Final Year Dissertation

Market Basket Analysis with Graph Theory

April 13, 2021

Sahil M. Pattni

Bachelor of Science with Honours in Computer Science

Supervised by Dr. Neamat El Gayar

Declaration

I, Sahil Manojkumar Pattni, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Date: April 13, 2021

Signed: Sahil Manojkumar Pattni

Abstract

In this digital age, data is being generated and collected at an unprecedented rate, with data analytics employed by corporations and small businesses alike to produce actionable insights, reduce costs, optimize operations and increase revenue. Association rules allow us to identify relationships between products that can provide insights into customer spending habits and product perception.

In this study, a minimum spanning tree (MST) will be generated from a transactional database such that only the strongest relationships between products remain. A clustering algorithm will be applied to this MST to identify high co-purchase segments, and association rules will then be extracted from these segments. [**ADD MORE HERE**]

Contents

Chapter 1

Introduction

1.1 Motivation

For businesses such as groceries, hypermarkets, and retail outlets that deal with the trade of heterogeneous physical assets, operations such as inventory management and product placement play an instrumental role in determining the business' financial success. These involve asking questions such as:

- Which products should be placed at the entrance of the store? Which should be placed closer to the exit?
- Which products will benefit the most by being placed at eye-level?
- Which products should be placed next to each other to maximize the purchase volume?

One way to find optimal solutions to such queries is to employ the use of Association Rule Mining (also known as Market Basket Analysis). This set of techniques assess frequent itemsets (e.g. from sales data) and generate association rules between products. A prime example of the utility of association rules is the urban legend of *"Beers and Pampers"*, where a company allegedly studied their point-of-sale data and found a strong association between the purchase of diapers and a particular brand of beer during a certain time. With *diapers* as the antecedent and *beer* as the consequent, this rule can be written as:

$$\{Diapers\} \rightarrow \{Beer\}$$

This is an example of a single-element rule, where both the antecedent and consequent are sets that contain only one element each. Several algorithms exist for association rule mining, most prominently the Apriori Algorithm [1] and FP-Growth [2], however these algorithms tend to generate an overwhelming amount of rules, rendering it inconvenient for the end-user to extract actionable information from the ruleset. This paper proposes to improve upon an existing method (see: Section 2.4.5) that derives association rules from minimum spanning trees.

1.2 Aims

The aim of this paper is to improve upon the aforementioned method by introducing a method that allows for the generation of multi-element association rules (i.e. rules where either/both the antecedent and consequent of the rule contain more than one element).

1.3 Objectives

The research objectives for this paper are as described below:

1. Acquire a suitable dataset upon which the study can be conducted.
2. Construct an affinity graph from the chosen dataset.
3. Explore and evaluate MST extraction algorithms.
4. Identify and evaluate methodologies for generating multi-element association rules from an MST.
5. Extract MST from the graph and generate association-rules using the chosen methodologies.
6. Evaluate the generated association rules against the rules generated by the Apriori Algorithm.

Chapter 2

Background

2.1 Graph Theory

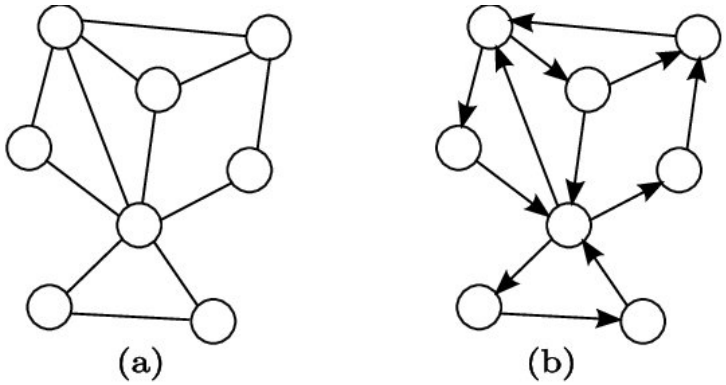

Figure 2.1: Undirected and Directed Graphs

In discrete mathematics and more specifically - graph theory, a graph is a data structure that contains a set of nodes (i.e. vertices) connected by lines (i.e. edges). These edges may be directed - such as in Figure 2.1:(a), or undirected - such as in Figure 2.1:(b). A graph G with a set of vertices V and a set of edges E can be represented via the notation $G = (V, E)$. For the scope of this project, we will be building undirected graphs, where the weight between two vertices is the same in both directions.

2.1.1 Minimum Spanning Trees

Given an undirected $G = (V, E)$, a *spanning tree* can be described as a subgraph that is a tree which includes all the vertices V of G with the minimum number of edges required. A *minimum spanning tree* (MST) is the spanning tree with the smallest sum of edge weights. This means that if the graph has n vertices, each spanning tree - including the minimum spanning tree - will have $n - 1$ edges. There are two widely used algorithms to extract the minimum spanning tree from a graph: Prim's algorithm and Kruskal's algorithm.

2.1.2 Prim's Algorithm

Independently discovered by three authors, Prim's algorithm [3][4][5] is a greedy algorithm[1] to find the minimum spanning three of an undirected, weighted graph G. To successfully implement the algorithm, three sets need to maintained: a set of *discovered* edges, and two sets of vertices: a set of *undiscovered* vertices, and a set of *discovered* vertices. Figure 2.2 illustrates Prim's algorithm being applied to a graph. The algorithm is as follows:
Initialize an empty set of discovered edges: E, and two sets of vertices: an empty set D of the discovered vertices, and UD as the set of undiscovered vertices.

- Pick an arbitrary vertex as a starting point (in the case of Figure 2.2, the top right node). Add this vertex to D and remove it from UD.
- While UD is not empty:
  - Find the edge e_i with the smallest weight such that it connects together a vertex V_1 in D and V_2 in UD (to avoid forming cycles).
  - Append V_2 to D and remove it from UD (i.e. V_2 is now discovered).
  - Append e_i to E.

Once D contains all the vertices of G, the algorithm terminates, and the set D represents the minimum spanning tree, and $\sum_{i=1}^{n} e_i$ is the weight of the MST. The time complexity of this algorithm is $O(V^2)$.

