Development of a Rule Based Decision Support System for Pilots

Using Network Analysis, Label Propagation and Association Rule Mining

Joel Henneberg
Master of Science Thesis in Computer Science

Development of a Rule Based Decision Support System for Pilots: Using Network Analysis, Label Propagation and Association Rule Mining

Joel Henneberg

LiTH-ISY-EX--23/5623--SE

Supervisor: David Nordlund
ISY, Linköpings universitet
Malin Linderstam
Saab, Aeronautics

Examiner: Danyo Danev
ISY, Linköpings universitet

Division of Communication Systems
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Copyright © 2023 Joel Henneberg
Abstract

Maneuvering a fighter jet is not easy, and new methods for minimizing the risk of pilots experiencing information overload are constantly under development. Saab is currently looking into the possibilities of creating a decision support system (DSS) for pilots in situations where multiple independent system failures occur simultaneously. In today’s JAS 39 Gripen E such failures generate multiple simultaneous alerts for the pilots to act upon, with each alert proposing a set of actions for the pilot to handle the current situation. These actions are displayed on separate pages for each of the alerts, requiring the pilot to manually swap between pages of actions to get an overview of what to do. When pilots must use a significant part of their cognitive abilities to maneuver the aircraft in critical situations, a DSS is needed to apprehend the information and guidelines from the alerts more quickly.

Our approach to developing such DSS is by representing each of the individual actions as nodes in a network, with links between nodes providing information about the relations between actions. The purpose of mapping the problem to become a networking problem is to use label propagation algorithms to divide the network into communities. With each node being assigned to a community and having the communities sorted on priority, getting a first draft of what a sorting algorithm can accomplish should be easy given a complex fault situation. We use rule mining to extract rules from the system as it is currently implemented to create a rule-based sorting within communities.

The conclusion regarding label propagation algorithms is that although they are great for prototyping, propagation algorithms have no place in creating a DSS with high reliability. We also conclude that inconsistencies in the data that yield cycles in the created networks limit the sorting algorithms since there is no way of knowing where the cycle begins, hence no way of knowing how to sort actions being part of a cycle. Despite this, we argue that mined rules can be very helpful for engineers at Saab to be more consistent in their future work. Because of this, the conclusion is that our prototype of the DSS is better suited for the engineers working with providing pilots with the correct information than for the pilots directly. Lastly, we provide suggestions for future work, focusing much on possible ways of dealing with the cycles. Suppose Saab wants to pursue further research into whether creating a good enough DSS the way we propose is possible. In that case, some filter has to be developed to reduce the amount of information in complex fault situations.
Acknowledgments

To begin with, I would like to thank my examiner Danyo Danev for believing in me and my thesis proposal and for letting me pursue my thesis with him. I would also like to thank my supervisor David Nordlund for all of his guidance and support during the project. The feedback David has given me has been invaluable and has taught me a lot.

Furthermore, I would like to thank Saab Aeronautics, Avionic Systems, for the opportunity to work on this thesis. Spending time with all of you in your facilities in Linköping has been a true pleasure. Special thanks to my supervisor at Saab, Malin Linderstam who has given me great feedback. I would also like to thank Torbjörn Fransson for sharing his expertise and experience with me throughout the project and for answering all of my technical questions.

I would also like to thank my family for their unconditional support throughout my studies at the university, especially during this thesis. It is your support that has led me to where I stand today.

Linköping, November 2023
Joel Henneberg
# Contents

List of Figures vii  
List of Tables viii  
Notation ix  

1 Introduction 1  
1.1 Motivation .......................... 3  
1.2 Aim ................................ 4  
1.3 Methodology ........................ 5  
1.4 Thesis disposition .................. 6  

2 Background 7  
2.1 Decision Support Systems ........ 7  
2.2 Complex Networks .................. 7  
2.3 Community Detection ................. 11  
2.4 Association Rule Mining .......... 13  

3 Data and Data Pre-Processing 19  
3.1 Data and Notation .................. 19  
3.2 Filtering the Data ................... 20  
3.3 Data Distribution ................... 20  

4 Implementation 23  
4.1 Creation of Networks .............. 23  
4.2 Identifying Connected Components 24  
4.3 Central Nodes ....................... 25  
4.4 Label Propagation ................... 27  
4.5 Sequential Rule Mining ........... 31  
4.6 Graphical User Interface .......... 32  

5 Results and Discussion 34  
5.1 Label Propagation ................... 34
5.2 Sequential Rules ................................................. 36
5.3 Cycles ............................................................... 36
5.4 Sorting with Sequential Rules .................................. 39
5.5 Results from Total-FMAN ....................................... 40

6 Conclusions and Future Work ................................. 44
  6.1 Conclusions ....................................................... 44
  6.2 Future Work ..................................................... 45

Bibliography .......................................................... 48
# List of Figures

1.1 Example of FMAN-page ........................................... 2

2.1 Community detection ............................................. 11

3.1 FMAN length distribution ........................................ 21
3.2 Action distribution ............................................... 21
3.3 Frequency of actions ............................................. 22
3.4 Frequency of actions, four actions removed .................... 22

4.1 Example of networks ............................................... 24
4.2 Network created after filtering. ............................... 25
4.3 Centrality scores ................................................ 26
4.4 The network of actions after initialization .................... 28
4.5 The graphical user interface .................................... 32

5.1 Network A – Propagation result ............................... 35
5.2 Network B – Propagation result ............................... 35
5.3 Network C – Propagation result ............................... 35
5.4 Example of a 2-cycle. ........................................... 38
5.5 Example of a k-cycle. ............................................ 39
5.6 Default alerts. .................................................. 42
5.7 Results for the different networks. ........................... 43

6.1 Example of two FMAN-pages with two actions each. ........ 47
List of Tables

2.1 Example database. .................................................. 13
2.2 Frequency table. .................................................. 13
2.3 All items with \textit{minsup} $\geq 0.6$. ............................... 15
2.4 Supersets of size two. ............................................ 16
2.5 Supersets of size two with \textit{minsup} $\geq 0.6$. ................. 16
2.6 Supersets of size three. ......................................... 16
2.7 Superset of size four. ............................................. 16
2.8 Rules generated with the Apriori algorithm in R. ................. 17
2.9 List of sequential rules ............................................ 18
2.10 List of sequential rules ........................................... 18

4.1 Labels and their priority. ........................................ 27

5.1 Label distribution after propagation. ............................. 34
5.2 List of sequential rules .......................................... 37
5.3 Top five and bottom five rules that make up 2-cycles. .......... 37

6.1 Table of generated rules. ........................................ 47
## Notation

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AoA</td>
<td>Angle of Attack</td>
</tr>
<tr>
<td>BC</td>
<td>Betweenness Centrality</td>
</tr>
<tr>
<td>BEOS</td>
<td>Back-up Emergency Oxygen System</td>
</tr>
<tr>
<td>CAS</td>
<td>Calibrated Airspeed</td>
</tr>
<tr>
<td>CC</td>
<td>Closeness Centrality</td>
</tr>
<tr>
<td>cSPADE</td>
<td>constrained Sequential Pattern Discovery using Equivalence classes</td>
</tr>
<tr>
<td>DAG</td>
<td>Directed Acyclic Graph</td>
</tr>
<tr>
<td>DC</td>
<td>Degree Centrality</td>
</tr>
<tr>
<td>DLPA</td>
<td>Directed Label Propagation Algorithm</td>
</tr>
<tr>
<td>DSS</td>
<td>Decision Support System</td>
</tr>
<tr>
<td>EC</td>
<td>Eigenvector Centrality</td>
</tr>
<tr>
<td>ECAM</td>
<td>Electronic Centralized Aircraft Monitoring</td>
</tr>
<tr>
<td>EMGY</td>
<td>Emergency</td>
</tr>
<tr>
<td>FMAN</td>
<td>Flight Manual</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>LPA</td>
<td>Label Propagation Algorithm</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>SAAB / Saab</td>
<td>Svenska Aeroplan Aktiebolaget</td>
</tr>
<tr>
<td>SRM</td>
<td>Sequential Rule Mining</td>
</tr>
<tr>
<td>Total-FMAN</td>
<td>Total Flight Manual</td>
</tr>
</tbody>
</table>
Maneuvering a fighter jet traveling about the speed of sound while making flawless decisions is not easy. Although pilots are trained to keep calm during critical fault situations and work in cognitively tough environments as their everyday job, each step towards a less stressful work environment must be encouraged. This is true for both military and commercial aircraft. In December 1991, a commercial aircraft with 129 people onboard crash-landed in Gottröra, Sweden. The plane, which was of type MD-81, split into three parts during the crash and even rolled over along its longitudinal axis. Miraculously, everyone survived [1]. In an investigation of the incident, cockpit crew members were interviewed. In one of these interviews, the co-pilot described how he experienced the stressful situation in the cockpit:

“All the lamps are blinking and there are a lot of warning sounds in the cockpit. It is a really terrible environment. It is not possible to manage all this information. With so many malfunctions you stop analyzing them and concentrate on the flying. That’s the only thing to do.” [2].

One way Saab is trying to facilitate a less stressful environment for their fighter jet pilots is to examine what information a pilot needs when multiple system failures occur. Fighter jets are advanced pieces of machinery with many systems that can fail either dependently or independently of one another due to either internal or external reasons. Hence, having a set of appropriate actions from the flight manual ready for all possible combinations of simultaneous service failures is unrealistic. This master’s thesis will identify which types of algorithms or ways of manipulating data are most suitable for making a general decision support system (DSS) for pilots maneuvering JAS 39 Gripen E, when multiple system failures occur simultaneously. Saab refers to this as having a Total-FMAN (total
Introduction

Figure 1.1: Example of FMAN-page, provided by a supervisor at Saab.

