithm. They are implemented in the MapReduce framework of the Hadoop platform to distribute the data in large clusters. The experimental results show that execution time was shortened with the help of multiple computing nodes. In other research works [31] and [32], a different platform called Spark is used to apply parallel processing of the Apriori. Compared to the proposals that work on Hadoop, results obtained from this proposal made on Spark can run faster due to using memory for caching and processing of the data.

Parallel execution is further improved in proposals such as in [33] and [34] by integrating the use of GPU (Graphics processing unit) to handle complex calculations that showed favorable results in total runtime. Even though parallelism improves the execution time for Apriori, these approaches have the drawback of relying on a separate processing platform to handle the parallel computing.

Another aspect of improving the Apriori algorithm is the utilization of different data structures. A method presented in [35] implements a hash tree structure. This resulted in more efficient counting of support and also reduced the frequent dataset scanning. The work proposed in [36] introduces a frequent matrix that stores the frequency of each item. Furthermore, in [37], the use of a boolean matrix structure is proposed, and then operations are performed on the compressed matrix row vectors. These proposals also resulted in scanning the dataset less and reduced the time complexity of the original Apriori.

Apart from the proposals that reduced scanning of the database as a side effect of implementing different data structures, there are also approaches to optimize Apriori directly by avoiding unnecessary scanning of the dataset based on basic properties. Some notable efforts presented in [38], [39], and [40] reduce the number of dataset scans by first splitting each itemset in frequent itemsets. Then, uses the TIDs of the split itemset which has a lower support value to scan the next candidate itemsets only in transactions with these specific TIDs instead of scanning in the whole database.

The methods proposed in [41] and [42], are based on skipping scanning the transaction if the transaction length is less than the current iteration number. Obviously, the runtime is shortened since scanning fewer transactions however it introduces a risk of losing potential valuable itemsets that are included in those transactions.

The research work presented in [43] takes a different approach where a new parameter infrequent support is introduced. This parameter is used instead of classical support measure to decide whether an itemset is found infrequent early in the scanning of the

database therefore there is no need to continue scanning for it anymore.

The method to reduce the scanning over the database is often an effective solution to shorten the runtime of Apriori. However, they might also yield reasonable results only on specific conditions such as in [43], the approach improves in the cases of using a higher minimum support threshold.

Some research works employ the use of an adaptive minimum support level instead of inputting a constant threshold value as in the original Apriori. In the proposal made in [44], the minimum support threshold is updated iteratively. The update is based on dividing the sum of support counts of all items and then dividing it by the number of unique transactions. Then, the updated minimum support threshold is used to prune candidate itemsets to generate frequent itemsets.

A different approach of using a dynamic minimum support level is investigated in [45] by utilizing the binomial distribution. An adaptive threshold is calculated at the start of finding frequent by first calculating the binomial distribution of each itemset. Then the final adaptive threshold is determined by multiplying the sum of all binomial distributions and total transactions, then dividing by the number of candidate sets.

Furthermore, recalculation of the minimum support level is done by the sum of support levels found in the candidate itemset divided by the size of the candidate itemset in [46]. The resulting new average threshold is then used to prune candidate itemsets instead of a static minimum support level. Dynamic determination of minimum support level is an effective method to reduce the execution time of Apriori but its effect on generating valuable association rules is not a well-researched aspect.

Applying efficient pruning techniques is another common way to optimize Apriori. A proposal made in [47] aims to extend the pruning process during frequent itemset generation step. At the first iteration, pruning is done only on minimum support threshold, and in the next iterations of frequent itemset generation, the rule is already generated and confidence is calculated. Itemset is marked as one if its confidence is higher than 1/2, then all items are subtracted from it and the items marked as zero are deleted.

Another effort to improve pruning is presented in [48]. It is based on an inference that in a frequent itemset, all of the itemsets should be checked if it is included in the transaction database. If not, it is pruned before continuing the generation of candidate itemset. In contrary to the original Apriori where the candidate itemset is generated first and then pruned, but some of these generated candidate itemsets might not be in the transactions.

Moreover, a proposal made in [49] uses a filtration approach instead of pruning to overcome unnecessary candidate generation by generating extra infrequent itemsets alongside the frequent itemsets. The infrequent itemsets are used to determine if an itemset in a previously generated candidate itemset is infrequent, then it does not need to be included in the next iteration and can be pruned.

A very elementary pruning application is proposed in [50] where the current number of iteration is compared with all of the frequent itemsets and the itemsets that have a support value less than that are pruned. Therefore, this results in a smaller size of candidate itemsets to be used in the next generations.

Reducing the redundant association rules is also a subject that has been researched in the literature. The proposal made in [51] proposes an algorithm named DKARM that eliminates itemsets that could potentially generate redundant rules. In another research work [52], the redundant association rules are identified and pruned in the post-processing step based on prior knowledge. However, these approaches are based on prior domain knowledge and require extra background information about the data which may not be available in all applications.

The research works in [53] and [54] state the definition of redundant association rule and propose a method to reduce redundant rules without relying on domain knowledge. Similarly, the proposals in [55] and [56] also develop methods to reduce redundant association rules but with different definitions. Nevertheless, all of these methods are applied as a post-processing step in addition to the main ARM algorithm. This means that the main ARM algorithm already generates the redundant association rules. However, these rules are later identified and pruned as an extra process by these proposed methods.

There are still limitations in the above-mentioned proposals that solely aim to improve one aspect of Apriori without considering the other aspects. Apriori-based algorithms suffer from generating many redundant rules that do not provide extra insight in the presence of another meaningful rule. The proposals made for improving execution time do not provide an optimization in the redundancy of association rules. Similarly, the methods aimed to reduce redundant association rules do not provide information about their effect on the execution time. Therefore, there is still a need to improve the apriori by proposing innovative solutions and approaches that enhance both of these aspects.

4. Methodology

The literature review done in the previous chapter has shown that the most costly step of Apriori-based algorithms is candidate generation, which adds complexity to the computational aspect and biggest contributor to long execution time. Analysis of original and improved versions of Apriori algorithms indicates that all of them have a limitation in that the pruning operation is handled by relying on only one metric which is support.

There are other metrics used in Apriori that can be utilized alongside support, such as confidence and lift. However, using any other metric in the pruning step will yield different results because the evaluation of an itemset varies from metric to metric, as well as favoring some of the itemsets [57].

So, using one metric may conclude that this itemset has a low metric and can be pruned, while another one may yield a higher metric thus keeping this itemset for the next iterations. As a result, there may be interesting and valuable itemsets that are pruned in this step that cannot be generated into strong association rules. In order to overcome this limitation, an improved version of Apriori based on *dominance relation* will be proposed in this chapter referred to as *apriori-D* algorithm.

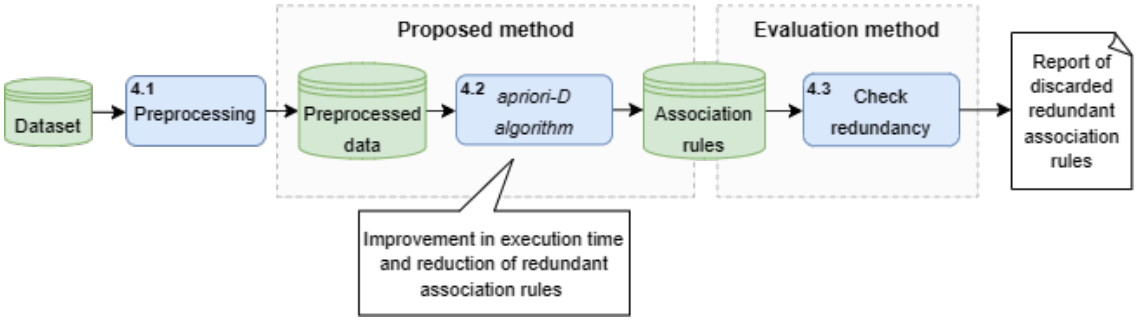

Figure 4. General flow of the methodology

The overall high-level process of the methodology and how the proposed algorithm is used is illustrated in Figure 4. The operation starting from the raw data to the generation and evaluation of association rules is comprised of several steps. First, the structure of input data has to be decided whether it is suitable to be processed in an algorithm directly or if it needs to go through a preprocessing step. Then the preprocessed data is used as an input for the proposed algorithm to generate association rules. After that, in the evaluation method, another separate algorithm will be used to check the redundancy of the association rules that *apriori-D* algorithm has generated.

In the following sections, each phase of the methodology will be elaborated.

4.1 Preprocessing of the Data

Often the datasets do not have the suitable structure to be able to be directly used as an input to the proposed algorithm. Because of that, for some of the datasets, a preprocessing step before the actual algorithm execution might be necessary including filtering and restructuring of the data.