[1] Selecting the locally optimal choice at each iteration of the solution

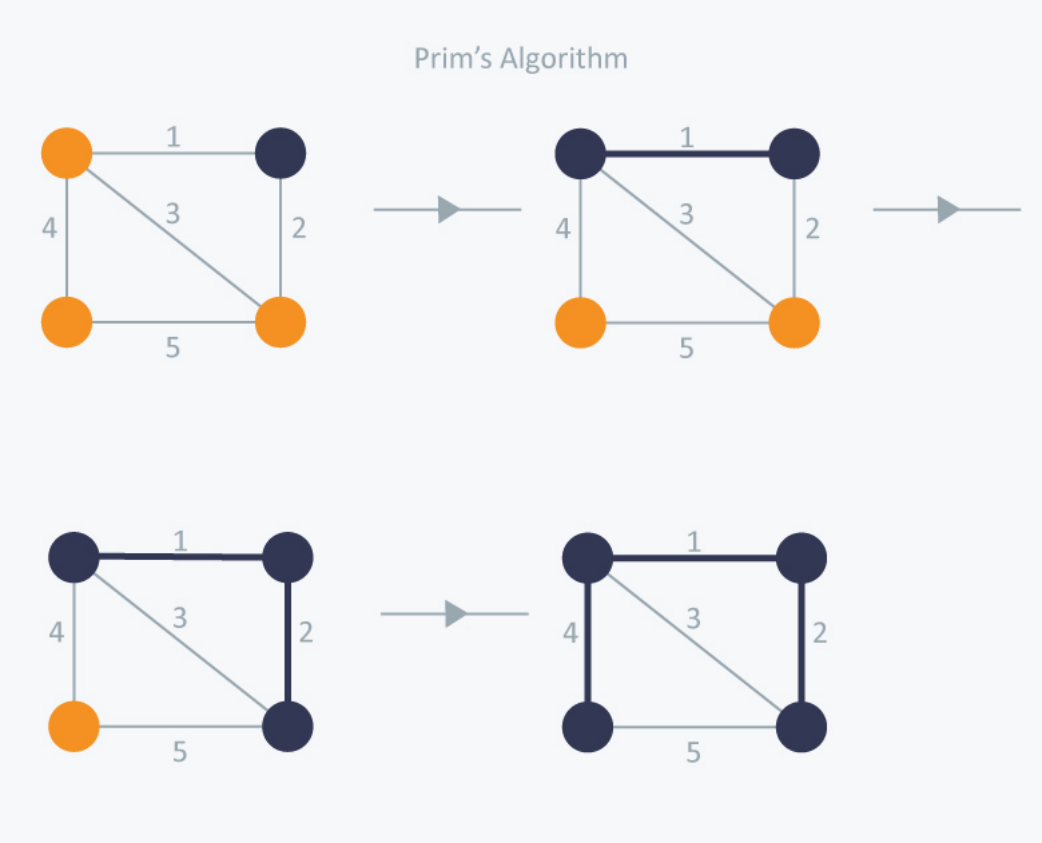

Figure 2.2: Prim's Algorithm applied to a graph [6]

2.1.3 Kruskal's Algorithm

Another greedy algorithm, Kruskal's algorithm [7] also extracts the MST from a graph. Unlike Prim's algorithm, Kruskal's doesn't select an edge that connects directly to the already built spanning tree, but rather picks the global optimal solution. The algorithm is as follows:
Maintaining a set of edges E, and an initially empty set of chosen edges C:

- Sort the set of edges E in ascending order.
- While the number of elements in C is not $n-1$:
  - Select the smallest edge e_i from E.
  - If adding it does not form a cycle with the spanning tree formed so far, append e_i to C.
  - Remove e_i from E.

The algorithm will terminate when $n-1$ edges have been selected. The time complexity for this algorithm is $O(ElogE)$.

2.2 Binary Purchase Vectors

A binary purchase vector is a vector with an element for each distinct product in our dataset, with a value of 1 if a given product was present in the purchase, and 0 otherwise. For example, consider a grocer who only sells five items: milk, eggs, bread, apples and oranges. If a customer purchases bread and oranges, the binary purchase vector for this transaction would be:

milk	eggs	bread	apples	oranges
0	0	1	0	1

2.3 Market Basket Analysis and Apriori Rule

Market Basket Analysis (MBA), also known as Affinity Analysis, is a data mining and analysis technique that identifies co-occurrence patterns between products purchased together, and produces association rules for these products as such: $\{A\} \rightarrow \{B\}$ which implies a strong relationship between the purchase of product A and product B. The contents of a purchase basket (i.e. the contents of a customer's basket at the time of transaction) is called a *itemset*, which - as the name suggests - is a set of all the items in the basket. For example, if a customer had bought bread, detergent and soda, the itemset would be $\{bread,\ detergent,\ soda\}$. Association rules are generated by looking at different combinations of the itemset (e.g. $\{bread,\ soda\} \rightarrow \{detergent\}$ and $\{soda\} \rightarrow \{bread\}$).

For an itemset of length $\boldsymbol{d}$, the equation [8] to calculate the number of rules that can be generated from this itemset is as follows:

$$number\ of\ rules = \sum_{k=1}^{d-1} \left(\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right) \tag{2.1}$$

To illustrate the exponential growth of the number of rules given the length of the itemset, Figure 2.3 illustrates the number of rules for itemsets whose length range from 2 to 14, plotted using Equation 2.1 above.

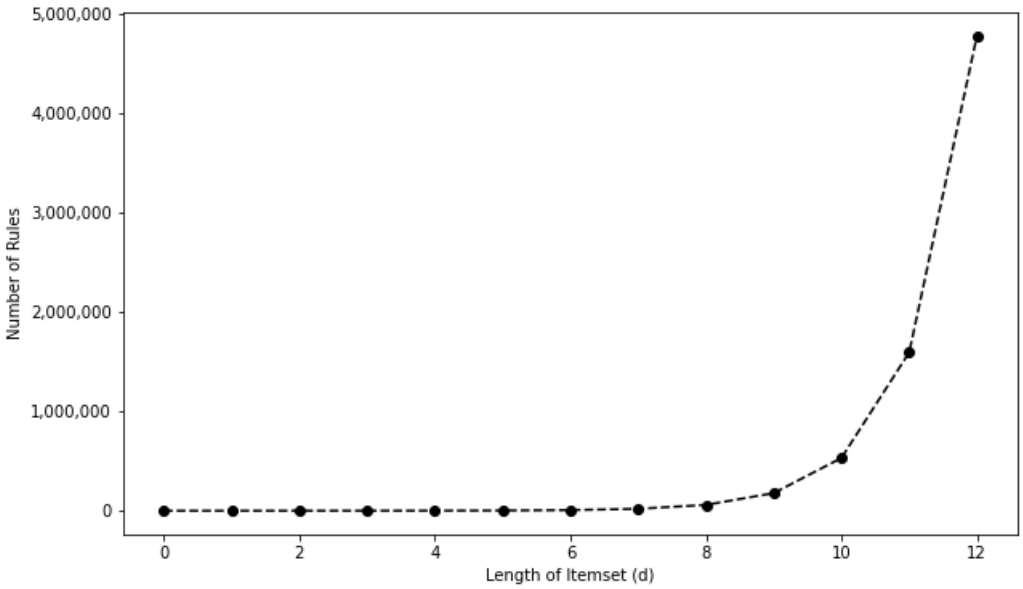

Figure 2.3: Number of Association Rules for an Itemset

Given this exponential growth in rules ($O(3^n)$), we can conclude that iterating through all the itemsets, and so a valid solution might be apply the Apriori Algorithm to prune the itemsets. To understand how the algorithm works, the following three metrics must be considered: The support of a set of products is the fraction of transactions in which these set of products are present. For example, for a list of transactions T, for the product X where $T(X)$ denotes that the set of transactions in which X was present:

$$support(\{A\}) = \frac{T(A)}{T}$$

and similarly,

$$support(\{A\} \rightarrow \{B\}) = \frac{T(A,B)}{T}$$

denotes the support for the rule $\{A\} \rightarrow \{B\}$. Rules with a low support score can be pruned as they indicate a rule does not occur enough to draw any reasonable conclusions from.