As seen in Figure 1.1, actions can be anything from Fly gently to Vacate A/C hazard zone. Actions have been carefully selected by pilots together with experienced system engineers at Saab for each FMAN to contain the most valuable information for any given system failure. The issue occurs when multiple systems fail simultaneously. So far, system engineers at Saab have identified some relations between system failures, which has led to them managing to add some dynamic features to some of the FMANs, but this needs to be further developed. It works now in the form of primary and secondary alerts. By identifying relations between different systems and their functionalities, FMANs have been somewhat merged to lower the workload for the pilots. In Figure 1.1 it can be seen that some actions are white and some are grey. The white actions either belong to the current alert or any of its active secondary alerts. Meanwhile, grey actions belong to any secondary alerts that are not active. The idea of a Total-FMAN is only to display the white actions and let them change dynamically as alerts become active and/or heal.
1.1 Motivation

NASA researchers Angela Harrivel and Terri McKay have studied ways to determine when a pilot is under too much stress and what causes it. They used functional near-infrared spectroscopy (fNIRS) [3]. Harrivel states that no matter how much training pilots have, they can suffer from a lack of situational awareness when too much is happening. When pilots do not have the cognitive ability to act upon a critical situation, the consequences can be catastrophic [4]. According to the final report on the accident involving the Airbus A330-203 on June 1st, 2009, operated by Air France, this is likely what happened. The A330 has a system called the ECAM (electronic centralized aircraft monitoring), which proposes alternative actions for pilots during a majority of failures and emergency cases. The final report on the accident states that the ECAM system took a lot of attention from the pilot and co-pilot while trying to resolve the problem. Unfortunately, the ECAM system did not provide critically essential or relevant information. This led to the system draining both pilots of time and mental resources, which could have been used elsewhere [5]. This shows room for improvement regarding flight safety and DSSs for pilots, which is why investigating new ways of developing these is of interest.

During the fall of 2022, Louise Nilsson and Malin Linderstam (previously Rahm) carried out their master’s thesis Dynamic Synthesis of Pilot Actions for Multiple Faults during Flight [6] at Saab, in which they examined how information from the flight manual should be presented to the pilot during multiple service failures. The focus of the thesis was on the interpretation and visualization of the information and not on the information itself. However, some machine learning (ML) models were examined in the later stages of the thesis to try and achieve more information-dense messages from the flight manual. Classification algorithms were used to classify actions into seven different classes. The idea was that this classification could determine a priority among actions when merging multiple FMANs. The classification did not have much success, with the best model only achieving an accuracy of 31%. The authors reason that the low accuracy could be due to not having a sufficient amount of data at hand. Since manually labeling specific actions in a flight manual as more critical than others takes considerable time (and is extremely difficult), it is interesting to explore ways of having a computer perform this type of labeling.

Knowing how to prioritize among actions and make good decisions is easier if there is a straightforward way of analyzing how actions are interconnected. If it is not apparent how or why interconnections arise and the amount of interconnected subjects is large, the system is called a complex network. Examples of complex networks are the human brain with synapses and nerves, the World Wide Web with websites and links, or transport infrastructure with roads and intersections. The science of complex networks is relatively new [7]. Therefore, it is interesting to identify new application areas in which the theory of complex networks can be helpful.
The label propagation algorithm (LPA) is an algorithm used to assign labels to unlabeled data points by propagating information about already known labels in the data set, much like epidemic contagion. This is accomplished by representing the data set as a network of relations between nodes between which the labels can propagate [8]. In this thesis, we use the same data as when Nilsson and Rahm conducted their master's thesis. Nilsson and Rahm mention in their discussion about future work that investigating classification algorithms that do not need any (or only a tiny amount of) manually labeled data to perform well could be an excellent way to approach the problem of classifying actions. Therefore, it is interesting to investigate whether or not LPA is a good way of spreading labels in a network created from relations between actions in a flight manual. If LPA works well on the data, Saab will have multiple ways of moving forward with the project of creating a Total-FMAN.

A DSS typically makes use of rules that it has to relate to. These rules will be mined with association rule mining [9]. General rules could help the development of a Total-FMAN since it would no longer require the engineers to have as much knowledge about the underlying systems in the plane as before. Of course, knowledge from experienced system engineers will always be invaluable. Still, general rules can speed up the development process and ensure that no trivial relationship between actions is overlooked in the future.

1.2 Aim

This thesis explores the possibility of generalizing the applications of complex networks and data mining such that it can be helpful in the development of a dynamic DSS for jet fighter pilots. The goal is for the DSS to schedule actions appropriately.

1.2.1 Research Questions

This master's thesis intends to answer the following research questions:

1. Can label propagation algorithms and rule mining algorithms be used to assist the creation of a decision support system capable of prioritizing actions coming from multiple pages in a flight manual?

2. What changes could be made to the flight manual to enhance the performance of a label propagation-based approach in the future?

3. Does FMAN-pages contain sufficient information for general rules to be mined from them?

1.2.2 Delimitations

Some delimitations have to be set so that the size and complexity of this master's thesis is manageable. Besides following the requirements from the already imple-
mented systems and the data at hand, the following delimitations are added to the thesis:

- The thesis only examines the performance of a DSS on the given data and does not consider how the result should be presented to the pilot.
- The thesis only focuses on action texts from the flight manual connected to flight.

1.2.3 System Restrictions

The screen presenting the FMAN-pages for the pilot can only display a fixed number of characters on its width and a fixed number of lines on its height. Optimally, the final result of the DSS should be within these limits independently of how many and which FMAN-pages have been merged. These hardware restrictions can be seen in Figure 1.1.

1.3 Methodology

The approach used for the thesis is to create a prototype of a DSS, focusing on complex networks, LPA, and association analysis. To begin with, a literature study will investigate what algorithms, centrality measures, and associations is most suitable for the data. The method focuses on extracting as much information from the current setup of FMAN-pages as possible and not on establishing new or modified FMAN-pages.

The idea is to represent actions $a_i$, where $i \in I = \{1, ..., 375\}$ as the nodes of a network, with edges representing relations between these actions. One can, for example, put edges between all actions on the same FMAN-page or between actions within a certain distance of one another within the same FMAN-page. One can also consider a network where edges are directed from $a_i$ to $a_j$ and added only if $a_i$ and $a_j$ are adjacent in a given FMAN. Directed, undirected, weighted, and unweighted networks will be considered.

LPAs will be used to identify communities and help to spread labels in the created networks. There are multiple ways of initializing LPAs and multiple ways of performing each propagation step [8]. This thesis will use a semi-supervised approach by manually labeling initial nodes based on prior knowledge. For this, various centrality measures will be used to identify central nodes. Pilots and experienced engineers will help assign appropriate labels to the extracted nodes.

There are many different ways to perform label propagation, some of which will be investigated and compared. The original LPA only works on simple, undirected, and unweighted graphs and produces flat, disjoint communities [8]. Hence, extensions on the original LPA will be used to handle the different types of networks created. The resulting communities extracted from the LPA will be used to roughly sort actions between any arbitrary combinations of FMAN-pages.
Association analysis will be used to draw conclusions about the interconnections between actions, similar to market basket analysis [9]. Takeaways from this analysis will be used to generate association rules.

Community detection and association analysis will be done separately. The final result will be a sorting algorithm which begins with roughly sorting the list of actions according to communities. Association rules will then be used to sort actions internally within each community. Therefore, community detection with LPAs and rule mining can be interpreted as two separate subsystems. We combine these subsystems to achieve our proposed Total-FMAN.

### 1.4 Thesis disposition

Chapter 2 presents relevant background theory on DSSs, complex networks, community detection, LPAs, and association rule mining. Chapter 3 discusses the data used and the pre-processing performed. A good understanding of the data set is necessary to understand the results and enhance reproducibility. In Chapter 4 we describe how the different networks are created and how the propagation is carried out in all of the networks with pseudocode. The initialization of the algorithms is also described together with an explanation of the classes used during the propagation. Chapter 5 provides label propagation and rule mining results. The issue of cycles in the data is discussed, and examples are given to clarify why cycles are problematic. Screenshots from the GUI provide examples from the final prototype developed throughout the project to enhance usability. Lastly, Chapter 6 answers the research questions posed in Section 1.2.1 and suggests some areas of interest for future work.
This chapter aims to give enough background on the theory and related work to understand the rest of the thesis.

2.1 Decision Support Systems

The purpose of a DSS is to analyze data to gather comprehensive information that can either be used to directly solve problems or support decision-making [10]. This process can be computerized or powered by humans but is often a combination. Some DSSs make their own decisions or display a set of possible decisions. In contrast, others aim to provide the user with helpful information to improve the speed and accuracy of the decision-making. As technology continues to advance, researchers manage to find new applications for DSSs each year. Also, since tools for data analysis continue to improve, the possibility of creating advanced DSSs has never been better [11].

2.2 Complex Networks

Complex networks is a branch in the field of networking science that is highly interdisciplinary, which is why the study of complex networks is so important. This has led to scientists with expertise in physics, mathematics, computer science, biology, and neuroscience often working side by side.

The field has drawn significant attention from the research community in the past twenty years because of the dramatic increase in computational power. The first two papers about complex networks were published in the late 1990s [12][13] and since then, the number of citations of these has grown exponentially. This
shows how popular complex networks has become and explains why researchers constantly find new applications for complex systems.

Principles and methods surrounding complex networks are primarily for analyzing the features of the networks. For a particular network, this could be extracting the most central nodes according to some specific centrality measure, sampling on small parts of the network to get an idea of the network structure, or dividing the network into communities, to mention a few. [7]

2.2.1 Definitions

The concept of a network can be mathematically interpreted as a graph \( G = (V, E) \), with a set \( V \) of \( n \) nodes and a set \( E \) of \( m \) edges. If an edge connects node \( u \) to node \( v \) but not vice versa, the edge is called directed, and we say that \( u \) is a parent of \( v \), and therefore we also say that \( v \) is a child of \( u \). Otherwise, it is called undirected. A graph \( G \) is said to be directed if all its edges are directed, and undirected if all its edges are undirected. Edges can have weights \( w(v, u) \), and the resulting graph is then called a weighted graph. If the graph is unweighted, all edges are assumed to have a unit weight. Two nodes that share an edge are adjacent to each other and are said to be neighbors. The amount of incoming edges to a node is called the nodes in-degree, and the amount of outgoing edges from a node is called the nodes out-degree. A self-loop is defined as an edge from a node to itself. If a graph contains no self-loops and is undirected and unweighted, we call it simple [8].

A walk consists of a sequence of traversed nodes from a source node to a destination node. A cycle is defined as a closed walk, which means that a cycle is a walk that ends in the same node as it begins. Furthermore, we define a \( k \)-cycle as a cycle of length \( k > 2 \), where \( k \) is the number of nodes included in the cycle. Furthermore, we refer to all cycles of length two as \( 2 \)-cycles. Lastly, a path is a walk in which no node is visited more than once. The distance between two nodes \( u \) and \( v \) in an undirected graph is equal to the length of the shortest path between \( u \) and \( v \). In a directed graph, the distance from \( u \) to \( v \) is equal to the shortest path from \( u \) to \( v \). A pair of nodes are said to be connected if a path exists between them. \( G \) is connected if all pairs of nodes are connected. Otherwise, it is said to be unconnected or disconnected. An undirected and disconnected graph can contain components, which themselves can be connected. We call these connected components [7].