TID	Item
$T100$	A
$T100$	B
$T200$	X
$T300$	C
$T300$	D
$T300$	E

Table 2. Raw data where each TID mapped to an item

TID	Items
$T100$	A, B
$T200$	X
$T300$	C, D, E

Table 3. Preprocessed data where each TID contains all of its items

Consider a dataset that has a structure with a TID mapped to one item per row as shown in Table 2. A preprocessing script will transform the raw data into a structure where each TID contains all of its items as shown in Table 3.

4.2 Improved Apriori Algorithm with Dominance - *apriori-D*

The proposed algorithm named *apriori-D*, uses three metrics support, confidence, and lift to enhance the pruning step of finding frequent itemsets by utilizing the dominance relation. Even though the use of different metrics is not limited, these three metrics are chosen as they are widely used in association rule mining methods. The notion of dominance relation will be explained in the following subsection.

4.2.1 Dominance Relation

Dominance is a novel approach for discovering association rules without favoring or excluding any of the used metrics [58]. In this thesis, it will integrated into finding frequent itemsets step of Apriori and will be utilized in the context of itemsets.

Considering that I is a collection of frequent itemsets where $I = \{i_1, i_2, ..., i_k\}$. All of these itemsets are evaluated based on a set M of different metrics such as support, confidence, and lift, where $M = \{m_1, m_2, ..., m_n\}$. Then the structure $\Omega = (I, M)$ is

created that consist of the itemsets in I and the metrics in M. The value of the metric m for the itemset i is denoted by $i[m]$, such that $i \in I$ and $m \in M$.

The dominance relation is based on the inference as follows; for all of the measures (*i.e.,* support, confidence, lift), if an itemset *i* is dominated by another itemset *i'*, then the itemset *i* is removed because it is not interesting according to the combination of all measures. Formally it is defined in two levels, value dominance and itemset dominance.

Definition 11. ***Value dominance***: Given two values of a measure *m* from *M* corresponding to two itemsets *i* and *i'* from *I*, $i[m]$ dominates $i'[m]$, denoted by $i[m] \geq i'[m]$, if $i[m]$ is preferred to $i'[m]$. If $i[m] \geq i'[m]$ and $i[m] \neq i'[m]$, then it is defined that $i[m]$ strictly dominates $i'[m]$, denoted as $i[m] > i'[m]$.

Definition 12. ***Itemset dominance***: Given two itemsets *i*, *i'* from *I*;

- *i* dominates *i'*, denoted as $i \geq i'$, iff $i[m] \geq i'[m]$, $\forall\, m \in M$.
- If $i \geq i'$ and $i' \geq i$, for example $i[m] = i'[m]\ \forall\, m \in M$ then *i* and *i'* are equivalent, denoted as $i \equiv i'$.
- If $i \geq i'$ and $\exists\, m \in M$ such that $i'[m] > i[m]$, then *i'* is strictly dominated by *i*, denoted as $i > i'$.

Definition 13. ***Reference itemset***: It is a fictitious itemset that dominates all the itemsets of *I*. Formally it is denoted as $\forall\, i \in I$ and $i^{\perp} \geq i$.

Definition 14. ***Degree of similiarity***: It is a numerical value that defines the strengths of associations between two variables [59], in this case the itemsets. It is calculated by the Equation 4.1. The sum operation is done for all of the metrics, starting from $n = 1$ up to k, which represents the total number of metrics used. The variables i and i' represent two itemsets, and $\hat{m_n}$ represents a value according to the metric n.

$$DegSim(i, i') = \frac{\sum_{n=1}^{k} |i[\hat{m_n}] - i'[\hat{m_n}]|}{k} \tag{4.1}$$

Detailed step-by-step execution of dominance relation is presented in Algorithm 4. The algorithm has one input as the set of itemsets and their corresponding measures. Where the variables used during the execution for accumulating the data are as follows:

- Variable *S* is initialized to an empty set that is used to keep track of undominated itemsets.
- Variable *C* is initialized to *I*, containing the set of current candidate itemsets to be qualified as undominated.

- Variable $\mathcal{E}$ is initialized to *I*, containing all current sets covering the undominated space of all undominated itemsets.

The lines between 4 and 26 of Algorithm 4, are executed iteratively while the set of candidate rules *C* is empty, if so the execution ends and the current undominated itemsets that are added to *S* are returned. Otherwise, each itemset in *C* is assumed to be an undominated itemset. Next, in lines 5 to 10, a degree of similarity calculation is performed. If the current itemset i, has a minimal degree of similarity with the reference itemset $i^{\perp}$, then i is said to be an undominated itemset, added to *S* and deleted from *C*. In lines 11 to 24, all of the itemsets are compared with undominated space $i*$, if $i*$ is dominated by i, then i is no longer a candidate and deleted from *C*. If not, $i*$ is still a candidate itemset and added to the undominated subspace of i. This process is done iteratively until there are no more candidate itemsets left in *C*.

Algorithm 4 Dominance algorithm

Input: $\Omega = (I, M)$, {set of itemsets and measures}

1: $S \leftarrow \emptyset$
2: $C \leftarrow I$
3: $\mathcal{E} \leftarrow \{I\}$
4: **while** $C \neq \emptyset$ **do**
5: $i* \leftarrow i \in C$ having $min(DegSim(i, i^{\perp}))$
6: $C \leftarrow C \setminus \{i*\}$
7: **for** $n = 1$ to k **do**
8: $s_n^{i*} \leftarrow \emptyset$
9: **end for**
10: $S \leftarrow S \cup \{i*\}$
11: **for all** $e \in \mathcal{E}$ such that $i* \in s$ **do**
12: **for all** $i \in e$ **do**
13: **if** $i* > i$ **then**
14: $C \leftarrow C \setminus \{i\}$
15: **else**
16: **for** $n = 1$ to k **do**
17: **if** $i[m_n] > i*[m_n]$ **then**
18: $s_n^{i*} \leftarrow s_n^{i*} \cup \{i\}$
19: **end if**
20: **end for**
21: **end if**
22: **end for**
23: $\mathcal{E} \leftarrow \mathcal{E} \setminus \{e\}$
24: **end for**
25: $\mathcal{E} \leftarrow \mathcal{E} \cup \{s_1^{i*}, ..., s_k^{i*}\}$
26: **end while**
27: **Return** *S*

Output: *S* = Set of undominated itemsets of Ω

4.2.2 Phases of the *apriori-D* Algorithm

The high-level overview execution of algorithm *apriori-D* is illustrated in Figure 5. In the first iterations, the algorithm behaves identically to Apriori, keeping only the itemsets that have support values bigger than the minimum support threshold. In the last 2 iterations, firstly the itemsets are pruned based on minimum support still. Then for all of the resulting itemsets, confidence and lift measures are also calculated to help the mining of more meaningful rules. These itemsets together with three calculated measures are fed through the dominance for further pruning.

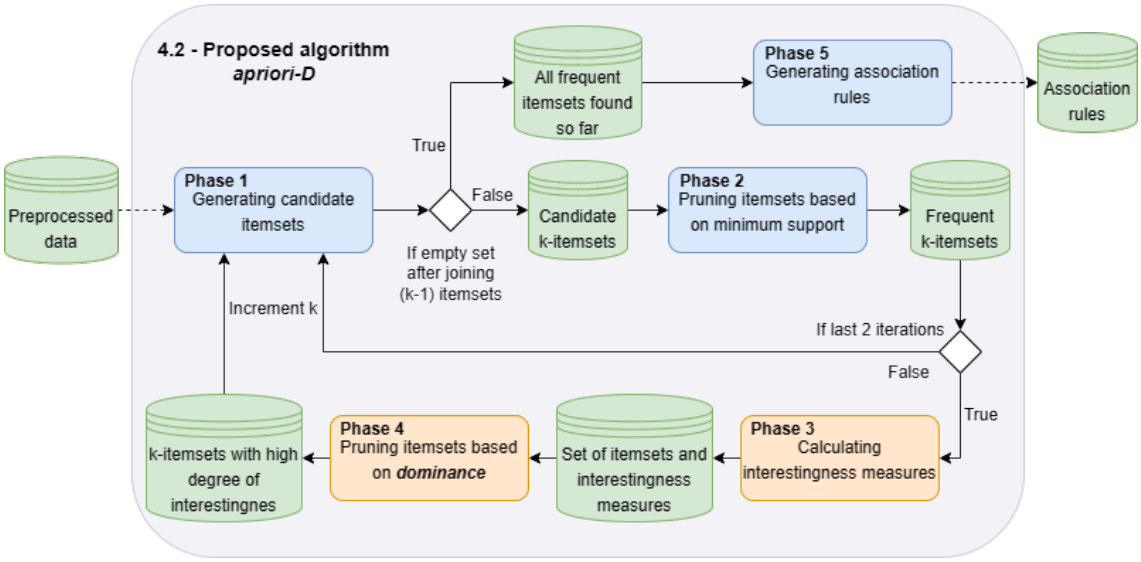

Figure 5. Overview of the algorithm *apriori-D* phases

Dominance is decided to be executed on the last 2 iterations after extensive testing with different starting conditions on a couple of iterations. Executing it in the very early iterations of the algorithm will potentially prune an excessive amount of itemsets. Obviously, it will result in a shorter execution time overall and fewer number of generated association rules. However, it may also indicate that many valuable association rules are lost during the process. Whereas executing it on the last iteration would not improve the algorithm much in terms of execution time. Therefore, running it on the last 2 iterations was found to be a balanced improvement between execution time and the generated number of rules.

