FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Comprehensive Analysis of Alarm Root Causes in an Optical Fiber Network

Lucas Leite Baptista

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Ana Cristina Costa Aguiar

October 23, 2023

Resumo

A análise da causa raiz é um processo crucial para identificar e abordar as razões subjacentes às falhas em sistemas complexos, como redes de fibra ótica e operações industriais de grande escala. As metodologias tradicionais de análise da causa raiz frequentemente enfrentam dificuldades para lidar com a grande quantidade de dados e complexidades presentes nesses sistemas, levando a análises demoradas e subjetivas. Para enfrentar esses desafios, esta pesquisa investiga a aplicação de técnicas de mineração de dados com o objetivo de agilizar e automatizar a análise da causa raiz.

O principal objetivo desta pesquisa é desenvolver um modelo baseado em mineração de dados que possa identificar e agrupar automaticamente alarmes, facilitando uma análise mais eficaz. Por meio de algoritmos avançados, o modelo automatiza a análise de dados de alarme, detecta padrões recorrentes e permite aos operadores de rede prioritizar questões críticas para atenção imediata. Técnicas de alinhamento de sequências e os cálculos de similaridade foram implementadas, permitindo obter perceções valiosas sobre as relações entre os alarmes e suas causas raiz.

O modelo proposto foi rigorosamente validado usando um conjunto abrangente de dados que abrange 7 dias de informações de alarme de uma rede real de fibra ótica, contendo mais de 5,5 milhões de alarmes. Ao implementar um algoritmo eficiente de deteção de *chattering*, aproximadamente 49% dos alarmes redundantes foram removidos com sucesso, resultando em um conjunto de dados mais refinado que melhorou significativamente a qualidade geral da análise.

Para identificar e medir com precisão as similaridades entre os alarmes, um novo algoritmo baseado no algoritmo de Smith-Waterman foi desenvolvido. Essa técnica avançada de alinhamento de sequências permitiu comparações e alinhamentos precisos entre os alarmes, fornecendo *insights* valiosos sobre as possíveis causas raiz e as relações entre diferentes alarmes. A aplicação de agrupamento hierárquico, com base na análise de similaridade, aprimorou ainda mais as capacidades do modelo, gerando grupos mais coesos e distintos de alarmes relacionados.

Esta pesquisa demonstra o potencial das técnicas de mineração de dados para revolucionar a Análise de Causa Raiz em sistemas complexos. Ao automatizar a identificação e análise de alarmes, as organizações podem alcançar garantia constante de serviço, resolução rápida de problemas e manter uma vantagem competitiva no mundo atual impulsionado pela tecnologia. O modelo proposto oferece uma abordagem proativa para a análise da causa raiz, tornando os sistemas mais resilientes e adaptáveis. Testes e validações extensivos em diversos conjuntos de dados e ambientes de rede garantirão a ampla aplicabilidade e eficácia do modelo. À medida que a tecnologia de mineração de dados evolui, o modelo continuará a se aprimorar, oferecendo às organizações uma ferramenta poderosa para aperfeiçoar suas capacidades de análise da causa raiz e garantir a confiabilidade e estabilidade de sistemas complexos.

Abstract

Root cause analysis is a crucial process for identifying and addressing the underlying reasons behind system failures in complex environments, such as fiber optic networks and large-scale industrial operations. Traditional root cause analysis methodologies often struggle to handle the vast amount of data and complexities present in these systems, leading to time-consuming and subjective analysis. To overcome these challenges, this research explores the application of data mining techniques to expedite and automate root cause analysis.

The main objective of this research is to develop a data mining-based model that can automatically identify and cluster alarm floods, enabling easier and more effective analysis. Leveraging advanced algorithms, the model automates the analysis of alarm data, detects recurrent patterns, and allows network operators to prioritize critical issues for prompt attention. Sequence alignment techniques and similarity calculations were implemented, allowing us to obtain valuable insights into the relationships between alarms and their root causes.

The proposed model has been rigorously validated using a comprehensive dataset that spans over 7 days of alarm data from a real-world fiber optic network, encompassing more than 5.5 million alarms. By implementing an efficient chattering detection algorithm, approximately 49% of redundant alarms were successfully removed, resulting in a streamlined and refined dataset that significantly improved the overall analysis quality.

To accurately identify and measure similarities between alarm floods, a novel algorithm based on the Smith-Waterman algorithm was developed. This advanced sequence alignment technique enabled precise comparison and alignment of alarm sequences, providing valuable insights into potential root causes and relationships between different alarms. The application of hierarchical clustering, based on similarity analysis, further enhanced the model's capabilities, generating more cohesive and distinct clusters of related alarms.

This research demonstrates the potential of data mining techniques to revolutionize Root Cause Analysis in complex systems. By automating the identification and analysis of alarm floods, organizations can achieve constant service assurance, swift issue resolution, and maintain a competitive edge in today's technology-driven world. The proposed model offers a proactive approach to root cause analysis, making systems more resilient and adaptive. Extensive testing and validation on diverse datasets and network environments will ensure the model's broader applicability and effectiveness. As data mining technology evolves, the model will continue to evolve, offering organizations a powerful tool to enhance their root cause analysis capabilities and ensure the reliability and stability of complex systems.

Acknowledgments

I want to thank everyone in general, including my friends and family, for the support they have provided throughout my academic journey. Without a doubt, I could only reach this moment thanks to them.

Special thanks to my colleagues at work, who have always been helpful in assisting with the development of this work.

I would also like to express my sincere gratitude to my advisor, Professor Ana Aguiar, for her dedicated guidance throughout the completion of this dissertation. Her mentorship and expertise were crucial to the success of this project.

To everyone, thank you so much!

Lucas Leite Baptista

"It is part of my thesis that all our knowledge grows only through the correcting of our mistakes."

Karl Popper

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AF	Alarm Flood BLAST
Basic Local Alignment Search Tool	
DTW	Dynamic Time Warping
FPM	Frequent Pattern Mining
FTTB	Fiber To The Building
FTTC	Fiber To The Cabinet
FTTH	Fiber To The Home
FTTN	Fiber To The Node
OLT	Optical Line Terminal
ONT	Optical Network Terminal
ONU	Optical Network Unit
PON	Passive Optical Network
RCA	Root Cause Analysis
SA	Sequence Alignment

Chapter 1

Introduction

This document is dedicated to Root Cause Analysis (RCA) in the context of a fiber optic network. We explore the field of data mining using sequence similarity analysis methods to gain a deeper understanding of the alarm data generated by the network.

In this introductory chapter, we provide a concise overview of the theme, emphasizing the importance of RCA in a fiber optic network scenario. We aim to leverage data mining techniques, particularly sequence similarity analysis, to enhance the RCA process and ensure the continuous and reliable operation of the network.

Furthermore, we present the structure of this document. After this introduction, we proceed with a detailed explanation of the thesis chapters, outlining the topics covered in each chapter. This structure is designed to provide a comprehensive exploration of our proposed methodology and its application to the fiber optic network's alarm data analysis.

1.1 Context

In the contemporary landscape, the operation of various industries, such as telecommunications and transportation, heavily relies on complex systems and networks [1, 2]. These systems are indispensable for ensuring uninterrupted operations. However, the growing complexity of these systems elevates the risk of potential failures and service disruptions [3]. Hence, the identification of the root causes of these failures is a pressing concern to maintain consistent and dependable services [4, 5].

1.2 Motivation

The increasing intricacy of these systems presents a significant challenge, particularly in the context of uncovering the root causes of these failures. This is crucial for preserving the continuity of reliable services [4, 5]. Traditional methods of RCA typically involve manual investigation and expert analysis, which can be both time-consuming and resource-intensive, often leading to subjective outcomes [6].

1.3 Root Cause Analysis Background

RCA is an established methodology used for identifying the fundamental reasons behind specific problems or failures in complex systems. Traditionally, this process involves manual investigations and expert analyses to discern the root causes of issues. While RCA has been indispensable in understanding and resolving problems, it can be time-consuming, resource-intensive, and prone to subjectivity, particularly in complex systems [6]. As technology and systems have evolved, the intricacy of these systems has increased significantly, demanding more efficient and precise RCA methods.

1.4 Problem Formulation

In the context of this thesis, we recognize that traditional RCA methodologies may not be sufficiently effective in complex systems, such as fiber optic networks or large-scale industrial operations. The vast number of variables and potential interactions present significant challenges in terms of data handling, pattern detection, and insight extraction. This can hinder the ability to promptly and accurately identify the root causes of failures, a crucial consideration in scenarios where continuous service assurance is vital.

1.5 Goals

In light of these challenges, the primary aim of this thesis is to provide innovative solutions. We seek to explore and implement data mining techniques to expedite and automate RCA in complex systems. Our specific objectives revolve around the development of cutting-edge algorithms and methodologies that can automate the identification of Alarm Floods (AFs). Alarm Floods, in this context, refer to sequences or clusters of similar alarms occurring within a specific timeframe. We aim to streamline the characterization of these AF clusters comprising similar alarms and effectively pinpoint the most representative sequences within each cluster. These efforts are aimed at rendering the RCA process more efficient and reliable, ensuring the continuity of services in our technology-driven world.

1.6 Structure

In this chapter, we provide an overview of the theme of this thesis and set the context for the research. We introduce the concept of RCA and highlight its significance in complex scenarios to ensure the constant service and stability of systems. We discuss the challenges faced while applying traditional RCA methods in the context of large-scale and dynamic systems. Additionally, we emphasize the importance of data mining techniques in enhancing RCA processes. Moreover, we briefly touch upon the significance of sequence similarity analysis in analyzing big data systems.

In Chapter 2, we delve into the state of the art in the field of RCA, emphasizing its foundational concepts and challenges. We explore the use of data mining approaches, specifically focusing on sequence similarity analysis, as a means to improve RCA in big data systems. A comprehensive study of the Smith-Waterman algorithm, which is a key element of our proposed methodology, is presented, highlighting its advantages and applications in SA and similarity analysis. Furthermore, we conduct a literature review to identify existing works with similar objectives and analyze their methodologies to derive insights and potential inspirations for our research.

Chapter 3 is dedicated to presenting our proposed methodology for RCA using pattern matching techniques. We begin by discussing the pre-processing steps involved in handling alarm data to prepare it for further analysis. Next, we introduce the SA algorithm, leveraging the Smith-Waterman method, as a powerful tool for identifying similar patterns in alarm sequences. Subsequently, we explore post-processing techniques, including clustering, to group similar AFs and identify distinct clusters. Additionally, we discuss the process of finding the archetype AF within each cluster, which serves as a representative sequence.

In Chapter 4, we present the results of applying our proposed RCA methodology to real-world datasets. We define relevant evaluation metrics and describe the experimental setup to measure the performance of the approach. We analyze the obtained results and discuss the effectiveness of our methodology in comparison to traditional RCA approaches. We also interpret the implications of our findings and identify any limitations encountered during the experimentation. Moreover, we discuss potential avenues for future enhancements and improvements.

The final chapter of this thesis summarizes the main contributions and accomplishments of our research. We reiterate the significance of data mining techniques, particularly sequence similarity analysis, in enhancing RCA for big data systems. We discuss the broader implications of our work and its potential applications in real-world scenarios. We provide insights into practical applications and offer suggestions for further research and development. Finally, we conclude by summarizing the key insights and highlighting the significance of our work in the context of complex and dynamic systems.

Chapter 2

Background and State of the art

In this chapter, our objective is twofold. Firstly, we analyze state-of-the-art techniques employed in analogous RCA problems. This examination provides valuable insights into problem-solving approaches and highlights existing solutions implemented in similar contexts. By addressing these aspects, we lay the groundwork for the project, ensuring that subsequent steps are built upon a solid understanding of the subject matter and the existing body of research in the field.

Secondly, we aim to establish a firm foundation for both the theoretical and practical facets of this project. To achieve this, we explore the fundamental concepts crucial for understanding and implementing the thesis. This exploration ensures a comprehensive grasp of the key concepts necessary for project development.

2.1 Related work on Root Cause Analysis

In the domain of RCA and fault management for complex systems, two key techniques, namely Sequence alignment (SA) and Frequent Pattern Mining (FPM), have emerged as powerful tools for analyzing large-scale alarm datasets. These methods play a vital role in detecting patterns, similarities, and correlations within alarm sequences, providing valuable insights into the behavior of systems and potential underlying issues.

In this related work section, we delve into the applications and advancements of SA and FPM in the context of AF similarity analysis.

2.1.1 Sequence alignment

SA is a fundamental technique in various fields, including bioinformatics, natural language processing, and data mining [7]. It involves the comparison of two or more sequences to identify regions of similarity or homology [4, 5, 6, 8, 9]. In the context of AF analysis and RCA in fiber optic networks, SA methods play a crucial role in identifying similar alarm patterns and understanding the relationships between alarms over time.

In the literature, SA methods have been extensively applied to AF similarity analysis [4, 5, 6, 7, 8, 9, 10]. These methods aim to match and align alarm sequences to reveal underlying patterns

and correlations. One of the prominent SA algorithms used in this domain is the Dynamic Time Warping (DTW) algorithm [11, 12, 13]. DTW allows for flexible alignment of sequences, accommodating time shifts and distortions, making it suitable for time-series data like alarm sequences.

Another well-known SA algorithm applied to AF analysis is the Smith-Waterman algorithm, which originates from bioinformatics [8, 14] but has been adapted to handle the temporal dimension of AFs [15, 16]. The Smith-Waterman algorithm calculates the similarity score between two alarm sequences by taking into account the order of alarms and their temporal relationships [5, 7, 14, 15, 16, 17]. It has become a benchmark in the literature due to its effectiveness in identifying similar alarm patterns and subsequences.

In addition to DTW and Smith-Waterman, other SA techniques like the Basic Local Alignment Search Tool (BLAST) [18] algorithm and match-based accelerated alignment algorithms [4, 9, 17] have also been utilized for AF similarity analysis. These methods further contribute to the identification of recurrent AFs and the detection of underlying patterns in large-scale alarm datasets.

SA methods have proved beneficial in identifying and removing redundancy in alarm data, allowing for more efficient alarm rationalization [19, 20, 21]. By aligning alarm sequences, operators can gain insights into the temporal patterns of alarms and better understand the dynamics of the network, aiding in the identification of root causes and troubleshooting of network issues.

Moreover, the application of SA in online operator support has shown promise [7, 19]. Techniques such as AF template extraction and online AF classification leverage SA to match ongoing AFs to historical patterns, assisting operators in real-time decision-making and problem resolution.

Overall, SA methods have demonstrated their utility in analyzing AFs and enhancing RCA in fiber optic networks. By aligning and comparing alarm sequences, these methods enable the extraction of valuable insights from large volumes of alarm data, leading to more efficient network maintenance and constant service assurance in complex fiber optic systems. As the volume of data in fiber optic networks continues to grow, the application of SA techniques will likely play an even more significant role in improving network performance and reliability.

2.1.2 Frequent Pattern Mining

FPM is a powerful data mining technique used to identify recurring patterns or itemsets that frequently co-occur in a given dataset [10, 18, 22, 23]. The concept of frequent patterns lies at the core of various data-driven applications, including market basket analysis, web usage mining, bioinformatics, and AF analysis in complex systems.