2.2.2 Centrality Metrics

Centrality measures are a good way of identifying key elements in a graph. One of the most important use cases of centrality measures is to characterize and rank the nodes of a network. This way, one can obtain a sorted list of nodes based on importance [7].

The following subsections will discuss different versions of centrality. Two centrality classes will be discussed, namely centrality based on node degree and centrality based on paths between pairs of nodes. Centrality measures based on node degree are degree centrality and eigenvector centrality. While measures for paths
between pairs of nodes are *closeness centrality* and *betweenness centrality*. When computing measures based on paths one must have a connected graph.

Furthermore, \( k_i^{\text{in}} \) and \( k_i^{\text{out}} \) denotes in-degree and out-degree for node \( i \in \{1, 2, ..., N\} \) respectively. The adjacency matrix \( A \) of a graph \( G \) has dimensions \( N \times N \), and its elements are equal to 1 or 0. Element \( a_{ij} \) is 1 if, and only if there is a directed edge from node \( i \) to node \( j \) in \( G \). Otherwise, the element is 0. All centrality measures covered are defined as by V. Latora et al. in [7].

### Degree Centrality

The easiest way to think of centrality measures is by looking at the *degree centrality* (DC). DC states that the higher in-degree a node has, the more information it has available. On the contrary, the higher the out-degree, the higher the influence on its neighbors. The DC of node \( i \) in an undirected network is computed according to

\[
DC_i = \sum_{j=1}^{N} a_{ij}.
\]  

(2.1)

### Eigenvector Centrality

Eigenvector centrality (EC) takes the centrality of neighboring nodes into account. Hence, high EC means a node is connected to nodes with high centrality scores. One can say that the centrality metric spreads among neighbors or that neighbors share centrality. This means that if two nodes share the same neighbors, not only do they have the same degree of centrality, but they also have the same eigenvector centrality. In mathematical terms, we get

\[
\lambda c^E_i = \sum_{j=1}^{N} a_{ij} c^E_j,
\]

(2.2)

where \( \lambda \) is a positive constant, \( c^E_i \) the EC for node \( i \), \( a_{ij} \) the element of the adjacency matrix which represents any connections between node \( i \) and node \( j \) and \( c^E_j \) is the EC of neighbor \( j \). In practice, however, this becomes a set of linear equations, with the number of equations directly proportional to the number of nodes. By rewriting Equation 2.2 to vector notation we obtain

\[
Ac^E = \lambda c^E,
\]

(2.3)

where \( A \) is the \( N \times N \) adjacency matrix. By identifying eigenvalues and their corresponding eigenvectors, we achieve multiple solutions to the above equation. However, we are only interested in one of the \( N \) possible solutions, preferably one
that yields all positive values $c_i^E$. If $A$ is a non-negative (that is, we do not allow negative weights) square matrix and $G$ is strongly connected, this is achieved by choosing the solution given by the eigenvector corresponding to the leading eigenvalue. The leading eigenvalue is positive and greater than or equal to all other eigenvalues. Once obtained, we get the EC of node $i$ as

$$EC_i = c_i^E,$$  \hspace{1cm} (2.4)

where $c$ is the eigenvector corresponding to the leading eigenvalue.

**Closeness Centrality**

Closeness centrality (CC) is based on the idea that a node close to other nodes is central because of its ability to interact with them quickly. To quantify this centrality, one can consider the sum of the distances from the given node to all other nodes in the network. Then, a node with a low sum distance is, on average, close to other nodes in the network. We let the CC of node $i$ for a connected graph be defined as

$$CC_i = \frac{1}{\sum_{j=1}^{N} d_{ij}},$$  \hspace{1cm} (2.5)

and we define the CC of node $i$ for a disconnected graph as

$$CC_i = \frac{n - 1}{\sum_{j=1}^{N} d_{ij}},$$  \hspace{1cm} (2.6)

where $d_{ij}$ is the distance between node $i$ and node $j$. For Equation 2.6, $n - 1$ is the number of reachable nodes from node $i$.

**Betweenness Centrality**

Betweenness centrality (BC) focuses on interactions between non-adjacent nodes in a network depending on nodes that lie on a path between them. Therefore, the nodes on the path between the nodes have a certain control or influence on which information can travel along that path. Hence, this measure shows nodes on the shortest path between many pairs of nodes as central. Assume that communications between nodes only travel along the shortest paths between them, we then define the BC measure as

$$BC_i = \sum_{j=1}^{N} \sum_{k=1}^{N} \frac{n_{jk}(i)}{n_{jk}},$$  \hspace{1cm} (2.7)
where \( n_{jk} \) is the number of shortest paths from node \( j \) to node \( k \), whereas \( n_{jk}(i) \) is the number of shortest paths from node \( j \) to node \( k \), containing node \( i \).

## 2.3 Community Detection

Detecting communities in networks has been of interest for as long as network- ing science has been around. One of the most fundamental questions regarding community detection is, "What is a community?". Different answers to this question will lead to different community detection methods. Thus, the user has to determine what should be regarded as a community in the given network and proceed from there [14]. In general, however, a community is a group of nodes with more edges "inside" the community than edges linking nodes of the community with the rest of the graph. In most cases, communities are algorithmically defined, meaning they are just the final product of an algorithm [15].

To understand the properties and behavior of a network correctly, it is essential to identify its communities [16]. Figure 2.1 shows a graph with five communities, with nodes colored according to community belonging.

![Figure 2.1: Illustration of communities in an undirected graph.](image)

### 2.3.1 Modularity

Modularity is a way of quantifying how well the partitioning of a graph into communities separates the communities from one another. Given a graph \( G \) with community partitions \( P_N = \{C_1, C_2, ..., C_M\} \) of its nodes into \( M \) sets, the modularity \( Q_P \) of a partition \( P_N \) is computed as

\[
Q_P = \sum_{m=1}^{M} \left[ \frac{K_{mm}}{K} - \left( \frac{K_m}{2K} \right)^2 \right].
\] (2.8)
Where $K_m$ is the total degree of set $C_m$, $K_{mm}$ is the number of internal edges within set $C_m$, and $2K$ is the total amount of edges in the graph. $Q_P$ is by definition smaller than one and can assume negative values.

The first term in each summand is the fraction of internal edges of set $C_m$, whereas the second term represents the expected fraction of internal edges of $C_m$ in a random graph with the same degree for each $C_m$ as in $G$. In a random graph, each node has an equal chance of having an edge going to any other node. The probability of an edge existing between nodes from different sets $C_m$ and $C_m'$ becomes proportional to the product of the degrees $K_m$ and $K_{m'}$ of the two sets. Therefore, the probability of an edge between nodes in the same set $C_m$ given a random graph is proportional to $K_m^2$. We say that the nodes in set $C_m$ form a real community if the number of edges inside $C_m$ is larger than the expected number of edges in $C_m$ in the random case. The nodes of $C_m$ are then more tightly connected than expected.

$Q_P \approx 0$ indicates that partitions are meaningless and that connections in the graph given the current partition are random and, therefore do not imply any notable community structure. $Q_P \approx 1$, implies that the network has more structure than one could expect by pure chance. $Q_P < 0$ indicates the existence of sets of nodes with more edges between the sets than within the sets [7].

### 2.3.2 Label Propagation

The label propagation algorithm for community detection uses a network structure to assign labels to previously unlabeled data points. The algorithm’s initialization requires a subset of the nodes in the network to have labels assigned to them. This subset of nodes is generally very small, which is one of the perks of the algorithm. During the course of the algorithm, labels propagate to unlabeled data points in an "epidemic" manner. At the end of the algorithm, connected nodes with the same label form a community [17].

#### Synchronous vs Asynchronous LPA

There exists both synchronous and asynchronous LPAs. In the synchronous versions, propagation can be performed on all nodes in parallel, while asynchronous versions only provide sequential propagation. Both algorithms have their drawbacks, the synchronous version does not necessarily guarantee convergence. While for the asynchronous model, performance can be worse. As might be expected, the synchronous version tends to be faster for large networks [18].

#### Directed Label Propagation Algorithm

When identifying communities, it is normal practice to make directed graphs undirected. Hence, the standard form of LPA is shaped for handling undirected networks. Although, [19] presents a modified version of the LPA, the directed label propagation algorithm (DLPA). The purpose of the DLPA is not to lose the information in the direction of the edges, yielding better results than LPA. Their
results show that DLPA outperformed LPA on both real and synthetic networks when detecting communities.

2.4 Association Rule Mining

Association rule mining is an important topic in data mining and is used to discover interesting relations between variables in large databases. Association rules are meant to discover rules that determine how and/or why items are connected [9]. In this section, one of the most common algorithms for finding such rules is discussed with important concepts.

An example database will be used to give some examples of association rule mining measures and algorithms. This database is given by Table 2.1 with its corresponding frequency table in Table 2.2. Note that the frequency table is sorted in descending order. The total set of items is given by \( I = \{ A, B, C, D, E, F, G, L, M, O, P \} \).

<table>
<thead>
<tr>
<th>ID</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{A}, {C}, {D}, {F}, {G}, {I}, {M}, {P}</td>
</tr>
<tr>
<td>2</td>
<td>{A}, {B}, {C}, {F}, {L}, {M}, {O}</td>
</tr>
<tr>
<td>3</td>
<td>{B}, {F}, {H}, {J}, {O}</td>
</tr>
<tr>
<td>4</td>
<td>{B}, {C}, {K}, {S}, {P}</td>
</tr>
<tr>
<td>5</td>
<td>{A}, {C}, {E}, {F}, {L}, {M}, {N}, {P}</td>
</tr>
</tbody>
</table>

*Table 2.1: Example database.*

<table>
<thead>
<tr>
<th>Action</th>
<th>Support count</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A]</td>
<td>3</td>
</tr>
<tr>
<td>[B]</td>
<td>3</td>
</tr>
<tr>
<td>[C]</td>
<td>4</td>
</tr>
<tr>
<td>[D]</td>
<td>1</td>
</tr>
<tr>
<td>[E]</td>
<td>1</td>
</tr>
<tr>
<td>[F]</td>
<td>4</td>
</tr>
<tr>
<td>[G]</td>
<td>1</td>
</tr>
<tr>
<td>[L]</td>
<td>2</td>
</tr>
<tr>
<td>[M]</td>
<td>3</td>
</tr>
<tr>
<td>[O]</td>
<td>2</td>
</tr>
<tr>
<td>[P]</td>
<td>3</td>
</tr>
</tbody>
</table>

*Table 2.2: Frequency table.*

2.4.1 Important Concepts

There are multiple ways of measuring the goodness of rules mined by association rule mining algorithms. These measures are used as thresholds such that only rules which satisfy specific criteria are generated. Some of these measures will be presented in this section with examples of computations based on the data in Table 2.1. This thesis will use the measures support, confidence, and lift, three of the most established measures for association rule mining [9].