Confidence

Confidence is the measure of how likely a product will be in a basket given that another product is in it. That is to say, the confidence of a rule $\{A\} \rightarrow \{B\}$ is the conditional probability that $\{B\}$ occurs in the

basket given that $\{A\}$ is present. The confidence of a rule can be denoted as:

$$\textit{confidence}(\{A\} \rightarrow \{B\}) = \frac{T(A,B)}{T(A)} \equiv \frac{\textit{support}(\{A\} \rightarrow \{B\})}{\textit{support}(\{A\})}$$

Lift

The lift of a rule $\{A\} \rightarrow \{B\}$ is the rise (i.e. **lift**) that $\{A\}$ gives to the *confidence*$(\{A\} \rightarrow \{B\})$.

$$\textit{lift}(\{A\} \rightarrow \{B\}) = \frac{confidence(\{A\} \rightarrow \{B\})}{\textit{support}(\{B\})}$$

For example, consider that *confidence*$(\{A\} \rightarrow \{B\}) = 0.5$, and *support*$(\{B\}) = 0.4$. This means that the presence of $\{A\}$ increases the probability of $\{B\}$ being in the same basket by 25% ($\frac{0.5}{0.4} = 1.25$), therefore providing us with a lift value of 1.25. A lift value below 1 would indicate that the occurrence of $\{A\}$ in a basket decreases the likelihood of $\{B\}$ occurring in the same basket (i.e. a low product association).
The Apriori principle states that *all subsets of a frequent itemset must be frequent.* The Apriori principle is a result of the *anti-monotone property of support* [9], which means for $\{A, B, C\}$, *support*$(\{A, B\}) \geq$ *support*$(\{A, B, C\})$. Therefore, the Apriori principle can be employed to prune the frequent itemsets much more efficiently, because if *support*$(\{A, B\}) < min_{support}$, then any itemset containing the set $\{A, B\}$ will also fall below $min_{support}$. Once the frequent itemsets have been pruned, association rules can be generated from the remaining itemsets. The resulting association rules can be even further pruned by removing those that fall below a confidence threshold $min_{confidence}$. Finally, the remaining rules can be ranked according to their lift to find the rules with the highest associations. Figure 2.4 is an illustration of how pruning one set $\{A, B\}$ leads to the pruning of all its children.

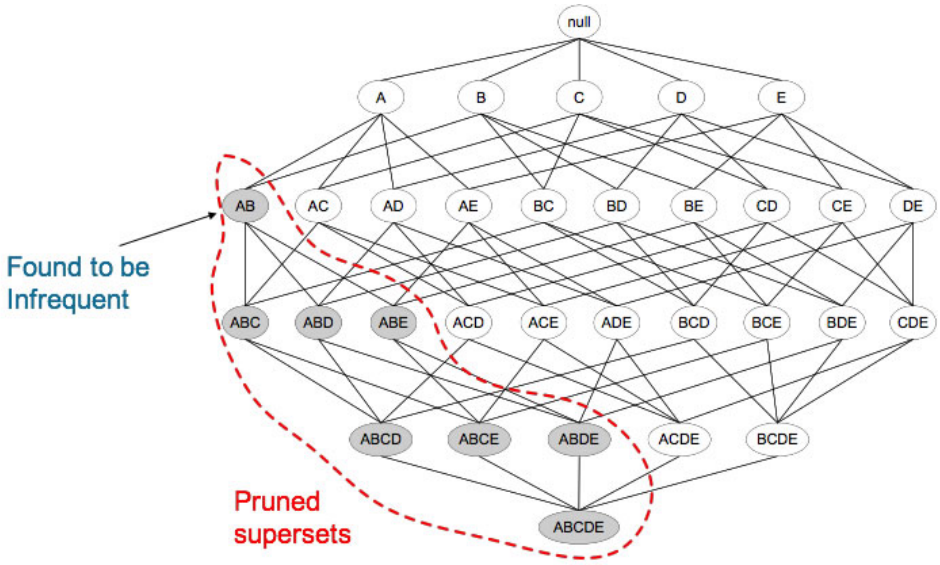

Figure 2.4: Pruning Infrequent Rules (Source: [10])

2.4 Related Work

2.4.1 Extracting Minimum Spanning Trees using K-Means

C. Zhong et al. [11] proposed a novel framework to extract the minimum spanning tree of a graph based on the K-Means clustering algorithm. Their methodology can be separated into two distinct phases. In the first phase, the data is partitioned into $\sqrt{n}$ clusters via K-Means and the Kruskal's algorithm is applied to each of the clusters individually. Once the $\sqrt{n}$ MSTs have been constructed, they are combined to form an approximate MST. In the second phase, new partitions are constructed based on the borders of the clusters identified in phase 1. Based on these new partitions, a second approximate MST is constructed. Finally, both graphs are merged such that the resulting graph has $2(n-1)$ edges. The Kruskal's algorithm is run on this graph to get the final approximation of the MST.

Critical Analysis

The authors have proposed an efficient way to approximate an MST, with their methodology having a complexity of $O(N^{1.5})$, which is faster than the standard MST algorithm which has a complexity of $O(N^2)$. For clarity, we have illustrated the disparity between the author's algorithm and the standard algorithm on Figure 2.5.

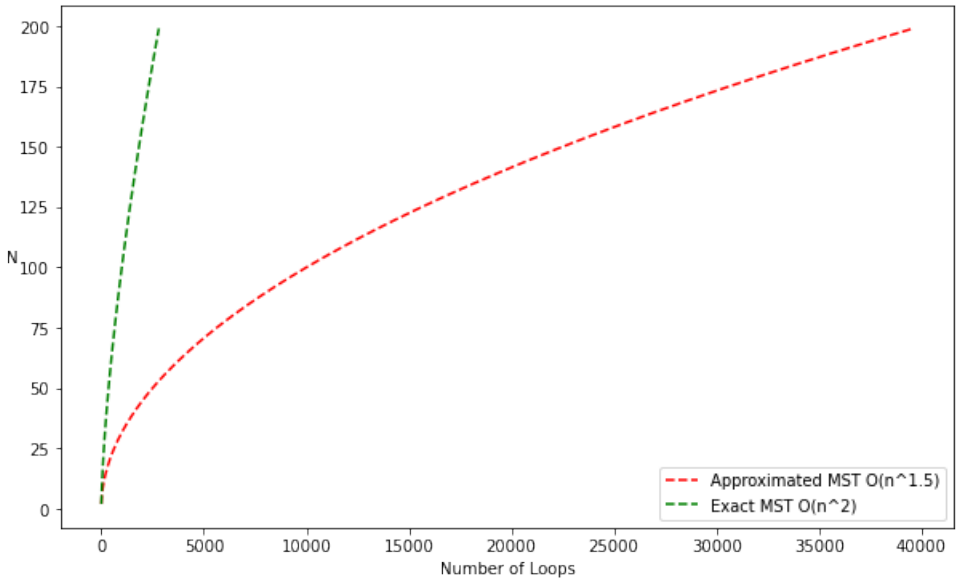

Figure 2.5: Efficiency of K-Means optimized MST vs. exact MST

The K-Means optimized algorithm is $\frac{\sqrt{n}-1}{\sqrt{n}}\%$ faster than the standard MST algorithm.