In the context of AF analysis, FPM methods have been employed to discover the most frequent combinations of alarms or alarm patterns within the alarm sequences. These methods aim to unveil the common alarm patterns that occur frequently in the network, providing valuable insights into the behavior and dynamics of the alarm system.

One of the well-known FPM algorithms is the Apriori [10, 24] algorithm, which is widely used for association rule mining in market basket analysis. In the context of AFs, Apriori identifies frequent itemsets, i.e., sets of alarms that often co-occur together. These itemsets can represent

recurrent alarm patterns that might be indicative of specific network issues or underlying system behaviors.

Another popular FPM approach is the FP-Growth algorithm [22, 23, 25, 26], which efficiently mines frequent patterns by building a compact data structure called the FP-tree. This algorithm is particularly suitable for large-scale alarm datasets, as it optimizes memory usage and reduces the number of database scans, thereby improving performance.

By discovering frequent alarm patterns, network operators can identify redundant or repetitive alarms, leading to a more concise and informative alarm system. This can significantly reduce alarm fatigue for operators, as they can focus on critical alarms and promptly address network anomalies.

Additionally, FPM has been applied to dynamic alarm suppression, a technique used to temporarily hide non-relevant alarms during AFs [6, 10, 18, 22, 23, 24]. By identifying frequently occurring alarms that are not directly related to the root cause, FPM helps optimize alarm presentation and improve operator response times during AFs.

Furthermore, FPM techniques have proven useful in discovering meaningful associations between alarms, providing valuable insights into network dependencies and relationships. This information can aid in the detection of cascading failures, helping network operators take proactive measures to prevent potential service disruptions.

In summary, FPM is a versatile and effective data mining technique that has found applications in diverse domains, including AF analysis in fiber optic networks. By identifying recurring alarm patterns, FPM contributes to alarm rationalization, dynamic alarm suppression, and the overall enhancement of RCA. As data volumes continue to grow in complex systems, FPM methods will remain instrumental in extracting valuable knowledge from alarm data, enabling constant service assurance and optimal network management.

2.1.3 Literature overview

SA algorithms excel in performing a fine-grained analysis of individual alarms and their temporal relationships. This level of detail is essential for gaining an accurate understanding of the precise sequence of events within AFs. FPM may not capture such nuances as effectively.

AFs often exhibit alarms in specific temporal sequences. SA algorithms excels in considering the temporal relationships between alarms. This temporal awareness is crucial for detecting patterns and similarities evolving over time, while FPM might oversimplify the analysis. The alarm sequences may also vary in length, making them less amenable to FPM. SA algorithms address this variability by accommodating gaps in sequences, aligning similar segments, and providing a scoring mechanism for the alignments.

SA algorithms offer the flexibility to define customized scoring criteria. This adaptability is vital for assigning varying weights to alarms or specific alarm components, reflecting their significance in the analysis. SA algorithms can also adapt to different gap penalties, which is valuable for handling AFs with varying temporal gaps between alarms.

FPM can be sensitive to noise and variations in alarm reporting times, potentially leading to less effective analysis. In contrast, SA algorithms are more resilient to noise, ensuring that even subtle patterns are captured accurately.

SA algorithms have a well-established track record in diverse fields like bioinformatics, showcasing their reliability and adaptability in identifying patterns in sequences.

In summary, SA algorithms offer distinct advantages in terms of granularity, temporal awareness, adaptability, and robustness, making them a robust choice for AF analysis.

2.2 Background concepts

This section delves into the theoretical and practical aspects of the project. To do so, we explore fundamental concepts that are essential for both understanding and implementing the thesis. This exploration is aimed at fostering a comprehensive understanding of the key concepts required for project development.

2.2.1 Fiber optic networks

Fiber optic networks are a type of telecommunication network that transmit data using pulses of light through optical fibers, which are thin strands of glass or plastic [27]. These networks are known for their high capacity, fast data transmission speeds, and long-distance capabilities.

In a fiber optic network, data is encoded into light signals, which are then transmitted through the optical fibers. The light signals travel through the fibers by repeatedly bouncing off the inner walls through a phenomenon called total internal reflection. This allows the signals to propagate over long distances without significant loss or degradation [27, 28].

Compared to traditional copper-based networks, fiber optic networks offer several advantages. Firstly, they have much higher bandwidth, enabling them to transmit large amounts of data at incredibly fast speeds. This makes them ideal for applications that require high data transfer rates, such as video streaming, cloud computing, and telecommunication services [14].

Additionally, fiber optic networks are less susceptible to electromagnetic interference and signal loss. They are immune to electromagnetic interference from nearby electrical cables or devices, ensuring reliable data transmission. The optical fibers also have lower signal attenuation, allowing data to be transmitted over longer distances without requiring frequent signal regeneration[28].

Fiber optic networks are widely used in various sectors, including telecommunications, internet service providers, data centers, and enterprise networks. They form the backbone of modern communication infrastructure, facilitating the efficient and reliable transmission of data across long distances.

Several types of fiber optic technologies are used in telecommunications, each designed to meet specific requirements and address various deployment scenarios. Some of the most common are Fiber To The Home (FTTH), Fiber To The Building (FTTB), Fiber To The Cabinet (FTTC), and Fiber To The Node (FTTN) [27].

The scenario presented in this dissertation follows an FTTH topology, which is a popular and efficient approach for delivering high-speed broadband services to end-users. This technology involves extending the fiber optic network directly to individual homes or buildings. FTTH offers the highest bandwidth and is capable of delivering gigabit or even multi-gigabit speeds to residential users. To ensure the successful deployment and operation of an FTTH network, several fundamental equipment components are utilized [27, 28]. Here are some of the key equipment used in an FTTH network:

1. **Optical Line Terminal**: The Optical Line Terminal (OLT) is a central device located in the service provider's facility or central office. It serves as the interface between the service provider's network and the subscriber's fiber optic connections. The OLT aggregates and manages multiple subscriber connections and handles the distribution of data, voice, and video services.

2. **Optical Network Unit/Terminal**: The Optical Network Unit (ONU) or Optical Network Terminal (ONT) is installed at the subscriber's premises. It is the endpoint of the FTTH network and is responsible for receiving and transmitting data over the fiber optic connection. The ONU/ONT converts the optical signal into electrical signals compatible with the customer's devices, such as computers, telephones, and televisions.

3. **Splitters and Couplers**: Splitters and couplers are passive devices used for dividing or combining optical signals in an FTTH network. Splitters are used to split the incoming fiber optic signal into multiple output ports, allowing multiple subscribers to share the same fiber connection. Couplers, on the other hand, combine or couple optical signals from different fibers into a single fiber.

2.2.2 Alarm Manager

The increasing complexity of network topologies has made the use of RCA tools crucial for ensuring their proper operation. However, manually defining rules for alarm aggregation and correction in these tools requires network knowledge and has become increasingly complex due to the growing number of network devices. To address this challenge, this project aims to leverage data mining techniques to extract new insights from the network, thereby assisting and enhancing the rule creation process.

In this case study, the RCA tool employed is Alarm Manager, developed by Altice Labs. Alarm Manager plays a vital role in generating and storing all alarm instances associated with the failure of customers' equipment. Before delving into further project details, it is important to gain a comprehensive understanding of Alarm Manager's functionality, workflow, and the data it retains for each alarm instance.

2.2.2.1 Description

Alarm management involves the process of monitoring, analyzing, and responding to alarms generated by various systems or processes. The overall goal of alarm management is to ensure that alarms are effective, reliable, and help operators take appropriate actions in response to abnormal conditions or events [1, 3, 29, 30].

AGORA, the Alarm Manager system developed by Altice Labs, is a comprehensive Fault Management tool designed to handle various aspects of alarm reception, storage, treatment, monitoring, and correlation. It serves as the central hub for managing states, severity levels, and other standardized alarm parameters. Additionally, Alarm Manager enables the definition of alarm cycles within the system and oversees the processing infrastructure along with multiple collection channels.

Drawing upon Altice Labs' expertise and insights from the telecommunications industry, Alarm Manager ensures efficient fault monitoring, problem detection, and the initiation of fault tickets. By streamlining processes, it enhances operational efficiency, allowing for prompt adjustments to align with business needs. Moreover, it offers flexibility in managing new platforms and network elements, empowering organizations to adapt swiftly to evolving requirements [3]. Through its capabilities, Alarm Manager delivers operational gains and facilitates agile business operations.

2.2.2.2 Functioning

The alarms are generated by monitoring systems that continuously monitor the status of devices, processes, or networks. These systems are designed to detect abnormal conditions, equipment failures, safety hazards, or other critical events. When a predefined threshold or condition is met, an alarm is triggered. Alarms are often assigned different levels of priority based on their significance and potential impact [2]. Prioritization helps operators focus on the most critical alarms and ensures that resources are allocated effectively. Priority levels may be determined based on factors such as safety implications, operational impact, or regulatory requirements [1].

Once the alarm is observed, it needs to be promptly communicated to the relevant operators or stakeholders. This can be done through various means such as visual indicators, audible alerts, or a message notification. It should be clear, and concise, and provide relevant information about the alarm, including its severity, location, and description. When operators receive an alarm notification, they need to acknowledge it to indicate that they are aware of the alarm. This helps in tracking the status of alarms and prevents them from being overlooked or ignored. Alarm acknowledgment can be done manually by the operator or automated within the alarm management system. Once an alarm is acknowledged, the operator initiates an analysis of the situation. They investigate the root cause of the alarm, assess its impact, and determine the appropriate response or action. This may involve referring to standard operating procedures, consulting with subject matter experts, or utilizing diagnostic tools to troubleshoot and resolve the issue [29, 30].

It is important to document the details of alarms, including their time of occurrence, duration, actions taken, and any additional information that can aid in future analysis or auditing [29]. After the alarm is resolved, proper documentation ensures traceability, and knowledge sharing, and enables continuous improvement of alarm management processes [1].

A periodic review of alarm performance and system behavior is essential to identify opportunities for optimization. This involves analyzing alarm patterns, evaluating alarm effectiveness, and adjusting alarm settings as needed. The aim is to minimize nuisance alarms, reduce AFs, and improve the overall efficiency and usability of the alarm management system [4].

2.2.2.3 Alarm Data

Given that the alarm serves as the central entity within the alarm manager, it comprises a multitude of fields aimed at describing all the relevant information. However, for the current project, many of these fields may lack helpful information, necessitating an initial assessment to determine the significance of each field for subsequent pre-processing and field removal phases.

Table 2.1 presents a comprehensive list of features associated with an alarm stored by the Alarm Manager. Each feature is accompanied by its respective meaning, which is crucial for comprehending the problem, facilitating its resolution, and understanding the potential range of values that each feature can assume.

By carefully examining the features presented in Table 2.1, operators can develop a more comprehensive perception of the problem, enabling them to identify relevant information for effective troubleshooting and resolution. This systematic approach to understanding alarm attributes lays the foundation for subsequent data processing and analysis, ensuring that only pertinent fields are retained while irrelevant or redundant ones are appropriately removed.

This thorough understanding of the alarm features enables effective problem-solving and decision-making processes by providing insights into the nature of the issue at hand. By carefully examining and analyzing these features, operators can identify pertinent information, eliminate irrelevant fields, and streamline subsequent data processing and analysis efforts. Chapter 3 will provide a more in-depth exploration of the discussed topics.

2.2.2.4 Alarm data mining

With the continuous advancements in alarm monitoring and control systems, the amount of data generated for analysis has significantly increased in both volume and complexity. This influx of data poses challenges for network operators who need to effectively process and interpret the information in a timely manner [4, 8, 9]. To address this issue, the application of data mining techniques has become a valuable solution.

Alarm data mining refers to the process of extracting meaningful insights and patterns from alarm data generated in various systems and processes. In many industries, such as manufacturing, telecommunications, and energy, alarm systems play a crucial role in monitoring and alerting operators about abnormal conditions or events that require attention.

Table 2.1: Alarm manager features

Feature	**Description**	**Range of Values** / Type
equipment_name	Name of the OLT where the alarm instance occurred	String (ex: olt401.hub.islpny.alticeusa.net)
equipment_type	Type of the corresponding OLT	String (ex: OLT2T4)
equipment_ manageddomain	OLT location	string (ex: New_York)
equipment_site	Area where the OLT is installed	string (ex: ISLPNY)
equipment_id	OLT identification	string (ex: 10.172.72.7)
alarmedresource_id	Device identification	string (ex: 10.172.72.7-20-101-2-1-2)
events	Number of occurrences	N>0 (ex: 1)
severity	Degree of severity of the alarm	string (ex: Critical)
raisedtime	Creation time of the alarm instance, measured by the Alarm Manager clock	timestamp (ex: 01/02/2023 01:59)
clearedtime	Clearing time of the alarm instance, measured by the Alarm Manager clock	timestamp (ex: 01/02/2023 01:59)
specificproblem	A brief summary of the alarm	string (ex: Deactivate failure of ONUi)
specificproblem-abbrev	Abbreviation of the alarm	string (ex: DFi)
ackstate	Whether the alarm was acknowledged or not	string (ex: Unacknowledged)
Actingurgency	Degree of urgency of the alarm, like severity	string (ex: Critical)
archivetime	Closing time of the alarm instance, measured by the Alarm Manager clock	timestamp (ex: 01/02/2023 01:59)
id	unique identifier of an alarm	N>0 (ex: 100400698)

Alarm data mining involves analyzing and exploring large volumes of alarm data to identify patterns, trends, anomalies, and relationships within the data. By doing so, it aims to improve the overall understanding of alarm behavior, optimize alarm settings, and enhance the efficiency and effectiveness of alarm management systems [1, 2]. By leveraging alarm data mining techniques, organizations can gain valuable insights into their systems' behavior, enhance operational efficiency, and improve decision-making processes related to alarm management.

Alarm data mining encompasses several common goals [6, 8, 9], which are:

- Alarm optimization: This involves the identification of redundant or nuisance alarms, reducing false positives, and improving the prioritization of alarms. The aim is to prevent

alarm overload and mitigate operator fatigue, ensuring that operators can focus on critical tasks effectively.

- Fault detection and diagnosis: Through alarm data mining, abnormal conditions can be detected, and the root causes of alarms can be diagnosed. This process provides valuable insights for troubleshooting and maintenance activities, enabling prompt resolution of issues.

- Performance improvement: By analyzing alarm response times, alarm resolution rates, and operator actions, alarm data mining helps identify areas for enhancing system reliability and operator performance. These insights facilitate the implementation of targeted improvements to optimize overall operational efficiency.

- Predictive analytics: Historical alarm data is utilized to build predictive models. These models enable the anticipation of potential failures or abnormal events, enabling proactive maintenance actions to be taken. By addressing issues before they escalate, downtime can be minimized, and system availability can be maximized.

By pursuing these goals, alarm data mining empowers organizations to optimize their alarm management systems, enhance operational efficiency, and improve decision-making processes. By uncovering hidden patterns and correlations, operators can gain actionable insights, streamline operations, and ensure the effective management of alarms in complex network environments [19, 20, 21].

