Development of Data-Driven Methods for Alarm Flood Monitoring and Analysis

by

Md Rezwan Parvez

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Control Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Alarm floods present substantial challenges to industrial process safety, given their diverse causes and potential for severe consequences. Modern process industries involve sophisticated networks of devices that are interconnected in both upstream and downstream directions. The interconnected nature of these units and devices is a result of complex industrial production processes and the necessity to effectively manage numerous variables. Therefore, when a fault or abnormal condition occurs in an upstream or downstream component, it can lead to fault propagation due to the interconnected nature of the units and the feedback mechanisms that exist to control and regulate the overall system. Thus, this phenomenon leads to an increased number of alarms being generated across the system, causing an alarm flood. As industrial processes become more complex, plant operators often find it increasingly difficult to respond effectively, particularly during an alarm flood. The increased rate of alarms during an alarm flood overwhelms plant operators, resulting in delayed response times and further deterioration of the situation. Consequently, decision supports in alarm flood situations are in great demand to assist plant operators in assessing the root causes and taking corrective actions in time. Therefore, this thesis focuses on developing data-driven methods to efficiently manage alarm flood situations and minimize their impacts.

Three research topics are considered. Firstly, early prediction of an incoming alarm flood sequence can provide valuable information to industrial operators, facilitating them to take corrective actions in time. A real-time pattern matching and ranking approach is proposed in this work to conduct similarity analysis under an online alarm flood situation and to export the results as a ranking list of historical alarm flood sequences. Unit and set-based pre-matching mechanisms are proposed to remove irrelevant sequences, and a set-based indexing and extension strategy is applied to further avoid unnecessary computation. Real-time decision supports in the form of ranking of similar historical alarm flood sequences are presented to the industrial operators.

Secondly, a novel association rule mining approach is proposed for real-time prediction of alarm events during an alarm flood situation. This approach integrates a modified compact prediction tree model with new features, namely, the timetable and co-occurrence matrix, and is constructed based on historical alarm sequences. An alarm relevancy detection strategy is designed to identify and eliminate irrelevant alarms from the ongoing alarm flood. Furthermore, the proposed approach provides confidence intervals of the time differences between the subsequent predicted alarm events for time prediction. Such real-time assistance during alarm flood situations can greatly simplify the decision-making process for industrial operators.

Finally, a reinforcement learning (RL) approach is proposed for early prediction of industrial alarm floods and to provide real-time guidance to plant operators in prompt mitigation of such situations. Based on various association rule metrics, irrelevant alarms are identified and eliminated to avoid inaccurate recommendations. A sequence reconstruction strategy is adopted to generate potential online scenarios by exploiting the alarm relations that exist in the historical sequences. Additionally, several criteria are introduced and implemented in the existing historical sequences to reformulate the training set for improved learning. To ensure accuracy and early recommendations, a double deep Q-network (DDQN) algorithm is incorporated into the proposed method.

The effectiveness of these proposed methods is demonstrated by industrial case studies based on real industrial data from an oil refinery plant. By adopting these proposed approaches, plant operators could handle alarm floods proactively, resulting in an improvement in operational efficiency and safety.

Preface

- Chapter 2 has been published as: Md. Rezwan Parvez, Wenkai Hu, and Tongwen Chen, Comparison of the Smith–Waterman and Needleman–Wunsch algorithms for online similarity analysis of industrial alarm floods, *IEEE Electric Power and Energy Conference (EPEC)*, Edmonton, AB, Canada, Nov. 2020, pp. 1–6.

- Chapter 3 has been published as: Md. Rezwan Parvez, Wenkai Hu, and Tongwen Chen, Real-time pattern matching and ranking for early prediction of industrial alarm floods, *Control Engineering Practice*, vol. 120, Art. no. 105004, Mar. 2022.

- Chapter 4 has been published as: Md. Rezwan Parvez, Wenkai Hu, and Tongwen Chen, An association rule mining approach to predict alarm events in industrial alarm floods, *Control Engineering Practice*, vol. 138, Art. no. 105617, Jul. 2023.

- Chapter 5 has been submitted for publication as: Md. Rezwan Parvez, Mohammad Hossein Roohi, and Fangwei Xu, A reinforcement learning approach for early prediction of industrial alarm floods, *ISA Transactions.*

Acknowledgements

I would like to extend my sincere gratitude to the individuals who have provided invaluable assistance and unwavering support throughout my Ph.D. program. A special acknowledgment goes to my supervisor, Professor Tongwen Chen, whose continuous encouragement, guidance, and support have been instrumental to my journey. Prof. Chen has not only granted me valuable opportunities but also allowed me the freedom to explore interesting research topics and engage in industrial collaborations for practical experience. The years spent under his supervision are unforgettable and irreplaceable in my life. I would also like to thank my committee members, Drs. Qing Zhao and Mahdi Tavakoli Afshari, for their precious time and valuable feedback. Furthermore, I am deeply thankful to those who have contributed to the collaborative research and discussions, with a special mention of Drs. Wenkai Hu (China University of Geosciences, China), Mohammad Hossein Roohi (Isfahan University of Technology, Iran), and Fangwei Xu (Suncor Energy Inc., Canada).

I also wish to acknowledge the financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) and industrial partners, including Enbridge, Suncor, and Syncrude. Moreover, I would like to thank all current and previous members of the research group, especially Jun Shang, Boyuan Zhou, Jing Zhou, Harikrishna Rao Mohan Rao, and Ziyi Guo, for their kindness, encouragement, and help.

Lastly, I would like to express my deepest gratitude to my family members and friends for their unwavering support and companionship, both emotionally and physically, throughout the years. I am eternally grateful to all individuals, whether I have named them or not, for their support and kindness.

Contents

List of Tables

List of Figures

List of Symbols

$\mathbb{N}^+$	Natural Numbers (Positive Integers)
H	Index Matrix
$S(A, B)$	Similarity Index Between Sequences A and B
$a_i^m = (e_i^m, t_i^m, u_i^m)$	Description of Alarm
$x_i^m(t)$	Alarm Binary Signal
$\mathbb{C}(A \to B)$	Confidence of an Association Rule $(A \to B)$
$\mathbb{I}(A \to B)$	Interest of an Association Rule $(A \to B)$
δ	Gap Penalty
ϕ	Match Score
μ	Mismatch Score
$\mathbb{U}$	Set of Plant Units
$\mathbb{H}$	Historical Database
W	Weighting Matrix
$\mathbb{R}$	Ranking List
C_M	Co-occurrence Matrix
$\aleph$	Prefix Length
Ψ_n	Pattern Index
B_λ	Varying Alarm Set
γ	Discounting Factor
$\mathbb{D}$	Training Set
$V(s)$	Value of State s
$Q(s, a)$	Action Value of State s with Action a

List of Acronyms

A&E	Alarm & Event
CI	Confidence Interval
CPT	Compact Prediction Tree
BPCS	Basic Process Control System
DDQN	Double Deep Q-Network
ESD	Emergency Shutdown
FSC	Frequent Subsequence Compression
HMI	Human-Machine Interface
MDP	Markov Decision Processes
NWA	Needleman-Wunsch Algorithm
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
SBC	Simple Branches Compression
SGD	Stochastic Gradient Descent
SIS	Safety Instrumented System
SWA	Smith-Waterman Algorithm

Chapter 1

Introduction

This section presents an overview of the research background related to alarm flood monitoring and provides a literature review that summarizes the recent advancements in alarm management, alarm flood monitoring, and analysis. Furthermore, the contributions of the thesis are outlined, followed by an overview of the thesis structure.

1.1 Research Background

Alarm systems play a critical role in ensuring the safety and efficiency of modern industrial plants, including oil refineries, petrochemical facilities, and power plants [63]. An alarm system [30] can be defined as a set of interconnected devices and components designed to detect and signal the occurrence of specific events or conditions. These events or conditions can include potential hazards, equipment failures, security breaches, or any other situations that require immediate attention or action. In the context of the process industry, alarms are triggered to notify plant operators of potential deviations of process variables from normal operating conditions or predetermined thresholds [30]. However, most industrial alarm systems exhibit inadequate performance, primarily due to the overwhelming number of alarms that exceed the operators' capacity to manage efficiently in control rooms. Such issues can arise from nuisance alarms, consequential alarms, incorrect alarm configurations, etc. Furthermore, in process industries where upstream and downstream devices are physically interconnected, abnormal situations can propagate, leading to alarm overloading, known as alarm flood [63].

Alarm floods are among the most difficult issues in industrial alarm management, as the causes of alarm floods are various and the consequences could be enormous. According to [30] an alarm flood is present when more than 10 alarms occur within a 10-minute time period per operator. Alarm floods could be caused by many factors, e.g., fault propagation, improper alarm system design, and operating state conditions. In practice, alarm floods should be limited to less than 1% of the total time period that an industrial alarm system is in operation [14]. In a practical scenario, alarm floods are too common in industrial alarm systems, where the industrial operators become overwhelmed with too many alarms and lose focus from the critical alarms that lead to major process shutdowns, equipment mulfunctions, etc. In such situations, industrial operators often need contextual guidance to proactively resolve the alarm flood situations in order to prevent major process upsets, equipment mulfunctions, etc.

1.2 Alarm Systems and Alarm Management

Alarm systems [44] serve as instruments to identify near misses, which are characterized as deviations from standard operational ranges for process variables followed by their subsequent return. A well-functioning alarm system helps the operator to address potentially dangerous situations proactively before the Emergency Shutdown (ESD) system is forced to intervene [14]. This improves plant availability and increases plant safety.