Support

Support indicates how frequent a set of items is in a given database. A mathematical formulation for the support of a set of items \( X = \{x_1, x_2, ..., x_n\} \), is given
by

\[ supp(X) = P(X), \]  

(2.9)

where \( P(X) \) is the joint probability of all items in \( X \). Hence, support is a measure of in how many rows all items in \( X \) exist. For example, given the database in Table 2.1, \( supp(F, M) = 0.6 \).

**Confidence**

The percentage of the rows in the database which satisfy \( X = \{x_1, x_2, ..., x_n\} \), that also satisfy the set of items \( Y = \{y_1, y_2, ..., y_m\} \), is called confidence and is defined as

\[ conf(X; Y) = P(X|Y) = \frac{P(X, Y)}{P(Y)}. \]  

(2.10)

For example, looking at the database in Table 2.1, the confidence that \( F \) is present given that \( M \) is present in the database is \( conf(F; M) = P(F|M) = 1 \). Also note that \( conf(M; F) = \frac{3}{4} \).

**Lift**

With \( X = \{x_1, x_2, ..., x_n\} \) and \( Y = \{y_1, y_2, ..., y_m\} \), we define the lift of a rule as

\[ lift(X, Y) = \frac{P(X, Y)}{P(X)P(Y)} = \frac{P(X|Y)}{P(X)} = \frac{P(Y|X)}{P(Y)}. \]  

(2.11)

If \( lift(X, Y) = 1 \), the occurrence of \( X \) and \( Y \) are independent. Hence, no rule can be drawn from this. However, \( lift(X, Y) > 1 \) implies that there is a dependency between the occurrence of \( X \) and \( Y \). More specifically, it means that \( X \)'s occurrence increases the probability of \( Y \) occurring and vice versa. Finally, \( lift(X, Y) < 1 \) implies that \( X \) and \( Y \) are each other's substitutes or that the occurrence of one item hurts the occurrence of the other item.

For example, since \( P(F, M) = \frac{3}{5} = 0.6 \), \( P(F) = \frac{4}{5} = 0.8 \) and \( P(M) = \frac{3}{5} = 0.6 \), we can compute \( lift(F, M) = \frac{0.6}{0.8 \cdot 0.6} = 1.25 \).

### 2.4.2 Apriori Algorithm

The Apriori algorithm is one of many algorithms used for association rule mining. It starts with identifying individual frequent items in the given database. Then, it proceeds by extending these to more significant and extensive sets of items as long as the created sets appear sufficiently often in the database [20].
Assume the database is given by 2.1. The Apriori algorithm starts with producing a frequency table containing the support count of all items. The frequency list for the given database can be found in Table 2.2. The algorithm takes as a parameter the minsup, which is the minimum support required for a set of items to be included in the solution. In our example, assume minsup = 0.6. This yields that neither D, E, G, L, nor O will be included since their support is less than minsup. The support of all remaining items is listed in Table 2.3.

<table>
<thead>
<tr>
<th>Action</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>0.6</td>
</tr>
<tr>
<td>{B}</td>
<td>0.6</td>
</tr>
<tr>
<td>{C}</td>
<td>0.8</td>
</tr>
<tr>
<td>{F}</td>
<td>0.8</td>
</tr>
<tr>
<td>{M}</td>
<td>0.6</td>
</tr>
<tr>
<td>{P}</td>
<td>0.6</td>
</tr>
</tbody>
</table>

*Table 2.3: All items with minsup ≥ 0.6.*

From this, the Apriori algorithm creates all unique supersets of size two and calculates the support count for each of these supersets. It is worth mentioning two fundamental aspects of the apriori algorithm:

- All subsets of a frequent itemset are frequent.
- If an itemset is infrequent, all its supersets are infrequent.

Once the support of all supersets has been obtained, the algorithm removes all supersets with support ≤ minsup. Table 2.4 and Table 2.5 shows these steps. These steps are iterated with bigger and bigger itemsets until no more frequent itemsets can be found. The algorithm then terminates. For this example, all iterations can be seen in Table 2.4 - 2.7.

From this, rules can be mined from supersets that satisfy the minsup-criterion. Rules are in the form of \{Antecedent\} ⇒ \{Consequence\}, where both the antecedent and consequence are sets of items. An example using R\(^1\) together with the arules\(^2\) package and the database given by Table 2.1 with minsup = 0.6 yields the output in Table 2.8.

The results show that many strong rules are generated from a relatively small database. Therefore, the difficult part is usually separating good and applicable rules from redundant and useless ones. A rule is said to be redundant if a more general rule with the same or higher confidence exists. Hence, in Table 2.8 for example, rule 26 is redundant because of rule 30, since rule 30 is more general and has equal confidence.

\(^1\)https://www.r-project.org/about.html
\(^2\)https://cran.r-project.org/web/packages/arules/index.html


### Table 2.4: Supersets of size two.  

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, B}</td>
<td>0.2</td>
</tr>
<tr>
<td>{A, C}</td>
<td>0.6</td>
</tr>
<tr>
<td>{A, F}</td>
<td>0.6</td>
</tr>
<tr>
<td>{A, M}</td>
<td>0.6</td>
</tr>
<tr>
<td>{A, P}</td>
<td>0.4</td>
</tr>
<tr>
<td>{B, C}</td>
<td>0.4</td>
</tr>
<tr>
<td>{B, F}</td>
<td>0.4</td>
</tr>
<tr>
<td>{B, M}</td>
<td>0.2</td>
</tr>
<tr>
<td>{B, P}</td>
<td>0.2</td>
</tr>
<tr>
<td>{C, F}</td>
<td>0.6</td>
</tr>
<tr>
<td>{C, M}</td>
<td>0.6</td>
</tr>
<tr>
<td>{C, P}</td>
<td>0.4</td>
</tr>
<tr>
<td>{F, M}</td>
<td>0.6</td>
</tr>
<tr>
<td>{F, P}</td>
<td>0.4</td>
</tr>
<tr>
<td>{M, P}</td>
<td>0.4</td>
</tr>
</tbody>
</table>

### Table 2.5: Supersets of size two with \( \text{minsup} \geq 0.6 \).  

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, C}</td>
<td>0.6</td>
</tr>
<tr>
<td>{A, F}</td>
<td>0.6</td>
</tr>
<tr>
<td>{A, M}</td>
<td>0.6</td>
</tr>
<tr>
<td>{C, F}</td>
<td>0.6</td>
</tr>
<tr>
<td>{C, M}</td>
<td>0.6</td>
</tr>
<tr>
<td>{F, M}</td>
<td>0.6</td>
</tr>
</tbody>
</table>

### Table 2.6: Supersets of size three.  

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, C, F}</td>
<td>0.6</td>
</tr>
<tr>
<td>{A, C, M}</td>
<td>0.6</td>
</tr>
<tr>
<td>{A, F, M}</td>
<td>0.6</td>
</tr>
<tr>
<td>{C, F, M}</td>
<td>0.6</td>
</tr>
</tbody>
</table>

### Table 2.7: Superset of size four.  

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, C, F, M}</td>
<td>0.6</td>
</tr>
</tbody>
</table>

### 2.4.3 Sequential Rule Mining  

Sequential rule mining (SRM) works very similar to association rule mining. The key difference is that sequential rules are mined from ordered sequences, while association rules are mined from transactions. Association rules do not consider order, while sequential rules do [21]. This thesis uses \( \text{cSPADE}^3 \) (constrained Sequential Pattern Discovery using Equivalence classes) [23] for mining sequential rules in R. Table 2.9 shows the resulting rules mined with the \( \text{cSPADE} \)-algorithm given \( \text{minsup} = 0.6 \) when used on the data provided by Table 2.1. The amount of mined rules is less than in 2.8, solely because of consideration given to the order in which items are given in the database. Table 2.9 is generated with the only constraint being \( \text{minsup} \). By applying the constrain \( \text{maxlen} = 2 \), only rules with one item as antecedent and one as a consequence will be considered. Table 2.10 shows the results after applying this constraint.

---

3[https://cran.r-project.org/web/packages/arulesSequences/index.html](https://cran.r-project.org/web/packages/arulesSequences/index.html)
Table 2.8: Rules generated with the Apriori algorithm in R.