Phase 1. Generating Candidate Itemsets

Generation of candidate itemsets is the first step of the algorithm where the dataset is scanned once to get support levels of each itemset and then they are used to create the candidate 1-itemsets $\{C_1\}$. The candidate 1-itemsets are assumed to be all the possible itemsets of length 1 that are found in the dataset.

Next, there is a statement to check if the candidate 1-itemsets found in the previous step is an empty set, which would reveal that the dataset is empty and there are no itemsets found. In this scenario, the algorithm terminates completely without being able to find any frequent itemsets to generate any association rules.

In the case that candidate 1-itemsets are not an empty set, they are passed to the next step of pruning the candidate 1-itemsets based on minimum support to generate frequent 1-itemsets. The frequent 1-itemsets found are used to create candidate 2-itemsets by joining the frequent 1-itemsets by itself. For instance, considering the frequent 1-itemsets, the $\{L_1\}$ is joined by itself to generate $\{C_2\}$.

This process is executed iteratively to generate candidate itemsets with each length. In order to generate candidate 3-itemsets $\{C_3\}$, the frequent 2-itemsets $\{L_2\}$ are joined with each other. In order to find candidate *k*-itemsets $\{C_k\}$, the frequent (*k*-1) itemsets are joined with each other as $\{L_{k-1}\}$ and so on.

After the generation of each candidate *k*-itemset, the statement checks again if it is an empty set to decide whether to move the generation of association rules step or to continue finding frequent itemsets. For example, there might be only one candidate 6-itemset. Since performing a union operation on it cannot generate any longer itemset with a length of 7, the result would be considered as an empty set.

If the result of candidate itemset generation is an empty set, the algorithm moves to the generation of association rules by using all of the found frequent itemsets up to this stage. In the case of the above example, it would use frequent itemsets with lengths 1, 2, 3, 4, 5, and 6 to generate association rules.

Phase 2. Pruning Itemsets Based on Minimum Support

After all the candidate *k*-itemsets are found, the next step is to prune them based on a minimum support threshold. All of the itemsets in the candidate *k*-itemsets are compared with a pre-defined input parameter minimum support. The itemsets that have lower support value than this threshold are removed from $\{C_k\}$ and the remaining itemsets are called frequent *k*-itemsets $\{L_k\}$.

Once the frequent *k*-itemsets are generated, the statement checks if the algorithm is within the last 2 iterations or not. If the algorithm is not within the last 2 iterations, it continues back to generating candidate itemsets step. If the execution is within the last 2 iterations, then it will move to another path that calculates multiple interestingness measures for each

subset of the itemsets.

Phase 3. Calculating Interestingness Measures

In this step, the confidence (Definition 8) and lift (Definition 10) metrics will be calculated for each subset of the itemsets and these measures will be given as input to the dominance relation. Support metric was calculated in the previous step and the frequent *k*-itemsets already have their associated support value, therefore there is no need to recalculate it again.

Originally, these two metrics were defined for association rules, where there is an antecedent and consequent. To be able to calculate these metrics for itemsets, the possible subsets of itemsets are generated and confidence and lift are calculated for these subsets first. Then, the average confidence and lift for the subsets are used to determine the final metrics for a specific itemset.

The result of this operation is a set of itemsets and three calculated measures corresponding to them.

Phase 4. Pruning Itemsets Based on Dominance

A set of itemsets and measures determined in the previous step are used as an input to the dominance. The main purpose of using dominance relation at this step is to compare the multiple measures of each of the itemsets and perform a further pruning operation on them. The remaining itemsets after this stage will have a higher degree of interestingness and less number that will be used to continue generating candidate itemsets.

An example to show the usage of dominance is examined in Table 4 where a set of itemsets and their respective three different measures are presented. Table 4 consists of four columns where the first column indicates the itemsets and the remaining three columns are different measures calculated for a specific itemset.

According to the Definition 11 and Definition 12, the itemset i_3 strictly dominates i_2 since all of its corresponding measures are bigger or equal, such as if $i_3[m_1] \geq i_2[m_1]$, $i_3[m_2] \geq i_2[m_2]$ and $i_3[m_3] > i_2[m_3]$. This relation is applied to all itemsets in Table 4. The execution of dominance results in only two undominated itemsets as shown in Table 5.

As a result, dominance ensures that only valuable itemsets are kept for the next iterations and pruning non-interesting itemsets to benefit in terms of computational aspect as well as reducing the execution time at this step.

Itemsets	Measure 1 (m_1) *Support*	Measure 2 (m_2) *Confidence*	Measure 3 (m_3) *Lift*
i_1	0.20	0.67	0.02
i_2	0.10	0.50	0.00
i_3	0.10	0.50	0.02
i_4	0.20	0.40	0.10
i_5	0.20	0.33	0.02
i_6	0.20	0.33	0.10
i_7	0.10	0.20	0.01
i_8	0.10	0.17	0.02

Table 4. Set of itemsets and their corresponding measures

Itemsets	Measure 1 (m_1) *Support*	Measure 2 (m_2) *Confidence*	Measure 3 (m_3) *Lift*
i_1	0.20	0.67	0.02
i_4	0.20	0.40	0.10

Table 5. Dominance result of the itemsets and their corresponding measures

Phase 5. Generating Association Rules

The generation of association rules is the final step and is presented in Algorithm 5. It is executed if the output of candidate itemsets is found to be an empty set. The algorithm has two inputs as minimum confidence and all of the discovered frequent itemsets.

In line 3 of Algorithm 5 the association rules are generated by antecedent and consequent parts of each non-empty subset for each itemset. For instance, if the frequent itemset is $[A, B, C]$, then the generated association rules are $(A, B) \rightarrow C$, $(A, C) \rightarrow B$, $(B, C) \rightarrow A$, $A \rightarrow (B, C)$, $B \rightarrow (A, C)$ and $C \rightarrow (A, B)$. Between lines 4 to 6, the confidence value of the generated rule is compared with minimum confidence. If it is higher, the association rule is considered as strong.

Algorithm 5 Generating association rules

Input: $\cup_k L_k$ = {all frequent itemsets}, $minimum\_confidence$

1: **for all** itemset $i \in L_k$ **do**
2: **for all** non-empty subset s of i **do**
3: **Generate** *rule* as $s \rightarrow (i - s)$
4: **Calculate** confidence of *rule*
5: **if** *rule*.confidence $\geq$ minimum_confidence **then**
6: {Association rules} $\leftarrow$ *rule*
7: **end if**
8: **end for**
9: **end for**

Output: {Association rules}

The whole procedure of *apriori-D* is presented in Algorithm 6. It has two inputs, the minimum support, and frequent 1-itemsets. The algorithm iteratively finds the frequent itemsets starting from the length of 2 and terminates when there are no more frequent itemsets to be generated. Line 2 handles the candidate generation step which was described previously.

Then, in lines 5 to 12, the transactions in the dataset are scanned for all of the candidate itemsets to determine their metric. Lines 6 to 10 describe which metrics to calculate depending on the current iteration. Next, lines 13 to 18 perform the pruning based on the current iteration, if it is within the last 2 iterations, dominance-based pruning is executed in addition to pruning based on the minimum support threshold. Otherwise, the candidate itemsets are pruned based on only the minimum support threshold.

Algorithm 6 Finding frequent itemsets in Apriori with Dominance - *apriori-D*

Input: L_1 = {frequent 1-itemsets}, $minimum\_support$

```
1:  for (k = 2; L_{k-1} ≠ ∅; k++) do
2:      C_k = apriori-gen(L_{k-1})                          ▷ candidate generation
3:      for all transactions t ∈ D do
4:          C_t = subset(C_k, t)
5:          for all candidates c ∈ C_t do
6:              if last 2 iterations then
7:                  Calculate measures {support, confidence, lift} of c
8:              else
9:                  Calculate support of c
10:             end if
11:         end for
12:     end for
13:     if last 2 iterations then
14:         L_k = {c ∈ C_k | c.support ≥ minimum_support}
15:         L_k = dominance(L_k, measures)                  ▷ execute dominance
16:     else
17:         L_k = {c ∈ C_k | c.support ≥ minimum_support}
18:     end if
19: end for
```

Output: $\cup_k L_k$

All of the discovered frequent itemsets are used to generate the association rules, which will be used in the next step to evaluate the redundancy of these association rules.