The primary function of an alarm system is to notify operators about abnormal process conditions or equipment malfunctions and facilitate their response [6, 50]. An alarm is triggered when a process measurement [14] violates a predefined threshold, indicating an undesirable or potentially unsafe situation. Fig. 1.1 illustrates how alarm and response data flow through the alarm system. Various types of sensors are deployed to monitor the industrial

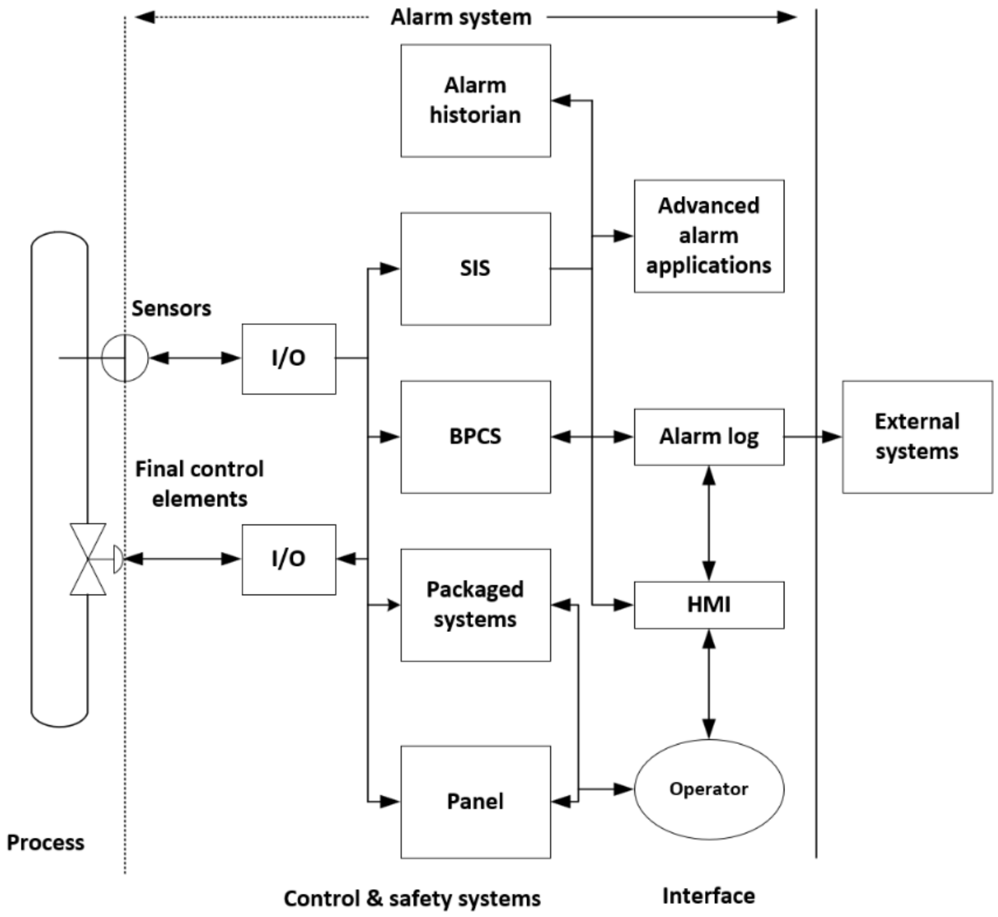


Figure 1.1: Alarm system dataflow [30].

process and collect data, which serves as input to the alarm system. Within an alarm system, the safety instrumented system (SIS), basic process control system (BPCS), and packaged systems play critical roles. The SIS ensures process safety by triggering emergency shutdowns during critical situations and typically generates safety-related alarms. The BPCS monitors process variables such as temperature and pressure and generates alarms when such variables deviate from normal operating conditions. In addition, package systems include specific stand-alone subsystems such as pumps and compressors and generate alarms if specific faults occur within the subsystems.

Human-Machine Interface (HMI) facilitates operator access to detailed alarm information and allows them to take corrective actions by interacting

Table 1.1: Alarm system performance survey [50]

Performance Measurement	**EEMUA**	**Oil & Gas**	**Petrochemical**	**Power**
Average alarms per hour	6	36	54	48
Average standing alarms	9	50	100	65
Peak alarms per hour	60	1320	1080	2100
Priority distribution % (low/med/high)	80/15/5	25/40/35	25/40/35	25/40/35

with the control system. HMI also offers comprehensive process visualization and the options to efficiently manage an alarm management system across the entire process. In contrast, a panel within the alarm system is a localized version of HMI, enabling operators to view and access information for a specific area. Real-time alarm events, including annunciation time, alarm tags, areas, and descriptions, are recorded in the alarm log. The alarm system also includes an alarm historian to store historical alarm data for future operator reference. Furthermore, an alarm system also includes advanced alarm applications and external systems to improve the overall efficiency of the alarm system.

Table 1.1 provides an overview of alarm system performance across diverse industries, using various performance metrics. This performance analysis is based on an investigation that involved 39 industrial plants ranging from oil and gas to petrochemical, power, and other industries. In Table 1.1, performance of different industries in various metrics is compared with the corresponding benchmarks in the guideline EEMUA-191 [14]. This statistic clearly shows notable differences between the performance of various industries and the industry benchmark. This indicates a substantial need for improvements in alarm systems across different industries to comply with industry benchmarks.

Many industries are subject to regulatory standards and guidelines that define criteria for effective alarm management. Understanding the alarm management lifecycle [30], as shown in Fig. 1.2, enables organizations to comply with these standards, avoiding legal issues. The alarm management lifecycle refers to the process of designing, maintaining, and continuously improving the alarm system within an industrial process facility. This lifecycle involves various stages and activities to ensure that alarms effectively serve their intended purpose of notifying operators about abnormal situations and facilitating appropriate responses.

Initially, the alarm philoshopy stage includes providing clear guidelines and principles for the design, implementation, and ongoing management of alarm systems in industrial processes. In the identification stage, all alarms are categorized, and relevant information, such as alarm set points and objectives, is documented for the use of subsequent stages. The alarm rationalization stage includes evaluating and prioritizing alarms based on the consequences, allowable response time, and the guidlines set in the alarm philosophy. This stage also ensures that only the essential alarms remain to reduce operator overload. Furthermore, the detailed design and implementation stage involves creating precise technical specifications [14] for alarms and accurately integrating them into the control system in accordance with the alarm philosophy. The core functions of the operation and maintenance phases are to guarantee the effectiveness and dependability of the alarm system and to confirm that it operates according to its design through regular maintenance activities. Finally, the management of change and audit stages propose changes in alignment with prior stages, assess the effectiveness of the alarm management process, and uphold the integrity of the alarm system.

A well-designed alarm system helps operators promptly identify and resolve process deviations, thereby reducing downtime and production setbacks. However, inadequate performance in industrial alarm systems, characterized by excessive alarms causing alarm overload [63], is attributed to issues like nuisance alarms, misconfigured variables, and abnormality propagation. Chattering alarms are the most commonly encountered nuisance alarms and may account for 10% - 60% of alarm occurrences [22]. According to the industrial standard ANSI/ISA-18.2 [30], a chattering alarm refers to a specific type of alarm that rapidly switches between active and inactive states within a short

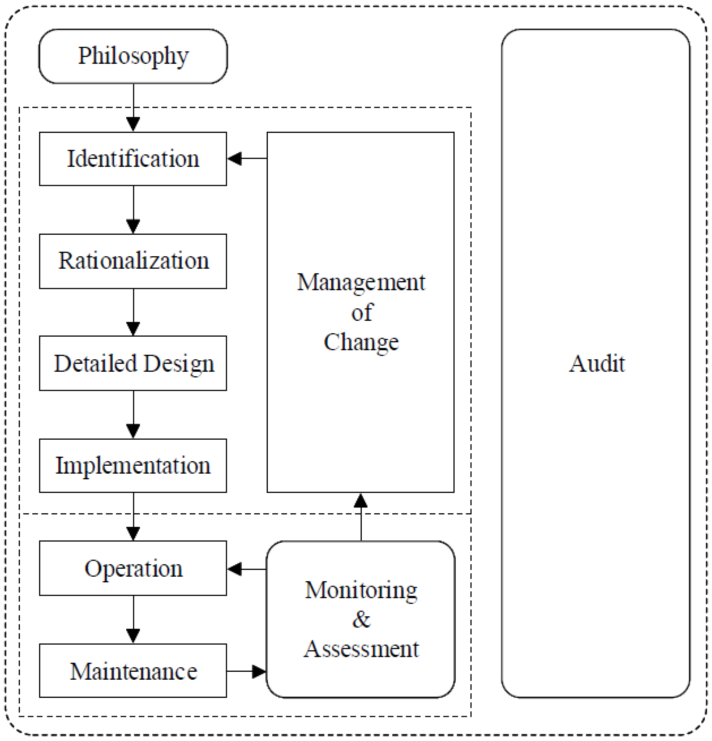


Figure 1.2: Alarm management lifecycle [30].

period of time, typically more than three times in a one-minute interval.

Additional types of nuisance alarms include stale alarms, repeating alarms, and fleeting alarms, all of which contribute to the deterioration of alarm system performance. Many of the stale alarms are nuisance alarms and remain in the alarm state for an extended period of time, such as more than 24 hours. [30]. Stale alarms that are not nuisance alarms need to be analyzed properly by operators to be aware of the ongoing situation [59]. Fleeting alarms are similar to chattering alarms but do not repetitively appear within a short timeframe [14]. Repeating alarms, on the other hand, cycle between annunciating and returning to the normal state over time [50].

Nuisance alarms often appear due to improper alarm trip points, sensor malfunctioning, control loop oscillations, and transient process conditions. In addition, correlated alarms and consequential alarms, if not logically designed, can lead to false alarms and alarm overload situations. Implementing measures such as appropriately justifying correlation logic and consolidating alarms can mitigate alarm overload and improve the operational efficiency of the operators.

In addition, alarm flood is a critical challenge for the modern industrial alarm system. According to [30] and [14], an alarm flood is known as a condition where the alarm rate is higher than what the operator can effectively manage, and the benchmark threshold is 10 alarms per 10 minutes for each operator. In an alarm flood situation, the plant operator is usually overwhelmed by a high number of alarms and may fail to identify the critical alarms and take the correct responses in time. The consequences include deterioration of the ongoing situation, equipment breakdowns, process malfunctions, and major plant shutdowns [27]. To address this, data-driven methods can be exploited to improve the alarm system's performance and provide real-time decision support to industrial operators during alarm floods.

1.3 Literature Survey

The thesis focuses on developing data-driven methods to provide real-time assistance in the form of predicting the incoming alarm flood sequence or the upcoming alarm events in alarm flood situations. Such assistance is intended to ease off the decision-making process for the plant operators during an ongoing alarm flood. Furthermore, data mining techniques are exploited to eliminate nuisance alarms, correlated alarms, and consequential alarms, thereby improving the accuracy of such real-time recommendations. In this section, a comprehensive literature survey on the recent developments of such methods is presented.

1.3.1 Recent Advances in the Methods to Reduce Nuisance Alarms, Correlated Alarms, and Consequential Alarms

Nuisance alarms present significant challenges to the optimal performance of alarm systems. Furthermore, the improper configuration of correlated and consequential alarms can negatively impact the effectiveness of an alarm system. Also, long-standing and mode-based alarms contribute to operator overload, affecting their ability to handle critical alarms on time. Thus, various approaches have been recently proposed to effectively suppress such alarms, ensure alarm rationalization, and improve the overall performance of the alarm system.