2.4.2 Markov Clustering

S. Dongen [12] introduced the novel Markov Clustering Algorithm (MCL), a graph clustering algorithm that favours sparse graphs (i.e. graphs where the average degree is smaller than the number of nodes). The reason for this follows the notion that dense regions in sparse graphs correspond with regions in which the number of k-length paths is relatively large - and therefore - paths whose beginnings and ends are in the same dense region have a higher probability for random walks[2] of length k as opposed to other paths. In other words, random walks originating from a node in a dense region have a high probability of ending in that same dense region. Given a non-negative weighted directional graph G, the MCL simulates flow in the graph. It does this by first mapping the graph G in a generic way onto a Markov matrix M_1. Once completed, the set of transition probabilities (TP's) are iteratively recomputed via expansion and contraction, resulting in an array of Markov Graphs. In the expansion stage, higher TP's are calculated, whereas for the contraction phase, a new Markov Graph is created by rewarding high TP's and penalizing low TP's. The belief is that by doing so, the flow between dense regions that are sparsely connected will be removed, leaving only distinct and unconnected dense regions (i.e. clusters).

[2] A random walk is a stochastic process of of successively random steps along a space, in this case: a directed graph.

2.4.3 Apriori Algorithm

R. Aggarwal et al. [13] proposed a novel algorithm to generate all statistically significant association rules between items in a database, laying the foundations for association rule mining. Given a set of items $I = I_1, I_2, I_3, ...I_m$, the authors define an association rule to be of the form $X \rightarrow I_j$ where X is a set of items such that $X \in I, I_j \notin X$. The hypothetical database stated was a list of transactions, T, where each transaction t was a binary vector of length m, representing a basket purchase, where $t[k] = 1$ if I_k been purchased in that basket. The authors stated that their methodology for association rule mining could be split into two steps: the generation of candidate itemsets, and the generation of statistically significant association rules from these itemsets.

To address the first subproblem, the authors provided the pseudo-code for their candidate itemset generation, where all itemsets possible were generated from tuples (samples) from the database, and those itemsets whose support score[3] is above the minimum support threshold are considered candidates (called *large itemsets*). Since a brute-force check would be sub-optimal (the authors note this could take up to 2^m passes of the database), the authors devised a methodology to check for candidate itemsets where on the k^{th} pass of the database, they would only check itemsets of length k, to see if they satisfied the support constraint. On the $(k+1)^{th}$ pass, they need only check those itemsets that are 1-extensions (i.e. itemsets extended by exactly one item) of the *large itemsets* found in the previous pass. This is because of what would later be known as the *Apriori Principle*, where if an itemset Y is *large*, then all subsets of Y must also be *large*. Therefore, if they found the itemset $\{A, B\}$ was *small* (i.e. did not satisfy the support constraint), then sets containing $\{A, B\}$ (e.g. $\{A, B, C\}, \{A, B, D\}, \{A, B, C, D\}$) would also be small, and need not be checked. This means, however, that if an itemset I is *large*, then another pass over the dataset would be required to check the support of the subsets of I. To avoid this, the authors devised a measure to calculate the expected support, $\bar{s}$, of an itemset, and use this to measure the support of itemsets $I = (X + I_j)$ not only where I is expected to be large, but also where $X + I_j$ is expected to be small but X is expected to be large, further helping them prune the number of itemsets to check. The authors proceed to define a method that allows the algorithm to be more memory efficient[4]. The authors also defined method to further prune itemsets from the search, namely *remaining tuples optimization* and *pruning function optimization.*

To address the second subprobem, the authors proposed the following methodology: for each large itemset

[3] see: Section 2.3

[4] This may no longer be required due to the advances in computational power in the the 27 years since this paper was written.

$Y = I_1, I_2, ...I_k, k \geq 2$ from the set of non-pruned large itemsets, generate a set of association rules of form $X \rightarrow I_j$ such that the consequent is I_j and the antecedent (i.e. the precedent set in the rule) is a subset X of Y such that X is of length $k-1$ and $I_j \notin X$. Therefore, each large itemset will produce k rules. From the generated rules, the authors discarded those rules whose confidence scores[5] fell below the minimum confidence threshold c.

The authors tested their methodology on a sales dataset with $46,783$ transactions, with 63 *items* (in this case, the department from which the customer bought an item). They used a configuration of a minimum support threshold of 1% and a minimum confidence threshold of 50%. The rules produced seemed to follow with general intuition, such as:

{Auto Accessories, Tires} $\rightarrow$ {Automotive Services}

Furthermore, the authors assessed the accuracy of their expectation method, by measuring the ratio of correctly estimated itemsets for both small and large, against various values for the minimum support threshold, and visualizing the result. To isolate the effect of their expectation method, they disabled their pruning optimization functions. They were able to conclude that their estimation accuracy was satisfactory, as their accuracy was above 98% for support thresholds except the first, where it was 96%. The authors also tested the effectiveness of their pruning optimization functions, namely the *remaining tuples* and the *pruning function* optimization functions, against multiple minimum support threshold values. They were able to conclude that their pruning efficiency increased as the support threshold increased.

Critical Analysis

The authors have proposed a novel methodology that has been the bedrock of numerous research publications, including most of the papers in this literature review. Their estimation function performed with high accuracy, meaning it can reduce the number of passes through a database the algorithm has to take by a significant amount. Additionally, their pruning techniques allowed them to eliminate a large proportion of itemsets from the space. Even after the significant pruning of rules, a major drawback of this methodology is the large number of rules produced, although one could argue that only the highest performing rules need be observed in further detail.