<table>
<thead>
<tr>
<th>Rule NR</th>
<th>Antecedent</th>
<th>Consequence</th>
<th>Support</th>
<th>Confidence</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{P}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>2</td>
<td>{C}</td>
<td>{P}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>3</td>
<td>{A}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>4</td>
<td>{M}</td>
<td>{A}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>5</td>
<td>{A}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>6</td>
<td>{F}</td>
<td>{A}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>7</td>
<td>{A}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>8</td>
<td>{C}</td>
<td>{A}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>9</td>
<td>{M}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>10</td>
<td>{F}</td>
<td>{M}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>11</td>
<td>{M}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>12</td>
<td>{C}</td>
<td>{M}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>13</td>
<td>{F}</td>
<td>{C}</td>
<td>0.60</td>
<td>0.75</td>
<td>0.94</td>
</tr>
<tr>
<td>14</td>
<td>{C}</td>
<td>{F}</td>
<td>0.60</td>
<td>0.75</td>
<td>0.94</td>
</tr>
<tr>
<td>15</td>
<td>{A, M}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>16</td>
<td>{A, F}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>17</td>
<td>{F, M}</td>
<td>{A}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>18</td>
<td>{A, M}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>19</td>
<td>{A, C}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>20</td>
<td>{C, M}</td>
<td>{A}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>21</td>
<td>{A, F}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>22</td>
<td>{A, C}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>23</td>
<td>{C, F}</td>
<td>{A}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>24</td>
<td>{F, M}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>25</td>
<td>{C, M}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>26</td>
<td>{C, F}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>27</td>
<td>{A, F, M}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>28</td>
<td>{A, C, M}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>29</td>
<td>{A, C, F}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>30</td>
<td>{C, F, M}</td>
<td>{A}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
</tbody>
</table>
Table 2.9: Rules mined with cSPADE, \textit{minsup} = 0.6.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Antecedent</th>
<th>Consequence</th>
<th>Support</th>
<th>Confidence</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{A}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>2</td>
<td>{C, F}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>3</td>
<td>{A, F}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>4</td>
<td>{A, C, F}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>5</td>
<td>{A, C}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>6</td>
<td>{A}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>7</td>
<td>{A}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>8</td>
<td>{A, C}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>9</td>
<td>{C}</td>
<td>{M}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>10</td>
<td>{F}</td>
<td>{M}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>11</td>
<td>{C}</td>
<td>{P}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>12</td>
<td>{C}</td>
<td>{F}</td>
<td>0.60</td>
<td>0.75</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table 2.10: Rules mined with cSPADE, \textit{minsup} = 0.6 and \textit{maxlen} = 2.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Antecedent</th>
<th>Consequence</th>
<th>Support</th>
<th>Confidence</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{A}</td>
<td>{M}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.67</td>
</tr>
<tr>
<td>2</td>
<td>{A}</td>
<td>{C}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>3</td>
<td>{A}</td>
<td>{F}</td>
<td>0.60</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>4</td>
<td>{C}</td>
<td>{M}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>5</td>
<td>{F}</td>
<td>{M}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>6</td>
<td>{C}</td>
<td>{P}</td>
<td>0.60</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>7</td>
<td>{C}</td>
<td>{F}</td>
<td>0.60</td>
<td>0.75</td>
<td>0.94</td>
</tr>
</tbody>
</table>
This chapter focuses on giving the reader some understanding of the structure of the data as well as describing the pre-processing. This chapter aims to increase reproducibility, get a better overview of the data characteristics, and motivate why some FMAN-pages are not used during implementation.

3.1 Data and Notation

The used data is classified, although a mapping from the real data onto a network structure will make the network nodes and edges anonymous. The data consists of 527 FMAN-pages $F_m$, $m \in M = \{1, \ldots, 527\}$ and 405 unique action texts $a_i$, $i \in I = \{1, \ldots, 405\}$. The set of all actions in $F_m$ is denoted $A_m$, where $m \in M$. Let $K_m$ be the total number of actions $a_i$ in $F_m$, and $k \in K = \{1, \ldots, K\}$. Then, if $a_i$ is action number $k$ in the sequence of actions in $F_m$, we say that $A_m(k) = a_i$.

We use the notation of $a_i$, $a_j$ for two unique actions, where $i, j \in I$, $i \neq j$. Similarly, for two unique FMAN-pages we have $F_m$, $F_n$, where $m, n \in M, m \neq n$. Action texts are partially ordered according to priority within each FMAN-page. We say $a_i > a_j$ if action $a_i$ is listed before action $a_j$ in the given FMAN-page. Furthermore, every $F_m$ is assigned a priority $p_m \in P$, where $P = \{1, 2, 3\}$. When the priority is necessary for context, we denote $F_m$ as $F_m^{p_m}$.

Some FMAN-pages are already known to be connected from prior knowledge about the systems, which can be taken advantage of. These connections are in the form of primaries and secondaries, which means that if a primary alert is active, its secondaries will most likely follow due to consequential errors. Let $F_m^{p_m}$ be primary to $F_n^{p_n}$, which is equivalent to letting $F_n^{p_n}$ being secondary to $F_m^{p_m}$. Properties of the priority state that for this dependency to be possible, it must hold
that \( p_m \geq p_n \). We denote this dependency as \( F_{n}^{p_n} \rightarrow F_{m}^{p_m} \), which may at first glance seem confusing. Note however that the dependency works such that the contents of \( F_n \) may alter the contents of \( F_m \), hence the direction of the dependency. An effect from this is that if both \( F_n \) & \( F_m \) are active, we display \( F_m(F_n) \) which content is pre-determined from prior knowledge. One can think of \( F_m(F_n) \) as a new FMAN-page, \( F_{m,n} \). Primaries and secondaries can be either direct or indirect. We say that an alert \( F_m \) is a direct primary to another alert \( F_n \) if \( F_n \rightarrow F_m \). Furthermore, we say that all primaries of \( F_m \) that are not direct primaries to \( F_m \) are indirect primaries to \( F_n \). It is also true that \( F_n \) is an indirect secondary alert to the primary alerts of \( F_m \) which are not direct primary alerts to \( F_n \). Simply put, all alerts being directly dependent on one another are called direct primaries/secondaries, while alerts being indirectly dependent on one another are called indirect primaries/secondaries.

### 3.2 Filtering the Data

One of the delimitations of the thesis is to only account for FMAN-pages that might be displayed for the pilot during flight. Because of this, FMAN-pages connected to start up-checks were removed. Furthermore, some FMAN-pages have been deemed so vital for the pilots, such that when one of these are displayed, all other FMAN-pages are suppressed. Therefore, these FMANs were also removed from the data since how they merge with other FMANs has already been decided. Once removed, only 470 FMAN-pages remain in the data, together with 359 action texts, compared to the 527 FMAN-pages and 405 action texts contained in the original data.

### 3.3 Data Distribution

Looking at Figure 3.1, one can see that most FMAN-pages have a length of three or fewer actions. In fact, 78% of all FMAN-pages have three actions or less. Figure 3.2 shows the distribution of actions among FMAN-pages in a better way. Note that FMAN-pages and actions have been mapped to integers for confidentiality reasons.
3.3 Data Distribution

It can be seen that actions 1, 66, 79, and 122 seem to occur frequently, which is better seen in Figure 3.3. These actions are declassified to *Abort mission*, *Choose normal runway*, *Complete mission at pilot discretion* and *Fly gently*, all of which are pretty general actions, which is why they occur so frequently. By removing these four most frequent actions, we acquire the histogram in Figure 3.4, where the frequencies of the less frequent actions are visible.
Figure 3.3: Histogram showing the frequency of each action.

Figure 3.4: Histogram showing the frequency of each action. Action 1, 66, 79, and 122 were removed.
This section covers how the theory and concepts described in Chapter 2 were used throughout the project. The label-propagation will sort the actions into different sets that must appear in the correct order. Then we apply rule mining to sort the actions within those sets.

### 4.1 Creation of Networks

Three different networks were created so that different versions of LPAs could be tested. All networks share the same nodes, but the edges are somewhat different. These networks will, from here on, be referred to as Network A, Network B and Network C. Figure 4.1 shows examples of creating each network.

#### 4.1.1 Network A

Network A has directed edges from each action to the action directly below it on an FMAN-page, and all edges have weights equal to one. By doing this for all FMAN-pages, a network with 609 edges is created.

#### 4.1.2 Network B

Each node in Network B has weighted edges to all actions below it on the same FMAN-page. We apply edge weights proportional to the distance between the pair of actions to which the edge connects. We end up with a network of 2873 edges, more than three times the number of edges for Network A.
4.1.3 Network C

Network C is created similarly to Network B but additionally contains edges from each action to the actions above it on the same FMAN-page. This results in Network C having 5746 edges, precisely double the number of edges of Network B.

4.2 Identifying Connected Components

For every unconnected component, it holds that actions in the given component have no connections outside the component. For an LPA to perform well, all nodes in the network must be connected. To solve the problem of having multiple components, one can remove the smaller components and focus on the main component, force connections between components, or run the LPA on each connected component individually. For this thesis, we choose not to remove the smaller components so that we can analyze as many of the actions as possible.

A total of fifteen components could be identified, and one of them is a large component connecting 334 out of the 359 nodes in the network, which equals 93% percent of the nodes. Figure 4.2 shows the created network of actions. Note that some of the smaller components consist of two or more nodes. For clarification, the smaller components consist of FMAN-pages that do not share their actions with the largest component.
4.3 Central Nodes

Centrality scores are computed for the undirected network illustrated by Figure 4.2. DC, EC, CC, and BC are calculated for all nodes, and the results are shown in Figure 4.3. Results are used to identify the most central nodes for the initialization part of the LPAs. We can see that some of the nodes score well for multiple measures. Nodes 1, 66, 79, and 122 are the highest-scoring nodes for DC, EC, and BC. Notice that since the network in Figure 4.2 is disconnected, we use CC as defined by Equation 2.6.

Furthermore, in the plot for the CC measure, we can see that some of the nodes have CC scores close to zero. These nodes are all 25 nodes being disconnected from the large network component. Since our way of defining CC for a disconnected network has the number of reachable nodes in the numerator, nodes being part of large components will tend to have higher scores, given that the average shortest distance for these nodes is not that much larger. Node 79 (Complete mission at pilot discretion) has the highest score in all measures and is the biggest node in the upper right part of the network in Figure 4.2. Looking at the figure, we see that most nodes on the right side of this big node must pass through it to reach any other node in the network. This indicates that node 79 also has a high BC score since it lies on the path of many shortest paths.
Figure 4.3: Centrality scores.
4.4 Label Propagation

Label propagation is used to help identify communities in the networks. By manually assigning labels to a subset of nodes and letting labeled nodes spread information about their label to neighbors, we end up with a network divided into communities. Eight labels representing different priorities were used; these are listed in Table 4.1. Note that priority zero, which is associated with Immediate, is the highest prioritized label. Experienced engineers and pilots at Saab carefully chose the labels. The amount of labels are not certain to be the optimal number for the network of interest. However, it is a good and sufficient example of labels.

<table>
<thead>
<tr>
<th>Label</th>
<th>Priority</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlabeled</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Immediate</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Mitigation</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Tactical</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Continuation</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Envelop</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Prepare to land</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Landing</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>On Ramp</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

The community detection allows for a rough ordering of the combined actions of multiple FMAN-pages. All actions in the Total-FMAN can be sorted based on their priority of the community to which they belong. This procedure cannot handle the scheduling of actions internally within each community; for this, we will use association rule mining.