Typically, strategies such as delay timers and dead-bands were frequently employed to address nuisance alarms [63]. Chattering alarms primarily cause a significant nuisance to the operators. A framework of generalized delay timers was introduced in [1] to handle false or nuisance alarms with specific criteria for raising and clearing alarms based on consecutive samples. Analytical expressions for performance matrices, including the false alarm rate (FAR), the missed alarm rate (MAR), and the expected detection delay (EDD), were provided to evaluate the performance of the framework. Design techniques of delay timers based on historical alarm data were investigated [32] and demonstrated that the combined use of off and on-delay timers performs better than the pure delay timers of the same length and the alarm latches. [41] studied the relation between alarm deadbands and optimal alarm limits and estimated the optimal threshold by analyzing the deadband and the characteristics of process variables.

In the alarm management life cycle, periodic assessment of alarm system performance is crucial, and one key aspect of this evaluation is the quantification of alarm chatter. [33] proposed an index to quantify the alarm chatter based on run-length distributions derived from the historical alarm data. [40] estimated the alarm chattering based on the distribution properties of both process variables and alarm parameters. [57] revised the index calculation for alarm chatter by considering the number of available data samples.

Advanced methods have been proposed to ensure alarm rationalization and prevent alarm overloading. [57, 58] introduced online techniques to eliminate chattering alarms by alarm shelving, adjusting alam thresholds, or implementing m-sample delay timers. [2] studied the time-deadband configurations for univariate alarm systems that introduce a specific time interval before triggering or deactivating alarms. A modified concept of generalized delay timer was introduced in [31] where additional alarm set points were introduced for process variables to activate or deactivate alarms alongside consecutive samples. [65] devolped a metric to assess the applicability of using either a delay timer or an alarm deadband to address nuisance alarms. In addition, a methodology for determining the optimal width of the deadband is introduced. Design of delay timers based on probabiltiy distributions of alarm durations [62], design of serially-connected alarm deadbands and delay timers [20], and design of deadbands based on maximum amplitude deviations and Bayesian estimation approach [61] are some of the recent approaches adopted to eliminate false alarms. [21] proposed some extensions to the work conducted in [62] where the design of both alarm trippoints and delay timers were considered to address false and missed alarms. In addition, cummulative probabilities of alarm durations was considered in this work.

Besides exploiting strategies involving delay timers and deadbands, there have been alternative approaches proposed to alleviate the issue of alarm overloading. A machine Learning-based approach was used in [55] to predict and quantify chattering alarms in chemical plants. This approach used three different models (Linear, Deep, Wide&Deep) which were trained using the results generated from the formulation of the dynamic chattering index based on historical data. Some of the long-standing alarms are nuisance alarms and significantly contribute to the overall high alarm rate. The main causes of long-standing alarms were discussed in [59] and a dynamic state-based strategy was adopted to suppress such alarms. State-based alarming has gained much attention for its efficiency in reducing nuisance alarms and preventing alarm floods. To simplify the implementation of state-based alarming, a data-driven approach was introduced in [25] for detecting association rules of mode-dependent alarms using A&E logs. Additionally, the design of ranking order filters and median filters are some of the techniques exploited to reduce false alarms and missed alarms.

Consequential or correlated alarms may also cause increased alarm rates. In [26], detection of correlated alarms based on the distribution of random occurrence delays and quantification of alarm correlation by converting binary alarm signals into continuous-valued pseudo-signals were demonstrated. The block matching similarity method was proposed in [70] to estimate the correlation coefficient from the time node sequences of alarms. Correlated alarms were grouped in [7] using a word embedding technique coupled with a novel clustering scheme and a multidimensional scaling method. A combination of alarm log, process data, and connectivity analysis was used in [48] to isolate consequential alarms originating from the same abnormality and to provide a causal alarm suggestion. Also, a weighted fuzzy association rule mining [60] approach to identify consequential alarms and data-driven trend analysis [36] are some of the effective methods to ensure alarm rationalization.

1.3.2 Recent Advances in Alarm Flood Analysis Methods

Industrial processes are significantly endangered by alarm floods. There are many causes of alarm floods. Nuisance alarms often co-exist with the true alarm pattern [22, 67] in alarm flood sequences and distract the plant operators. Through alarm rationalization or effective alarm system design, alarm floods consisting of nuisance or false alarms can be effectively averted. The industrial standard ANSI/ISA-18.2 [14] says that an alarm flood is present when more than 10 alarms occur within a 10-minute time period for each operator. However, alarm floods are often falsely detected due to the influence of chattering alarms and long-standing alarms. An algorithm based on a new criterion was proposed in [64] to accurately detect alarm floods by eliminating the influence of chattering and long-standing alarms in both online and offline scenarios.

In recent years, research on effectively managing alarm floods has gained much interest, leading to the exploitation of various data-driven and machine-learning approaches. Frequent pattern mining methods, in particular, have gained much attention. Such methods are used to discover interesting alarm patterns that can be exploited in real-time for conducting root cause analysis [3], implementing dynamic suppression of alarms, and facilitating decision-making. For instance, a frequent pattern mining algorithm was implemented in [56] to identify patterns of causally dependent notifications and remove redundant information in decision-making. An itemset mining method was proposed in [25] to detect frequent alarm patterns as candidates for dynamic alarm suppression. [11] introduced an algorithm to discover closed frequent temporal alarm patterns for the suppression of alarm floods. A sequential pattern mining algorithm was proposed in [43] to facilitate root cause analysis by identifying alarm sequential patterns. In [72], a modified CloFAST algorithm was developed to extract closed sequential patterns from alarm flood sequences.

Similarity-based approaches are commonly employed to recognize patterns, extract contextual details, and predict alarm floods in advance. A modified Smith-Waterman algorithm was developed in [10] to find local alignments between alarm flood sequences. To improve the computational efficiency, accelerated sequence alignment for alarm floods with set-based pre-matching, priority-based scoring, and a seeding and extending strategy was proposed in [27]. An accelerated sequence alignment method based on alarm match analysis was proposed to assess the similarity of industrial alarm flood sequences [19]. [35] extended the pairwise sequence alignment algorithm to the comparison of multiple alarm flood sequences. To compare alarm floods in cross-analogous processes, [71] invented a generalized pattern-matching approach based on word processing. Fault templates based on the presence of fault-specific alarms [8] and based on weighted similarity measures [9] were formed by pattern matching to provide fault-specific information to industrial operators. Classification methods, such as the Bayesian classifier [38], decision trees [12], deep recurrent neural networks (RNN) [13], and exponentially attenuated component analysis (EACA) [51] were applied to conduct classification of alarm floods and fault diagnosis.

Additionally, analyzing the root causes of the abnormal situations provides valuable insights, ultimately leading to improved operational efficiency for plant operators. Bayesian networks were exploited in [16] to model the underlying plant for capturing the root cause of faults. [28] introduced the ideas of normalized TE (NTE) and normalized DTE (NDTE) to help infer causal relationships from binary alarm data. [49] identified root causes using a causal Bayesian network, where faults were considered interventions (manipulation of variables by an external agent). [29] introduced a novel method for identifying the root causes of alarm floods by adopting a few-shot learning framework, an adaptive weighting strategy, and a similarity analysis approach.

Most of the above techniques are more suitable for offline analysis of alarm floods. In practice, assisting operators during an ongoing alarm flood situation (where alarms appear one by one in a sequential order) is a critical task and also a challenging issue. The online analysis of alarm floods mainly aims at providing decision support for plant operators to judge the root cause, predict incoming alarms, and make correct responses. Providing real-time assistance to operators with context-specific guidance to manage alarm floods is a challenging task and requires a systematic approach. To address this, an online pattern-matching algorithm with an incremental dynamic programming strategy was utilized in [34] to extract similar alarm patterns from the historical database as the incoming alarm flood sequence. Another approach presented in [37] was the classification of an incoming alarm flood based on alarm coactivations. [69] exploited similar historical sequences to predict upcoming alarms based on the Bayesian estimator. Classification based on a unique representation of alarm floods was proposed in [51] where higher weights are assigned to earlier alarms. [68] proposed a novel approach based on maximum entropy to predict upcoming alarms during alarm floods. A semi-supervised approach based on the Gaussian mixture model was proposed in [4] for early classification of alarm floods. An operator assistance system based on the early classification of alarm floods and a strategy for ranking alarms was proposed in [5].

During alarm floods, it is difficult to respond efficiently without essential information on the current and upcoming alarms, especially when the alarm rate is significantly high. Moreover, the presence of nuisance alarms may hide the true alarm pattern of an underlying abnormality. Additionally, some alarms associated with an alarm pattern may not appear in some scenarios. Accordingly, the alarm floods caused by the same underlying abnormality may have discrepancies in their sequences of alarms. Such misleading information may suggest wrong corrective actions during an ongoing alarm flood. Further, the ongoing situation may deviate from the existing historical sequences that were initially deemed similar by including a different set of alarms. In such cases, the plant operators may not have the necessary information to efficiently respond, leading to a significant deterioration of the current situation. Also, accuracy and earliness in corrective actions are important for the operational efficiency of the plant operators in such situations. Therefore, real-time decision support in alarm flood situations is in great demand, as it assists plant operators in judging the root causes and taking corrective actions. However, considering these challenges, the current state of research on online alarm flood analysis is still limited, with several aspects yet to be explored. This motivated us to pursue further research in this area.

1.4 Thesis Contributions

To address the challenges associated with online alarm flood analysis, this thesis focuses on developing data-driven approaches to provide real-time assistance to industrial operators during alarm floods. The contributions of this thesis are outlined as follows:

- In Chapter 2, we propose an online similarity analysis of alarm flood sequences based on the Smith-Waterman and Needleman-Wunsch algorithms. To address the ambiguity in alarm orders caused by small time gaps, the order ambiguity tolerance strategy from [10] is integrated into both methods. An online similarity with an incremental strategy is developed. The differences between the two methods are provided, and the application conditions for the two basic sequence alignment methods in the context of alarm system management are summarized.

- In Chapter 3, we propose a real-time similarity analysis method to compare an incoming alarm flood with historical sequences in an incremental manner. To avoid unnecessary computation for irrelevant alarms, an online set-based indexing and extension strategy is proposed. Online pre-matching mechanisms based on plant units and alarm sets are proposed to exclude irrelevant alarm flood sequences. Additionally, a ranking strategy is proposed to export and update the list of similar alarm flood sequences from historical database.

- In Chapter 4, we propose an online alarm prediction algorithm to predict upcoming alarms at the early stage of the ongoing alarm flood. A Compact Prediction Tree (CPT) model is modified with new features, namely, the timetable and co-occurrence matrix, and constructed based on historical alarm sequences. An alarm relevancy detection strategy is designed to identify and eliminate irrelevant alarms from the ongoing alarm flood. The confidence intervals of the time differences between the subsequent predicted alarm events are determined for time prediction.