2.2.3 Pattern Similarity Algorithms

This section delves into various pattern similarity algorithms used in the context of AF analysis, including the Levenshtein distance algorithm, Jaccard similarity, Needleman-Wunsch, and the Smith-Waterman algorithm. We will discuss their features and highlight why the Smith-Waterman algorithm is the most suitable choice for this thesis.

The Levenshtein distance algorithm, often referred to as the edit distance algorithm, quantifies the dissimilarity between two strings by determining the smallest count of single-character modifications needed to convert one string into another [31, 32]. While it serves as a valuable tool for evaluating text-based data, like textual alarm descriptions or codes, its effectiveness diminishes when confronted with temporal patterns or sequences, a frequent occurrence in AF analysis [32].

Jaccard similarity is a metric employed to gauge the resemblance between two sets. In alarm analysis, it proves valuable for assessing the similarity between alarms appearing in distinct sequences. This metric excels at recognizing shared elements but falls short in accounting for the chronological order of occurrences. Consequently, it may not capture essential patterns associated with the temporal sequencing of alarms within AFs [33, 34].

The Needleman-Wunsch algorithm, commonly employed in bioinformatics, is a global SA algorithm primarily designed for comparing sequences such as DNA or protein sequences [35].

This algorithm calculates the optimal alignment of two sequences by comprehensively evaluating matches, mismatches, gaps, and similarities spanning the entire length of the sequences [35, 36]. It assigns specific scores to matches, mismatches, and gaps, with the overarching objective of maximizing the alignment's overall score [37].

Both Needleman-Wunsch and Smith-Waterman algorithms are very similar. However, the Smith-Waterman algorithm is designed to find the best local alignment, meaning it identifies the most similar subsequence within two sequences [15]. This is particularly useful when you are interested in pinpointing specific regions of high similarity within longer sequences. In contrast, the Needleman-Wunsch algorithm aims to align the entire sequences globally [35], which can mask local similarities. The Smith-Waterman algorithm performs better when comparing sequences with significant dissimilarities. It doesn't enforce a global alignment like Needleman-Wunsch, which can sometimes align sequences that are too different.

In summary, the selection of the Smith-Waterman algorithm over other pattern similarity algorithms is underpinned by the algorithm's superior suitability for AF analysis. Its granularity, ability to handle temporal relationships, adaptability, robustness, and efficiency make it the most fitting choice for the comprehensive analysis of AF patterns and their underlying causes. In AF scenarios, where understanding precise temporal sequences is paramount, the Smith-Waterman algorithm emerges as the ideal solution to unearth complex relationships among alarms and capture subtle patterns efficiently.

2.2.4 Smith-Waterman Algorithm

The Smith-Waterman algorithm, initially proposed in [7], serves the purpose of finding a pair of segments, one from each of two long sequences, that exhibit the highest similarity (homology) compared to any other pair of segments. Originally developed in 1981 for the purpose of identifying similarities among molecular sequences, the Smith-Waterman Algorithm rapidly gained popularity and found broader applications, including the analysis of industrial alarm sequences [15, 16].

As a local SA method, the Smith-Waterman algorithm operates on the concept of local alignment. To elucidate this concept, let us consider a pair of symbolic segments, each obtained from two symbolic sequences [5]. By introducing gap symbols ('-') into one or both segments, their lengths can be equalized (if the sequences have the same length, there might be no gaps). Consequently, every symbol in one segment aligns with a corresponding symbol in the other segment at the same position, establishing an alignment.

Given that the two symbolic segments represent contiguous subsequences of their respective symbolic sequences, this alignment is referred to as the local alignment of the two sequences. In other words, the Smith-Waterman algorithm identifies the specific regions within the sequences where the highest similarity occurs rather than attempting to align the sequences in their entirety.

By utilizing dynamic programming techniques, the Smith-Waterman algorithm efficiently calculates a similarity score for each possible alignment position, taking into account the match/mismatch

between symbols and the introduction of gaps. The algorithm identifies the alignment with the highest score, representing the most significant local similarity between the sequences.

If we consider two sequence examples:

$$X = [3143112]$$

$$Y = [214123]$$

A possible alignment of this pair of sequences is:

$$X' = [3143112-]$$

$$Y' = [214-1-23]$$

After alignment, both sequences should have the same length. Each element in the aligned segment X' corresponds to an element in the aligned segment Y'. For instance, the first pair in the alignment is (3, 2), and the second pair is (1, 1). It is important to note that this is just an example of how the alignment could be performed, as there are countless possible combinations. However, the algorithm's goal is to determine the optimal alignment, which requires distinguishing between good and poor alignments.

The objective of the alignment process is to identify the alignment that maximizes the similarity between the sequences. This involves assigning scores to different alignments and selecting the one with the highest score as the optimal alignment. Good alignments are characterized by a high similarity score, indicating a strong match or homology between the aligned segments. On the other hand, poor alignments have lower similarity scores and may indicate a weaker or insignificant match between the segments. To achieve this, the Smith-Waterman algorithm utilizes a scoring system that considers various factors such as match/mismatch scores and gap penalties.

Consider a symbolic pair (a, b) from two aligned sequences, where a and b represent two alarms. If neither of them corresponds to a gap ("-"), a similarity score function, s(a, b), is calculated. The value of this similarity function is positive if the alarms match (a == b), and a non-positive value is assigned if a $\neq$ b. Typically, in the case of pair matching where no prior information about the alarm types is available, a score of 1 is assigned for matching pairs, and a negative value μ is assigned for non-matching pairs. However, these scores can vary. Therefore, the result of the similarity function would be 1 for the pair (1, 1) and μ for the pair (3, 2).

In cases where a gap is introduced in the sequences, and the pair (a, b) includes a gap, a penalty is always added to the similarity score function. A constant penalty δ is assigned for all gaps inserted in the SA. For example, in the pair (3, -), the score of the function would be δ.

Finally, to calculate the overall similarity of the alignment between the two sequences, the sum of all similarity scores of all pairs included in the alignment is computed. Referring back to the initial example alignment of the segments X' and Y', considering a penalty of δ = -0.4 for introducing a gap and a non-matching penalty μ = -0.6, the following similarity function is obtained:

$$X' = [3143112-]$$

$$Y' = [214-1-23]$$

$$S(X',Y') = s(3,2)+s(1,1)+s(4,4)+\delta+s(1,1)+\delta+s(2,2)+\delta$$

$$S(X',Y') = (-0.6)+1+1+(-0.4)+1+(-0.4)+1+(-0.4) = 2.2$$

As it is evident, the higher the result of S(X', Y'), the greater the degree of similarity. The alignment between X and Y that has the highest similarity score function is considered the optimal alignment between the two sequences.

It is important to note that the similarity score function provides a quantitative measure of how well the two sequences align with each other. By comparing different alignments and their corresponding similarity scores, we can determine which alignment captures the highest level of similarity between the sequences. This optimal alignment represents the best matching pattern between the AF sequences, indicating the strongest correlations and patterns within the data.

Now that we have a thorough understanding of SA, it becomes much simpler to formulate the problem addressed by the Smith-Waterman Algorithm. To solve the problem of local alignment, we consider two symbolic sequences, A and B, with arbitrary lengths:

$$A = [a_1, a_2, \cdots, a_M]$$

$$B = [b_1, b_2, \cdots, b_N]$$

Let Σ be the set of elements, and a_m and b_n are elements from sequences A and B, respectively, where $m = 1,2,\ldots,M$ and $n = 1,2,\ldots,N$.

We can define segments or contiguous subsequences of A and B as $A_{i:p}$ and $B_{j:q}$, respectively, where $1 \leq i \leq p \leq M$ and $1 \leq j \leq q \leq N$.

$$A_{i:p} = [a_i, a_{i+1}, \cdots, a_p]$$

$$B_{j:q} = [b_j b_{j+1}, \cdots, b_q]$$

The similarity index, $I(A_{i:p}, B_{j:q})$, represents the similarity between the segment pair $(A_{i:p}, B_{j:q})$. The goal of the local alignment problem is to find the optimal segment pair that has the highest similarity index among all possible segment pairs. We denote this highest similarity index as S(A, B). However, if the two sequences, A and B, are completely different and all segment pairs have a negative similarity index, S(A, B) is set to 0 to indicate no similarity. The similarity index S(A, B) can be calculated using Equation 2.1.

$$S(A,B) = \max_{1 \leq i \leq p \leq M, 1 \leq j \leq q \leq N} (I(A_{i:p}, B_{j:q}), 0) \tag{2.1}$$

To determine S(A, B) and the corresponding local alignment, we use the Smith-Waterman algorithm. This algorithm constructs an index matrix H, where each element $H_{p+1,q+1}$ represents the maximum positive similarity index of segment pairs ending in a_p and b_q. If there is no positive similarity index, $H_{p+1,q+1}$ is set to 0. The value of $H_{p+1,q+1}$ can be recursively computed using Equation 2.2, considering a uniform penalty δ, when a gap is inserted into the sequence.

$$H_{p+1,q+1} = \max\{H_{p,q} + s(a_p, b_q), H_{p,q+1} + \delta, H_{p+1,q} + \delta, 0\} \tag{2.2}$$

The dynamic programming algorithm based on this equation can be summarized in the following steps:

1. Build a matrix H of size (M+1)×(N+1) and initialize the first row and column of H as 0.
2. Calculate the values of the remaining entries in H using the recursive equation.
3. Find the highest value in matrix H, which represents the similarity index S(A, B) of the two sequences.
4. Trace back from this highest value until reaching an entry with a value of 0. The path obtained during this traceback represents the optimal local alignment.

By applying this dynamic programming algorithm, we can efficiently solve the problem of local alignment and determine the optimal alignment and similarity index between the two sequences, A and B.

Let us consider the example of the two sequences mentioned earlier: A = [3 1 4 3 1 1 2] and B = [2 1 4 1 2 3]. We have a gap penalty δ = -0.4, a match score of 1, and a mismatch penalty μ = -0.6. To find the optimal local alignment, we first construct the matrix H. Each element H(i, j) in the matrix represents the maximum similarity index of segment pairs ending at position i in sequence A and position j in sequence B. After initializing the first column and row of the matrix H with 0, we proceed to calculate the remaining values of H by considering the neighboring elements and applying Equation 2.2. This iterative process allows us to determine the maximum similarity index for each position in the matrix. Figure 2.1 represents the matrix H of the alignment.

By performing iterative calculations of the values in matrix H, we can identify the highest similarity index. In our example, the maximum value is found at H(8, 6), which is 3.2. This value is determined by considering the contributions from H(7, 5) + 1, H(7, 6) + δ, H(8, 5) + δ, and 0. Upon closer examination, it becomes evident that the maximum similarity score of the segments is 3.2 rather than 2.2, as previously mentioned. This implies that the alignment example provided earlier does not represent the optimal local alignment.

To obtain the optimal local alignment, we perform a backward path search starting from the element with the highest value in H. We follow the path by considering the neighboring elements with increasing indices until we reach an entry with a value of 0. The path that represents the alignment of the segments can be observed in Figure 2.2.

	A	**3**	**1**	**4**	**3**	**1**	**1**	**2**
B	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1
1	0	0	1	0.6	0.2	1	1	0.6
4	0	0	0	2	1.6	1.2	0.8	0.4
1	0	0	1	1.6	1.4	2.6	2.2	1.8
2	0	0	0.6	1.2	1	2.2	2	3.2
3	0	1	0.6	0.8	2.2	1.8	1.6	2.8

Figure 2.1: Matrix H for example sequences A and B

After obtaining the path, we proceed forward to complete the alignment. Each diagonal move corresponds to adding a symbol from both aligned segments. Each horizontal move involves adding a symbol from sequence B to its aligned segment and inserting a gap in sequence A's aligned segment. Similarly, each vertical move includes adding a symbol from sequence A to its aligned segment and inserting a gap in sequence B's aligned segment. In Figure 2.3, the alignment score of the two segments is displayed. The alignment score represents a numerical value that quantifies the overall similarity between the two sequences being aligned.

Following these rules, we finalize the optimal local alignment for our example: [1 4 3 1 1 2] and [1 4 - 1 - 2].

	A	3	1	4	3	1	1	2
B	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1
1	0	0	1	0.6	0.2	1	1	0.6
4	0	0	0	2	1.6	1.2	0.8	0.4
1	0	0	1	1.6	1.4	2.6	2.2	1.8
2	0	0	0.6	1.2	1	2.2	2	3.2
3	0	1	0.6	0.8	2.2	1.8	1.6	2.8

Figure 2.2: Optimal path of the aligment

$A_{2:7}$:	1		4		3		1		1		2	
Score:	1	+	1	+	-0.4	+	1	+	-0.4	+	1	= 3.2
$B_{2:5}$:	1		4		—		1		—		2	

Figure 2.3: Score of the optimal alignment

Chapter 3

Pattern Matching of Alarm Floods Sequences

The AF pattern matching problem revolves around measuring the similarity between different AFs and identifying which floods should be grouped together based on this similarity. Think of an AF as a sequence of alarms that occur within a specific time frame. The goal is to find patterns or common segments among these AFs that can act as distinguishing features.

By calculating a similarity index, we can quantify the resemblance between AFs. If two floods share a significant portion of alarms or exhibit similar patterns, they are considered more alike and likely to belong to the same group or pattern. These shared segments within a pattern serve as symptomatic characteristics that aid in determining whether a new incoming flood should be classified into an existing pattern.

In essence, the AF pattern matching problem involves identifying similarities between floods, grouping them based on these similarities, and utilizing common segments as indicators to classify new floods. This process enables the detection and categorization of AFs, facilitating more effective analysis and decision-making in alarm management systems.

In this chapter, we will delve into data processing methods to optimize the speed and efficacy of analysis. Additionally, we will explore a possible algorithmic approach for Pattern Matching of AF Sequences, drawing inspiration from the Smith-Waterman Algorithm.

3.1 Pre-processing

In the field of data analysis and machine learning, the process of preparing data for modeling and analysis plays a critical role in achieving accurate and reliable results. Pre-processing techniques are employed to transform raw data into a suitable format that can be effectively utilized by machine learning algorithms. This stage involves several steps, including feature selection, addressing chattering, and organizing the dataset into temporal sequences.

Feature selection is a crucial pre-processing step that aims to identify and retain the most relevant and informative features from the available data. By selecting a subset of features that

have the most significant impact on the target variable, we can reduce computational complexity, enhance model interpretability, and potentially improve the overall performance of the predictive models.

Chattering refers to the occurrence of repeated or fluctuating values in a dataset, often caused by measurement errors or noise [38, 39, 40]. It can have a detrimental effect on model accuracy, as it introduces unwanted variations and instability. Pre-processing techniques, such as smoothing or filtering, are applied to mitigate chattering and create more consistent and reliable data.

Furthermore, in many applications, data is collected over time and exhibits temporal dependencies. To capture these temporal relationships and patterns, it is essential to organize the dataset into sequential segments or time series [29]. By dividing the data into temporal sequences, we can leverage the inherent temporal information to improve the predictive power of our models.