4.3 Evaluation of the Redundancy of Association Rules

The final generated association rules will be compared and evaluated between Apriori and the proposed algorithm to determine which rules the proposed algorithm is able to

find redundant and discard. The redundant association rules are the rules that do not provide any extra insight in the presence of another more meaningful rule. Even though association rules are also pruned and only those that exceed minimum confidence and minimum support threshold are considered to be included in the output, often it does not provide enough insight to determine whether these association rules are valuable.

For this purpose, another method named *redundancyRulesChecker* is developed and used to identify redundant association rules from the output of algorithms. The Algorithm 7 has two inputs as the association rules of original *apriori* and proposed algorithm *apriori-D*.

Algorithm 7 Identify redundant association rules - *redundancyRulesChecker*

Input: (rules_of_apriori), (rules_of_proposed_algorithm)
1: redundantRules $\leftarrow \emptyset$
2: redundantRuleCount $\leftarrow 0$
3: different_rules $\leftarrow$ (rules_of_apriori) - (rules_of_proposed_algorithm)
4: **for all** $r \in$ different_rules **do**
5: **Parse** rule: *r.ant*, *r.cons* ▷ Parse as antecedent and consequent
6: **for all** $k \in$ rules_of_proposed_algorithm **do**
7: **Parse** rule: *k.ant*, *k.const* ▷ Parse as antecedent and consequent
8: **if** (*r.ant* is subset *k.ant*) and (*r.cons* is subset *k.cons*) **then**
9: redundantRules $\leftarrow r$ ▷ Flag *r* as a redundant rule
10: redundantRuleCount++
11: **end if**
12: **end for**
13: **end for**
Output: redundantRules, redundantRuleCount

In line 3 of Algorithm 7, the different rules are determined by subtracting all the rules of original *apriori* and proposed algorithm *apriori-D*. Next, the lines 5 and 7 parses each rule of *different_rules* and *rules_of_proposed_algorithm* as antecedent and consequent. Finally, between lines 8 and 11, the redundancy is identified if both the consequent and antecedent of rule in *different_rules* are a subset of an association rule in proposed algorithm output *rules_of_proposed_algorithm*. In that case, this specific rule is flagged as redundant and the *redundantRuleCount* variable is incremented. An example of identifying a redundant rule will be explained below.

Consider an association rule of $(X,Y) \rightarrow (A,B)$, which appears in the difference set and does not appear in the output of the proposed algorithm. This association rule can be identified as redundant if there is an association rule present in output of proposed algorithm such as $(X,Y,Z) \rightarrow (A,B)$, $(X,Y) \rightarrow (A,B,C)$ or $(X,Y,Z) \rightarrow (A,B,C)$. The reason is that there is another rule that is a superset of the former one in the final output. It means that the dominance can get rid of itemsets that might produce this redundant association rule at the end.

5. Experimental Results

The experimentations are performed on an Intel Core i7-6600U CPU (Central processing unit) with 2.60 GHz clock rate, and 8 GB of RAM (Random access memory) running Windows 10 - 64 bit as the OS (Operating system). For the algorithms, Python programming language is used to implement an original Apriori algorithm without any modifications named as *apriori*, and an improved algorithm based on dominance relation named as *apriori-D*.

The output of the algorithms includes information regarding the number of frequent itemsets found at each step, the number of generated association rules, and the total execution time. For calculation of execution time Python standard *time* module is used and takes into account the time between the algorithm starts running until it completely terminates. The original and proposed algorithms are going to be compared in terms of these aspects as well as the number of found redundant rules in this chapter.

5.1 Dataset Properties

Three different datasets have been selected and used in the experimentations. All of the datasets have the CSV (Comma-separated values) format. Each of them has different sizes in terms of total transactions, unique items, and lengths per transaction.

Dataset name	Transaction count	Unique items	Avg. item count per transaction
retail_small	1349	2337	18.8
retail_large	2183	3232	19.7
groceries	9835	331	4.4

Table 6. Properties of the datasets after preprocessing

Table 6 contains the information of different attributes of each dataset. It consists of four columns and three rows. The first column represents the name of the dataset, the second column represents the total transaction number of the dataset, the third column represents how many unique items the dataset has and the last column represents how many items each transaction contains on average in the dataset. Each row of the table has the name of the dataset and three of their attributes respectively.

The first dataset named *retail_small*, contains online retail transactions taken from the UCI (University of California, Irvine) Machine Learning Repository. It has several columns

including invoice number, item description, and country of origin. It is filtered only to keep transactions with countries other than the United Kingdom to reduce the transaction count. Since this dataset has the structure that all items belong to the same invoice number in different rows, it needs to be preprocessed as described in Section 4.1, so that it can be used by the algorithms. After the filtering and preprocessing steps, it has 1349 transactions, 2337 unique items, and on average 18.8 items per transaction.

The second dataset is also a dataset containing retail transactions and it is from the UCI Machine Learning Repository. It is named as *retail_large*. It has the same structure as *retail_small* but with a higher transaction count and overall longer transactions with more unique items. It is also filtered to keep transactions with countries other than the United Kingdom. Also, a preprocessing step is needed as described in Section 4.1 to be able to use this dataset. After preprocessing, it has 2183 transactions, 3232 unique items, and on average 19.7 items per transaction.

The last dataset that is named *groceries* contains transactions of a grocery market basket taken from Kaggle. This dataset is already in a suitable structure where it has each transaction mapped to all of the items that it contains. So, no preprocessing is needed for this dataset. It has 9835 transactions, 331 unique items, and on average 4.4 items per transaction.

5.2 Comparison of Results In Terms of Execution Time

The algorithms are executed on three different datasets with four different minimum support thresholds for each of them. On the other hand, minimum confidence levels of 0.6, 0.8, and 0.4 are used in *retail_small*, *retail_large*, and *groceries* datasets respectively for the association rule generation step. The reasoning behind using different minimum support and confidence levels for different datasets is because of the diverse properties of the datasets.

Using the same minimum support and minimum confidence thresholds for different datasets often results in extreme outcomes such as generating a huge number of association rules or generating no association rules at all in some cases. So, after a series of tests, these values are found to provide a balance in both execution time and association rule generation aspects.

Table 7 is composed of five columns where the first one shows the name of the dataset, the second one shows the minimum confidence threshold, the third one shows the minimum support thresholds used for these datasets, the fourth and the fifth ones show the execution

Datasets	Minimum confidence	Minimum support	Total execution time (sec.)	
			Original algorithm *apriori*	Proposed algorithm *apriori-D*
retail_small	0.6	0.02	4.2	3.7
		0.015	8.2	7.8
		0.01	21.5	21.4
		0.008	41.7	40.9
retail_large	0.8	0.01	28.2	27.6
		0.008	47.8	47.2
		0.006	103.9	98.3
		0.005	154.9	142.6
groceries	0.4	0.003	43.1	42.5
		0.002	81.4	79.2
		0.0018	98.2	92.7
		0.0015	164.3	151.2

Table 7. Execution time comparison of *apriori* and proposed algorithm *apriori-D*

time in seconds for *apriori* and *apriori-D* respectively. It has three main rows that represent the results for each dataset and another four rows per dataset that represent the minimum support levels used and execution time in both algorithms.

The improvement of execution time was achieved on the lowest minimum support levels for all of the datasets. For the dataset *retail_small* on minimum support level 0.008, there was a 0.8 seconds improvement. For the dataset *retail_large* on minimum support level 0.005, a 12.3 seconds shorter execution is achieved. Lastly for the dataset *groceries* on minimum support level 0.0015, the proposed algorithm has 13.1 seconds shorter execution time.