- In Chapter 5, we propose a reinforcement learning approach for the early prediction of industrial alarm floods. The early prediction of industrial alarm floods is framed as a partially observable Markov decision process (POMDP). To optimize the accuracy and effectiveness, the double deep-Q network (DDQN) algorithm is adopted with a modified learning process. Furthermore, we propose a sequence reconstruction strategy based on association rule mining to eliminate irrelevant alarms and generate potential online scenarios based on the existing alarm relations present in the historical sequences. The training set is formulated based on several novel criteria for effective learning of the algorithm.

1.5 Thesis Outline

The remainder of the thesis is organized as follows: Chapter 2 presents an online similarity analysis of alarm floods using the Smith-Waterman and Needleman-Wunsch algorithms, highlighting their distinctions and application conditions. In Chapter 3, a real-time pattern-matching approach is introduced for the early prediction of alarm floods by searching for similar alarm flood sequences and generating a ranking list based on similarity scores. Chapter 4 introduces a novel association rule mining approach for real-time prediction of alarm events and their corresponding annunciation times during alarm flood situations. Chapter 5 proposes a reinforcement learning (RL) approach to predict incoming alarm floods and provide real-time assistance to plant operators in the early stages of an ongoing alarm flood. Finally, Chapter 6 concludes the thesis and presents potential future research directions.

Chapter 2

Comparison of the Smith-Waterman and Needleman-Wunsch Algorithms for Online Similarity Analysis of Industrial Alarm Floods

In this chapter, we propose an online similarity analysis of industrial alarm floods based on sequence alignment algorithms: the Smith-Waterman and Needleman-Wunsch algorithms. During an alarm flood situation, industrial operators often get confused by too many alarms and thus have difficulties in observing and handling critical alarms. In recent years, sequence alignment based similarity analysis has emerged as an effective way to handle alarm floods. Alarm floods caused by the same fault are very likely to consist of the same group of alarms in a certain sequential order. Conducting real-time sequence alignment of industrial alarm floods can help operators quickly recall the root cause and take prompt corrective actions. This chapter presents the online similarity analysis of alarm floods based on the Smith-Waterman and Needleman-Wunsch algorithms and compares their differences and application conditions. Case studies are provided to illustrate the proposed online similarity analysis methods and the differences between the two sequence alignment algorithms.

2.1 Similarity Analysis of Alarm Floods

This section introduces the problem of similarity analysis of alarm floods, provides basics on the Smith-Waterman algorithm and the Needleman-Wunsch algorithm, presents the online implementation of the sequence alignment of alarm floods, and summarizes the differences and application conditions of the two methods.

2.1.1 Problem Description

An alarm flood consists of an influx of sequentially ordered alarms; each one is described by a unique tag name, alarm identifier, priority, and time stamp. A pair of alarm flood sequences can be represented by $A = [a_1, a_2, ..., a_M]$ and $B = [b_1, b_2, ..., b_N]$, where M and N indicate the numbers of alarms appeared in sequences A and B, respectively. Each element in A or B can be represented by its unique alarm tag, timestamp, and priority. For example, a_i indicates the ith alarm of the alarm flood A, and can be represented by its attributes as $a_i = [e_i, t_i]$ where e_i and t_i are the alarm tag and time stamp of a_i, respectively.

In the context of alarm management, similarity analysis is a process of quantifying the similarity between alarm flood sequences by locating the similar segments or aligning the common alarm tags from both sequences. Basic sequence alignment methods, such as the Smith-Waterman algorithm and the Needleman-Wunsch algorithm, provide optimal local and global alignments along with the similarity scores between alarm flood sequences. A high similarity score indicates the presence of a large number of common alarm tags in both alarm flood sequences in the same chronological order. In practice, the reoccurrence of a fault usually leads to similar alarm floods consisting of almost the same group of alarms in a certain sequential order. By performing online similarity analysis, similar alarm floods can be extracted from the historical database and grouped to assist industrial operators in root cause analysis and early fault diagnosis.

Given an incoming alarm flood, the problem is how to compare it with historical sequences and find the most similar one, so as to predict alarms in the alarm flood and also to help with the root cause analysis. Different from offline applications, the sequence of an alarm flood increases with time. Therefore, the online similarity analysis is not a one-time comparison; instead, the comparison needs to be conducted iteratively with the increments of alarms. Therefore, this chapter proposes the online similarity analysis of alarm flood sequences based on the Smith-Waterman and Needleman-Wunsch algorithms. The ambiguity tolerance strategy in [10] is integrated with the two methods to tolerate the ambiguity of alarm orders caused by small time gaps. An online similarity with an increment strategy is developed. The differences between the two methods are provided, and the application conditions for the two basic sequence alignment methods in the context of alarm system management are summarized.

2.1.2 Sequence Alignment Algorithms

This subsection introduces the basics of the Smith-Waterman algorithm (SWA) and the Needleman-Wunsch algorithm (NWA), as well as the time ambiguity tolerance strategy.

Smith-Waterman Algorithm

The Smith-Waterman algorithm is a dynamic programming method that provides optimal local alignment. It was first proposed in [53] and modified in [10] by integrating a time ambiguity tolerance strategy for offline alarm flood pattern matching. As the original Smith-Waterman algorithm does not require time stamps, alarm floods can be represented as sequences of alarm tags, namely, $A = [a_1, a_2, ..., a_M]$ and $B = [b_1, b_2, ..., b_N]$, where a_p and b_q are alarm tags with $p = 1, 2, \cdots, M$, $q = 1, 2, \cdots, N$; M and N are the lengths of sequences A and B. Symbols $A_{i:m}$ and $B_{i:n}$ denote the segments of A and B, respectively. The algorithm identifies the optimal segment pair which has the highest similarity score, i.e., $S(A, B)$ defined as

$$S(A, B) = \max_{1 \leqslant i \leqslant m \leqslant M, 1 \leqslant j \leqslant n \leqslant N} (W(A_{i:m}, B_{j:n}), 0) \tag{2.1}$$

where $W(A_{i:m} : B_{j:n})$ indicates the similarity index of the pair of segments $(A_{i:m}, B_{j:n})$. The SWA generates an index matrix H with each element $H_{p+1,q+1}$ for a pair of alarms a_p and b_q calculated by

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

where $p = 1, 2, \cdots, M$, $q = 1, 2, \cdots, N$, and δ is the gap penalty score. The initial values in the first row and the first column of H are set as $H_{1,1} = 0, H_{p+1,1} = 0, H_{1,q+1} = 0$. The highest value of H is regarded as the similarity index between A and B. The optimal local alignment is found through a trace-back procedure that starts from the position with the highest value in H and ends with the position with zero in H.

Needleman-Wunsch Algorithm

The Needleman-Wunsch algorithm (NWA) was originally proposed in [42]. Unlike the SWA, it provides optimal global alignment, namely, end-to-end alignment between two sequences. The algorithm generates an index matrix H with each element $H_{p+1,q+1}$ for a pair of alarms a_p and b_q calculated by

$$H_{p+1,q+1} = \max\{H_{p,q} + S(a_p, b_q), H_{p,q+1} + \delta, H_{p+1,q} + \delta\}, \tag{2.3}$$

where $p = 1, 2, \cdots, M$ and $q = 1, 2, \cdots, N$. The initial values in the first row and the first column of H are set as $H_{1,1} = 0$, $H_{p+1,1} = p\delta$ and $H_{1,q+1} = q\delta$. Following the calculation of H, the trace-back procedure starts from the lower rightmost corner of H, which might not necessarily be the highest value in H. It switches to the upper diagonal, left, or upper vertical element depending on which of the elements has the highest value. In this way, the trace-back procedure continues until reaching the upper leftmost element. Moving forward through the identified elements from the trace-back procedure, the optimal global alignment is found.

Time Ambiguity Tolerance Strategy

Due to random detection delays, the orders of the physically connected alarms might be different in alarm flood sequences. To remove the discrepancy of orders, a time ambiguity tolerance strategy was proposed in [10]. In this strategy, a time distance vector for the n^{th} alarm a_n in the alarm flood sequence A is $\mathbf{d}_n = [d_{n1}d_{n2}....d_{nK}]^T$ with each element given by

$$d_n^k = \begin{cases} \min\limits_{1\leqslant j\leqslant m} \quad \{|t_n - t_j| : e_j^a = k\} & \text{if the set is not empty} \\ \infty, & \text{otherwise} \end{cases} \tag{2.4}$$

where $k = 1, 2, \cdots, K$ represents the index of a unique alarm type and K denotes the total number of unique alarm types. In the time distance vector, d_n^k denotes the time gap between the nth alarm and the nearest alarm with alarm type k. The time weight vector for the nth alarm is $w_n = [w_n^1 w_n^2....w_n^k]^T$, where $w_n^k = f(d_n^k)$. In the comparison of two sequences, the weighting functions to calculate the time weight vectors are different. For one sequence, the weighting function $f(\cdot)$ can be selected as a scaled Gaussian function $f(d_n^k) = e^{\frac{{d_n^k}^2}{2\sigma^2}}$, where σ denotes the kernel bandwidth in terms of the time difference. For the other sequence, the weighting function $f(\cdot)$ is $f(d_n^k) = 1$ if $d_n^k = 0$ and $f(d_n^k) = 0$ if $d_n^k = 1$. The similarity score $S(a_m, b_n)$ for an alarm pair a_m and b_n can be redefined by incorporating the time weights of corresponding alarms in both sequences [10]:

$$S((e_m, t_m), (e_n, t_n)) = \max_{1\leqslant k\leqslant K} [w_m^k \times w_n^k](1-\mu) + \mu, \tag{2.5}$$

where w_m^k and w_n^k are the time weight vectors of alarm tags e_m and e_n; μ is the mismatch score. Replacing the original similarity score in eqs (2.2) and (2.3) with the redefined similarity score of eq (2.5), both the SWA and NWA are capable of tolerating the ambiguity of orders caused by short time differences.

2.1.3 Online Similarity Analysis of Alarm Floods

This subsection presents a systematic online similarity analysis method, which exploits the sequence alignment algorithms to compare an incoming alarm flood sequence with each historical alarm flood sequence. Different from the offline comparison, the length of online alarm flood sequence is increasing.

The comparison starts with the triggering of an alarm flood. The detection of an alarm flood is based on the counting of annunciated alarms in a sliding window of 10 min. The alarm rate over a 10 min time window can be denoted as ε. Then, an alarm flood is said to be present if $\varepsilon \geq \varepsilon_{th}$, where ε_{th} denotes the threshold to identify alarm floods. According to the ANSI/ISA-18.2 standard [30], ε_{th} can be set as 10 alarms over a 10 min time window for each operator, i.e., $\varepsilon_{th} = 10$.