As discussed in section 2.2.2, this dissertation utilizes data sourced from an alarm manager managed by Altice Labs, which is responsible for capturing alarm information in a fiber optic network. To develop and study the algorithm, a comprehensive dataset consisting of a full day's worth of alarms from February 1, 2023, was provided. This dataset encompasses a significant volume of data, comprising a total of 534,859 alarm entries. The extensive and representative nature of this dataset ensures that the algorithm can be thoroughly tested and evaluated for its effectiveness in handling real-world alarm scenarios.

3.1.1 Model features

As a first step in the pre-processing phase, it is necessary to select the features that are relevant to the model. This involves identifying and choosing the features that can provide valuable insights and information. In the context of the problem under investigation in this dissertation, it has been determined that the majority of features within the dataset hold little relevance to the system. To ensure simplicity and reduce computational workload, it is essential to remove these irrelevant features from the model.

The primary objective of this analysis is to explore the relationships among various alarms within the system. A key factor in the data that plays a crucial role is the correlation between failures, with a specific emphasis on accurately identifying the specific problem that occurred. This identification is represented by the variable "specificProblem" throughout the study.

To provide a visual representation of the occurrence frequencies for each problem, the accompanying Figure 3.1 illustrates the number of times each problem has been recorded within the top 20 occurrences over a full day. This visual depiction offers valuable insights into the relative significance and distribution of different problems within the system, aiding in prioritization and understanding of the most prevalent issues.

In order to achieve more accurate and effective identification of correlations among AF sequences, relying solely on the alarm description parameter is not sufficient. It is also beneficial to incorporate the "raisedtime" feature, which represents the time at which the alarm occurred. By including this temporal information, we not only identify and order the alarms but also capture

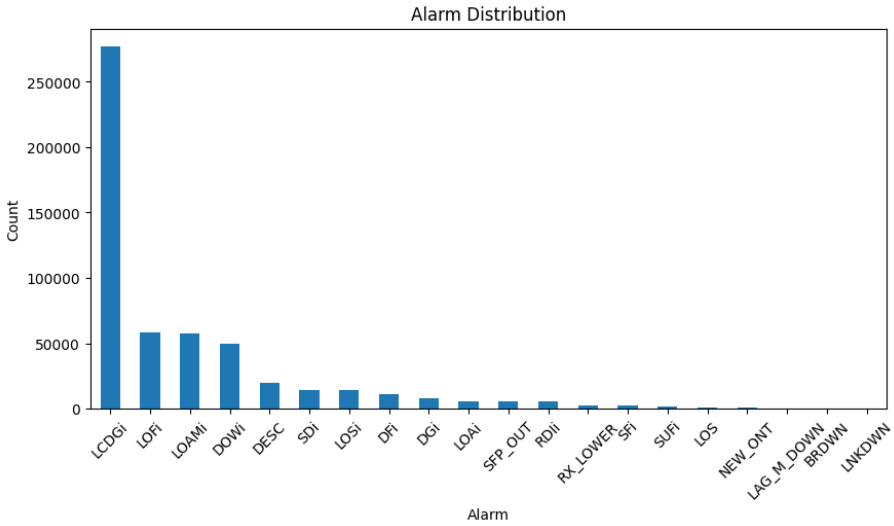

Figure 3.1: Top 20 alarm distribution

the temporal distance between them. This temporal aspect can provide more precise indications of similarities between sequences.

Furthermore, the "alarmedresource_id" feature, which indicates the equipment where the alarm occurred, has also been considered. By leveraging this feature, we can divide the analysis into regions, enabling the identification of physical distances between equipment. This approach helps to eliminate potential correlations between different infrastructures.

To enhance the analysis, the combination of the “specificproblem”, “raisedtime”, and “alarmedresource_id” features allows for a more comprehensive exploration of AFs. This multi-dimensional perspective facilitates a deeper understanding of the relationships and patterns within the data, enabling more accurate identification and analysis of correlated sequences.

Another feature that could potentially be useful is "clearedtime," which represents the time when an alarm is cleared or resolved. This feature provides information about the duration of each alarm, which could offer valuable insights. However, upon analyzing the distribution of alarm durations, it was observed that the majority of alarms have very short durations. Specifically, around 90% of alarms have a duration of less than one minute, with over 50% of alarms having a duration of 0 seconds.

Considering this distribution pattern, it was determined that the "clearedtime" feature may not contribute significant discriminatory power to the analysis. Including it in the model may introduce noise or irrelevant information. To visually illustrate this distribution, a graph can be observed in Figure 3.2.

3.1.2 Chattering

Implementing a method to remove chattering alarms is highly recommended as they do not provide any valuable information for analysis [38, 41]. Although there is no definitive criterion to identify

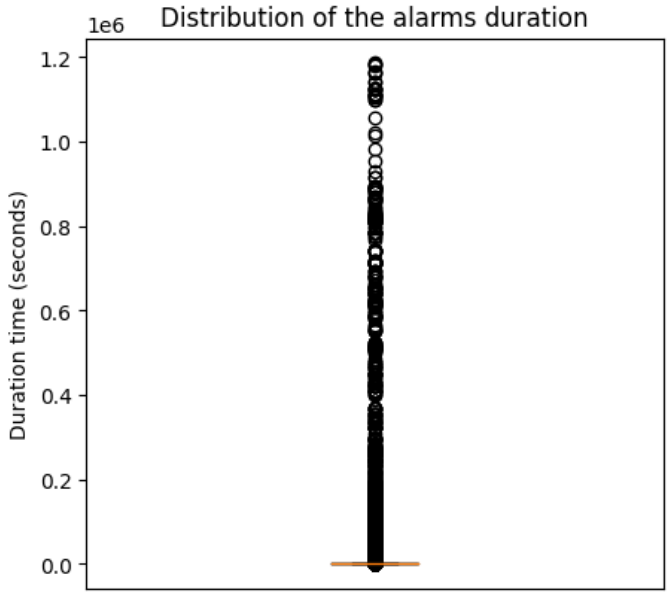

Figure 3.2: Alarms duration distribution

chattering alarms, various techniques are used, with their effectiveness depending on the specific system being analyzed. The main objective of this step is to eliminate repeated alarms and prepare the dataset accordingly [42].

In many reported cases, including the present one, the dataset primarily consists of point-based data, which means that the alarms only contain information about the occurrence timestamp and not their duration. Typically, a fixed time window (Tw) is employed in these scenarios, representing a specific temporal interval in which only one occurrence of each alarm is allowed. There are two common approaches to accomplish this:

- Fixed Data Frame: In this method, once an alarm is triggered, no new occurrences of the same alarm are accepted during the time window Tw. For instance, if we have an alarm sequence $a_i < t_1, t_2, \cdots, t_{n-1}, t_n >$ with $n = \|a_i\|$, any timestamp t_{i+x} that satisfies $t_{i+x} - t_i < Tw$ (for x > 0) corresponds to the same alarm and will be filtered out. A new alarm of that type is only accepted after $t_i + Tw$, and the process repeats accordingly.

- Moving Data Frame: This approach is similar to the fixed data frame method, with the distinction that the Tw period is calculated based on the previous alarm in the original dataset, rather than the most recent alarm classified as non-chattering. In other words, if the time difference between t_i and the previous alarm t_{i-1} is less than Tw, the alarm is considered chattering and is omitted during the pre-processing stage.

Figure 3.3 displays the two analyzed methods of chattering using a window time frame of 1 minute. Both methods can be implemented, and there is no clear superiority of one over the other. The choice should be based on the specific use case being processed. However, despite the fact

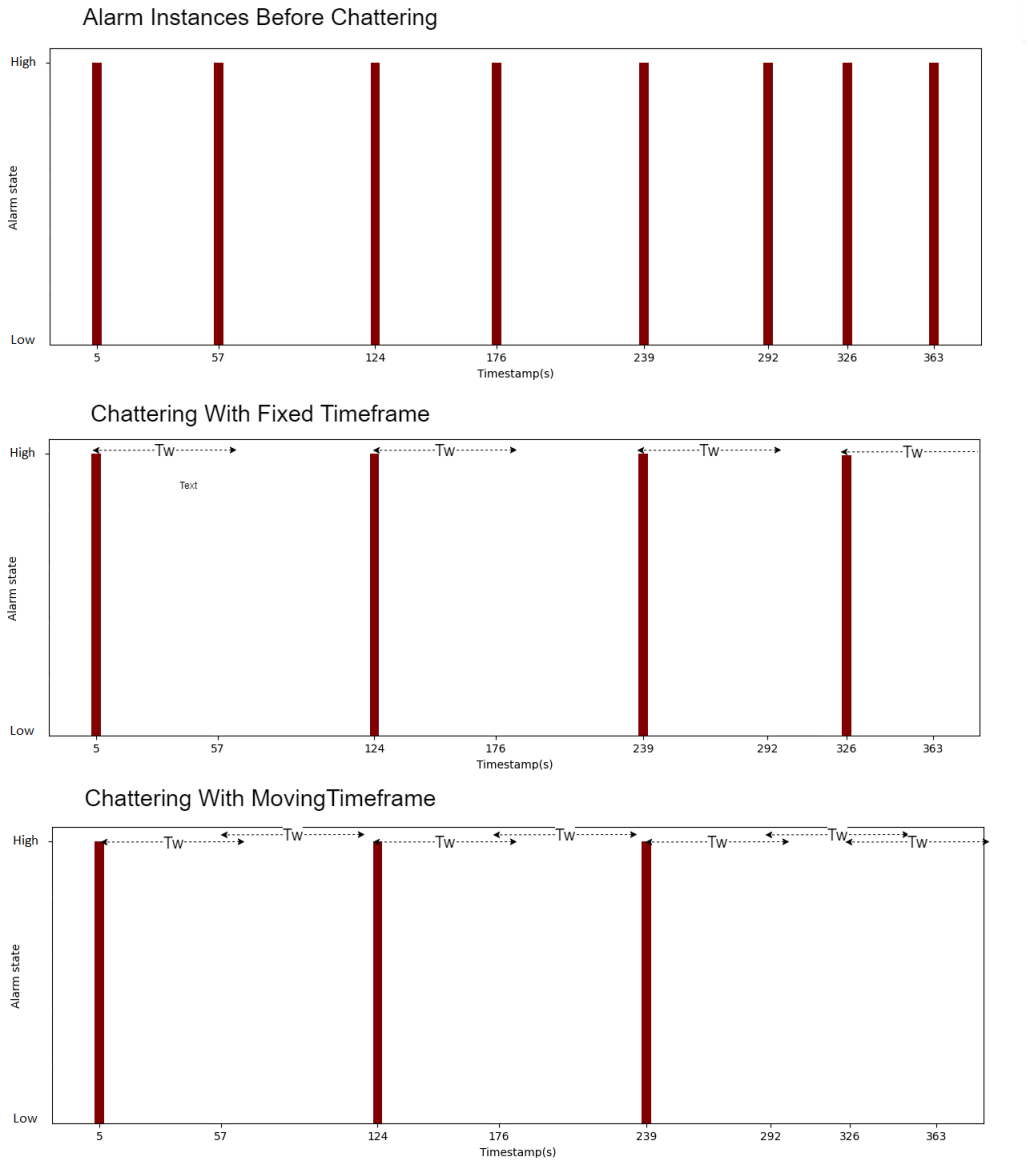

Figure 3.3: Alarm Chattering: Fixed Vs Moving Data Frame ($Tw = 60s$)

that the fixed time frame method removes fewer alarms, it has been more widely adopted. This is because while chattering alarms should not be accepted, data analysts prefer not to remove all these alarms for fear of losing important information. It is important to carefully consider the trade-offs between removing chattering alarms and potentially losing important information.

The selection of an appropriate value for the alarm removal period, Tw, is crucial for the efficiency of the chattering process [38, 43]. This value can be fixed for the entire system or configured specifically for each alarm tag. According to the ISA-18.2 standard [42], an alarm is considered chattering if it repeats three or more times within a period of less than one minute. However, this restriction period is often extended to ensure the removal of all potential chattering alarms. Although there are no direct constraints on the chosen value, the period is typically between 60 and

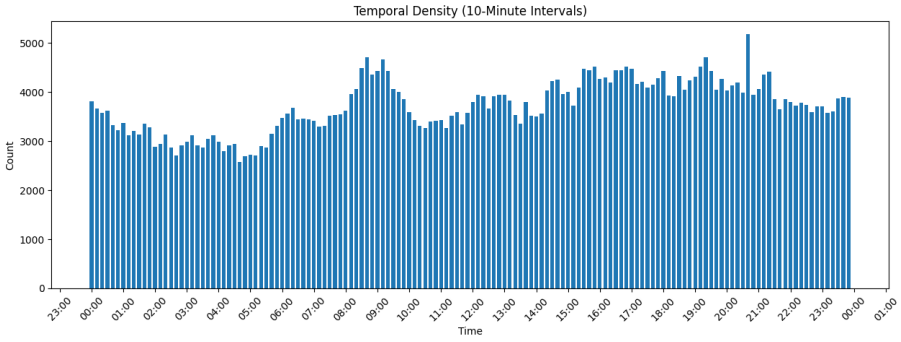

Figure 3.4: Alarm frequency in original dataset

300 seconds, with a one-minute time frame being the most common choice.

To process the dataset effectively, a time window of two minutes was applied, using the fixed data frame method. This approach was chosen to remove chattering alarms, resulting in a significant reduction in the dataset size. The initial dataset, consisting of over half a million alarms, was pruned down to approximately 150,000 alarms after the removal process.

To visualize the impact of the preprocessing step, Figures 3.4 and 3.5 provide a comparison of alarm occurrences in 10-minute intervals before and after the chattering alarm removal. The graph highlights the effectiveness of the filtering process in reducing alarm clutter and enhancing the overall quality of the dataset.

Overall, the application of the fixed data frame method and subsequent removal of chattering alarms has streamlined the dataset, creating a more manageable and insightful dataset for subsequent analysis.

3.1.3 Alarm Floods

Now that the dataset has undergone the desired filtering process, we proceed with the final step: dividing it into alarm sequences. However, before we proceed with the temporal segmentation, we also perform a spatial division. This involves separating the alarms based on the specific devices in which they occurred. Since the alarm manager consists of multiple OLTs, it is crucial to separate the reported alarms for each OLT. This separation allows us to analyze the behavior of each network segment individually, providing a clearer understanding of the most frequent problems encountered in each OLT.