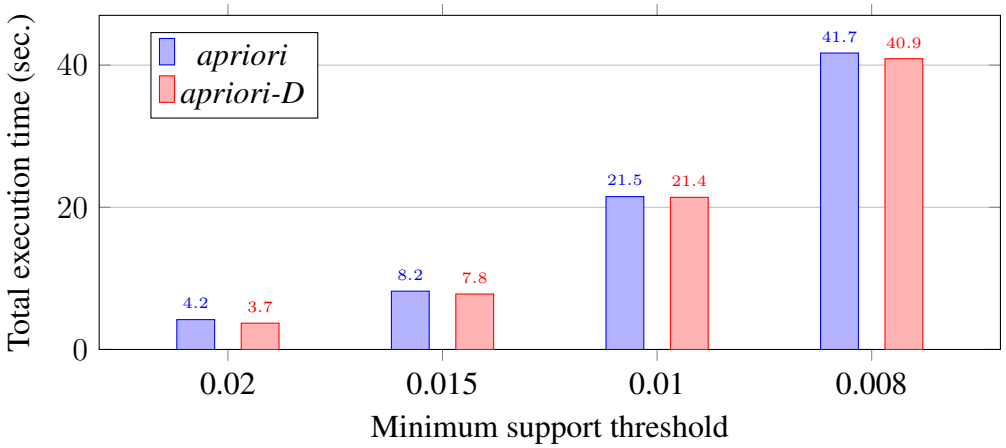

Figure 6. Execution time comparison of *apriori* and proposed algorithm *apriori-D* using dataset *retail_small*

The first graph illustrated in 6, shows the total execution time for both the original algorithm

apriori and the proposed algorithm *apriori-D* executed on dataset *retail_small*, on four minimum support thresholds as 0.02, 0.015, 0.01 and 0.008. The execution time grows as the minimum support level is decreased, due to finding many more frequent itemsets during the first stage of both algorithms that contribute to the computational aspect. The results show that the proposed algorithm is clearly able to improve execution time on all minimum support levels on this dataset.

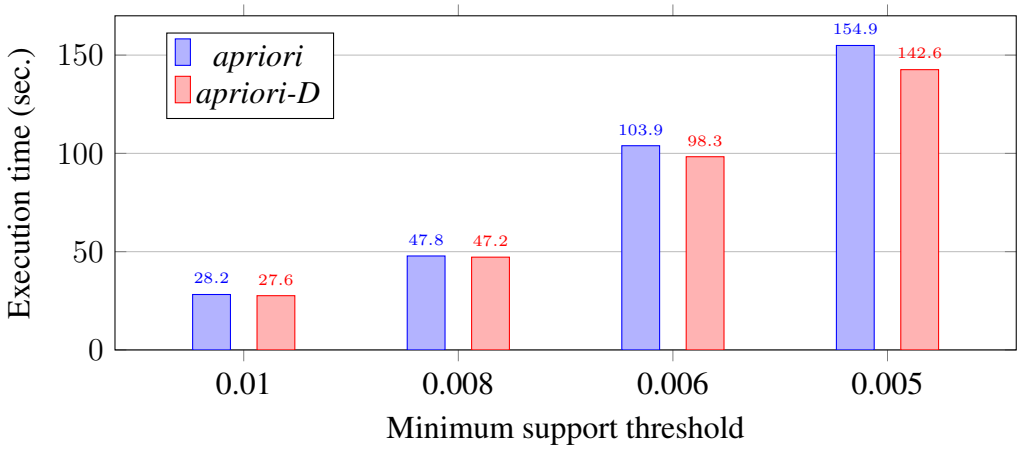

Figure 7. Execution time comparison of *apriori* and proposed algorithm *apriori-D* using dataset *retail_large*

The results illustrated in Figure 7 also show a similar trend on the dataset *retail_large*. It represents the total execution time for both the original *apriori* and proposed algorithm *apriori-D* on four minimum support levels as 0.01, 0.008, 0.006, and 0.005. Again, the proposed algorithm is able to surpass the original one in all minimum support thresholds by achieving a shorter execution time. The difference between the two algorithms gets even wider on lower minimum support levels, where 12.3 seconds of improvement was achieved on 0.005 minimum support level.

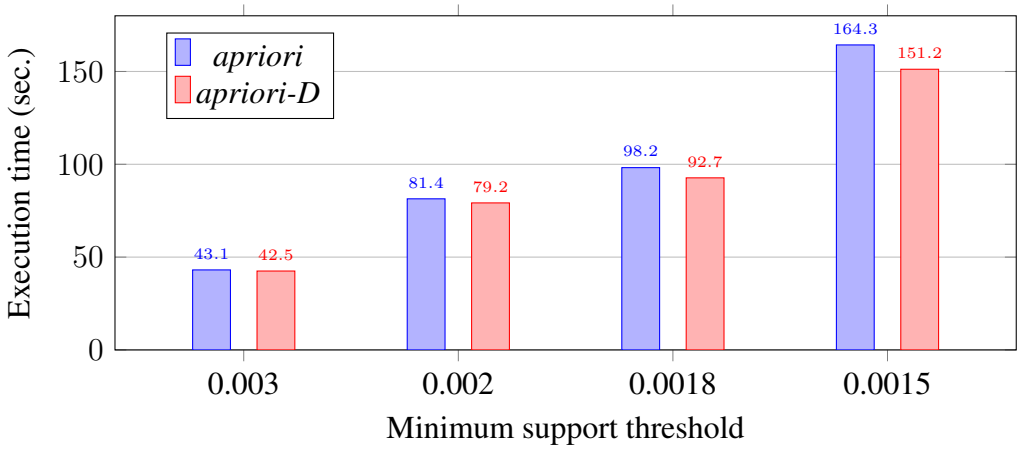

Figure 8. Execution time comparison of *apriori* and proposed algorithm *apriori-D* using dataset *groceries*

Lastly, the total execution time for the original algorithm *apriori* and proposed algorithm *apriori-D* tested on minimum support thresholds as 0.003, 0.002, 0.0018, and 0.0015 using the dataset *groceries* are illustrated in Figure 8. The proposed algorithm provides shorter execution time on all minimum support thresholds but the most prominent one is on 0.0015 where a 13.1 seconds improvement is achieved.

Overall, it can be clearly observed that the proposed algorithm *apriori-D* provides an improvement compared to the original algorithm *apriori* by accomplishing superior total execution time across all datasets and minimum support threshold inputs. These results prove that the dominance relation is performing well during the finding of frequent itemsets step by efficiently pruning the itemsets.

5.3 Comparison of Results In Terms of Generated Rules

In this section, the number of generated association rules for the original algorithm *apriori* and proposed algorithm *apriori-D* is compared.

Datasets	Minimum confidence	Minimum support	Generated association rules	
			Original algorithm *apriori*	Proposed algorithm *apriori-D*
retail_small	0.6	0.02	284	189
		0.015	933	415
		0.01	2835	2632
		0.008	5303	5100
retail_large	0.8	0.01	669	594
		0.008	2051	1158
		0.006	9809	9195
		0.005	14428	13737
groceries	0.4	0.003	817	643
		0.002	1894	1175
		0.0018	2327	1356
		0.0015	3610	1733

Table 8. Generated association rules comparison of *apriori* and proposed algorithm *apriori-D*

Table 8 represents the generated association rules of both original algorithm *apriori* and proposed algorithm *apriori-D* on three datasets and four minimum support levels on each of them. The table contains five columns where the first column represents the name of the dataset, the second column represents the minimum confidence threshold, the third column represents the different minimum support thresholds used, and the fourth and fifth columns represent the number of generated association rules for original and proposed algorithms. It has three main rows that show the results for each dataset and another four

rows per dataset that show the minimum support levels used and generated association rules in both algorithms respectively.

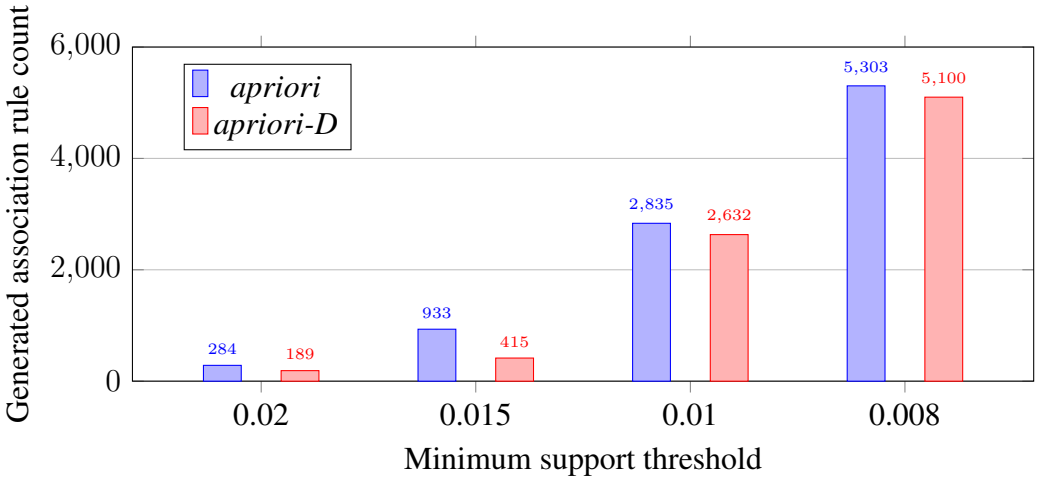

Figure 9. Generated association rule count comparison of *apriori* and proposed algorithm *apriori-D* using dataset *retail_small*

The data in Figure 9 illustrates the number of generated association rules in the original algorithm *apriori* and the proposed algorithm *apriori-D* executed on dataset *retail_small*, on four minimum support thresholds as 0.02, 0.015, 0.01 and 0.008.