With the triggering of an alarm flood, the initial online alarm flood sequence A_f is obtained by including all alarms in the past 10 min time window before the triggering time point. To compare A_f of length $\tilde{M}$ with a historical alarm flood sequence B of length N, the sequence alignment algorithms can be applied. Given the initial sequence A_f, the initial index matrix H_f of size $\tilde{M} \times N$ is obtained based on eqs (2.2) and (2.3). Then, once a new alarm $\tilde{a}$ is annunciated, it will be added to A_f, i.e., $A_f = [A_f, \tilde{a}]$. Using the SWA to update the matrix H, an initial value $H_{\tilde{M}+1,1} = 0$ is firstly set. Then, each element in the $\tilde{M}+1$ row of H is calculated by

$$H_{\tilde{M}+1,q+1} = \max\{H_{\tilde{M},q} + S(\tilde{a}, b_q), H_{\tilde{M},q+1} + \delta, H_{\tilde{M}+1,q} + \delta, 0\}, \tag{2.6}$$

where $q = 1, 2, \cdots, N$. Analogously, using the NWA to update the matrix H, an initial value $H_{\tilde{M}+1,1} = \tilde{M}\delta$ is firstly set. Then, each element in the $\tilde{M}+1$ row of H is calculated by

$$H_{\tilde{M}+1,q+1} = \max\{H_{\tilde{M},q} + S(\tilde{a}, b_q), H_{\tilde{M},q+1} + \delta, H_{\tilde{M}+1,q} + \delta\}, \tag{2.7}$$

The calculation proceeds to the next iteration when another new alarm appears. The length of the online alarm flood sequence A_f is updated by $\tilde{M} = \tilde{M} + 1$.

The above iterative calculation is based on the assumption that the new alarms are annunciated and added to the sequence one by one. However, it is not uncommon to see that several alarms may appear almost simultaneously.

Or, it is also possible that the next new alarm appears before the calculation of $H_{\tilde{M}+1,q+1}$ for the current alarm $\tilde{a}$ is completed. In such cases, denote the group of new alarms as $[\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_L]$. Then, they will be added to A_f, i.e., $A_f = [A_f, \tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_L]$. To update the matrix H, the new added part is calculated for $[\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_L]$ and B. The values of $H_{\tilde{M},1}, H_{\tilde{M},2}, \cdots, H_{\tilde{M},N}$ are known in H. The initial values of $H_{\tilde{M}+1:\tilde{M}+L,1}$ are set as $H_{\tilde{M}+1,1} = 0, H_{\tilde{M}+2,1} = 0, \cdots, H_{\tilde{M}+L,1} = 0$ for the SWA, and set as $H_{\tilde{M}+1,1} = \tilde{M}\delta, H_{\tilde{M}+2,1} = (\tilde{M}+1)\delta, \cdots, H_{\tilde{M}+L,1} = (\tilde{M}+L-1)\delta$ for the NWA. Using the SWA, the matrix H is then updated by adding L rows with each element calculated by

$$\begin{aligned} H_{\tilde{M}+p,q+1} = & \max\{H_{\tilde{M}+p-1,q} + S(\tilde{a}_p, b_q), H_{\tilde{M}+p-1,q+1} + \delta, \\ & H_{\tilde{M}+p,q} + \delta, 0\}, \end{aligned} \tag{2.8}$$

where $q = 1, 2, \cdots, L$ and $q = 1, 2, \cdots, N$. Analogously, using the NWA, the matrix H is updated by

$$\begin{aligned} H_{\tilde{M}+p,q+1} = & \max\{H_{\tilde{M}+p-1,q} + S(\tilde{a}_p, b_q), H_{\tilde{M}+p-1,q+1} + \delta, \\ & H_{\tilde{M}+p,q} + \delta\}, \end{aligned} \tag{2.9}$$

Then, the calculation proceeds to the next iteration when new alarms appear, and the length of the online alarm flood sequence A_f is updated by $\tilde{M} = \tilde{M}+L$. It can be seen that the updating strategies in eqs. (2.6) and (2.7) are special cases of those in eqs. (2.8) and (2.9), respectively. The iterative calculation continues until the alarm rate drops to a low value, i.e., less than or equal to 5 alarms in a 10-minute time period according to the ANSI/ISA-18.2 standard [30].

2.1.4 Comparisons of Methods

The major differences between the two sequence alignment algorithms (namely, the SWA and the NWA) for similarity analysis of alarm flood sequences lie in the following aspects:

1. *Way of optimal alignment*: The SWA performs local alignment, i.e., finding the optimal local sub-sequence pair between two alarm flood

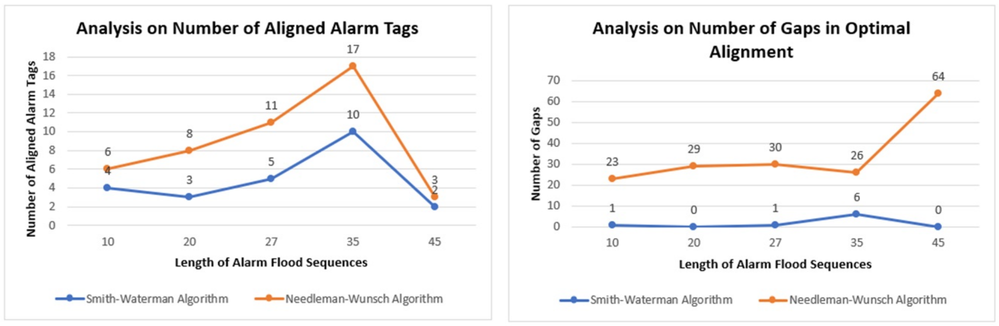


Figure 2.1: Analysis on sequence alignment by comparing one alarm flood sequence with other five sequences of different lengths.

sequences. The NWA, on the other hand, provides the optimal global alignment, i.e., end-to-end alignment.

2. *Initialization*: The SWA initializes the scoring matrix by assigning zero values to its first row and first column. The NWA, on the other hand, adds a gap for each shift to the right in the first row and each shift downward in the first column.

3. *Similarity score*: In the NWA, the similarity score between the alarm tags can be either positive or negative. For the SWA, the negative similarity score is set to zero.

4. *Length of optimal alignment*: For a pair of alarm flood sequences, the length of the optimal local alignment using the SWA will always be smaller than that of the optimal global alignment using the NWA. It should be noted that there could be more gaps in the alignment results using the NWA.

5. *Trace back procedure*: To find the optimal local alignment using the SWA, the trace-back starts from the maximum value and ends with an element equal to zero in the score index matrix. To find the optimal global alignment using the NWA, the trace-back starts from the lower rightmost element and ends at the upper leftmost element of the score index matrix.

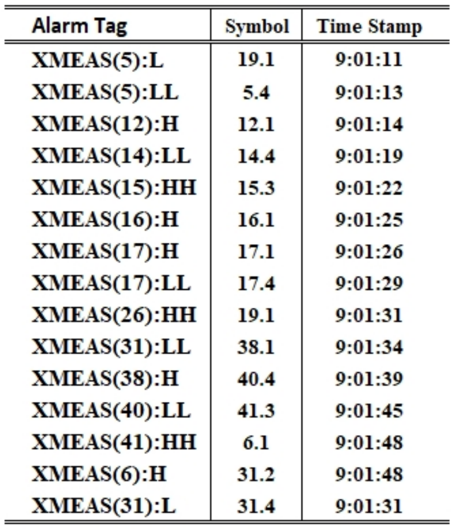
6. *Application*: The NWA is more useful for alarm flood sequences of approximately similar lengths. The SWA can locally align any pair of alarm flood sequences, but comparatively, more suitable for distantly related alarm flood sequences. Recent applications to alarm system management using the SWA and the NWA are briefly discussed in [27], [10], [34], and [9].

Based on the above comparisons of the two basic sequence alignment algorithms, each one has its merits and limitations. In real applications, one must be careful in choosing the suitable method, especially for online similarity analysis of alarm floods. The following aspects can be considered:

1) The sequence size may have a significant influence on the final alignment results. In Fig. 2.1, an alarm flood sequence of length 26 was selected from the historical database. Similarity analysis was performed with alarm flood sequences of lengths 10, 11, 24, 35, and 45, respectively. It can be seen that in the five cases, the numbers of aligned alarms using the NWA were larger than those using the SWA. Meanwhile, the numbers of gaps in the aligned sequence pair using the NWA were also much greater than those using the SWA. In other words, the NWA performed much better than the SWA with respect to the number of aligned alarms, but performed much worse than the SWA with respect to the number of gaps. In fact, the SWA provides good local alignments with significantly fewer gaps, and it will be easier to locate similar alarm tags from optimal local alignment between alarm flood sequences of extreme lengths. Given two alarm flood sequences with very different lengths, the SWA usually performs better in locating the common subsequence if the short alarm flood sequence is actually a portion of another alarm flood sequence caused by multiple faults or involving a larger area. Given that two alarm flood sequences do not have very different lengths, the NWA could be more suitable if the purpose is to find a maximum number of common alarms in a certain sequential order regardless of gaps.

2) The number of repeating alarms in the alarm flood may influence the *quality of sequence alignment.* Repeating alarms may exist in the alarm flood sequences due to the effect of random noises or disturbances in the process signals. Due to the presence of repeating alarms, it becomes difficult for the SWA to find a good local alignment between the sequences because the repeating alarms may lead to a large number of gaps, which makes the score in the index matrix H quickly drops to zero. Therefore, the SWA may only find a short segment of aligned subsequence pair between two alarm floods. By contrast, the NWA can output an end-to-end sequence alignment result. Even if there are many gaps caused by the presence of repeating alarms, the NWA can find all aligned alarms in the final alignment result. Therefore, the influence of the repeating alarms in the sequence alignment using the NWA is minor, making the NWA more suitable in the application of such cases.

Table 2.1: Alarm Flood Sequence



Alarm Tag	Symbol	Time Stamp
XMEAS(5):L	19.1	9:01:11
XMEAS(5):LL	5.4	9:01:13
XMEAS(12):H	12.1	9:01:14
XMEAS(14):LL	14.4	9:01:19
XMEAS(15):HH	15.3	9:01:22
XMEAS(16):H	16.1	9:01:25
XMEAS(17):H	17.1	9:01:26
XMEAS(17):LL	17.4	9:01:29
XMEAS(26):HH	19.1	9:01:31
XMEAS(31):LL	38.1	9:01:34
XMEAS(38):H	40.4	9:01:39
XMEAS(40):LL	41.3	9:01:45
XMEAS(41):HH	6.1	9:01:48
XMEAS(6):H	31.2	9:01:48
XMEAS(31):L	31.4	9:01:31

2.2 Case Studies

This section provides case studies to illustrate the online similarity analysis of alarm floods and the differences between the two sequence alignment algorithms in applications. The well-known Tennessee Eastman Process (TEP) was used to produce the alarm flood data for simulations. Table 2.1 shows an example of an alarm flood sequence consisting of alarm tags and their time stamps. For the ease of computation, the alarm tags in the alarm flood sequence are encoded into numeric symbols. The two numbers before and after the decimal point represent the process tag and the alarm identifier, respectively.