This separation is based on the value of the "alarmedresource_id" field, which identifies the location in the network where the alarm occurred. After discussions with a team knowledgeable about the network, it was decided that the division would be based on the Passive Optical Network (PON) associated with the OLT where the alarm was reported. In other words, the division is performed by examining only the first three values of the "alarmedresource_id" field. The first value identifies the OLT, the second value represents the respective card, and the third value indicates the downlink PON of the card. This decision was made considering the low correlation

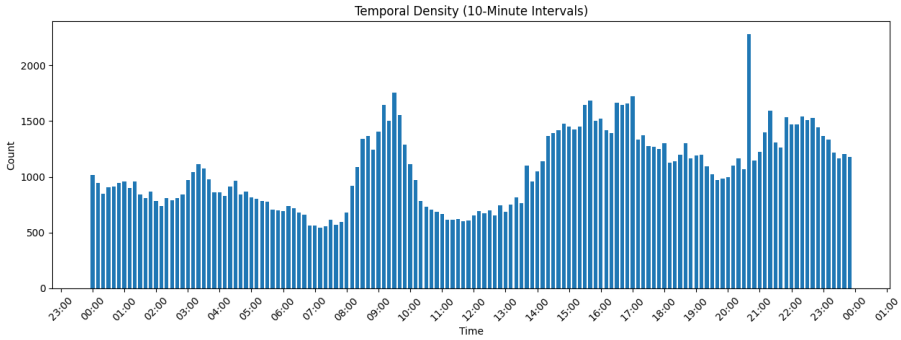

Figure 3.5: Alarm frequency after chattering ($Tw = 120s$)

between devices in different PONs and the algorithm's computational load for similarity analysis. By dividing the dataset in this manner, we significantly reduced the training time required for the model. Additionally, this division aligns with the network's structure, allowing for a more focused analysis of each PONs behavior.

This spatial segmentation provides valuable insights into the specific performance and behavior of individual PONs within the network. By isolating alarms reported from different PONs, we can better understand any unique characteristics or issues associated with each segment. This approach enhances the efficiency and effectiveness of subsequent analyses, enabling more targeted troubleshooting and problem-solving efforts.

Once the spatial separation is completed, we can focus on the temporal segmentation by identifying AFs. The objective is to determine the root cause of alarm sequences, as correlated alarms often occur in close proximity in terms of time. To achieve this, we follow the guidelines provided by the ISA18.2 standard. According to these guidelines, an AF is defined as follows: it begins when the alarm rate exceeds 10 alarms per 10 minutes within a 10-minute interval, and it ends when the alarm rate falls below 5 alarms per 10 minutes within a regular 10-minute interval.

Applying these criteria to the dataset, we initially identified a total of 154 AFs. However, upon closer analysis, it became evident that 42 of these floods had considerably different sequence lengths compared to the majority of the floods. To ensure consistency and coherency in the subsequent analysis, we made the decision to remove these 42 AFs from the dataset. This pruning process primarily aimed to expedite the algorithm's training time, as processing time scales quadratically with the length of AFs.

By segmenting the remaining AFs, we can isolate and analyze sequences of alarms that are temporally close and likely related. This segmentation enables a more focused examination of the root cause within alarm patterns, allowing for a deeper understanding of the underlying issues. With the dataset now organized into AFs, it is ready for further analysis and the application of the pattern matching algorithm to identify and explore meaningful relationships among the alarm sequences.

3.2 Adapted Smith-Waterman Algorithm

The AF pattern matching problem involves calculating a similarity index between AFs and determining which floods should be classified into the same group or pattern based on this index. The goal is to identify common segments among AFs within a pattern, as these segments serve as indicative symptoms of such AFs. By analyzing the common segments, it becomes possible to assess whether an incoming flood belongs to a particular pattern.

This approach, proposed in [4, 10], builds upon the principles of the Smith-Waterman Algorithm, described in 2.2.4, for similarity analysis of segments and introduces a novel element: temporal information. By incorporating time into the analysis, this approach allows for a more flexible and nuanced interpretation of alarm sequences. For instance, if two alarms occur in close proximity, this method does not differentiate between which one was reported first, thereby accommodating variations in reporting order. Moreover, it can effectively discern alarm pairs that occur with significant temporal differences, indicating a lower level of correlation between them. By considering the temporal dimension, this approach provides a richer understanding of alarm patterns and their interrelationships, enabling more accurate and insightful alarm management in complex systems.

To gain a better understanding of the algorithm, let's consider a scenario where we have a set of alarms, denoted by K, and define a time-stamped AF as follows:

$$A =< a_1, a_2, \cdots, a_M >$$

$$a_m = (e_m, t_m), m = 1, 2, \cdots, M.$$

Here, e_m represents the alarm type from the set K, and t_m represents the corresponding time stamp of the alarm occurrence for each alarm a_m.

In order to explain the new method of calculating the similarity score for pairs of time-stamped alarms, it is necessary to calculate the temporal distance between each alarm in relation to their time stamps. This is achieved by introducing the concept of a time distance vector for each AF:

$$d_m = [d_m^1, d_m^2, \cdots, d_m^K]^T$$

$$d_m^K = \begin{cases} \min_{1 \le i \le M}\{|t_m - t_i| : e_i = k\} & \text{if the set is not empty} \\ \infty & \text{otherwise} \end{cases}. \tag{3.1}$$

Each entry (d_m^K) in the time distance vector (d_m) provides valuable information about the time gap between the m-th alarm and the nearest alarm of type k on the time axis. If there are no alarms of type k present in the alarm sequence (i.e., none of the alarms have the alarm type k), the corresponding time gap is considered infinite (∞).

To illustrate this further, let us consider an example. Imagine that we have an alarm sequence where each alarm is associated with a specific type, such as temperature or pressure. The time

distance vector for a particular alarm, let us say the m-th alarm (a_m), captures the time gaps between this alarm and the nearest alarm of type k (d_m^K). If there are no alarms of type k present in the sequence, the time gap is considered infinite because there is no reference alarm of that type to compare it with.

It's important to note that the m-th alarm itself has the alarm type e_m, so the corresponding entry ($d_{e_m}^m$) in the time distance vector is naturally zero since the time gap between an alarm and itself is zero.

Once the distances have been calculated, they need to be converted into time weights. This conversion is achieved by applying a monotonically decreasing function that maps the distances to weights. This function should satisfy the properties of $f(0) = 1$ and $f(\infty) = 0$. By applying this function to each distance, we obtain the corresponding time weight. The resulting time weight vector captures the relative importance or influence of each alarm type based on their temporal proximity. The time weight vector can be obtained as follows:

$$\begin{aligned} W_m &= [w_m^1, w_m^2, \cdots, w_m^K]^T \\ &= [f(d_m^1), f(d_m^2), \cdots, f(d_m^K)]^T. \end{aligned} \tag{3.2}$$

As a result, the $e_m - th$ entry of the time weight vector W_m is 1, indicating that the m-th alarm belongs to its corresponding alarm type. When an alarm event occurs very close to the m-th alarm on the time axis, it receives a high weight in the time weight vector. Conversely, if a specific alarm type is not present in the vicinity of the m-th alarm on the time axis, it is assigned a low weight. This weight gradually diminishes and approaches zero as the time gap increases. By using the time weight vector, an alarm event is not restricted to a single alarm type but is associated with all alarm types with varying weights, reflecting the temporal relationship between alarms. For the implementation of the algorithm, the scaled Gaussian function was chosen:

$$f(x) = e^{-\frac{x^2}{2\sigma^2}}. \tag{3.3}$$

The value σ controls the time weight according to the distance. A higher value of σ results in a wider curve for the function, leading to a larger or equal similarity index. On the other hand, if σ is set to 0, the curve becomes non-existent, and the algorithm behaves the same as the standard Smith-Waterman algorithm.

By adjusting the value of σ, you can fine-tune the sensitivity of the algorithm to capture different levels of temporal proximity between alarm events. A higher σ value allows for a broader acceptance of time gaps, potentially capturing more similar patterns. Conversely, a lower σ value narrows the acceptance window, focusing on more immediate temporal relationships.

It's important to experiment with different σ values to find the optimal balance between capturing relevant temporal relationships and avoiding false matches. The choice of σ will depend on the specific characteristics of your alarm data and the desired level of flexibility in the similarity assessment.

When comparing two sequences, it is not recommended to apply the same function to both sequences. This is because the concept of temporal difference is relative rather than absolute. Applying a Gaussian function to both sequences can lead to duplicate matches if there are many closely spaced alarms, which can result in an increase in the similarity index. In [9], to address this, a different function is applied to the second sequence:

$$f(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

However, this can lead to undesired alignments, particularly when multiple different alarms occur at the same timestamp. To counteract this tendency, the time weight vector was calculated as follows:

$$w_m^k = \begin{cases} 1 & \text{if } e_m = k \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

Now, let us redefine the similarity score, denoted as s(a, b), for a pair of time-stamped alarms $s((e_a,t_a),(e_b,t_b))$. In the original Smith-Waterman algorithm, the similarity score function only had two possible values: a match score of 1, indicating that the alarms are similar, or a mismatch penalty denoted as μ, indicating a dissimilarity between the alarms.

However, in the modified algorithm, a more flexible approach is introduced. The similarity score is now calculated as a linear combination of the match score and the mismatch penalty. This means that the score value is determined by a weighted combination of these two factors. The specific weights or ratios used in the combination are based on the time weight vectors, W_a and W_b, associated with each alarm. The new similarity score can be obtained as follows:

$$s((e_a,t_a),(e_b,t_b)) = \max_{1\leq k\leq K}[w_a^k \times w_b^k](1-\mu)+\mu. \tag{3.6}$$

After obtaining the similarity indexes for all pairs, the next step is to construct the H matrix, which follows a similar approach to the original Smith-Waterman algorithm. The H matrix is built using Equation 2.2.

After obtaining the complete matrix, the similarity index is represented by the maximum value in the matrix and the optimal local alignment is obtained through a backward path search starting from that element.

It is important to note that, unlike the Smith-Waterman algorithm, the similarity score between two sequences, S(A, B), is not necessarily equal to S(B, A). This is because a Gaussian function is applied only to the temporal distance of one of the sequences and not to both. As a result, both similarity indexes between the two AFs should be calculated by inverting the order of the sequences. The similarity index between the two sequences is then defined as the maximum value between these two indexes.

Table 3.1: AFs examples

AF A			AF B	
Alarm	**Timestamp**		**Alarm**	**Timestamp**
DESC	20:10:24		LOSi	15:54:07
RX_LOWER	20:10:24		DESC	15:54:07
LOFi	20:10:24		LOFi	15:54:24
DFi	20:10:28		LOAMi	15:54:40
LOAMi	20:10:28		LOFi	15:56:51
LOFi	20:13:03		DFi	15:57:17
			LOAMi	16:01:05
			LOSi	16:01:06

Following the matrix calculation formula, the maximum possible similarity value between two sequences is determined by the length of the shorter sequence. This means that as the sequences become longer, the similarity index between them increases, which may not be desirable. To address this, the similarity index can be normalized between 0 and 1. This can be achieved by dividing the maximum value in the H matrix by the length of the shorter sequence. Therefore, if two segments have exactly the same pattern, the normalized similarity value will always be 1, regardless of the lengths of the sequences.

By normalizing the similarity index, it allows for fair comparisons between sequences of different lengths and ensures that the similarity score reflects the similarity of the patterns rather than the sequence lengths.

We take as an example two partial AFs presented in Table 3.1 obtained from the provided dataset. The set of unique alarms consists of 6 alarms, denoted as $\Sigma = \{'DESC','RX\_LOWER', 'LOFi','DFi','LOAMi','LOSi'\}$. Segment A has a length of 7, while Segment B has a length of 9.

To begin, we need to calculate the time distance vectors. Each alarm instance will have a matrix size of 7x6, where 7 represents the length of the segment, and 6 represents the number of unique alarms. Let us consider the first alarm in AF A as an example. It belongs to the $'DESC'$ alarm type, resulting in a distance of 0 to the $'DESC'$ alarm. For instance, the minimum distance to a $'LOAMi'$ alarm is 4 seconds. As there are no $'LOSi'$ alarms in sequence A, all instances will have an infinite distance to this particular alarm type. Thus, the time distance vectors for segment A yield the results in Table 3.2.

By applying the formula previously presented in Equation 3.3, we obtain the following time weight vectors shown in Table 3.3.

By setting the algorithm parameters $\delta = -0.4$ and $\mu = -0.6$, we can calculate the matrix H, representing the similarity index of alarm sequences A and B, along with the optimal local alignment. This calculation process closely follows the classical Smith-Waterman algorithm, with the key difference lying in the calculation of the similarity score.

For example, when determining the value of $H_{4,8}$, we need to evaluate the similarity score between the 3rd alarm in sequence A and the 7th alarm in sequence B. To accomplish this, we utilize the time weight vectors associated with each alarm. In this case, the time weight vector

Table 3.2: Time distance between alarm types

AF A	DESC	RX_L	LOFi	DFi	LOAMi	LOFi	LOAMi
DESC	0	0	0	4	4	159	164
RX_LOWER	0	0	0	4	4	159	164
LOFi	0	0	0	4	4	0	5
DFi	4	4	4	0	0	155	160
LOAMi	4	4	4	0	0	5	0
LOSi	∞	∞	∞	∞	∞	∞	∞

Table 3.3: Time weight between alarm types

AF A	DESC	RX_L	LOFi	DFi	LOAMi	LOFi	LOAMi
DESC	1	1	1	0.1353	0.1353	0	0
RX_LOWER	1	1	1	0.1353	0.1353	0	0
LOFi	1	1	1	0.1353	0.1353	1	0.0439
DFi	0.1353	0.1353	0.1353	1	1	0	0
LOAMi	0.1353	0.1353	0.1353	1	1	0.0439	1
LOSi	0	0	0	0	0	0	0

for the 3rd alarm in sequence A is given as $w_3 = [1\ 1\ 1\ 0.1353\ 0.1353\ 0]^T$, while the time weight vector for the 7th alarm in sequence B is $w_7 = [0\ 0\ 0\ 1\ 0\ 0]^T$, obtained using the time weighting function. By performing an element-wise multiplication of these two vectors and selecting the largest element, which is 0.1353, we can calculate the similarity score based on Equation 3.6:

$$0.1353(1+0.6)-0.6=-0.3835.$$

Consequently, the value of $H_{4,8}$ can be determined using Equation 2.2:

$$H_{4,8} = max(H_{3,7}+(-0.3835), H_{3,8}-0.4, H_{4,7}-0.4, 0)$$

$$H_{4,8} = max(1.6+(-0.3835), 1.2-0.4, 3-0.4, 0) = 2.6.$$

By calculating the entire matrix using this approach, we obtain Table 3.4.

As mentioned earlier, since the indexes are not commutative (S(A, B) $\neq$ S(B, A)), it is also necessary to calculate the H matrix with the sequences swapped. By applying the same calculation process for the H matrix, we obtain the Table 3.5.