2.4.4 Subjective Measurement of Association Rules

M. Zekić-Sušac et al. [14] proposed a novel measure for the *interestingness* of association rules, identifying that a dominant, universally used measure did not exist. The authors' goal was to combine

[5] see: Section 2.3

objective measures such as the support, confidence and lift scores with more subjective measures. Instead of the Apriori approach, their methodology has them generate association rules via the *tree-building technique* - which compresses a large database into a Frequent-Pattern tree, citing that this technique was more efficient than the Apriori algorithm. The author's employed the heuristical unexpectedness measure(i.e. significantly contradicting a user's prior beliefs) and the heuristical actionability measure (i.e. if the end user can use the information to their advantage(e.g. a promotion)) as their subjective measures, and a minimum confidence threshold of 51% as their subjective measure. Since a subjective measure would require a human subject, the authors' used the estimations of a sales manager from a Croatian retail chain, and stored his responses in binary format for the subjective measures (i.e. 0 if a rule was unexpected else 1, 0 if a rule is not useful else 1). The dataset used for this paper was a real transactional dataset with 14,012 transactions and a set of 1,230 unique items (which was later pruned to 7,006 transactions and a set of 278 products) from the same Croatian retail chain their test subject worked at. The authors then generated association rules from the first-level hierarchical grouping of items from the dataset (items with a minimum support of 25%), of which 36 rules were identified as statistically significant. From this set of rules, only two rules satisfied both subjective measures and the confidence constraint, and therefore these two rules were identified as highly interesting. The authors then generated association rules from the second-level hierarchical grouping of items, where items that represented the same product (but had different a manufacturer, brand etc.) were grouped together. Of the rules generated, 15 satisfied their confidence constraints and had a support value able 10%. 5 rules from this set satisfied both their subjective and objective measures, more than the previous experiment The authors were able to conclude that the increase in accuracy and number of interesting rules resulted from the second level of grouping which generalized the products.

Critical Analysis

Wheras the original measure of statistical significance introduced by R. Aggarwal et al. was purely objective, the authors of this paper have presented a well thought out approach to combining the subjective metrics with objective ones to produce a human-verified association rule set. A few caveats to note, however: their study only involved one subject, which is rarely regarded to be statistically acceptable. An ideal study would require multiple, randomly chosen subjects to offset any bias that the singular subject would have had, and in addition, the larger their subject size, the closer their collective estimations will model the total population's. Another drawback of their approach is that by using human intuition as a metric, they're promoting association rules that satisfy pre-existing notions about human behavior (e.g. if someone buys milk, they'll *probably* get eggs too), however these types of rules are usually regarded as

common knowledge, whereas the beauty of association rule mining is in its ability to surface association rules that - while true - seem unintuitive, and therefore are less likely to be known by the management of such organizations.

2.4.5 Association Rules from Minimum Spanning Trees

M. A. Valle et al. [15] proposed a novel methodology to study the structure and behavior of consumer market baskets from the topology of a minimum spanning tree which represented the interdependencies between products, and use this information to complement the association rule generation process. The input to their proposed methodology was a correlation matrix between the set of all one-hot encoded purchase vectors such that each vector denoted the presence or non-presence of each product from the dataset in that vector. The dataset used for the MST construction was a list of $1,046,804$ transactions containing a set of $3,240$ unique products from a large supermarket chain branch in Santiago, Chile. When building this correlation matrix, the authors opted to use the Pearson's Coefficient (which is equivalent to the coefficient ϕ for binary data such as theirs) over the traditionally used Jaccard distance to compute the similarity between the binary product vectors, as the former provides both a positive and negative association between products. Additionally, they used the distance function $d_{ij} = \sqrt{2(1-\phi_{ij}}$ to transform the correlation matrix into a distance measurement (i.e. the weight of the edges). The authors constructed a MST for 220 product subcategories, and noted that there was a significant level of grouping between product sub-categories that belonged to the same parent category. To remove edges from the MST that were not statistically significant, the authors used the mutual information [16] measure $\sum_{x,y} log_2 \frac{r(x,y)}{p(x)q(y)}$ between product subcategories p and q, and were able to prune 14 edges, all of which were connected to a terminal node, therefore effectively pruning 14 vertices from the MST too. To identify the most influential regions of the MST, the authors defined an influence zone of distances that were in the 10^{th} percentile. To generate meaningful association rules, for each MST product i, the authors ran a search for the set of all association rules R_i such that $P_i \rightarrow P_j (i \neq q)$. Then from the resulting set of rules, they searched for rules that obeyed $P_i \rightarrow P_m$ where m a product node connected to the product i in the minimum spanning tree. For both resulting sets of rules for each product, the mean of their lift scores were observed, and the authors determined that the rules that were reinforced by the MST had a higher mean, and that a majority of these rules had a lift score above the 90^{th} percentile.

To identify the clusters each of the products should be identified under, the authors constructed a hierarchical tree using the average linkage clustering method, and by using an unspecified cut distance, they were able to produce 17 taxonomic groups (i.e. clusters). Cross-referencing their results with the

actual parent categories of the products, they were able to conclude that the MST did indeed categorize the product sub-categories into clusters with a reasonable degree of accuracy. The authors then compared their MST to another methodology, namely the structured association map (SAM) [17], using the Jaccard distance as a measure of similarity, and were able to generate interesting 2x2 rules (i.e. $\{A, B\} \rightarrow \{C, D\}$), all with lift scores above 1.0, with one rule even having a lift score of 106.46. They concluded that while both approaches provided different information, they both visually identify the strongest relationships between the products, and provide useful information to reduce the search space for association rules.

Critical Analysis

The authors' approach seems to be novel, thorough and well structured. Their methodology successfully employed the use of minimum spanning trees to complement the association rule generation process with sound results. One caveat of their approach is that they only used the MST to generate single-element rules (i.e. $\{A\} \rightarrow \{B\}$, where the antecedent and consequent contain only one element each). Using the distance score in conjunction with the importance function they defined (i.e. $\sum_{k \in K_u} \frac{1}{w_{uk}}$), they could have defined a system to produce multi-element rules (i.e. where either/both the antecedent and the consequent have more than one element), then rank them using their respective lifts. While single-element rules are easily understandable and tend to have high lift values when extracted from the MST, multi-element rules would provide an additional layer of insight as to how a range of products (perhaps a cluster) related to another.

2.4.6 Summary

In conclusion, **R. Aggarwal et al.** introduced a formal system for association rules, and a method to generate them from a transactional database - the Apriori Algorithm [1]. However, while these papers only based the interestingness of an association rule on objective measures such as the support, confidence and lift scores of these rules, **M. Zekić-Sušac et al.** proposed a methodology where subjective human input was used to validate the interestingness of these rules. A drawback of the above methods, however, is the large number of rules produced. **M. A. Valle et al.** introduced a methodology to extract high value association rules from a minimum spanning tree, used to complement the rules produced by the Apriori algorithm. A caveat of this approach, however, is that it can only produce rules such that the antecedent and consequent are sets with one element each. Additionally, the methodology described involves building the MST using Prim's algorithm, which can be relatively slow. **C. Zhong et al.** proposed a solution to this, where the Kruskal's algorithm was optimized using K-Means, leading to a significant performance increase as illustrated in Figure 2.5. The methodology proposed in this paper was a framework of sorts,

where Kruskal's could be substituted with any MST algorithm, such as Prim's. All the research conducted above has served as the inspiration for this project.