For the online simulation, alarms in the online alarm flood sequence were raised in a chronological order; the scoring matrices, along with the time distance and time weighting matrices, were updated simultaneously for both algorithms. To avoid mismatches in the alignment results, set $|\delta| < 0.5|\mu|$, such that gaps are included to replace mismatches. In the following case studies, the optimal and global alignments between various types of alarm flood sequences using SWA and NWA are presented.

2.2.1 Case I: Online Similarity Analysis between Alarm Flood Sequences of Different Lengths

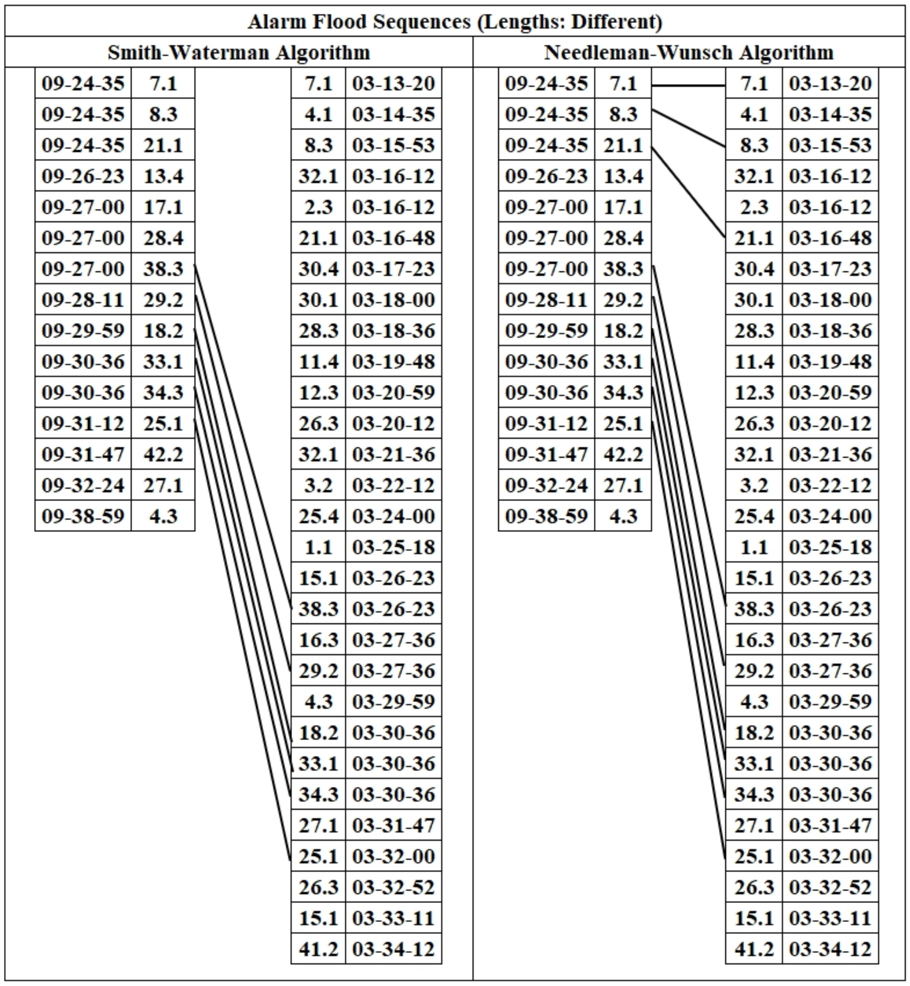
Fig 2.2 shows the optimal local and global alignment between the alarm flood sequences of different lengths. In the optimal global alignment using the NWA, it can be observed that the end-to-end alignment between the alarm flood sequences found more matched alarm tags, compared to that of the optimal local alignment using the SWA. However, NWA introduced a large number of gaps to provide the end-to-end alignment, which significantly reduced the similarity score and increased the length of the optimal global alignment. On the contrary, the SWA provides a good local alignment with significantly fewer gaps. Thus, if the preference is to find a good local pattern with alarms appearing in a tight sequential order, the SWA is more suitable. If preference is to extract maximum number of matched alarms regardless of gaps, the NWA performs much better in view of a much larger number of matched alarms.

2.2.2 Case II: Online Similarity Analysis between Alarm Flood Sequences with Repeating Alarms

Optimal local and global alignments between alarm flood sequences with repeating alarms using online similarity analysis are illustrated in Fig 2.3. In both alarm flood sequences, alarm tag 19.1 has multiple appearances. It can be observed that, compared to the optimal local alignment, more matched alarm tags were found in the optimal global alignment using the NWA. Both the SWA and the NWA, with the time ambiguity tolerance strategy, include alarm tags 19.1 and 19.4 in the optimal alignment results even though the orders of appearance were different in each sequence. If the preference is given to a type of alignment which shows the end-to-end alignment along with the pattern of repeating alarms between the alarm flood sequences, then finding an optimal global alignment using NWA is more suitable in such a case.

2.3 Summary

This chapter addressed the problem of selecting the suitable sequence alignment algorithm for the comparison of alarm floods in practice, especially for online implementation. Motivated by this question, an online similarity analysis of alarm floods based on the Smith-Waterman and Needleman-Wunsch algorithms is presented in this chapter. Initially, the chapter introduced the challenges of similarity analysis for alarm floods and provided the basics on the Smith-Waterman and the Needleman-Wunsch algorithms. Subsequently, a systematic method for online similarity analysis is presented, which is based on these sequence alignment algorithms to compare an incoming alarm flood sequence with each historical alarm flood sequence. Furthermore, the chapter briefly discusses the key differences between the Smith-Waterman and the Needleman-Wunsch algorithms, along with their respective suitability for similarity analysis of alarm flood sequences under different conditions. The effectiveness of each method in online similarity analysis is demonstrated through case studies, considering various criteria for alarm flood sequences.



Alarm Flood Sequences (Lengths: Different)

Smith-Waterman Algorithm

09-24-35	7.1
09-24-35	8.3
09-24-35	21.1
09-26-23	13.4
09-27-00	17.1
09-27-00	28.4
09-27-00	38.3
09-28-11	29.2
09-29-59	18.2
09-30-36	33.1
09-30-36	34.3
09-31-12	25.1
09-31-47	42.2
09-32-24	27.1
09-38-59	4.3

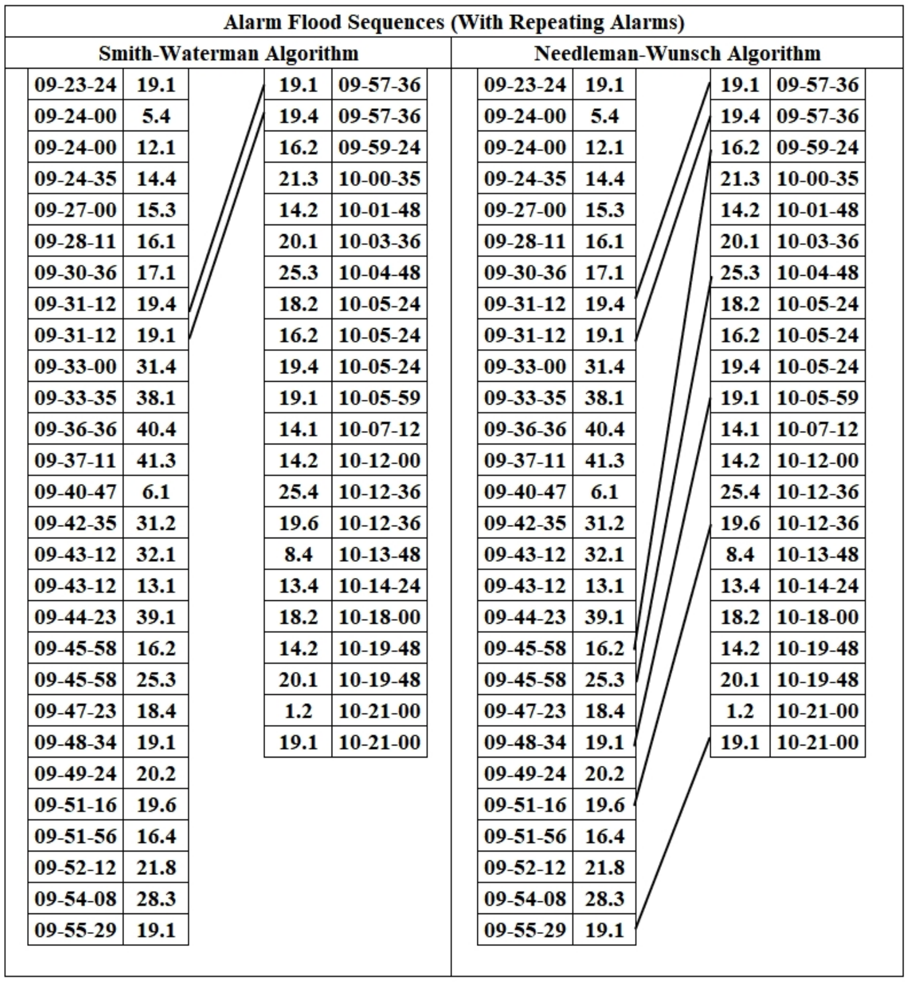
7.1	03-13-20
4.1	03-14-35
8.3	03-15-53
32.1	03-16-12
2.3	03-16-12
21.1	03-16-48
30.4	03-17-23
30.1	03-18-00
28.3	03-18-36
11.4	03-19-48
12.3	03-20-59
26.3	03-20-12
32.1	03-21-36
3.2	03-22-12
25.4	03-24-00
1.1	03-25-18
15.1	03-26-23
38.3	03-26-23
16.3	03-27-36
29.2	03-27-36
4.3	03-29-59
18.2	03-30-36
33.1	03-30-36
34.3	03-30-36
27.1	03-31-47
25.1	03-32-00
26.3	03-32-52
15.1	03-33-11
41.2	03-34-12

Needleman-Wunsch Algorithm

09-24-35	7.1
09-24-35	8.3
09-24-35	21.1
09-26-23	13.4
09-27-00	17.1
09-27-00	28.4
09-27-00	38.3
09-28-11	29.2
09-29-59	18.2
09-30-36	33.1
09-30-36	34.3
09-31-12	25.1
09-31-47	42.2
09-32-24	27.1
09-38-59	4.3

7.1	03-13-20
4.1	03-14-35
8.3	03-15-53
32.1	03-16-12
2.3	03-16-12
21.1	03-16-48
30.4	03-17-23
30.1	03-18-00
28.3	03-18-36
11.4	03-19-48
12.3	03-20-59
26.3	03-20-12
32.1	03-21-36
3.2	03-22-12
25.4	03-24-00
1.1	03-25-18
15.1	03-26-23
38.3	03-26-23
16.3	03-27-36
29.2	03-27-36
4.3	03-29-59
18.2	03-30-36
33.1	03-30-36
34.3	03-30-36
27.1	03-31-47
25.1	03-32-00
26.3	03-32-52
15.1	03-33-11
41.2	03-34-12

Figure 2.2: Alarm flood sequence alignment results for two alarm flood sequences of different lengths.