Similar to execution time, decreasing the minimum support level results in a higher number of association rules in the output. This is an expected behavior since the number of found frequent itemsets increases due to smaller minimum support levels and these frequent itemsets are later used to generate association rules. On all conditions, it can be observed that the proposed algorithm generates a smaller number of association rules as an outcome of using the dominance relation.

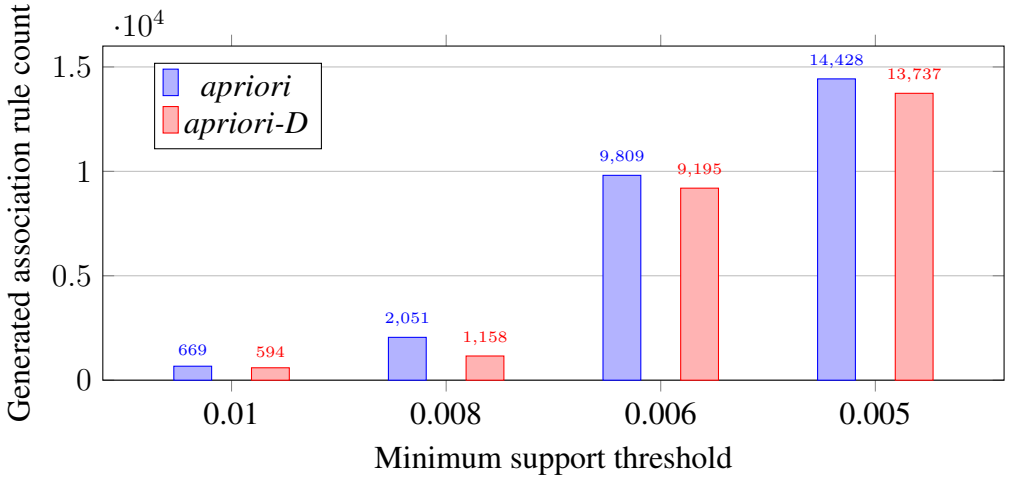

Figure 10. Generated association rule count comparison of *apriori* and proposed algorithm *apriori-D* using dataset *retail_large*

In Figure 10, the number of association rules using the dataset *retail_large* on minimum support levels as 0.01, 0.008, 0.006, and 0.005 are represented. There is a big difference in terms of the generated number of association rules in minimum support thresholds 0.008 and 0.006. The reason behind these results is that there are a very big number of frequent itemsets found between these two levels for this particular dataset.

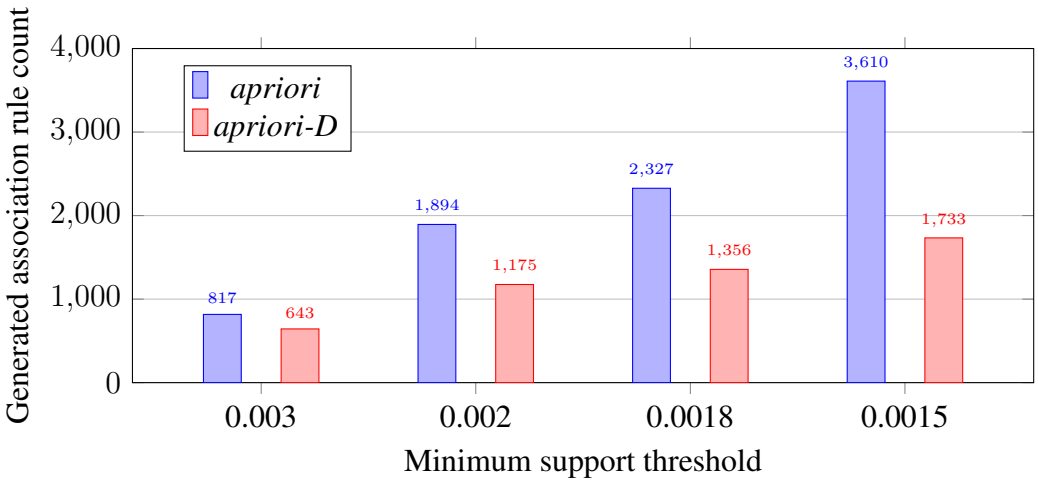

Figure 11. Generated association rule count comparison of *apriori* and proposed algorithm *apriori-D* using dataset *groceries*

Lastly, the results shown in Figure 11 illustrate the number of generated association rules from the dataset *groceries*, for the original algorithm *apriori* and proposed algorithm *apriori-D*. The minimum support levels are 0.003, 0.002, 0.0018 and 0.0015. The reason for selecting very low thresholds on this dataset is because of the sparse distribution of data and the short length of transactions compared to other datasets.

5.4 Comparison of Results In Terms of Discarded Redundant Rules

The very high number of association rules is often inconvenient to analyze. Therefore, having less number of association rules with more relevance is a desirable result. However, it introduces the question of whether this reduction results in association rules with more relevance or not.

In order to evaluate the efficiency of reduction in association rules, the difference of rules in original algorithm *apriori* and proposed algorithm *apriori-D* will be compared, and the number of association rules that are found redundant and eventually discarded in the proposed algorithm will be identified.

The data represented in Table 9 has five columns where the first column show the name of the dataset. The second and third columns show the minimum confidence and minimum

support levels respectively. The fourth column represents the number of different rules in *apriori* and *apriori-D*. The last column shows the number of redundant association rules found in the original algorithm's output *apriori* and discarded in the output of the proposed algorithm *apriori-D*. It has three main rows that represent the results for each dataset and another four rows per dataset that represent the minimum support levels used, the rule difference, and discarded redundant rules in algorithm *apriori-D* respectively.

Datasets	Minimum confidence	Minimum support	Rule difference in *apriori* and *apriori-D*	Discarded redundant rules in algorithm *apriori-D*
retail_small	0.6	0.02	95	6
		0.015	518	10
		0.01	203	203
		0.008	203	203
retail_large	0.8	0.01	75	23
		0.008	893	70
		0.006	614	614
		0.005	691	614
groceries	0.4	0.003	174	6
		0.002	719	6
		0.0018	971	6
		0.0015	1877	6

Table 9. Number of discarded redundant rules in proposed algorithm *apriori-D*

As it can be observed in Table 9, in all scenarios of experimentation, the proposed algorithm *apriori-D* is able to identify and discard redundant association rules. For the dataset *groceries*, the number of discarded rules is the lowest due to the sparse distribution of data.

On the other hand, where the data is distributed densely with overall longer transactions such as *retail_small* and *retail_large*, the proposed algorithm identifies and discards many more redundant association rules. This is especially relevant at lower minimum support thresholds where in some cases all of the rules different in *apriori* and *apriori-D* were found to be redundant and discarded completely which is a very desirable result.

For instance, the execution on the minimum support level as 0.01 and 0.008 on the dataset *retail_small*, it is apparent that 203 rules were different in the output of *apriori* and *apriori-D*, and proposed algorithm found all 203 rules to be redundant and discarded them. Similarly on the execution with the minimum support level 0.006 on the dataset *retail_large*, 614 rules were found to be different, and all of them were identified as redundant and thus discarded.

It is clear that by reducing the minimum support threshold, the number of rule difference

gets bigger and there is a higher probability of finding redundant rules with the proposed algorithm *apriori-D*.

Whereas in the executions with higher minimum support thresholds, there are more frequent itemsets found with a shorter length and bigger number. This results in a scenario where dominance prunes most of these frequent itemsets with shorter lengths, therefore the possibility of finding redundant rules at the end is minimized. This is not an apparent problem on lower minimum support thresholds by pruning the lower number of frequent itemsets with longer lengths.

6. Conclusion

In this thesis, an improved algorithm named *apriori-D* was proposed to enhance the pruning step of finding the frequent itemsets phase of the original Apriori algorithm. The novel approach called dominance relation is utilized to use multiple metrics to extend the pruning process instead of solely relying on the support measure.

The experimentation results have shown that the proposed algorithm *apriori-D* outperforms the original Apriori in terms of execution time on all of the tested conditions. Apart from improvement in the execution time, the number of generated rules aspect was also optimized by achieving fewer number of rules by discarding the redundant ones will make the rule analysis much more convenient for data analysts.

In future work, various metrics can be utilized by extending the proposed algorithm *apriori-D*, in order to evaluate the effect of using different measures on the redundancy of generated association rules.