As we can see, the first matrix, representing the similarity from A to B, has a value of 5.2, while the second matrix has a value of 4.2. This indicates that the similarity between the two sequences is 5.2. To normalize the similarity index, we divide the value by the length of the shorter segment, resulting in the following result:

$$S(A,B) = \frac{\max(5.2, 4.2)}{\min(7,9)} = 0.743.$$

By following the same approach on the dataset from February 1, 2023, as presented earlier, we conducted pattern matching for all 112 pairs of AFs. The outcomes were used to construct a

Table 3.4: Matrix H between A and B

	A	DESC	RX_L	LOFi	DFi	LOAMi	LOFi
B	0	0	0	0	0	0	0
LOSi	0	0.2	0.65	0.25	0	0	0
DESC	0	0.9	1.47	1.07	0.67	0.27	0
LOFi	0	0.5	1.6	1.2	1.61	1.21	0.98
LOAMi	0	0.6	1.2	2.5	2.1	1.7	1.4
LOFi	0	0.24	1.12	2.1	3.5	3.1	2.7
LOAMi	0	0	0.72	1.7	3.1	4.25	3.85
DFi	0	0	0.32	1.6	2.7	2.7	5.2
LOAMi	0	0	0	1.2	2.3	3.23	4.8
LOSi	0	0	0	0.8	1.9	2.83	4.4

Table 3.5: Matrix H between B and A

	B	LOSi	DESC	LOFi	LOAMi	LOFi	LOAMi	DFi	LOAMi	LOSi
A	0	0	0	0	0	0	0	0	0	0
DESC	0	0	0.87	0	0	0	0	0	0	0
RX_L	0	0	0.47	1.6	1.2	1.9	1.5	1.4	1	0.8
LOFi	0	0.3	0.07	0.68	2.55	2.15	1.75	1.35	0.6	0.4
DFi	0	0	0	0	0.86	1.56	3.35	2.95	2.6	2.2
LOAMi	0	0	0	0	0.46	1.6	2.1	3.9	3.5	3.1
LOFi	0	0.2	0	0	0	0	0	3.5	4.2	3.8

heatmap that visualizes the similarities between these AFs, depicted in Figure 3.6.

3.3 Post-processing

After calculating the similarity indexes between all pairs of AFs, it is necessary to implement methods for post-processing and analyzing this data.

Post-processing refers to additional steps or operations performed on data or results after an initial analysis or computation. Its purpose is to refine, enhance, or transform the output in order to improve its quality, usefulness, or compatibility with further processes.

In the context of AF analysis, post-processing plays a crucial role in refining the obtained similarity indexes and making them more meaningful and actionable. This involves applying techniques to reduce noise, eliminate outliers, and enhance the overall quality of the data. The goal is to extract relevant information and insights that can be used for decision-making or further analysis.

Since analyzing AFs individually would require a significant amount of time and human resources, clustering AFs based on their similarity allows for classification and study to be performed on clusters. To achieve this, a method called hierarchical clustering was utilized. This approach facilitates the grouping of AFs with similar characteristics, enabling more efficient and effective analysis of the data.

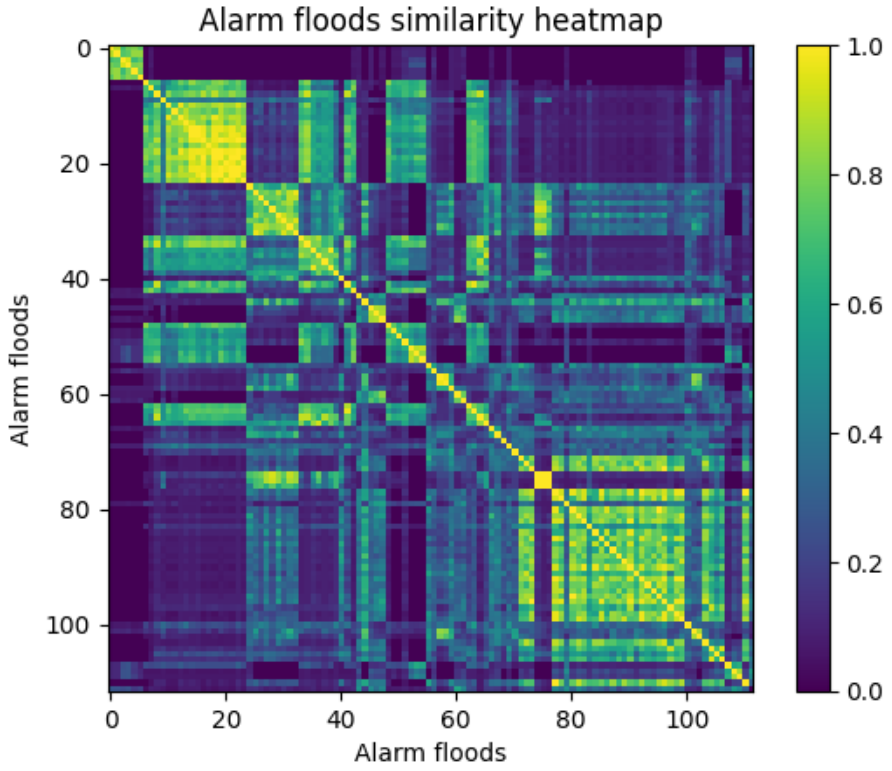

Figure 3.6: Similarity heatmap

Furthermore, after the AFs are grouped using clustering, a method was developed to characterize each cluster through an exemplar sequence. In other words, the AF that is closest to the centroid of the cluster is selected as the representative sequence, known as the archetype.

In the following sections, this topic will be further discussed in detail, providing insights into the post-processing techniques and the characterization of clusters.

3.3.1 Alarm flood clustering

AF clustering is a technique used to group similar AFs together based on their patterns, characteristics, or similarities. It is a post-processing method that follows the calculation of similarity indexes between AFs. The goal of clustering is to identify common patterns or behaviors within the AFs and to group them into meaningful clusters.

In the context of alarm management and analysis, clustering can help uncover hidden structures, relationships, and trends among the alarms. By clustering AFs, we can gain a better understanding of their underlying causes, identify recurring patterns, and classify them into distinct groups or categories.

Various clustering algorithms can be employed for AF analysis, such as hierarchical clustering, k-means clustering, or density-based clustering. These algorithms consider different measures of

Table 3.6: Hieralchical clustering - similarity matrix

	AF1	AF2	AF3	AF4	AF5	AF6
AF1	1	0.2	0.85	0.7	0.4	0.8
AF2	0.2	1	0.3	0.2	0.5	0.4
AF3	0.8	0.3	1	0.6	0.3	0.7
AF4	0.7	0.2	0.6	1	0.9	0.2
AF5	0.4	0.5	0.3	0.9	1	0.2
AF6	0.8	0.4	0.8	0.2	0.2	1

similarity or dissimilarity between AFs and group them based on their proximity or similarity in feature space.

In the specific case studied in this dissertation, where we have a pairwise similarity scores matrix, a post-processing method based on hierarchical clustering was chosen. Hierarchical clustering is a technique that groups similar data points into clusters based on their proximity or similarity. It allows for the identification of patterns or structures within the AFs, enabling a deeper understanding of their relationships and characteristics.

The method involves transforming the similarity table, which contains the similarity values between all pairs of AFs, into a distance matrix. This is done by subtracting each similarity value from 1, resulting in a distance value. The purpose of this transformation is to measure the dissimilarity or distance between AFs. This information is crucial for performing hierarchical clustering and determining the optimal clusters based on the defined distance threshold.

Initially, each AF is assigned to its own individual cluster. For example, if there are 10 AFs, the algorithm starts with 10 separate clusters. The next step is to merge the clusters that have the smallest distance between them. There are different approaches to determining the distance between clusters during the merging process:

- **Single-linkage**: The distance between the merged cluster and the remaining clusters is determined by the smallest distance between any two AFs in the merged cluster.
- **Complete-linkage**: The distance between the merged cluster and the remaining clusters is determined by the largest distance between any two AFs in the merged cluster.
- **Average-linkage**: The distance between the merged cluster and the remaining clusters is calculated as the average distance between each AF in the merged cluster and every AF in the remaining clusters.

The distance matrix is updated after each merge, and the process continues until a certain distance threshold is reached. If no distances below the threshold are found during the merging process, the cluster aggregation stops and the final clusters are determined.

To illustrate this process, let us consider an example with six AFs and a maximum distance threshold of 0.4. Using the similarity table shown in Table 3.6 as an example, we can demonstrate how the clusters are formed.

Table 3.7: Hieralchical clustering - distance matrix part 1

	AF1	AF2	AF3	AF4	AF5	AF6
AF1	0	0.8	0.15	0.3	0.6	0.2
AF2	0.8	0	0.7	0.8	0.5	0.6
AF3	0.2	0.7	0	0.4	0.7	0.3
AF4	0.3	0.8	0.4	0	0.1	0.8
AF5	0.6	0.5	0.7	0.1	0	0.8
AF6	0.2	0.6	0.2	0.8	0.8	0

Table 3.8: Hieralchical clustering - distance matrix part 2

	AF1	AF2	AF3	AF4/AF5	AF6
AF1	0	0.8	0.15	0.6	0.2
AF2	0.8	0	0.7	0.8	0.6
AF3	0.2	0.7	0	0.7	0.3
AF4/AF5	0.6	0.8	0.7	0	0.8
AF6	0.2	0.6	0.2	0.8	0

Table 3.9: Hieralchical clustering - distance matrix part 3

	AF1/AF3	AF2	AF4/AF5	AF6
AF1/AF3	0	0.8	0.7	0.3
AF2	0.8	0	0.8	0.6
AF4/AF5	0.7	0.8	0	0.8
AF6	0.2	0.6	0.8	0

Table 3.10: Hieralchical clustering - distance matrix part 4

	AF1/AF3/AF6	AF2	AF4/AF5
AF1/AF3/AF6	0	0.8	0.8
AF2	0.8	0	0.8
AF4/AF5	0.8	0.8	0

By converting the matrix into distances, we obtain Table 3.7.

The two clusters with the lowest distance are AF4 and AF5, with a distance of 0.1. For the merging of these two clusters, the complete-link method was chosen. This decision is based on the fact that in the studied context, it is important to consider the maximum distance between each element of the cluster to ensure that no new AFs are associated that are not similar to all elements of the cluster. By merging these two clusters using the complete-link method, we obtain the Table 3.8.

The table visually represents the merged cluster, showing the connection between AF4 and AF5. This merging process helps to create more cohesive clusters by grouping together AFs that exhibit high similarity.

Repeating the process, we now merge the cluster AF1 with AF3, resulting in the Table 3.9. The merging of AF1 and AF3 follows the same methodology as before, using the selected linkage

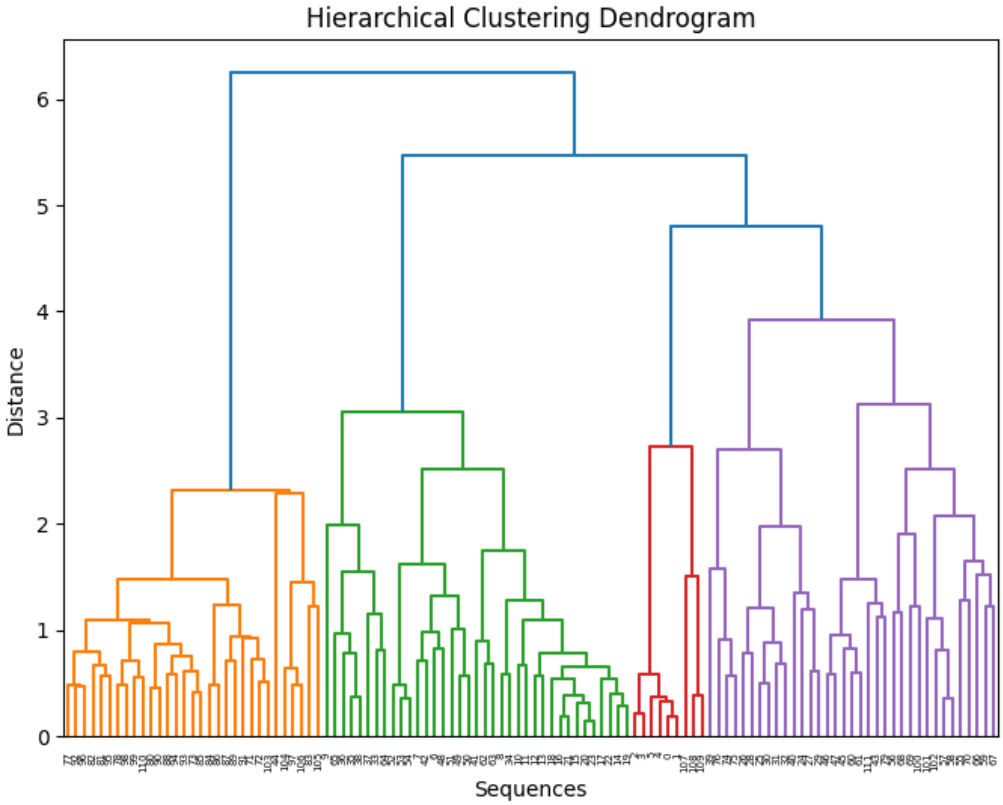

Figure 3.7: Hierarchical clustering

method to determine the distance between the merged cluster and the remaining clusters.

As there is still a similarity distance below the threshold, it is possible to add sequence AF6 to the cluster AF1/AF3. This results in the final clusters, as shown in the Table 3.10.

Once the algorithm finishes, it results in three distinct clusters, each containing a group of closely related AFs. The first cluster comprises AFs 1, 3 and 6, indicating their high similarity. The second cluster consists of AF 2, which stands alone as a distinct sequence. Finally, the last cluster combines sequences 4 and 5, highlighting their close relationship.

For a clearer understanding of the clustering analysis performed, Figure 3.7 depicts the hierarchical clustering based on the similarity heatmap shown in Figure 3.6.

This diagram illustrates the distances between all sequences, making it easier to visualize the boundaries of each cluster in the model. By using this diagram, we can better understand the groupings of sequences and their relationships within the dataset.

For the dataset under analysis, a minimum similarity threshold of 0.5 was chosen for sequences within a cluster. This resulted in the identification of 11 clusters, encompassing a total of 95 AFs.

The clustering process ensures that each cluster maintains a maximum level of similarity, with the dissimilarity between any two AFs within a cluster being below a specified threshold. This guarantees that the clusters are formed based on strong similarities, promoting the identification of coherent patterns and meaningful associations among AFs.

With the clusters defined, the next step is to select an archetype AF for each cluster. The archetype represents a representative sequence that encapsulates the essential characteristics of its respective cluster, providing a concise and informative representation of the clustered data.

3.3.2 Clusters Archetype Alarm Floods

Once the AFs are grouped into clusters, it is necessary to obtain a representative segment for each cluster. This step aims to reduce computational load. Instead of calculating the similarity index for all elements within the cluster when new data is presented for analysis, it is only calculated for the AF that represents the cluster.

This process of finding a representative segment helps streamline the analysis by reducing the number of calculations required. By using a single representative AF to characterize the cluster, the computational burden is significantly reduced. This approach enables faster and more efficient analysis of new data, as the similarity index computation is focused on the representative AF rather than the entire cluster.

There are different approaches to selecting the archetype of a cluster, depending on the clustering algorithm and the data. Some common methods include:

- **Centroid**: The centroid is the mean position of all data points in the cluster. It represents the central tendency of the cluster.
- **Medoid**: The medoid is the data point in the cluster that has the smallest average dissimilarity to all other points. It is a real data point from the cluster.
- **Exemplar**: The exemplar is a representative data point that best exemplifies the characteristics of the cluster. It can be chosen based on criteria such as proximity to the cluster center or the highest density within the cluster.

The centroid method is commonly used due to its accuracy in obtaining the reference point of the cluster. However, in the case of complex cluster elements with categorical variables, it was not possible to find a solution to exactly determine a sequence that represents the centroid of the cluster. To overcome this issue, the medoid method was chosen instead.