Alarm Flood Sequences (With Repeating Alarms)

Smith-Waterman Algorithm

09-23-24	19.1	19.1	09-57-36
09-24-00	5.4	19.4	09-57-36
09-24-00	12.1	16.2	09-59-24
09-24-35	14.4	21.3	10-00-35
09-27-00	15.3	14.2	10-01-48
09-28-11	16.1	20.1	10-03-36
09-30-36	17.1	25.3	10-04-48
09-31-12	19.4	18.2	10-05-24
09-31-12	19.1	16.2	10-05-24
09-33-00	31.4	19.4	10-05-24
09-33-35	38.1	19.1	10-05-59
09-36-36	40.4	14.1	10-07-12
09-37-11	41.3	14.2	10-12-00
09-40-47	6.1	25.4	10-12-36
09-42-35	31.2	19.6	10-12-36
09-43-12	32.1	8.4	10-13-48
09-43-12	13.1	13.4	10-14-24
09-44-23	39.1	18.2	10-18-00
09-45-58	16.2	14.2	10-19-48
09-45-58	25.3	20.1	10-19-48
09-47-23	18.4	1.2	10-21-00
09-48-34	19.1	19.1	10-21-00
09-49-24	20.2		
09-51-16	19.6		
09-51-56	16.4		
09-52-12	21.8		
09-54-08	28.3		
09-55-29	19.1		

Needleman-Wunsch Algorithm

09-23-24	19.1	19.1	09-57-36
09-24-00	5.4	19.4	09-57-36
09-24-00	12.1	16.2	09-59-24
09-24-35	14.4	21.3	10-00-35
09-27-00	15.3	14.2	10-01-48
09-28-11	16.1	20.1	10-03-36
09-30-36	17.1	25.3	10-04-48
09-31-12	19.4	18.2	10-05-24
09-31-12	19.1	16.2	10-05-24
09-33-00	31.4	19.4	10-05-24
09-33-35	38.1	19.1	10-05-59
09-36-36	40.4	14.1	10-07-12
09-37-11	41.3	14.2	10-12-00
09-40-47	6.1	25.4	10-12-36
09-42-35	31.2	19.6	10-12-36
09-43-12	32.1	8.4	10-13-48
09-43-12	13.1	13.4	10-14-24
09-44-23	39.1	18.2	10-18-00
09-45-58	16.2	14.2	10-19-48
09-45-58	25.3	20.1	10-19-48
09-47-23	18.4	1.2	10-21-00
09-48-34	19.1	19.1	10-21-00
09-49-24	20.2		
09-51-16	19.6		
09-51-56	16.4		
09-52-12	21.8		
09-54-08	28.3		
09-55-29	19.1		

Figure 2.3: Alarm flood sequence alignment results for two alarm flood sequences with repeating alarms.

Chapter 3

Real-time Pattern Matching and Ranking for Early Prediction of Industrial Alarm Floods

In this chapter, we propose a real-time pattern matching approach for the early prediction of alarm floods by searching for similar alarm flood sequences and exporting a ranking list based on the similarity score. An alarm flood is known as a condition in which the number of alarms is more than the operator can manage effectively. As a result, an operator has difficulties in identifying and responding to critical alarms. If corrective actions are not taken timely, especially at the early stage of an alarm flood, the situation may get worse and cause more serious consequences. Therefore, how to assist operators during the ongoing alarm flood situation is a critical problem to solve. Accordingly, this chapter presents a new method to search for similar alarm floods and export a ranking list, so as to provide decision supports for plant operators to judge the root cause and make corrective responses based on the knowledge about the most similar historical alarm floods. Initially, online pre-matching mechanisms based on plant units and alarm sets are proposed to exclude irrelevant alarm flood sequences. Then, a real-time similarity analysis method is developed to compare an incoming alarm flood with historical sequences in an incremental manner. To avoid unnecessary computation for irrelevant alarms, an online set-based indexing and extension strategy is proposed. Finally, a ranking strategy is proposed to export and update the list of similar alarm flood sequences from historical database. The effectiveness of the proposed method is demonstrated by an industrial case study based on real alarm & event logs from a refinery plant.

3.1 Problem Description

This section introduces the alarm flood problem and the framework of real-time pattern matching to provide decision supports, and also presents the detection and representation of alarm flood sequences.

3.1.1 Industrial Alarm Floods

In an alarm system, each alarm is typically described by several attributes, such as the tag name, alarm identifier, and plant unit, which are usually all known and well configured in the system. Whenever an abnormality occurs, the associated alarms are annunciated and presented to the operators, and meanwhile are historized as time-stamped events in an Alarm & Event (A&E) log. In this work, each alarm event a_i^m is described by three attributes as:

$$a_i^m = (e_i^m, t_i^m, u_i^m), \tag{3.1}$$

where $e_i^m \in \mathbb{A}$, t_i^m, and $u_i^m \in \mathbb{U}$ indicate the alarm tag, time stamp, and plant unit of a_i^m, respectively. Here, $\mathbb{A}$ represents the set of all unique configured alarm tags in an alarm system; $\mathbb{U}$ denotes the set of associated plant units for all alarms in $\mathbb{A}$, and each unique alarm tag e in $\mathbb{A}$ must be configured with a certain unit u in $\mathbb{U}$ to indicate where it is from.

Alarm floods can be identified from the historical A&E log by comparing the alarm rate with the benchmark thresholds in [30]. Then, the periods of alarm floods are found, and the corresponding sequences are extracted [25]. Specifically, the alarm rate $\eta(t)$ at any time instant t is calculated as the count of alarm occurrences within a time window $[t - \beta + 1, t]$, where β is 600 s. An alarm flood is detected to start at t if the alarm system is not under an alarm flood situation at $t-1$, and $\eta(t) \geq \eta_1$, where η_1 denotes the threshold to mark the beginning of an alarm flood, namely, $\eta_1 = 10$ alarms in a period of 10 min [14, 30]. The alarm flood situation exists until $\eta(t) < \eta_2$, where $\eta_2 = 5$ alarms over a time window of 10 min. Here, an alarm flood sequence is obtained and denoted by

$$A_m =< a_1^m, a_2^m, ..., a_{|A_m|}^m >, \tag{3.2}$$

where m stands for the numeric identifier of the alarm flood, and a_i^m denotes the ith ($i = 1, 2, \cdots, |A_m|$) alarm event in A_m. The symbol $|\cdot|$ represents the size of a sequence or set; here, $|A_m|$ is the number of alarm events in A_m. Meanwhile, related information such as the start time τ_m^s, the time length τ_m^d, and the set U_m of involved plant units, can also be extracted and historized.

It should be noted that the plant unit can be defined in different ways depending on the context and functionality. This study assumes that the plant unit is a piece of known information, which usually holds as plant units are generally defined in system configuration and can be retrieved from the historical data set. Meanwhile, each alarm in an alarm system is usually assigned with the unit information as a key attribute. Then, the plant units for an alarm flood is obtained as a set of plant units for all included alarms in this flood sequence. Therefore, the involved unit set U_m for alarm flood m is a set of plant units given by

$$U_m = \bigcup_{i=1}^{|A_m|} \{u_i^m\}. \tag{3.3}$$

The database of all historical alarm flood sequences is then represented by $\mathbb{H} = \{A_m : m = 1, 2, \cdots, M\}$, where M is the number of alarm floods. For each alarm flood sequence A_m, other related data attributes, such as the start time τ_m^s, the time length τ_m^d, and the set of involved plant units U_m, are also historized and used for real-time pattern matching of alarm floods.

It is noteworthy that chattering alarms must be reduced first prior to detecting alarm floods because chattering alarms can lead to a high alarm rate and thus cause false detection of alarm floods. There exist many ways to reduce chattering alarms, such as using delay timers, which are well-established techniques [25, 34]. Thus, this work is not going to discuss how to reduce chattering alarms, and in the rest of this work, it is assumed that all chattering alarms are reduced.

3.1.2 Problem Formulation of Real-time Pattern Matching

In the presence of an ongoing alarm flood, the plant operators could be distracted and stressed, because they have to face a large number of alarm messages but have difficulty in identifying critical alarms from non-important or irrelevant ones. If critical alarms are not correctly responded, the situation may get worse. Thus, it is in great demand to provide real-time assistance for plant operators to make corrective decisions in identifying and responding to critical alarms on time. This work presents a real-time pattern matching and ranking approach to find similar alarm floods from historical database and export a ranking list based on the similarity score. As a result, the operator can handle the ongoing alarm flood situation and mitigate its influence in the early stage, by referring to the experience and key information in similar historical alarm floods.

An ongoing alarm flood is formed at time instant t_0 when the alarm rate $\eta(t_0)$ reaches 10 alarms over a time window of 10 min. Denote the initial ongoing alarm flood sequence as

$$B =< b_1, b_2, ..., b_L >, \tag{3.4}$$

where $b_i = (e_i^b, t_i^b, u_i^b), i = 1, 2, \cdots, L$; L denotes the number of alarm occurrences in $[t_0 - 599\text{s}, t_0]$ of the ongoing alarm flood. As more alarms are coming on, the ongoing alarm flood sequence is increasing. Given a sequence of newly appeared alarms till time instant t_1 after t_0 as $\tilde{B} =< b'_1, b'_2, \cdots, b'_l >$, the ongoing alarm flood sequence B is updated by

$$\begin{aligned} B &= B \oplus \tilde{B} \\ &= < b_1, b_2, ..., b_L, b'_1, b'_2, \cdots, b'_l >, \end{aligned} \tag{3.5}$$

where $\oplus$ denotes the extension of a sequence and l represents the number of newly appeared alarms.

Then, the problem is how to find which alarm flood sequences from the historical database $\mathbb{H}$ are similar to the ongoing alarm flood B. Hereby, the similarity score $S(B, A_m)$ between B and each historical alarm flood sequence A_m must be calculated. According to many existing studies for offline alarm flood similarity analysis in [9, 10, 27, 35, 71], such a similarity score can be computed via sequence alignment approaches. To ensure the efficiency and reliability of the real-time pattern matching of alarm floods, the following issues must be handled:

Issue 1 The analysis is conducted online and thus is very time-sensitive. To perform early prediction of alarm floods, the real-time pattern matching must be executed efficiently.