7. Summary

Association rule mining is a foundational method for data mining that has been widely researched in the current literature. The necessary background regarding data mining, association rule mining as well as its most known algorithms was explained in Chapter 2. The widespread adaption of Apriori has been discussed alongside the main drawbacks it brings and proposals made to overcome these limitations in Chapter 3. Following that, based on the gaps found in the literature, the notion of dominance relation is introduced in Chapter 4 together with a proposed algorithm by this thesis using the dominance relation named as *apriori-D* to be able to utilize multiple metrics during the frequent itemset generation process. The experimentation results performed on several datasets have been presented in Chapter 5. The results demonstrate the proposed algorithm was able to further improve the original Apriori in terms of execution time and the number of generated valuable association rules. Finally, Chapter 6 provided an overall conclusion with possible advancements for a future research work.

References

[1] Manoj Kumar Gupta and Pravin Chandra. "A comprehensive survey of data mining". In: *International Journal of Information Technology* 12.4 (Feb. 2020), pp. 1243–1257. ISSN: 2511-2112. DOI: 10.1007/s41870-020-00427-7.

[2] Min Chen, Shiwen Mao, and Yunhao Liu. "Big Data: A Survey". In: *Mobile Networks and Applications* 19.2 (Jan. 2014), pp. 171–209. ISSN: 1572-8153. DOI: 10.1007/s11036-013-0489-0.

[3] Tim Kraska. "Finding the Needle in the Big Data Systems Haystack". In: *IEEE Internet Computing* 17.1 (Jan. 2013), pp. 84–86. ISSN: 1089-7801. DOI: 10.1109/mic.2013.10.

[4] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. "Knowledge Discovery in Databases: An Overview". In: *AI Magazine* 13.3 (Sept. 1992), p. 57. DOI: 10.1609/aimag.v13i3.1011. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1011.

[5] M. Moulet. "From machine learning towards knowledge discovery in databases". In: *IEE Colloquium on Knowledge Discovery in Databases*. IEE, 1995. DOI: 10.1049/ic:19950116.

[6] Majid Ramzan and Majid Ahmad. "Evolution of data mining: An overview". In: *2014 Conference on IT in Business, Industry and Government (CSIBIG)*. IEEE, Mar. 2014. DOI: 10.1109/csibig.2014.7056947.

[7] Surbhi K. Solanki and Jalpa T. Patel. "A Survey on Association Rule Mining". In: *2015 Fifth International Conference on Advanced Computing and Communication Technologies*. IEEE, Feb. 2015. DOI: 10.1109/acct.2015.69.

[8] Dion H. Goh and Rebecca P. Ang. "An introduction to association rule mining: An application in counseling and help-seeking behavior of adolescents". In: *Behavior Research Methods* 39.2 (May 2007), pp. 259–266. ISSN: 1554-3528. DOI: 10.3758/bf03193156.

[9] Kamran Shaukat Dar, Sana Zaheer, and Iqra Nawaz. "Association Rule Mining: An Application Perspective". In: *International Journal of Computer Science and Innovation* 1 (Nov. 2015), pp. 29–38.

[10] Trupti A. Kumbhare and Santosh V. Chobe. "An Overview of Association Rule Mining Algorithms". In: 2014. URL: https://api.semanticscholar.org/CorpusID:6316475.

[11] Dongmei Ai, Hongfei Pan, Xiaoxin Li, Yingxin Gao, and Di He. “Association rule mining algorithms on high-dimensional datasets”. In: *Artificial Life and Robotics* 23.3 (May 2018), pp. 420–427. ISSN: 1614-7456. DOI: 10.1007/s10015-018-0437-y.

[12] Sandhya Harikumar and Divya Usha Dilipkumar. “Apriori algorithm for association rule mining in high dimensional data”. In: *2016 International Conference on Data Science and Engineering (ICDSE)*. IEEE, Aug. 2016. DOI: 10.1109/icdse.2016.7823952.

[13] Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. “Mining Minimal Non-redundant Association Rules Using Frequent Closed Itemsets”. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000, pp. 972–986. ISBN: 9783540449577. DOI: 10.1007/3-540-44957-4_65.

[14] Aimin Yang, Wei Zhang, Jiahao Wang, Ke Yang, Yang Han, and Limin Zhang. “Review on the Application of Machine Learning Algorithms in the Sequence Data Mining of DNA”. In: *Frontiers in Bioengineering and Biotechnology* 8 (Sept. 2020). ISSN: 2296-4185. DOI: 10.3389/fbioe.2020.01032.

[15] Rashi Rastogi and Mamta Bansal. “Diabetes prediction model using data mining techniques”. In: *Measurement: Sensors* 25 (Feb. 2023), p. 100605. ISSN: 2665-9174. DOI: 10.1016/j.measen.2022.100605.

[16] Ritu Sharma. “Study of Supervised Learning and Unsupervised Learning”. In: *International Journal for Research in Applied Science and Engineering Technology* 8.6 (June 2020), pp. 588–593. ISSN: 2321-9653. DOI: 10.22214/ijraset.2020.6095.

[17] Jagmeet Kaur and Neena Madan. “Association Rule Mining: A Survey”. In: *International Journal of Hybrid Information Technology* 8.7 (July 2015), pp. 239–242. ISSN: 1738-9968. DOI: 10.14257/ijhit.2015.8.7.22.

[18] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. 3rd ed. Elsevier, 2012. ISBN: 9780123814791. DOI: 10.1016/c2009-0-61819-5.

[19] Jose Maria Luna, Philippe Fournier-Viger, and Sebastian Ventura. “Frequent itemset mining: A 25 years review”. In: *WIREs Data Mining and Knowledge Discovery* 9.6 (July 2019). ISSN: 1942-4795. DOI: 10.1002/widm.1329.

[20] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. “Mining association rules between sets of items in large databases”. In: *ACM SIGMOD Record* 22.2 (June 1993), pp. 207–216. ISSN: 0163-5808. DOI: 10.1145/170036.170072.

[21] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. “Dynamic itemset counting and implication rules for market basket data”. In: *ACM SIGMOD Record* 26.2 (June 1997), pp. 255–264. ISSN: 0163-5808. DOI: 10.1145/253262.253325.

[22] Nada Hussein, Abdallah Alashqur, and Bilal Sowan. “Using the interestingness measure lift to generate association rules”. In: *Journal of Advanced Computer Science & Technology* 4.1 (Apr. 2015), pp. 156–162. ISSN: 2227-4332. DOI: 10.14419/jacst.v4i1.4398.

[23] Pang Ning Tan and Vipin Kumar. “Interestingness Measures for Association Patterns: A Perspective”. In: (Aug. 2000).

[24] Rakesh Agrawal and Ramakrishnan Srikant. “Fast Algorithms for Mining Association Rules in Large Databases”. In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB ’94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. ISBN: 1558601538.

[25] Jiawei Han, Jian Pei, and Yiwen Yin. “Mining frequent patterns without candidate generation”. In: *ACM SIGMOD Record* 29.2 (May 2000), pp. 1–12. ISSN: 0163-5808. DOI: 10.1145/335191.335372.

[26] M.J. Zaki. “Scalable algorithms for association mining”. In: *IEEE Transactions on Knowledge and Data Engineering* 12.3 (2000), pp. 372–390. ISSN: 1041-4347. DOI: 10.1109/69.846291.

[27] Otmane Stit, Jamal Riffi, Ali Yahyaouy, and Hamid Tairi. “Comparative Study of Different Association Rule Methods”. In: *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*. IEEE, Oct. 2018. DOI: 10.1109/cist.2018.8596670.

[28] Mohammed J. Zaki and Karam Gouda. “Fast vertical mining using diffsets”. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD03. ACM, Aug. 2003. DOI: 10.1145/956750.956788.

[29] Seema A. Tribhuvan, Nitin R. Gavai, and Bharti P. Vasgi. “Frequent Itemset Mining Using Improved Apriori Algorithm with MapReduce”. In: *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*. IEEE, Aug. 2017. DOI: 10.1109/iccubea.2017.8463915.

[30] Hongqin Wang, Huiyong Jiang, Hongxia Wang, and Lina Yuan. “Research on an improved algorithm of Apriori based on Hadoop”. In: *2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*. IEEE, Aug. 2020. DOI: 10.1109/ispds51347.2020.00057.

[31] Shaosong Yang, Guoyan Xu, Zhijian Wang, and Fachao Zhou. “The Parallel Improved Apriori Algorithm Research Based on Spark”. In: *2015 Ninth International Conference on Frontier of Computer Science and Technology*. IEEE, Aug. 2015. DOI: 10.1109/fcst.2015.28.