4.4.1 Initialization

The initialization is crucial for the propagation of labels. Each label must be associated with at least one node after initialization. Otherwise, other nodes in the network will have no way of being assigned that label. First, the most central nodes were identified by looking at the top fifteen nodes from all centrality measures and storing the common nodes. This resulted in a list of ten nodes, which were all labeled by the experienced engineers according to our list of available labels. Unfortunately, On Ramp could not be assigned to any of these nodes, so we had to identify a node we knew could be associated with this label. Hence, eleven nodes in the big component were initially labeled to cover the range of all labels. Since labels can not propagate from one network component to another, a decision was made to manually label all 25 nodes not being connected to the largest component. This resulted in the network having a total of 36 nodes being labeled manually, eleven in the main component of the network and 25 outside
4 Implementation

Figure 4.4: The network of actions after initialization.

of this component. Figure 4.4 shows the network after initialization, with colors according to Table 4.1.

4.4.2 Propagation – Network A

Network A is the simplest of the three networks, also true for its propagation. Pseudocode for the algorithm that propagates labels in Network A is given by Algorithm 1. It begins with defining a Drawpile from the initialized nodes and continues running as long as the Drawpile is not empty. Initialized nodes have 1 in the position corresponding to the priority of their label and 0 otherwise. For example, if a node is initialized as Immediate, its label vector is initialized as $[1, 0, 0, 0, 0, 0, 0, 0]$. Each iteration copies a random node $u$ from Drawpile and takes a random outgoing edge $e$ from this node pointing to node $v$. If $v$ is not already labeled, it inherits $u$’s label vector, and $v$ is added to the Drawpile. If $v$ was already labeled, however, it will only inherit the label vector if the priority of its current label is larger ($p(v) > p(u)$) than the priority of $u$’s label. It then deletes the edge $e$ from the graph. If $e$ was the last outgoing edge from $u$, then $u$ is removed from the Drawpile. Once the algorithm has finished we can retrieve the label associated with a node by looking at what index of its labelvector contains the maximum value.
Algorithm 1 Pseudocode for propagating labels in Network A

**Input:** \( G(V, E) \) and a list of labeled nodes \( L \)
**Output:** Labeled nodes

**Definition 1:** Let \( l(v) \) notate the label vector of node \( v \)

**Definition 2:** Let the index of \( \arg \max(l(v)) = p(v) \) be the priority of \( v \)'s label

1: Drawpile ← copy of \( L \)
2: while Drawpile ≠ [] do
3: \( u \) ← copy of a random node from Drawpile
4: if \( u \) has no outgoing edges then
5: remove \( u \) from Drawpile
6: continue
7: end if
8: \( e \) ← copy of a random edge from \( u \)
9: \( v \) ← copy of node which \( e \) points to
10: if \( v \not\in L \) then
11: \( l(v) \leftarrow l(u) \)
12: append \( v \) to \( L \)
13: else
14: if \( p(v) > p(u) \) then
15: \( l(v) \leftarrow l(u) \)
16: end if
17: end if
18: if \( v \) not in Drawpile and \( v \) has at least one outgoing edge then
19: append \( v \) to Drawpile
20: end if
21: remove \( e \) from \( G_A \)
22: end while

### 4.4.3 Propagation – Network B & C

The propagation for Network B and Network C is carried out using the same algorithm. Both of these networks have edge weights based on the distance between nodes in the network, which allows propagation of a certain fraction of a label vector, with the fraction being inversely proportional to the weight of the edge. Hence, if an edge between a pair of nodes has a weight equal to one, we get similar propagation to Network A. Meanwhile, we get less impact from the node from which we propagate the label information for weights larger than one. The pseudocode for this propagation is given in Algorithm 2. Although the same algorithm is used for Network B and Network C, only Network C will be able to associate all of its nodes with a label once the algorithm terminates, and this is simply because of how we choose to create the networks. Since the nodes in Network B only have outgoing edges pointing downwards, some nodes will never be able to inherit any label information. For example, if we look at Figure 4.1(b) we can see that label information cannot propagate upwards because of the direction of edges. As for Algorithm 1, we can retrieve the label associated with a node by...
looking at what index of its labelvector contains the maximum value. Note that for this algorithm, a fraction of the labelvector of \( u \) propagates to \( v \) regardless of the current label of \( v \), which is different from Algorithm 1.

**Algorithm 2** Pseudocode for propagating labels in Network B and Network C

**Input:** \( G(V, E) \), a list of labeled nodes \( L \) and \( \alpha > 0 \).

**Output:** Labeled nodes

**Definition 1:** Let \( l(v) \) notate the label vector of node \( v \).

**Definition 2:** Let \( w(e) \) notate the weight on edge \( e \).

1: Drawpile ← copy of \( L \)
2: while Drawpile ≠ [] do
3:   \( u \) ← copy of a random node from Drawpile
4:   if \( u \) has no outgoing edges then
5:     remove \( u \) from Drawpile
6:     continue
7:   end if
8:   \( e \) ← copy of a random edge from \( u \)
9:   \( v \) ← copy of node which \( e \) points to
10:  if \( v \notin L \) then
11:    \( l(v) \leftarrow l(u) \times \frac{1}{w(e)} \times \alpha \)
12:    append \( v \) to \( L \)
13:  else
14:    \( l(v) \leftarrow l(v) + l(u) \times \frac{1}{w(e)} \times \alpha \)
15:  end if
16:  if \( v \) not in Drawpile and has at least one outgoing edge then
17:    append \( v \) to Drawpile
18:  end if
19:  remove \( e \) from \( G_A \)
20:  if \( u \) have no more outgoing edges then
21:    remove \( u \) from Drawpile
22:  end if
23: end while

4.4.4 Completing the propagation of Networks A & B

Since both Network A and Network B can only propagate label information downwards according to Figure 4.1, an algorithm for propagating labels such that all nodes can be assigned labels had to be implemented. The pseudocode for this algorithm is given in Algorithm 3. This algorithm is very similar to Algorithm 1, with the main difference being that it uses incoming edges to backpropagate label information. It begins with defining the Drawpile as a copy of the already labeled nodes \( L \). It also has a list of all unlabeled nodes \( U \) and runs as long as there are still nodes in \( U \). Each iteration takes a random node \( u \) and stores all incoming edges to \( u \) which have weight = 1 in the variable \( E_{in} \). If there are no such edges, it removes \( u \) from Drawpile and continues with the next iteration. Otherwise, it
takes a random edge $e$ from $E_{in}$ and lets $v$ be a copy of the node which points to $u$ via $e$. If $v$ is unlabeled, it inherits $u$’s label vector, and $v$ is moved from the list of unlabeled nodes $U$ to the list of labeled nodes $L$. If $v$ already is assigned a label, however, the label vector is only inherited if the priority of its current label is lower than the priority of $u$’s label. The edge $e$ is then removed from $G(V, E)$ and if $v$ is not already in Drawpile and has incoming edges, it is appended to Drawpile.

Algorithm 3 Pseudocode for completing the propagation of $G_A$ and $G_B$

Input: $G(V, E)$ a list of labeled nodes $L$, and a list of unlabeled nodes $U$
Output: Labeled nodes

Definition 1: Let $l(v)$ notate the label vector of node $v$
Definition 2: Let the index of $\arg\max(l(v)) = p(v)$ be the priority of $v$’s label

1: Drawpile ← copy of $L$
2: while $U \neq []$ do
3: $u ←$ random node from Drawpile
4: $E_{in} ←$ all incoming edges to $u$ with weight = 1
5: if $E_{in} = \emptyset$ then
6: remove $u$ from Drawpile
7: continue
8: end if
9: $e ←$ random edge from $E_{in}$
10: $v ←$ copy of node which point to $u$ via $e$
11: if $v \notin L$ then
12: $l(v) ← l(u)$
13: append $v$ to $L$
14: remove $v$ from $U$
15: else
16: if $p(v) > p(u)$ then
17: $l(v) ← l(u)$
18: end if
19: end if
20: remove $e$ from $G$
21: if $v \notin$ Drawpile and has incoming edges then
22: append $v$ to Drawpile
23: end if
24: end while

4.5 Sequential Rule Mining

Sequential rules are mined using the cSPADE-algorithm with the restriction of minimum support being $0.002 < \frac{1}{470}$, and $maxlen = 2$. This means we allow a rule to have enough support as long as it appears once. We store all rules together with their support, confidence, and lift. Note that although label propagation is carried out with three networks, rule mining is done once and generated rules have nothing to do with networks described in Section 4.1.
4.6 **Graphical User Interface**

A Graphical User Interface (GUI) was created so that a user can quickly test the system. The GUI was divided into five frames, as shown in Figure 4.5.

![Figure 4.5: The GUI at start up, divided into frames.](image)

4.6.1 **Frame 1 – User Information**

The first frame is allocated space for user information. This frame explains how to use the other frames and clarifies how the underlying system works.

4.6.2 **Frame 2 – Display**

The second frame is where the result of the Total-FMAN is displayed. One can interact with this frame by using the three upmost positioned buttons on the right side of the frame. The button at the top is for refreshing the frame to see if any alerts have recovered or been triggered, and the remaining two buttons are for scrolling the page up and down if the number of actions is too large to fit within one page. Any newly activated actions are displayed as turquoise, while older actions are shown as white.

4.6.3 **Frame 3 – Alert Selection**

In frame three, the user can add alerts by selecting any alert in the drop-down menu at the top of the frame. The set alert gets added to a list of chosen alerts as a check button that marks the alert as active when checked. Furthermore, when adding an alert, all its direct and indirect secondaries automatically get added to
a similar list of chosen alerts. An alert can only be in one of those lists at a time, and when added to any of the lists, it is removed from the drop-down menu. Both lists have buttons for activating/deactivating all alerts from the given list. There is also a button for adding default alerts. By pressing this button, six pre-selected alerts get added to the list of chosen alerts, and these are the same alerts being experimented with by Rahm and Nilsson in their thesis during 2022 [6]. The red button at the very bottom of the frame is for resetting the GUI to its initial state.

4.6.4 Frame 4 – Alert Preview

To keep track of which alert contains what actions, a frame for previewing alerts and understanding their place in the network was created. This frame contains an identical drop-down menu as frame three, from which one can choose any alert of interest to display its actions. The selected alert also shows all its indirect and direct primaries and indirect and direct secondaries. This way, by setting an alert and previewing it, one can better understand the content and context of the alert.