Chapter 3

Methodology

3.1 Dataset Pre-Processing

The dataset used in this study is the transactional sales data from a chain of Brazilian gas-station stores [18]. Each row in the dataset represents the purchase of a specific product as part of a transaction - and as such, each row corresponds to the following columns:

- Company Code
- Order Number
- Employee
- Product
- Product Category
- Client
- Sale Date Time
- Product Cost
- Discount Amount

All personal and corporate names were exchanged for fictitious names by the author of the dataset in order to preserve the anonymity of those whose who could have otherwise been identified through the dataset. Before employing the dataset, sanitary procedures were carried out to ensure that the dataset was error-free and in a format suitable for graph generation. The steps have been detailed below.

1. **Transaction Identifier**

   The *Order Number* field showed discrepancies, where a given order number could reference distinct transactions in different stores and cities, and at different dates and times. This could be due to

the stores maintaining their own order numbers, and also because the order numbers may reset after a predetermined limit. A unique transaction identifier - named `basket_id` - was created by concatenating the order number and the date, thereby mitigating the occurrence of a identifier that references multiple transactions.

2. **Binary Purchase Vector transformation**

   The dataset was then transformed such that each transaction was represented by a binary purchase vector - as described in Section 2.2 - wherein each column represents a product category. The product categories were chosen for the graph representation over the products themselves as it would give a more generalized view on the associations between them, and the categories themselves were deemed specific enough that they would not be parent to children of significant variance.

3.2 MST Generation

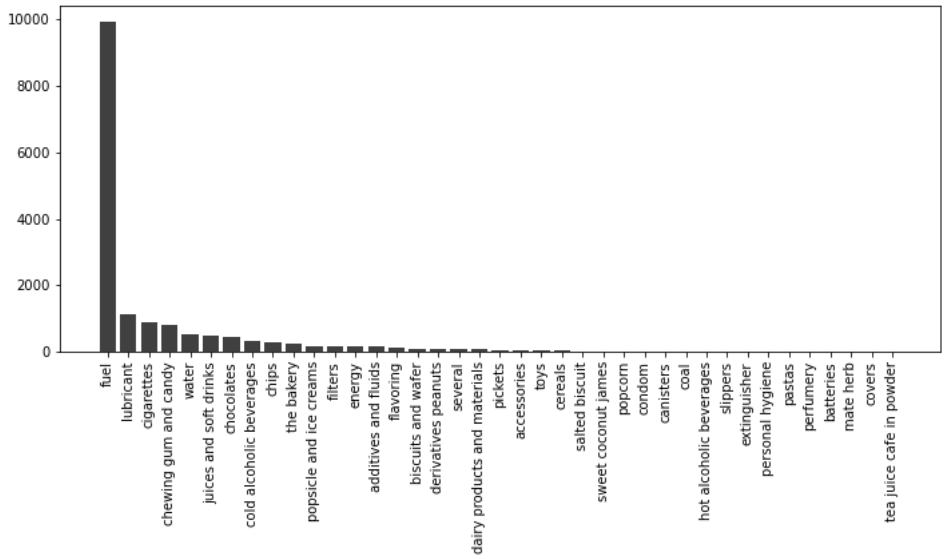

Figure 3.1: Category Distribution

The metrics used to assess the association rules - support, lift and confidence - are based on the proportional presence of a given itemset in the transactions. Since our dataset is from a gas-station store chain, fuel products dominate the transactional presence by a significant factor. Figure 3.1 highlights the disparity between the presence of *fuel* products and the others, with *fuel* being present in 99.28% of all transactions. To avoid the association rules being dominated by the *fuel* category - which should inherently understood

to be a key product for gas stations - the fuel category was purged from the dataset.

Using the Pearson's Correlation Coefficient [19] (Equation 3.1), a correlation matrix was derived from the binary purchase vector dataset. In this instance - because the correlation is of binary variables - the Pearson's Correlation Coefficient is equivalent to the Phi Coefficient (ϕ) [20]. The resulting correlation matrix is also diagonally symmetrical (i.e. $\phi_{ij} = \phi_{ji}$), and therefore the illustration of the correlation matrix in Figure 3.2 displays only those values that are below the diagonal.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}} \tag{3.1}$$

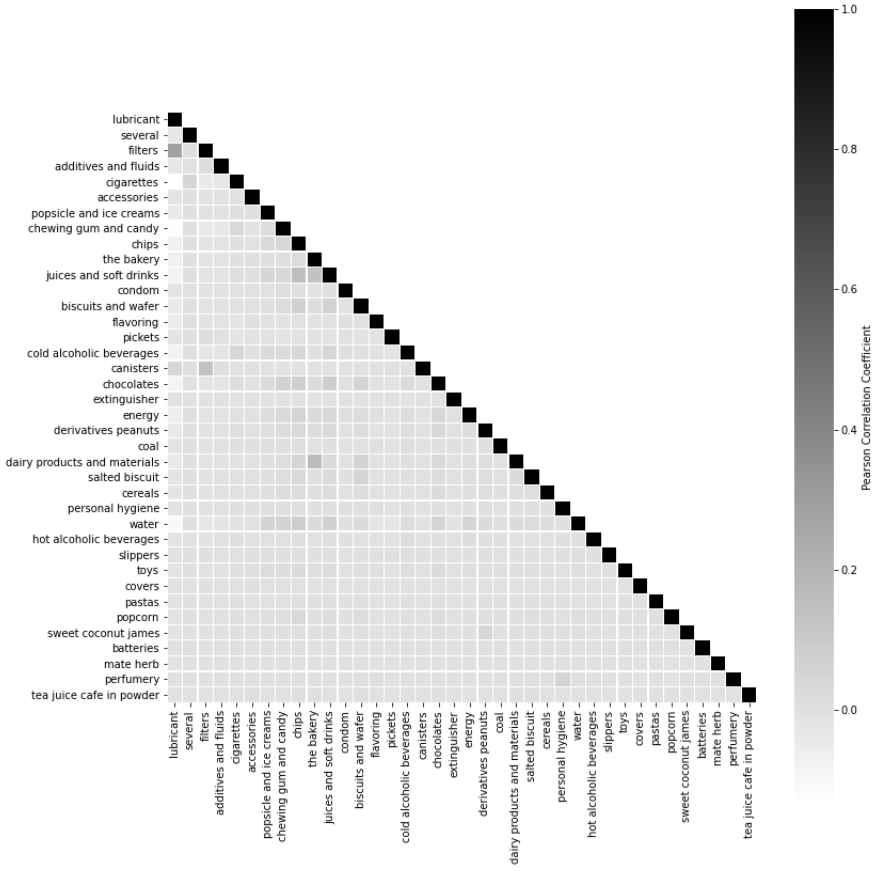

Figure 3.2: Correlation Matrix from Binary Purchase Vectors

A minimum spanning tree is a graph composed of the *shortest* path connecting all vertices, and therefore the values in the correlation matrix would need to be transformed via a function such that the higher the correlation value, the lower the output value. This would allow the MST to capture the