[32] Sanjay Rathee, Manohar Kaul, and Arti Kashyap. “R-Apriori: An Efficient Apriori based Algorithm on Spark”. In: *Proceedings of the 8th Workshop on Ph.D. Workshop in Information and Knowledge Management*. CIKM’15. ACM, Oct. 2015. DOI: 10.1145/2809890.2809893.

[33] Mayank Tiwary, Abhaya Kumar Sahoo, and Rachita Misra. “Efficient implementation of apriori algorithm on HDFS using GPU”. In: *2014 International Conference on High Performance Computing and Applications (ICHPCA)*. IEEE, Dec. 2014. DOI: 10.1109/ichpca.2014.7045323.

[34] Yuxin Wang, Tongkun Xu, Shiqing Xue, and Yanming Shen. “D2P-Apriori: A deep parallel frequent itemset mining algorithm with dynamic queue”. In: *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, Mar. 2018. DOI: 10.1109/icaci.2018.8377536.

[35] Sara Tanha, Richard Tirtho Biswas, Tonosree Roy Ritu, and Shaheena Sultana. “Improved Apriori Algorithm Using Hash Technique”. In: *2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)*. IEEE, June 2023. DOI: 10.1109/ncim59001.2023.10212624.

[36] Kun Niu, Haizhen Jiao, Zhipeng Gao, Cheng Chen, and Huiyang Zhang. “A developed apriori algorithm based on frequent matrix”. In: *Proceedings of the 5th International Conference on Bioinformatics and Computational Biology*. ICBCB ’17. ACM, Jan. 2017. DOI: 10.1145/3035012.3035019.

[37] Liu Shuwen and Xiao Jiyi. “An Improved Apriori Algorithm Based on Matrix”. In: *2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*. IEEE, Feb. 2020. DOI: 10.1109/icmtma50254.2020.00111.

[38] Mohammed Al-Maolegi and Bassam Arkok. *An Improved Apriori Algorithm for Association Rules*. 2014. DOI: 10.48550/ARXIV.1403.3948.

[39] Akshita Bhandari, Ashutosh Gupta, and Debasis Das. “Improvised Apriori Algorithm Using Frequent Pattern Tree for Real Time Applications in Data Mining”. In: *Procedia Computer Science* 46 (2015), pp. 644–651. ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.02.115.

[40] Xuexian Qiu, Shiyong Ning, Shihui Zhang, and Zhuorui Yang. “Research and Application of an Improved Apriori Algorithm in Market Basket Data”. In: *2023 4th International Conference on Machine Learning and Computer Application*. ICMLCA 2023. ACM, Oct. 2023. DOI: 10.1145/3650215.3650319.

[41] Jai Puneet Singh and Hari Ram. “Improving Efficiency of Apriori Algorithm Using Transaction Reduction”. In: 2013. URL: https://api.semanticscholar.org/CorpusID:17968176.

[42] Sakshi Aggarwal and Ritu Sindhu. “An approach to improve the efficiency of apriori algorithm”. In: (June 2015). DOI: 10.7287/peerj.preprints.1159v1.

[43] Shyam Kumar Singh and Preetham Kumar. “I2Apriori: An improved apriori algorithm based on infrequent count”. In: *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. 2016, pp. 1281–1285. DOI: 10.1109/ICEEOT.2016.7754889.

[44] Fei Gao, Ashutosh Khandelwal, and Jiangjiang Liu. “Mining Frequent Itemsets Using Improved Apriori on Spark”. In: *Proceedings of the 2019 3rd International Conference on Information System and Data Mining*. ICISDM 2019. ACM, Apr. 2019. DOI: 10.1145/3325917.3325925.

[45] Md. Mahamud Hasan and Sadia Zaman Mishu. “An Adaptive Method for Mining Frequent Itemsets Based on Apriori And FP Growth Algorithm”. In: *2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*. IEEE, Feb. 2018. DOI: 10.1109/ic4me2.2018.8465499.

[46] Jovita Vani Sequeira and Zahid Ahmed Ansari. “Analysis on Improved Pruning in Apriori Algorithm”. In: 2015. URL: https://api.semanticscholar.org/CorpusID:61252211.

[47] Meng Xiao, Yong Yin, Yunyao Zhou, and Shengzhi Pan. “Research on improvement of apriori algorithm based on marked transaction compression”. In: *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, Mar. 2017. DOI: 10.1109/iaeac.2017.8054177.

[48] Ke Zhang, Jianhuan Liu, Yi Chai, Jiayi Zhou, and Yi Li. “A Method to Optimize Apriori Algorithm for Frequent Items Mining”. In: *2014 Seventh International Symposium on Computational Intelligence and Design*. IEEE, Dec. 2014. DOI: 10.1109/iscid.2014.233.

[49] Lalit Mohan Goyal and M. M. Sufyan Beg. “Evaluation of filtration and pruning approach for Apriori algorithm”. In: *2014 International Conference on Computer and Communication Technology (ICCCT)*. IEEE, Sept. 2014. DOI: 10.1109/iccct.2014.7001464.

[50] Jiao Yabing. “Research of an Improved Apriori Algorithm in Data Mining Association Rules”. In: *International Journal of Computer and Communication Engineering* (2013), pp. 25–27. ISSN: 2010-3743. DOI: 10.7763/ijcce.2013.v2.128.

[51] Jing Zhang, Bin Zhang, Zihua Wang, and Lijun Shi. “Elimination Algorithm of Redundant Association Rules Based on Domain Knowledge”. In: *2010 Seventh Web Information Systems and Applications Conference*. IEEE, Aug. 2010. DOI: 10.1109/wisa.2010.23.

[52] Julio César Díaz Vera, Guillermo Manuel Negrín Ortiz, Carlos Molina, and María Amparo Vila. “Knowledge redundancy approach to reduce size in association rules”. In: *Informatica* 44.2 (June 2020). ISSN: 0350-5596. DOI: 10.31449/inf.v44i2.2839.

[53] Mafruz Zaman Ashrafi, David Taniar, and Kate Smith. “A New Approach of Eliminating Redundant Association Rules”. In: *Database and Expert Systems Applications*. Springer Berlin Heidelberg, 2004, pp. 465–474. ISBN: 9783540300755. DOI: 10.1007/978-3-540-30075-5_45.

[54] Mafruz Zaman Ashrafi, David Taniar, and Kate Smith. “Redundant Association Rules Reduction Techniques”. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 254–263. ISBN: 9783540316527. DOI: 10.1007/11589990_28.

[55] Ye Xin, Wang Na, and Wang Chunyu. “A New Method for Eliminating Redundant Association Rules”. In: *2010 International Conference on Intelligent Computation Technology and Automation*. IEEE, May 2010. DOI: 10.1109/icicta.2010.129.

[56] Rafael Garcia Leonel Miani and Estevam Rafael Hruschka Junior. “Eliminating Redundant and Irrelevant Association Rules in Large Knowledge Bases”. In: *Proceedings of the 20th International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications, 2018. DOI: 10.5220/0006668800170028.

[57] Mohammad Reza Heidari Iman, Jaan Raik, Maksim Jenihhin, Gert Jervan, and Tara Ghasempouri. “An automated method for mining high-quality assertion sets”. In: *Microprocessors and Microsystems* 97 (Mar. 2023), p. 104773. ISSN: 0141-9331. DOI: 10.1016/j.micpro.2023.104773.

[58] S. Bouker, R. Saidi, S. B. Yahia, and E. M. Nguifo. “Ranking and Selecting Association Rules Based on Dominance Relationship”. In: *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*. IEEE, Nov. 2012. DOI: 10.1109/ictai.2012.94.

[59] Dirk Ifenthaler. “Measures of Similarity”. In: *Encyclopedia of the Sciences of Learning*. Springer US, 2012, pp. 2147–2150. DOI: 10.1007/978-1-4419-1428-6_503.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

I Cem Şamiloğlu

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Improvement in the Apriori Algorithm to Enhance the Efficiency of Association Rule Mining Techniques", supervised by Mohammadreza Heidari Iman and Tara Ghasempouri
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

05.05.2024

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - Sources of Datasets and Implementations

The source of dataset *retail_small* to be found at:

`https://archive.ics.uci.edu/dataset/352/online+retail`

The source of dataset *retail_large* to be found at:

`https://archive.ics.uci.edu/dataset/502/online+retail+ii`

The source of dataset *groceries* to be found at:

`https://www.kaggle.com/datasets/irfanasrullah/groceries`

The implementation of the original algorithm *apriori* is to be found at:

`https://github.com/cemsam/apriori-dominance/blob/master/apriori-original.py`

The implementation of the proposed algorithm *apriori-D* is to be found at:

`https://github.com/cemsam/apriori-dominance/blob/master/apriori-dominance.py`