In the practical case under study, the medoid method selects an actual AF from the cluster rather than calculating a fictitious sequence that would be at the center of the cluster. The chosen AF is the one with the smallest distance to the other alarms in the cluster. This approach ensures that the selected AF is a real data point within the cluster and is representative of its characteristics.

To determine the medoid from the distance matrix, one can calculate the sum of distances for each row or column, which represents the dissimilarity between each AF and the others. The medoid is then identified as the row or column with the lowest sum, indicating the AF that exhibits the smallest overall dissimilarity to the remaining alarms in its cluster.

Using the AF1/AF3/AF6 cluster as an example from the clustering chapter, we have obtained the Table 3.11.

Table 3.11: Distance matrix of the cluster AF1/AF3/AF6

	AF1	AF3	AF6	Σ
AF1	0	0.15	0.2	0.35
AF3	0.15	0	0.3	0.45
AF6	0.2	0.3	0	0.5

By calculating the sum of distances for each row, we can determine the medoid of the cluster. In this case, when examining the sums of distances for each row in the distance matrix, we observe that the row corresponding to AF1 has the smallest sum. This indicates that AF1 is the AF that exhibits the least dissimilarity compared to the other AFs in the cluster.

Being the medoid of the cluster, AF1 serves as the representative sequence that best captures the overall characteristics of the AFs within the cluster. It can be considered the most typical or central sequence, as it demonstrates the closest similarity to the rest of the AFs.

Chapter 4

Results

Throughout the development of this work, the dataset covering a 24-hour period on February 1, 2023, was utilized. However, for model validation, a new dataset comprising a week's worth of alarm records, totaling over 5.5 million alarms, was made available. 70% of the alarms were used for training the model, enabling the generation of AF clusters. The remaining alarms were inputted into an evaluation model, which involved comparing the newly obtained AFs with the clusters generated during the training phase.

In this chapter, we will discuss the results obtained from the analysis of the provided data. Special attention was given to the efficiency and model speed, with the aim of determining the feasibility of applying a real-time system. The challenges and opportunities of deploying the model in a real-time setting will be carefully examined and assessed. Moreover, the insights gained from the evaluation process will shed light on the model's capability to handle large-scale datasets and provide meaningful RCA in a time-sensitive manner.

4.1 Chattering

Regardless of whether the data is used for model training or testing, it needs to undergo pre-processing. Although the chattering pre-processing step is relatively simple to implement, its efficiency in processing speed should be taken into consideration. With a dataset of this size, there is a significant processing load.

Efficient pre-processing is of paramount importance, particularly when dealing with large datasets, as it can significantly influence the overall performance and speed of subsequent analysis and modeling tasks. By streamlining the chattering detection process, computational resources are utilized more effectively, resulting in quicker and more dependable RCA outcomes. Without a fast and efficient model, applying online RCA becomes unfeasible. Therefore, optimizing the pre-processing steps is essential to enable real-time or near-real-time RCA in dynamic and complex systems.

Initially, an algorithm was created that simply took the original dataset and, for each alarm iteration, analyzed a two-minute time window to check for any instances of the same type of

alarm occurring on the same device. While this approach worked as expected, it proved to be highly inefficient as it traversed alarms that were unrelated to the filtering process.

To address this issue, the dataset was efficiently subdivided into pairs of combinations based on "alarmedresource_id" and "specificalarmabbrev," resulting in approximately four hundred thousand combinations. By applying the chattering detection method to each of these smaller subsets, we were able to drastically reduce the processing time compared to the original approach. This optimization allowed us to process around half a million alarms within 30 minutes, whereas the previous method took approximately 4 hours.

In the dataset under analysis, the chattering algorithm successfully removed approximately 2.7 million (49%) reported alarms. This outcome highlights the repetitive nature of alarm reporting, which is not very informative and does not provide new insights for problem analysis. By effectively eliminating chattering alarms, the dataset becomes more refined and focused, ensuring that the subsequent analysis and modeling tasks are based on meaningful and relevant data.

In Figure 4.1, we can observe the top pairs of reported alarms in the dataset before and after applying the chattering detection method. Pre-chattering, we can clearly see an excessive number of alarms, particularly those of type *LCGDi*. However, after the chattering process, these alarms almost entirely disappear from the dataset's top alarm pairs. This observation underscores that the alarm manager generates a considerable number of repeated alarms, especially of type *LCGDi*.

This finding also reinforces the importance of pre-processing techniques in RCA, as it helps reduce unnecessary noise and improves the overall efficiency of the analysis process. With a cleaner dataset, the subsequent steps of pattern matching and clustering can yield more accurate results, leading to a more effective RCA outcome.

Additionally, the removal of chattering alarms from the dataset has implications for the application of real-time systems. By reducing the number of redundant alarms, a real-time RCA system can operate with greater speed and precision, allowing for timely detection and resolution of critical issues in complex systems.

4.2 Alarm Flood similarity

Another fundamental aspect of pre-processing is the detection of AFs. The definition of an AF can vary depending on the system under analysis. For this particular problem, given the limited knowledge of the dataset, it was decided to implement the norm defined by ISA-18.2. Thus, an AF is considered to start when there are more than ten alarms within a 10-minute period and ends when the number of alarms within the same period drops below five. Following this approach, in the seven-day dataset, we identified 3363 AFs, comprising 542534, 23.6% of all alarms (after chattering).

After identifying all the AFs in the dataset, they were divided into two groups. As mentioned earlier, 70% of the AFs were used for the training model, while the remaining 30% were used for validation. Thus, in this dataset, 1750 AFs were used for training, and 760 for validation. The AFs selected for training were then subjected to the proposed similarity calculation between them.

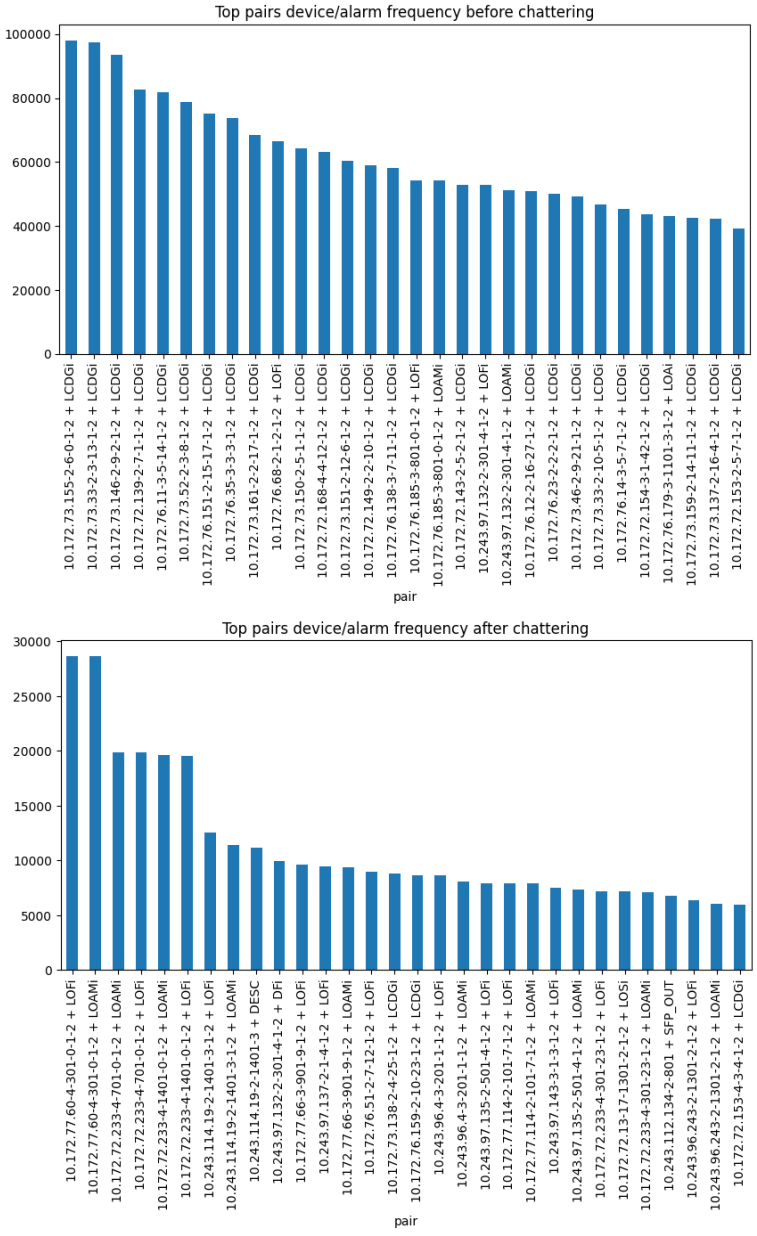

Figure 4.1: Alarm pairs difference with chattering

This partitioning of the dataset into training and validation sets allows for a robust evaluation of the model's performance. By using a significant portion of the data for training, the model can learn patterns and similarities within the AFs, improving its ability to generalize and identify root causes in unseen data.

The only parameter that influences the computation time of each similarity is the length of the AF sequence. Since longer sequences require more processing time, and it is not very common for AFs to exceed 30 alarms, all AFs with a size greater than 60 alarms were excluded from the similarity calculation. This decision aimed to expedite the training process without compromising

the selected sample's value.

By removing the longer AFs from the similarity calculations, we can significantly reduce the computational overhead during training while still capturing relevant patterns and behaviors. This optimization ensures a more efficient and streamlined analysis, allowing us to focus on the most representative and informative AFs within the dataset.

Additionally, this approach reflects a practical consideration, as very long AFs may not be as common or relevant to the specific analysis at hand. By focusing on a subset of AFs that are more representative of typical scenarios, we can obtain more accurate and actionable results from the subsequent data mining techniques.

Given the substantial number of AFs in the dataset, calculating the similarity index for all pairs would be time-consuming and result in unnecessary comparisons between dissimilar AFs. To optimize the process, a divide and conquer strategy was implemented, dividing the AFs based on the number of common alarm types they share. This approach enables the formation of initial clusters that are less refined but can be processed rapidly. Subsequently, each of these clusters is further subdivided into more detailed clusters, allowing for faster and more focused analysis.

This division and clustering strategy efficiently groups similar AFs together, reducing the time and computational resources required for similarity analysis. By first creating smaller clusters and then refining them, the model can quickly identify patterns and similarities between AFs, which is crucial for effective RCA in fiber optic networks. For this particular system, a threshold of at least three common alarm types was chosen. This value was carefully considered to maximize the separation of more distant AFs while ensuring that highly similar sequences were not split apart.

The main issue with this approach is that the relationship between the number of alarms and computational load is quadratic. This means that if the number of AFs doubles, the number of similarity calculation iterations will quadruple, resulting in four times more time consumption. Consequently, this approach lacks efficiency when dealing with the scalability of the training dataset. To address or at least minimize this inherent algorithmic nature, optimizing the similarity calculation and implementing a multi-threaded algorithm that allows multiple iterations to run simultaneously during model training would help alleviate processing time.

4.3 Clustering and archtype

A visual representation of the AFs' distances within the training model can be observed in Figure 4.2.

The application of hierarchical clustering to this model proves to be a highly effective method for identifying distinct clusters of AFs. It efficiently groups similar AFs together, enabling them to be analyzed collectively. By employing the complete-link method to merge individual clusters into larger ones, we ensure that all AFs within the same cluster are never more than 0.4 index distance apart, as defined by the distance threshold.

In this research, the chosen threshold of 0.4 was deemed suitable, as it strikes a balance between creating a sufficient number of sequences within each cluster while maintaining granularity.

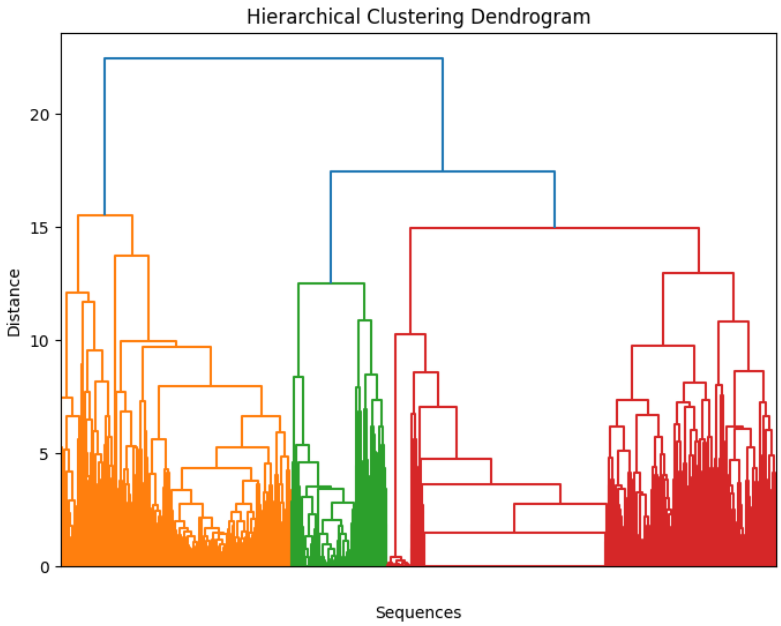

Figure 4.2: Hieralchical clustering of the 7 days

A higher threshold would result in larger clusters, but they would be less refined. Conversely, a lower threshold would yield more detailed clusters, but a considerable number of AFs might not fit into any cluster, leading to data loss. The selected threshold provides a meaningful trade-off between cluster size and precision, facilitating accurate and insightful RCA.

With this approach, a total of 25 clusters were obtained. However, clusters with less than 10 AFs were removed from the analysis, as they represent a very small sample size and might lead to erroneous conclusions for network operators. After applying this filter, the remaining clusters were found to be mostly composed of AFs with more than 30 alarms, indicating the detection of numerous similar sequences in the dataset. Only 904 (26.9%) AFs were not associated with any final cluster, suggesting that a majority of the system's issues can now be understood more efficiently, as the most common problems have been identified.

As previously mentioned, the archetype AF within each cluster represents the closest similarity to all other AFs within that cluster. By reducing all the AFs in a cluster to a single archetype, the analysis for network operators becomes significantly faster and more straightforward.

Following a detailed analysis of each AF, the 25 archetypes were selected. It was observed that each archetype is distinct from the others, suggesting the identification of various types of problems that may have different root causes. This outcome underscores the model's ability to uncover diverse issues in the system and provides valuable insights for resolving and preventing them effectively. The identified archetypes serve as representative examples, facilitating quicker problem diagnosis and resolution for network operators.