strongest associations between the product categories. As used by **M. A. Valle et al.** in [15], the distance function (Equation 3.2 was used to make this transformation - where ϕ_{ij} refers to the correlation value for two given product categories i and j in Figure 3.2). However, whereas **M. A. Valle et al.** left their ϕ_{ij} unaltered, we have converted it to an absolute value (i.e. $|\phi_{ij}|$). The reason behind this is to capture the strongest relationships between product categories, positive and negative alike. By leaving the value unaltered, the MST would not capture strong negative associations as they would have the highest weights after the distance function was applied. Figure 3.2 illustrates the effect of this function, where the stronger the correlation, the lower the distance function's output. The figure also demonstrates that because the Pearson's Correlation Coefficient is bound to the interval $[-1, 1]$, the distance function's output is therefore bound to the interval $[0, \sqrt{2}]$.

$$d_{ij} = \sqrt{2(1 - |\phi_{ij}|)} \tag{3.2}$$

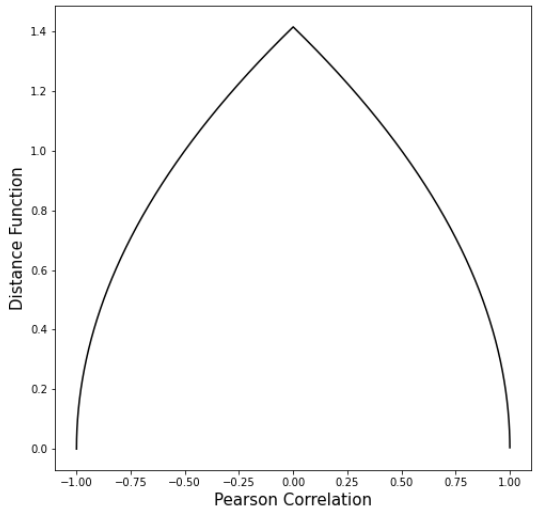

Figure 3.3: The effect of the distance function $\sqrt{2(1 - |x|)}$

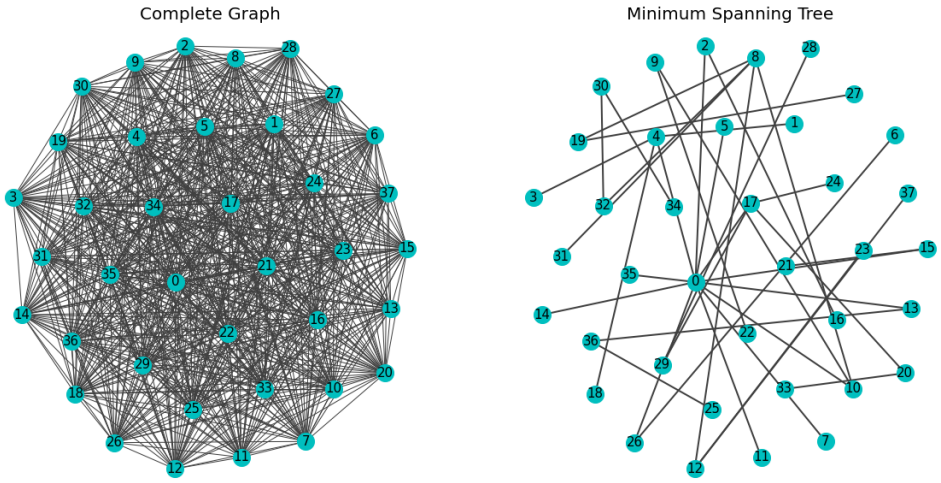

Figure 3.4: Product Category Graph and MST

With the correlation matrix transformed using Equation 3.2, a graph $G = (V, E)$ was generated such that the vertices V represent the product categories, and the edges E the associations between them. Employing Kruskal's algorithm, a minimum spanning tree was then extracted from this graph. Both the complete graph and the MST are illustrated in Figure 3.4. The value of each node is an integer, which corresponds with the index of the product category in the binary purchase vector dataset. The length of each edge is directly proportionate to its value, such that the greater the value, the greater the length of the edge.

3.3 Markov Clustering

The MST was then clustered using the Markov Clustering algorithm. To identify the most modular clustering configuration (modularity being the minimality of connectedness between dense clusters), an array of inflation scores between 1.5 and 2.5 (inclusive) were tested at increments of 0.1, leading us to determine that an inflation score of 1.6 resulted in the most optimal modularity. The Markov Clustering was performed using this inflation score, and the clustering results are illustrated in Figure 3.5 (note that while the disposition of the nodes is different from that illustrated in Figure 3.4, the values of all nodes and edges are the same).

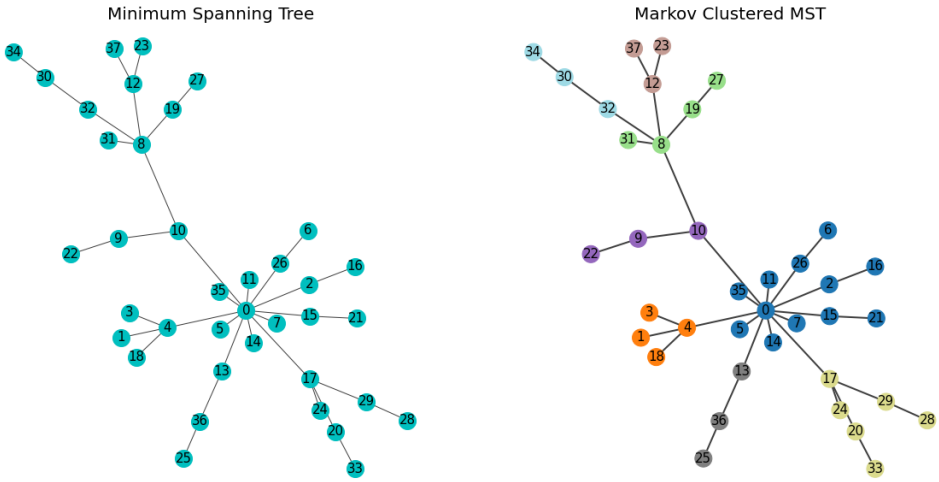

Figure 3.5: MST before and after Markov Clustering

The Markov Clustering algorithm segmented the nodes into 8 distinct clusters. Figure 3.6 illustrates the names of the product categories in each cluster, color-coded in accordance with the graphs in Figure 3.5.