4.6.5 Frame 5 – Network Visualization

The final frame is for visualizing the network in use. Hovering the pointer over any node displays what action text that node represents. When alerts have been selected and the Total-FMAN is displayed in frame 2. Only the active nodes and their connections to one another will be shown in the network. The colors of the nodes are the same as their corresponding texts in frame 2. By following the connections between active nodes in the network, one can understand why the list of actions is sorted the way it is.
This chapter contains the results from both the label propagation and the SRM. We talk about cycle detection and how results from the SRM are used for identifying and deleting cycles. Lastly, we provide examples from the Total-FMAN and discuss the results.

5.1 Label Propagation

The results of propagating labels in the networks can be seen in Figures 5.1 – 5.3, and Table 5.1 shows the distribution of labels in the networks.

Table 5.1: Label distribution after propagation.

<table>
<thead>
<tr>
<th>Label</th>
<th>Network A</th>
<th>Network B</th>
<th>Network C</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td>3.62%</td>
<td>59.61%</td>
<td>2.51%</td>
<td></td>
</tr>
<tr>
<td>Mitigation</td>
<td>52.92%</td>
<td>3.06%</td>
<td>9.75%</td>
<td></td>
</tr>
<tr>
<td>Tactical</td>
<td>0.84%</td>
<td>0.84%</td>
<td>0.84%</td>
<td></td>
</tr>
<tr>
<td>Continuation</td>
<td>25.35%</td>
<td>24.23%</td>
<td>36.21%</td>
<td></td>
</tr>
<tr>
<td>Envelop</td>
<td>4.46%</td>
<td>3.62%</td>
<td>1.95%</td>
<td></td>
</tr>
<tr>
<td>Prepare to land</td>
<td>0.00%</td>
<td>6.69%</td>
<td>26.18%</td>
<td></td>
</tr>
<tr>
<td>Landing</td>
<td>11.98%</td>
<td>1.39%</td>
<td>7.24%</td>
<td></td>
</tr>
<tr>
<td>On Ramp</td>
<td>0.84%</td>
<td>0.56%</td>
<td>15.32%</td>
<td></td>
</tr>
</tbody>
</table>
5.1 Label Propagation

**Figure 5.1:** Result from the propagation of Network A.

**Figure 5.2:** Result from the propagation of Network B.

**Figure 5.3:** Result from the propagation of Network C.
As one can see in Figures 5.1 – 5.3, the result from the propagation varies depending on which network we choose to propagate labels in. Although, we can see that each network has nodes labeled as Continuation in a cluster of nodes located at the top-right part of the networks, so there are some similarities in the results. Also, by looking in Table 5.1, we can confirm that the number of nodes labeled as Continuation are about the same for all networks. This is especially true for networks A and B. By comparing networks A and B further, we see that nodes labeled as Mitigation in Network A are labeled as Immediate in Network B. Furthermore, many of the nodes labeled as Landing in Network A are instead labeled as Prepare to land in Network B. These differences are relatively small since Immediate and Landing are priority levels right beside each other. Furthermore, all networks have the same three nodes labeled as Tactical.

Comparing Network C to Network A and Network B, we notice that the labels are more evenly distributed in Network C. In Network A, Mitigation takes up almost 53% of all nodes, while for Network B, Immediate contains nearly 60% of nodes. In Network C, however, the most significant community is the one associated with nodes labeled as Continuation, which means more nodes are left for the remaining communities. Network A and Network B end up with big communities with high priority because of Algorithm 1 & 3 in Section 4.4.4. In these algorithms high-priority labels will spread much easier than low-priority labels. Since Network C can propagate labels to all of its nodes without having bias towards certain labels, it will not have the same risk of ending up with these big high-priority communities. Here, all labels are allowed to propagate independently, and the most probable label (the element in the label vector with the largest value at the end of the algorithm), is chosen.

5.2 Sequential Rules

2141 rules are generated with the minimum support set to 0.002. The top five and the bottom five sequential rules are listed in Table 5.2. One can see that the most supported rule has a support of 20.6% while the second most supported rule has a support of only 7.4%. Furthermore, when we reach the 10th most common rule, the support has already dropped to only 2.5%. As one can see, the last five rules all have support = 0.002, confidence = 0.004 and lift < 1.

5.3 Cycles

A large part of the results is the identification of cycles that are created from contradictory rules. This chapter will cover how these cycles can be identified and the consequences of trying to eliminate them. The existence of cycles is proven to be problematic. How can we decide which order to put actions in the Total-FMAN if rules contradict each other?
Table 5.2: Top five and bottom five sequential rules sorted primarily on support, followed by confidence and lift.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Antecedent</th>
<th>Consequence</th>
<th>Support</th>
<th>Confidence</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>122</td>
<td>1</td>
<td>0.206</td>
<td>0.752</td>
<td>2.506</td>
</tr>
<tr>
<td>2</td>
<td>122</td>
<td>66</td>
<td>0.074</td>
<td>0.271</td>
<td>2.452</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>66</td>
<td>0.070</td>
<td>0.234</td>
<td>2.115</td>
</tr>
<tr>
<td>4</td>
<td>122</td>
<td>244</td>
<td>0.034</td>
<td>0.124</td>
<td>2.429</td>
</tr>
<tr>
<td>5</td>
<td>122</td>
<td>299</td>
<td>0.030</td>
<td>0.109</td>
<td>3.643</td>
</tr>
</tbody>
</table>

| ...  | ...        | ...         | ...     | ...        | ...   |

| 2138 | 79         | 279         | 0.002   | 0.004      | 0.918 |
| 2139 | 79         | 273         | 0.002   | 0.004      | 0.229 |
| 2140 | 79         | 125         | 0.002   | 0.004      | 0.167 |
| 2141 | 79         | 4           | 0.002   | 0.004      | 0.122 |
| 2142 | 79         | 269         | 0.002   | 0.004      | 0.097 |

5.3.1 2-Cycles

From Table 5.2 we generate a new table of rules by only keeping pairs of rules that are contradictory. Hence, we only keep pairs of rules for which the antecedent and consequence have switched places. By doing this, we obtain a list of 126 rules (63 pairs of rules), all being part of 2-cycles. The result can be seen in Table 5.3 and an illustration of what causes a 2-cycle can be seen in Figure 5.4.

Table 5.3: Top five and bottom five rules that make up 2-cycles.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Antecedent</th>
<th>Consequence</th>
<th>Support</th>
<th>Confidence</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>1</td>
<td>0.021</td>
<td>0.556</td>
<td>1.852</td>
</tr>
<tr>
<td>2</td>
<td>269</td>
<td>79</td>
<td>0.021</td>
<td>0.526</td>
<td>0.966</td>
</tr>
<tr>
<td>3</td>
<td>46</td>
<td>1</td>
<td>0.015</td>
<td>0.583</td>
<td>1.944</td>
</tr>
<tr>
<td>4</td>
<td>66</td>
<td>273</td>
<td>0.015</td>
<td>0.135</td>
<td>7.909</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>271</td>
<td>0.015</td>
<td>0.050</td>
<td>2.333</td>
</tr>
</tbody>
</table>

| ...  | ...        | ...         | ...     | ...        | ...   |

| 122  | 1          | 261         | 0.002   | 0.007      | 0.667 |
| 123  | 1          | 324         | 0.002   | 0.007      | 0.417 |
| 124  | 1          | 17          | 0.002   | 0.007      | 0.185 |
| 125  | 79         | 125         | 0.002   | 0.004      | 0.167 |
| 126  | 79         | 269         | 0.002   | 0.004      | 0.097 |
Results and Discussion

(a) Two contradictory rules. (b) The resulting 2-cycle created when creating a network of rules.

Figure 5.4: Example of a 2-cycle.

For example, by looking at rules 1 and 124 in Table 5.3, we can see that the support and confidence for rule 1 are greater than for rule 124. By multiplying the support of both rules by the number of FMAN-pages in the data, we get the number of occurrences for both rules. This gives us that rule 1 occurs 10 times while rule 124 occurs only once. It could be the case that for the FMAN-page, which supports rule 124, action 1 must be before action 17 for a legitimate reason. However, since the positioning of the actions is swapped in 10 out of the 11 cases, it is more likely that rule 124 only exists because of human inconsistency while creating the FMAN-pages.

Deleting 2-Cycles

Since cycles proved to be a big issue, an effort was made to delete as many cycles as possible, cleaning up the data. When asking supervisors at Saab to look at the 2-cycles, it was clear that the vast majority existed solely because engineers have not had any software supporting them in being consistent with the arrangement of actions when creating FMAN-pages. By providing them a list of cycles together with information stating between which FMAN-pages these cycles occur, they immediately began deleting cycles from the data by rearranging actions in certain FMAN-pages to cancel out these cycles. Since rearranging actions sometimes yields new cycles, some iterations of this procedure were needed. After three iterations, only eight 2-cycled remained. This is a considerable improvement compared to the 67 2-cycles in the original data. The remaining cycles could not be removed due to some orders of actions essentially having to be contradictory for each FMAN-page to contain the correct information given a specific fault. Hence, deleting the remaining cycles by rearranging actions is likely possible, but it would not benefit flight safety.
5.3.2 k-Cycles

Unlike 2-cycles, it is more difficult to find and get rid of k-cycles. This is partly because k-cycles usually span several FMAN-pages and partly due to the cycles including more actions. An illustration of what causes a k-cycle can be seen in Figure 5.5. An essential aspect of k-cycles is that at least three actions must be involved in the cycle. 50 k-cycles were identified with the built-in function `chordless_cycles` from NetworkX\textsuperscript{4}. When running the `chordless_cycles`–function 2-cycles will also be detected. Therefore, removing all 2-cycles from the result is essential to only obtain k-cycles. Extensive work is needed to solve k-cycles and this was not a priority for this project. This will be further discussed in the final chapter of this thesis.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{k-cycle_example.png}
\caption{(a) Three rules. No pair of rules being contradictory. (b) The resulting k-cycle when creating a network of the rules.}
\end{figure}

5.4 Sorting with Sequential Rules

The main problem with relying on a set of rules for sorting is that there are not rules for all pairs of actions. Hence, we might end up in a situation where we have a set of actions to sort but no rules.

On the other hand, too many rules make it difficult for sorting algorithms to converge because of cycles. Trying to sort actions within the same cycle is the same as answering the question: “Where does a cycle begin?”. Since rules have certain support, confidence, and lift, it is sometimes easy to determine which rules to use in the case of a contradiction. In this thesis, the statistical approach is taken, therefore always choosing the rule with the highest lift, and if lifts are equal, choosing the one with the highest confidence. Remember that a high lift value indicates that the rule is significant and that the presence of the consequence

\textsuperscript{4}https://networkx.org/
Results and Discussion

strongly depends on the presence of the antecedent. Using this method of prioritizing rules always yields the most accurate results according to the data. Note, however, that flight safety and pilot preference are not considered when using the statistical approach.