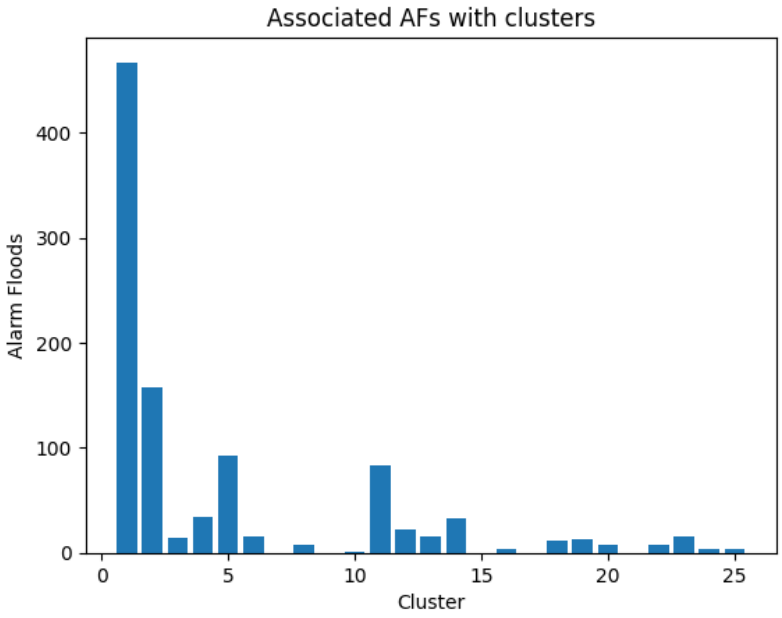

Figure 4.3: Association of the new data with the clusters

4.4 Test model

After selecting all the archetype AFs from the training clusters, the next step involves analyzing the AFs from the validation dataset. These new sequences are subjected to the proposed similarity algorithm, but instead of comparing each AF with all others, each one is compared with all the archetypes generated during training.

Upon calculating the similarities, we found a total of 542 AFs from the test dataset that had a similarity index of 0.6 or higher with at least one of the clusters, representing 71.3% of the AFs associated with a known cluster.

This result indicates that a significant portion of the new dataset contains alarm sequences previously identified and represented by the archetypes. Consequently, these sequences can be considered as known AFs, enabling the network operator to focus more on identifying the root causes of the remaining AFs. By recognizing familiar AFs, the operator can allocate resources efficiently and prioritize the analysis of unfamiliar alarm patterns, contributing to quicker and more effective root cause identification.

Figure 4.3 illustrates the distribution of AFs across different clusters. Notably, cluster 1 exhibits a significantly higher number of AFs compared to other clusters, suggesting that a substantial portion of the detected issues share a common root cause.

Chapter 5

Conclusions and Future Work

By utilizing the modified Smith-Waterman Algorithm, this research aimed to enhance the accuracy and effectiveness of pattern matching in AF sequences. The algorithm's ability to handle local alignments and its sensitivity to gaps and mismatches make it well-suited for capturing intricate patterns and subtle variations in alarm occurrences.

Through the implementation, this dissertation explored the potential of utilizing advanced SA techniques in the context of alarm management. By applying this algorithm to AF sequences, researchers can uncover meaningful patterns, clusters, and trends that may go unnoticed using traditional approaches.

Moreover, the study extended beyond theoretical exploration and delved into the practical implementation studied algorithm. The algorithm has been adapted to handle the specific characteristics and requirements of alarm data, considering factors such as alarm timing, and contextual information. This ensures the algorithm's relevance and effectiveness in the AF analysis domain.

By leveraging the Smith-Waterman Algorithm and its adaptations, this dissertation contributed to the advancement of pattern matching techniques for AF sequences. The results obtained through this research endeavor hold the potential to enhance alarm management strategies, improve fault detection and diagnosis, and ultimately optimize system performance and reliability.

5.1 Future Work

In the context of future work, there are several key tasks to enhance the developed system. One of the primary objectives is to create an online AFs similarity model capable of efficiently identifying correlations between new AFs and previously trained clusters. This real-time or near-real-time model would automatically enhance the AFs, enabling network operators to dedicate more time to investigating unknown sequences of alarms.

To further optimize the system, it is crucial to improve the distance calculation between AFs, reducing the processing time required for training new AFs. This enhancement would significantly increase the system's capacity to handle larger volumes of data, ensuring scalability and robustness.

Additionally, the implementation of multi-threading could significantly boost processing speed, further streamlining the entire analysis process. This improvement would lead to faster results and more agile decision-making, enhancing the overall efficiency of the RCA.

Furthermore, exploring advanced machine learning techniques and algorithms could potentially lead to more accurate and precise clustering of AFs, enabling deeper insights into the underlying patterns and correlations. Embracing the latest advancements in data mining and artificial intelligence would undoubtedly strengthen the system's capability to handle complex scenarios and deliver more reliable results.

Finally, conducting extensive real-world testing and validation on different datasets and network environments would be crucial to ensure the model's generalizability and effectiveness in diverse operational settings. By continuously refining and updating the model based on new data and insights, organizations can maintain a proactive approach to RCA and continuously improve the reliability and resilience of their systems.

References

[1] Raúl Costa, Nuno Cachulo, and Paulo Cortez. An intelligent alarm management system for large-scale telecommunication companies. pages 386–399, 10 2009. doi:10.1007/978-3-642-04686-5_32.

[2] Abiola Salau, Chika Yinka-Banjo, Sanjay Misra, Adewole Adewumi, Ravin Ahuja, and Rytis Maskeliunas. *Design and Implementation of a Fault Management System*, pages 495–505. 05 2019. doi:10.1007/978-3-030-16681-6_49.

[3] Deep Medhi and Jane Zupan. An alarm management approach in the management of multi-layered networks great plains environment for network innovation (gpeni) view project 5g nr-unlicensed view project an alarm management approach in the management of multi-layered networks. 2003. URL: https://www.researchgate.net/publication/4047822, doi:10.1109/IPOM.2003.1251227.

[4] Shiqi Lai, Fan Yang, Tongwen Chen, and Liang Cao. Accelerated multiple alarm flood sequence alignment for abnormality pattern mining. *Journal of Process Control*, 82:44–57, 10 2019. doi:10.1016/J.JPROCONT.2019.06.004.

[5] Wenkai Hu, Jiandong Wang, and Tongwen Chen. A local alignment approach to similarity analysis of industrial alarm flood sequences. *Control Engineering Practice*, 55:13–25, 10 2016. doi:10.1016/J.CONENGPRAC.2016.05.021.

[6] Matthieu Lucke, Moncef Chioua, Chriss Grimholt, Martin Hollender, and Nina F. Thornhill. Advances in alarm data analysis with a practical application to online alarm flood classification. *Journal of Process Control*, 79:56–71, 7 2019. doi:10.1016/J.JPROCONT.2019.04.010.

[7] Fa Zhang, Xiang-Zhen Qiao, and Zhi-Yong Liu. A parallel smith-waterman algorithm based on divide and conquer. In *Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002. Proceedings.*, pages 162–169, 2002. doi:10.1109/ICAPP.2002.1173568.

[8] Marco Wiltgen. Algorithms for structure comparison and analysis: Homology modelling of proteins. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, 1-3:38–61, 1 2018. doi:10.1016/B978-0-12-809633-8.20484-6.

[9] Cen Guo, Wenkai Hu, Shiqi Lai, Fan Yang, and Tongwen Chen. An accelerated alignment method for analyzing time sequences of industrial alarm floods. *Journal of Process Control*, 57:102–115, 9 2017. doi:10.1016/J.JPROCONT.2017.06.019.

[10] Boyuan Zhou, Wenkai Hu, Kevin Brown, and Tongwen Chen. Generalized pattern matching of industrial alarm flood sequences via word processing and sequence alignment. *IEEE*

Transactions on Industrial Electronics, 68(10):10171–10179, 2021. doi:10.1109/TIE.2020.3026287.

[11] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

[12] Pavel Senin. Dynamic time warping algorithm review. 2008. URL: https://api.semanticscholar.org/CorpusID:16629907.

[13] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11:561–580, 2007.

[14] William R. Pearson. Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the smith-waterman and fasta algorithms. *Genomics*, 11:635–650, 11 1991. doi:10.1016/0888-7543(91)90071-L.

[15] Lukasz Ligowski and Witold Rudnicki. An efficient implementation of smith waterman algorithm on gpu using cuda, for massively parallel scanning of sequence databases. In *2009 IEEE International Symposium on Parallel Distributed Processing*, pages 1–8, 2009. doi:10.1109/IPDPS.2009.5160931.

[16] Ali Khajeh-Saeed, Stephen Poole, and J. Blair Perot. Acceleration of the smith–waterman algorithm using single and multiple graphics processors. *Journal of Computational Physics*, 229:4247–4258, 6 2010. doi:10.1016/J.JCP.2010.02.009.

[17] Wenkai Hu, Jiandong Wang, and Tongwen Chen. Fast sequence alignment for comparing industrial alarm floods. *IFAC-PapersOnLine*, 48:647–652, 1 2015. doi:10.1016/J.IFACOL.2015.09.041.

[18] Yifan Xu, Wen Tan, and Tingshun Li. An alarm flood pattern matching algorithm based on modified blast with leveshtein distance. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–6, 2016. doi:10.1109/ICARCV.2016.7838817.

[19] Shiqi Lai, Fan Yang, and Tongwen Chen. Online pattern matching and prediction of incoming alarm floods. *Journal of Process Control*, 56:69–78, 8 2017. doi:10.1016/J.JPROCONT.2017.01.003.

[20] Md Rezwan Parvez, Wenkai Hu, and Tongwen Chen. Real-time pattern matching and ranking for early prediction of industrial alarm floods. *Control Engineering Practice*, 120:105004, 3 2022. doi:10.1016/J.CONENGPRAC.2021.105004.

[21] Shiqi Lai and Tongwen Chen. Methodology and application of pattern mining in multiple alarm flood sequences. *IFAC-PapersOnLine*, 48:657–662, 1 2015. doi:10.1016/J.IFACOL.2015.09.043.

[22] Christian Borgelt. An implementation of the fp-growth algorithm. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, OSDM '05, page 1–5, New York, NY, USA, 2005. Association for Computing Machinery. URL: https://doi.org/10.1145/1133905.1133907, doi:10.1145/1133905.1133907.

[23] Yi Zeng, Shiqun Yin, Jiangyue Liu, and Miao Zhang. Research of improved fp-growth algorithm in association rules mining. *Scientific Programming*, 2015, 2015. URL: https://www.hindawi.com/journals/sp/2015/910281/, doi:10.1155/2015/910281.

[24] Wenkai Hu, Zhuang Wang, and Jiandong Wang. A priority-aware sequential pattern mining method for detection of compact patterns from alarm floods. *Journal of Process Control*, 129:103041, 9 2023. doi:10.1016/J.JPROCONT.2023.103041.

[25] Santhosh Kumar, BSanthosh Kumar, and abstract. Implementation of web usage mining using apriori and fp growth algorithms. 2009. URL: https://www.researchgate.net/publication/266012367.

[26] Shivam Sidhu, Upendra Kumar Meena, Aditya Nawani, Himanshu Gupta, and Narina Thakur. Fp growth algorithm implementation. *International Journal of Computer Applications*, 93:975–8887, 2014.

[27] Mahmoud M Al-Quzwini. Design and implementation of a fiber to the home ftth access network based on gpon. *International Journal of Computer Applications*, 92:975–8887, 2014.

[28] Josep Segarra, Vicent Sales, and Josep Prat. Planning and designing ftth networks: Elements, tools and practical issues. In *2012 14th International Conference on Transparent Optical Networks (ICTON)*, pages 1–6, 2012. doi:10.1109/ICTON.2012.6254486.

[29] Dae Sun Kim, Hiroyuki Shinbo, and Hidetoshi Yokota. An alarm correlation algorithm for network management based on root cause analysis. In *13th International Conference on Advanced Communication Technology (ICACT2011)*, pages 1233–1238, 2011.

[30] Mika Klemettinen. A knowledge discovery methodology for telecommunication network alarm databases. 1999. URL: https://www.researchgate.net/publication/2303503.

[31] Shengnan Zhang, Yan Hu, and Guangrong Bian. Research on string similarity algorithm based on levenshtein distance. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 2247–2251, 2017. doi:10.1109/IAEAC.2017.8054419.

[32] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095, 2007. doi:10.1109/TPAMI.2007.1078.

[33] Sujoy Bag, Sri Krishna Kumar, and Manoj Kumar Tiwari. An efficient recommendation generation using relevant jaccard similarity. *Information Sciences*, 483:53–64, 2019. URL: https://www.sciencedirect.com/science/article/pii/S0020025519300325, doi:https://doi.org/10.1016/j.ins.2019.01.023.

[34] Hangyu Yan and Yan Tang. Collaborative filtering based on gaussian mixture model and improved jaccard similarity. *IEEE Access*, 7:118690–118701, 2019. doi:10.1109/ACCESS.2019.2936630.

[35] Maroš Čavojský, Martin Drozda, and Zoltán Balogh. Analysis and experimental evaluation of the needleman-wunsch algorithm for trajectory comparison. *Expert Systems with Applications*, 165:114068, 2021. URL: https:

//www.sciencedirect.com/science/article/pii/S0957417420308307, doi:https://doi.org/10.1016/j.eswa.2020.114068.

[36] Emma Aspland, Paul R. Harper, Daniel Gartner, Philip Webb, and Peter Barrett-Lee. Modified needleman–wunsch algorithm for clinical pathway clustering. *Journal of Biomedical Informatics*, 115:103668, 2021. URL: https://www.sciencedirect.com/science/article/pii/S1532046420302963, doi:https://doi.org/10.1016/j.jbi.2020.103668.

[37] Amr Ezz El-Din Rashed, Hanan M. Amer, Mervat El-Seddek, and Hossam El-Din Moustafa. Sequence alignment using machine learning-based needleman–wunsch algorithm. *IEEE Access*, 9:109522–109535, 2021. doi:10.1109/ACCESS.2021.3100408.

[38] Jiandong Wang and Tongwen Chen. An online method for detection and reduction of chattering alarms due to oscillation. *Computers Chemical Engineering*, 54:140–150, 7 2013. doi:10.1016/J.COMPCHEMENG.2013.03.025.

[39] Elham Naghoosi, Iman Izadi, and Tongwen Chen. Estimation of alarm chattering. *Journal of Process Control*, 21:1243–1249, 10 2011. doi:10.1016/J.JPROCONT.2011.07.015.

[40] Understanding and applying the ansi/ isa 18.2 alarm management standard. 2010. URL: http://www.iec.ch/cgi-bin/procgi.pl/www/iecwww.p?wwwlang=e&wwwprog=pro-det.p&progdb=db1&He=IEC&Pu=62682&Pa=&Se=&Am=&Fr=&TR=&Ed=1.0.

[41] Jiandong Wang and Tongwen Chen. An online method to remove chattering and repeating alarms based on alarm durations and intervals. *Computers Chemical Engineering*, 67:43–52, 8 2014. doi:10.1016/J.COMPCHEMENG.2014.03.018.

[42] Todd Stauffer, Marketing Manager, Exida Sellersville, Nicholas Pa, Sands, and Donald Dunn. Alarm management and isa-18 – a journey, not a destination. 01 2010.

[43] Alexander Pinkham. Effective alarm management requires planning, maintenance: Isa 18.2 specification offers a roadmap to successful implementation. *Plant Engineering*, 66:39–42, 5 2012. URL: https://go.gale.com/ps/i.do?p=AONE&sw=w&issn=0032082X&v=2.1&it=r&id=GALE%7CA342768185&sid=googleScholar&linkaccess=fulltexthttps://go.gale.com/ps/i.do?p=AONE&sw=w&issn=0032082X&v=2.1&it=r&id=GALE%7CA342768185&sid=googleScholar&linkaccess=abs.