Figure 3.6: Product categories by cluster

3.4 Rule Generation

3.4.1 Itemset Generation

For every cluster identified, all possible itemsets are generated for the n product categories in a given cluster such that the length of the itemsets range from 1 to $n-1$. Listing 3.1 is the excerpt of code that generates all possible itemsets for a cluster.

```
# Source: arm_cython.pyx
from itertools import combinations
...
cpdef __generate_itemsets_by_cluster():
    global itemsets_by_cluster
    cdef set items
    for index, cluster in enumerate(clusters):
        items = set()
        for set_size in range(1, len(cluster)):
            items.update(combinations(cluster, set_size))
        itemsets_by_cluster[index] = items
```

Listing 3.1: Cluster Itemset Generation

Once the minimum spanning tree has been successfully clustered, association rules can be generated from the clusters in two distinct ways.

3.4.2 Bi-Cluster Rule Generation

Bi-cluster rules are those where the antecedent and consequent originate from distinct and separate clusters. All possible bi-cluster permutations ${}^{n}P_{2}$ are calculated for n clusters. Possible association rules are then generated for these combinations of clusters.

```
# Source: arm_cython.pyx
cpdef list generate_bicluster_rules(min_support=0.005, min_confidence=0.6):
    cdef list rules = []
    cdef list cluster_combinations = list(combinations(list(range(len(clusters))), 2))

    for comb in cluster_combinations:
        a_items = itemsets_by_cluster[comb[0]]
        b_items = itemsets_by_cluster[comb[1]]
        current_rules = list(product(a_items, b_items))
```

```
10          rules.extend(current_rules)
11
12      rules = __add_reverse(rules)
13      return __filter_rules(rules, min_support=min_support, min_confidence=min_confidence)
```

Listing 3.2: Cluster Itemset Generation

Listing 3.2 is the excerpt of code used to generate all the bi-cluster rules. On line 4, we have used the `combinations` function instead of `permutations`.

If we were to use permutations, we would have to generate each itemset i such that:

$$\{i_1 \in A\} \rightarrow \{i_2 \in B\}$$

for clusters A and B, and then later:

$$\{i_1 \in B\} \rightarrow \{i_2 \in B\}$$

which would essentially be the same combinations in a different configuration. Instead, by using `combinations` on line 4 of Listing 3.2, we only compute the former, and then re-add the rule to the ruleset with the antecedent and consequent swapped (line 12). This method essentially reduces our computational time by half.

3.4.3 Intra-Cluster Rule Generation

Intra-cluster rules are those where the both antecedent and consequent originate from the same cluster. Similar to the bi-cluster rules, the intra-cluster rules used the itemsets generated via the `__generate_itemsets_by_cluster()` function in Listing 3.1, however with the constraint:

$$\{i_1\} \rightarrow \{i_2\}; i \in A$$

for every itemset i in cluster A.

3.4.4 Rule Pruning

As described in Section 2.3, the *anti-monotone property of support* implies that the support for a subset of a given set should be equal to or greater than the support of the set:

$$\text{support}(\{x, y\}) \geq \text{support}(\{x, y, z\})$$

To take advantage of this, we first sort our rules in ascending order by the number of elements in the antecedent. When iterating through the potential rules, if either the antecedent or consequent is found to have a support score below the pre-defined minimum tolerance, the itemsets are added to a set `below_threshold`. Any future rules iterated are then cross-checked against `below_threshold`, and if it is found that any set in `below_threshold` is a subsset of the rule, the rule is discarded as we know that its support is below our minimum tolerance. This allows us to avoid computing the support for several rules - the time complexity of which $O(n)$ at worst, thereby significantly reducing computation time.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of the 20th International Conference on Very Large Databases*, pp. 487–489, 1994.

[2] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," 2, vol. 29, New York, NY, USA: Association for Computing Machinery, May 2000, pp. 1–12. DOI: 10.1145/335191.335372. [Online]. Available: https://doi.org/10.1145/335191.335372.

[3] R. C. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 6, pp. 1389–1401, 6 1957. DOI: 10.1002/j.1538-7305.1957.tb01515.x.

[4] V. Jarník, "O jistém problému minimálním," *Práce Moravské Přírodovědecké Společnosti*, pp. 57–63, 1930.

[5] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, 1959, ISSN: 1. DOI: 10.1007/BF01386390.

[6] O. K. A. Abdelnabi. (2018). "Minimum spanning tree," [Online]. Available: https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/.

[7] J. B. K. Jr., "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, 1956. DOI: 10.1090/S0002-9939-1956-0078686-7.

[8] J. C. Pennsylvania. (2020). "Generating association rules," Junita College Pennsylvania, [Online]. Available: http://faculty.juniata.edu/rhodes/ml/assocRules.html.

[9] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, "Definition 6.2," in *Introduction to Data Mining*, 2nd ed. 2019, ch. 6, p. 334, ISBN: 9780134080284.

[10] C.-L. Hsu. (2017). "Frequent itemset generation using apriori algorithm," [Online]. Available: https://chih-ling-hsu.github.io/2017/03/25/apriori.

[11] C. Zhong, M. Malinen, D. Miao, and P. Fränti, "A fast minimum spanning tree algorithm based on k-means," *Information Sciences*, vol. 295, pp. 1–17, 2015, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2014.10.012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025514009943.

[12] S. M. van Dongen, "Graph clustering by flow simulation," Dutch, Dissertation, Universiteit Utrecht, Sep. 1969, ch. 1, p. 5. [Online]. Available: http://www.library.uu.nl/digiarchief/dip/diss/1895620/full.pdf.

[13] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '93, Washington, D.C., USA: Association for Computing Machinery, 1993, pp. 207–216, ISBN: 0897915925. DOI: 10.1145/170035.170072. [Online]. Available: https://doi.org/10.1145/170035.170072.

[14] M. Zekic-Susac and A. Has, "Discovering market basket patterns using hierarchical association rules," *Croatian Operational Research Review*, vol. 6, pp. 475–487, Oct. 2015. DOI: 10.17535/crorr.2015.0036.

[15] M. A. Valle, G. A. Ruz, and R. Morrśs, "Market basket analysis: Complementing association rules with minimum spanning trees," *Expert Systems with Applications*, vol. 97, pp. 146–162, May 2018. DOI: 10.1016/j.eswa.2017.12.028.

[16] T. M. Cover and J. A. Thomas, *Elements of information theory*. 2006.

[17] J. W. Kim, "Construction and evaluation of structured association map for visual exploration of association rules," *Expert Systems with Applications*, vol. 74, 2017. DOI: 10.1016/j.eswa.2017.01.007.

[18] Kaggle. (2020). "Sales data for a chain of brazilian stores," Kaggle, [Online]. Available: https://www.kaggle.com/marcio486/sales-data-for-a-chain-of-brazilian-stores.

[19] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, pp. 240–242, 1895, ISSN: 03701662.

[20] J. Ernest C. Davenport and N. A. El-Sanhurry, "Phi/phimax: Review and synthesis," *Educational and Psychological Measurement*, vol. 51, no. 4, pp. 821–828, 1991. DOI: 10.1177/001316449105100403.