5.5 Results from Total-FMAN

Using the GUI described in Section 4.6 for all three networks while activating the same alerts, given by Figure 5.6, we obtain the results in Figure 5.7. Remember that actions with blue font color originates from the most recently activated alert, and are the ones most recently added to the Total-FMAN. We can tell by the figure that not all actions with blue font color are inserted at the same index in the list of actions for the three networks. More precisely, the action texts "At CAS 270 kt / AoA 8: Extend gear" and "If 3305 GEAR HI SPEED also set: - EMGY gear extension when CAS < 215 kt" are placed differently depending on which network is in use. We note that they are placed at rows 12 and 13 for Network A, rows 16 and 17 for Network B, and at rows 19 and 20 for Network C. The difference in placement of these actions has to do with the communities to which they are assigned in the three networks. Since the same rules are used for sorting actions within communities in Figure 5.7, the only way of having actions not being placed on the same rows for each of the results is if their community belonging varies. We know for a fact, according to Figures 5.1 – 5.3, that the communities varies a lot for the three networks. Hence, making this result reasonable.

This being said, one of the three lists should be arguably more reasonable than the others. By talking to experienced engineers we learned that both of the texts should be labeled as Landing, hence at least being placed after "Choose most suitable runway" which belongs to the phase Prepare to Land. We can tell by Figure 5.7(a) that this is not fulfilled for Network A.

Furthermore, by analyzing the meaning of "Use rudder and wheel brakes to steer", it is reasonable to assume that this text should be at the very end of the list since, for the wheel brakes to make any difference, the plane should already have touched the ground. This implies that any guidelines regarding CAS should be given before the text regarding steering while back on the ground. Only the result from Network B given by Figure 5.7(b) fulfills the logical reasoning in this section.

Additionally, we take a thorough look at where actions from alert 3206 in Figure 5.6(a) are placed in the Total-FMAN by looking at any of the sub-figures to Figure 5.7. We notice that Complete mission at pilot discretion is removed, and this is due to the presence of Land ASAP. Although creating a filter to remove redundant actions is outside the scope for this thesis, we chose to implement solutions to conflicts between actions already identified by Linderstam and Nilsson in [6]. In their thesis, they learned that in a situation where both Land ASAP and Complete mission at pilot discretion are present in the Total-FMAN, only Land ASAP should be displayed. Furthermore, 3206 has three actions dealing with BEOS (Back-up Emergency Oxygen System). Here, the order must not be compromised for the
actions to make sense. Since the order is known to be important, each occurrence of these actions in an FMAN-page have the same ordering. This results in strong rules being mined between the actions, making sure that the order is not compromised when merged into the Total-FMAN as long as the label propagation assigns these to the same community. Notice also that $Alt < 20000 \text{ ft}$ has the same relative position to the BEOS-actions in both alert 3206 and the Total-FMAN.

It is challenging to objectively evaluate the performance of the system. However, the algorithms seem to generate a Total-FMAN that at least preserve many important aspects of the individual FMAN-pages.
Results and Discussion

(a) Alert 3206.

(b) Alert 3520.

(c) Alert 6201.

(d) Alert 6361.

Figure 5.6: Default alerts.
5.5 Results from Total-FMAN

(a) Results using Network A.

(b) Results using network Network B.

(c) Results using Network C.

Figure 5.7: Results for the different networks. All alerts in Figure 5.6 activated.
This chapter answers the research questions posed in Section 1.2.1 by concluding the results presented in Chapter 5. Additionally, it brings attention to the potential directions of interest for future work.

6.1 Conclusions

This section provides conclusions from Section 5 about the research questions.

6.1.1 Label Propagation

Although LPAs are convenient for speeding up the process of labeling nodes and getting a first draft of a prototype, since the Total-FMAN is supposed to be a DSS for pilots in critical fault situations, it must be flawless. Hence, we conclude that LPAs are suitable for prototyping but not suitable for creating a reliable DSS for pilots. For the example in Section 5.5 we conclude that using Network B provides the best result. It is important to note that this is not a general conclusion, but only our conclusion for this specific example. However, the example shows what effect changes in community partitions has on the results.

Suppose Saab wants to proceed with creating a Total-FMAN of the kind we propose. In that case, we suggest that Saab put time and energy into making sure that by carefully and thoughtfully labeling nodes, all nodes are assigned the correct label.

At the end of the project, we had the pleasure of joining our supervisor and experienced engineers on a meeting with two aviation security officers from the Swedish armed forces. The agenda was to discuss the future of FMAN-pages and
propose the Total-FMAN for the officers. It is important to note that the Total-FMAN has been developed with FMAN-pages from JAS 39 Gripen E, while the officers only have experience with the FMAN-pages given by JAS 39 Gripen C/D. However, without any prior knowledge of our work and especially the concept of assigning labels to actions to divide pages into phases, they had prepared a presentation themselves with a more or less identical idea with their examples from FMAN-pages for JAS 39 Gripen C. Learning that the customers wish for something that is already under development, without talking to them beforehand, is very positive for the continuation of the project with creating a Total-FMAN. Even though LPAs might not be the optimal way moving forward for safety and control reasons, the knowledge that it might be worth putting hours into manually getting a correct set of labeled nodes is very motivational.

6.1.2 Sequential Rule Mining

Using SRM to mine rules from FMAN-pages is proven to be a good way of understanding the general structure of an FMAN-page. Without knowledge about systems and understanding of abbreviations used in action texts, one can use these rules to help schedule actions in complex fault situations.

Support System

While a well-functioning Total-FMAN is supposed to support pilots during flight, our proposed approach is currently better suited for the engineers working with FMAN-pages on a daily basis. One could argue that the mined sequential rules could substitute for experience gathered from the engineers while working on creating FMAN-pages. Given a set of actions and, it is difficult for any inexperienced person to determine how to schedule the actions to make up a reasonable FMAN. Although, with a set of rules to rely on, this task would be much easier for anyone to accomplish. Rules will also help with consistency in scheduling actions for which the order is not that important. As pilots need to get as little variation in the FMAN-pages as possible, rules are a good support tool for the engineers creating FMAN-pages to ensure this variance is as small as possible.

6.2 Future Work

This was the second thesis at Saab to investigate the possibilities of creating a Total-FMAN. In our work, we focused on a methodology that does not require any prior knowledge about the systems in JAS 39 Gripen E but is solely reliant on information mined from the already implemented systems. We can conclude that cycles restrict the system’s performance and will suggest alternative ways to deal with cycles for future work while also sharing gathered thoughts and ideas regarding the continuation of the Total-FMAN project gathered during the thesis. The problem of dealing with cycles is as interesting as it is difficult, and if Saab wants to continue with our suggested approach in their future work to create a
Conclusions and Future Work

Total-FMAN, some systematic handling of the cycles will be needed. Below, we will list our suggestions on how to approach dealing with the cycles.

1. Investigate whether or not it is possible to determine if there is a way to rearrange the actions (where needed) to get a network representation completely free from cycles. This is a well-defined problem and will take many hours to do by hand - if it is even possible. Therefore, it is important to investigate algorithms for determining if it is possible to create a directed acyclic graph (DAG) by rearranging the actions within the FMAN-pages without deleting or adding any actions.

2. If the conclusion is that it is impossible to make the given graph a DAG solely by rearranging actions, algorithms such as the Feedback Arc Set (FAS) could be of interest to investigate as an alternative way to make the graph a DAG. This would not be optimal for flight safety since this would force links to be deleted, but it could, similar to the conclusion of this thesis, help the engineers create a network of actions free from cycles in the future. A problem with using FAS for the given problem is the time complexity. Although, we would still recommend looking into versions of this algorithm.

3. Another way of approaching the issue of cycles in the future could be to create the network in another way. In our work, we represent each action as a node, adding edges as new relations between actions are found within FMAN-pages. Another way of creating the network could be to follow the same procedure, but when a cycle is about to be created, instead of adding a link to a previously created node, we create a new node and create a link to it. This way, we end up with a network containing more nodes but being free from cycles. We will also have multiple nodes representing the same action. This could be a good thing for capturing where actions must be taken in different order depending on the situation.

4. A final suggestion that gives the engineers complete control of the outcome and takes their experience and expertise into account is to rewrite the FMAN-pages. More precisely, we would suggest creating a new set of FMAN-pages by adding the old FMAN-pages to this new set of pages one by one. For each of the added FMAN-pages running a cycle-detection algorithm, one can be certain when a cycle is introduced. When a cycle is introduced, it should be easy to review the currently added FMAN-pages and rearrange actions so that any cycle is no longer introduced in that step. One can then add further FMAN-pages until a new cycle is detected. With this methodology, the problem would feel more approachable since one would only have to pay respect to one cycle at a time, with the only goal being to try and fix it without introducing new cycles. One could try and brute force this approach to minimize the number of cycles in the network instead of getting a network completely free from cycles. Again – in some cases, it might be absolutely critical that actions are taken in an order contradictory to another FMAN-page. Hence, a system without cycles might not be good from a flight safety perspective, even if we could solve it arithmetically.
Our work focuses on scheduling all actions from the active alerts to the pilot without filtering away redundant actions. Hence, in the case of a complex fault situation involving a large number of actions, scheduling the actions is a weak consolation if we cannot remove redundant actions since pilots will still suffer from information overload. With the help of SRM, we can determine which actions are likely to be presented as pairs of actions, but we lack information on what actions act as each other’s complement. Therefore, investigating ways of implementing a filter to remove redundant actions in complex fault situations is interesting. Simply put, regardless of how well a sorting algorithm performs for the given system, the pilots will eventually suffer from information overload if too many alerts become active simultaneously.

In this thesis, we have proposed a method of mining sequential rules that has the restriction of only mining rules between pairs of actions on the same FMAN-page. By instead mining sequential rules that extend over multiple FMAN-pages, one could achieve even more exciting results and learn more from the data. Figure 6.1 and Table 6.1 give an example of how this could work. In this example, we mine three instead of two rules, as we would have done in this thesis.

![Figure 6.1: Example of two FMAN-pages with two actions each.](image)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Antecedent</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

Table 6.1: Table of generated rules.