Issue 2 The pattern matching is applied to an ongoing alarm flood sequence that is increasing with more new alarms emerging over time. Thus, the pattern matching must be conducted in an incremental manner.

Issue 3 Considering that alarm floods may assemble each other locally based on different common subsequences, identifying only one historical alarm flood holding the highest similarity score sometimes might be unauthentic.

In view of the above problems, this work proposes a systematic method to export reliable results as decision supports for operators under an ongoing alarm flood situation. The framework of the proposed method is as follows:

Step 1 Plant unit based screening is applied to exclude irrelevant alarm floods that originated from plant units different from the ongoing alarm flood B.

Step 2 Alarm set based pre-matching is applied to quickly judge which historical alarm floods are similar to B, based on the principle that two alarm flood sequences cannot be similar to each other given that their alarm sets are not similar. This pre-matching step can further screen out a significant number of dissimilar alarm floods.

Step 3 Incremental sequence alignment is devised to compare the ongoing alarm flood sequence with screened historical sequences in an incremental manner without needing to compare the complete sequences over again whenever new alarms appear. To accelerate the computation by avoiding unnecessary alignment, an online set based indexing and extension strategy is proposed to exclude irrelevant segments of alarms from the pattern matching.

Step 4 Dynamic ranking of similar alarm floods is proposed to update and export the ranking list of similar historical alarm floods based on their calculated similarity scores.

The first two steps solve the first issue, the 3rd step copes with issue 2, and the 4th step addresses issue 3. The proposed method is executed whenever $\eta(t) \geq \eta_1$. A ranking list of similar historical alarm floods is exported as the final result and is updated whenever new alarms are annunciated along with time. Such results are presented to plant operators to provide decision supports for them to recall the root causes and make corrective responses. The iterative computation is completed once $\eta(t) < \eta_2$ or the alarm flood problem is solved. Detailed methods of each step are presented in the next section.

3.2 The Proposed Method

This section presents the proposed method, including the plant unit based screening, alarm set based pre-matching, incremental sequence alignment, online set based indexing and extension, as well as dynamic ranking of similar alarm floods.

3.2.1 Plant Unit Based Screening

In an industrial plant, a fault is usually originated from a specific place, and causes an alarm flood consisting of consequential alarms from a few related plant units rather than from the whole plant. Therefore, every alarm flood has its own associated plant units. To measure whether alarm floods are similar or not, one can firstly compare the associated units between two alarm floods, as the set of unique units is much smaller than the set of unique alarms. If the units are different, it is unnecessary to conduct sequence alignment, which is more computationally expensive. Thus, in this study, the purpose for exploiting the "plant unit" information is to screen out irrelevant alarm floods as a pre-matching step, so as to avoid unnecessary sequence alignment computation in later steps. Accordingly, given an online alarm flood sequence B, it is unnecessary to directly compare it with all alarm floods in the historical database $\mathbb{H}$. Plant unit-based screening finds and excludes irrelevant historical alarm flood sequences, by comparing the unit set U_B of B with those of floods in $\mathbb{H}$.

At time instant t_0 when an online alarm flood B is firstly triggered, the associated unit set U_b of B is obtained as

$$U_b = \bigcup_{i=1}^{|B|} \{u_i^b\}, \tag{3.6}$$

where u_i^b denotes the plant unit of each alarm b_i in B. Given a sequence of newly appeared alarms $\tilde{B} =< b'_1, b'_2, \cdots, b'_l >$ added to B at t_1, the associated unit set U_b of B is updated by

$$U_b = U_b \cup \bigcup_{i=1}^{l} \{\tilde{u}_i^b\}, \tag{3.7}$$

where $\tilde{u}_i^b$ is the plant unit of b'_i in $\tilde{B}$.

Then, the unit-based similarity score, i.e., $S_{unit}(A_m, B)$ between U_b of B

and U_m of $A_m \in \mathbb{H}$ is formulated as

$$S_{unit}(A_m, B) = \frac{|U_b \cap U_m|}{|U_b \cup U_m|}. \tag{3.8}$$

However, this equation ignores the frequency of alarm occurrences, i.e., each plant unit is only counted once even if there are multiple alarms originated from it. In view of this problem, an improved formulation of S_{unit} between U_b of B and U_m of $A_m \in \mathbb{H}$ is formulated as

$$S_{unit}(A_m, B) = 0, \text{ if } U_b \cap U_m = \emptyset, \tag{3.9}$$

else if $U_b \cap U_m \neq \emptyset$,

$$S_{unit}(A_m, B) = \sqrt{\frac{\sum_{i=1}^{|B|} I_i^b \cdot \sum_{j=1}^{|A_m|} I_j^m}{|B| \cdot |A_m|}}, \tag{3.10}$$

where

$$I_i^b = \begin{cases} 1 & \text{if } u_i^b \in U_b \cap U_m, \\ 0 & \text{o.w.,} \end{cases} \quad I_j^m = \begin{cases} 1 & \text{if } u_j^m \in U_b \cap U_m, \\ 0 & \text{o.w.} \end{cases}$$

The unit-based similarity score $S_{unit}(A_m, B)$ ranges from 0 to 1. A larger value of $S_{unit}(A_m, B)$ indicates a higher similarity in terms of plant units between A_m and B. In other words, the historical alarm flood sequence A_m is said to be similar to B in terms of plant units if $S_{unit}(A_m, B) > \Gamma_{unit}$, where Γ_{unit} is a user-defined threshold. A default value $\Gamma_{unit} = 0$ can be used as the simplest case, implying that the comparison will proceed to the next step only if there is at least one least one alarm in B coming from the same plant unit of alarms in A_m. The collection of historical alarm flood sequences from units with B is obtained as

$$\mathbb{H}_{unit} = \{A_m \in \mathbb{H}, \text{ s.t., } S_{unit}(A_m, B) > \Gamma_{unit}\}. \tag{3.11}$$

With the addition of incoming alarms to the online alarm flood sequence B, the unit-based similarity score $S_{unit}(A_m, B)$ will be recalculated and $\mathbb{H}_{unit}$ will be updated.

3.2.2 Alarm Set Based Pre-matching

The plant unit-based screening narrowed the comparison down to a smaller set of historical alarm flood sequences that have alarms coming from the same plant units as those in the online alarm flood B. Then, an alarm set based pre-matching strategy is used to further find and exclude historical alarm flood sequences that bear little similarity in terms of alarm tags with B. Given $A_m \in \mathbb{H}_{unit}$ and B, their corresponding alarm sets are V_m and V_b. Then, two binary indexes J_j^m and J_i^b are given as

$$J_i^b = \begin{cases} 1 & \text{if } e_i^b \in V_b \cap V_m, \\ 0 & \text{o.w.,} \end{cases} \quad J_j^m = \begin{cases} 1 & \text{if } e_j^m \in V_b \cap V_m, \\ 0 & \text{o.w.} \end{cases}$$

where e_i^b (e_j^m) is the alarm tag of the ith (jth) alarm event in B (A_m); $i = 1, 2, \cdots, |B|$ and $j = 1, 2, \cdots, |A_m|$. Then, the alarm set-based similarity score S_{set} between V_b of B and V_m of $A_m \in \mathbb{H}_{unit}$ is formulated as

$$S_{set}(A_m, B) = 0, \text{ if } V_b \cap V_m = \emptyset, \tag{3.12}$$

else if $V_b \cap V_m \neq \emptyset$,

$$S_{set}(A_m, B) = \sqrt{\frac{\sum_{i=1}^{|B|} J_i^b \cdot \sum_{j=1}^{|A_m|} J_j^m}{|B| \cdot |A_m|}}. \tag{3.13}$$

The alarm set-based similarity score $S_{set}(A_m, B)$ ranges from 0 to 1. A larger value of $S_{set}(A_m, B)$ indicates a higher similarity in terms of alarm sets between A_m and B. In other words, A_m is said to be similar to B in terms of alarm sets if $S_{set}(A_m, B) > \Gamma_{set}$, where Γ_{set} is a user-defined threshold. A default value $\Gamma_{set} = 0$ can be used as the simplest case, implying that the comparison will proceed to the next step if B shares at least one common unique alarm with A_m. The collection of historical alarm flood sequences sharing common alarms with B is obtained as

$$\mathbb{H}_{set} = \left\{A_m \in \mathbb{H}_{unit}, \text{ s.t., } S_{set}(A_m, B) > \Gamma_{set}\right\}. \tag{3.14}$$

With the addition of incoming alarms to the online alarm flood sequence B, the alarm set-based similarity score $S_{set}(A_m, B)$ will be recalculated and $\mathbb{H}_{set}$ will be updated.

3.2.3 Incremental Sequence Alignment

To query B in $\mathbb{H}_{set}$ and measure the similarities, the sequence alignment needs to be conducted online. The Smith-Waterman (SW) algorithm [52] and Needleman-Wunsch (NW) algorithm [42] are the two basic sequence alignment approaches. Both of them can be used to measure the similarities between alarm flood sequences. The former one performs local alignment, while the latter provides optimal global alignment. In most existing studies [9, 10, 27, 35, 71], alarm flood sequences are complete and the sequence alignment is conducted offline. In online applications, such methods cannot be directly applied since the online alarm flood sequence B is increasing with time. Thereby, this subsection proposes an incremental dynamic programming strategy for sequence alignment of industrial alarm floods based on the SW and NW algorithms.

The sequence alignment starts with the triggering of the online alarm flood B at time instant t_0 when it is detected $\eta(t_0) \geq \eta_1$. Following the alarm set-based pre-matching, B is compared with every historical sequence A_m in $A_m \in \mathbb{H}_{set}$. Then, an index matrix H of dimension $(|B|+1) \times (|A_m|+1)$ is initialized. Based on the SW algorithm, each element in H, namely, $H_{p+1,q+1}$ is calculated as

$$H_{p+1,q+1} = \max\{H_{p,q} + s(b_p, a_q^m), H_{p,q+1} + \delta, H_{p+1,q} + \delta, 0\}, \tag{3.15}$$

where $p = 1, 2, ..., |B|$ and $q = 1, 2, ..., |A_m|$. The initial values of H are $H_{1,1} = 0$, $H_{p+1,1} = 0$ $(p = 1, 2, ..., |B|)$, and $H_{1,q+1} = 0$ $(q = 1, 2, ..., |A_m|)$. $s(b_p, a_q^m)$ is a basic similarity score between the pth alarm in B and qth alarm in A_m; $s(b_p, a_q^m) = \phi$ if $e_p^b = e_q^m$, and $s(b_p, a_q^m) = \mu$ if $e_p^b \neq e_q^m$, where ϕ and μ are the match and mismatch scores, respectively. The symbol δ is the gap penalty score. Typically, $\phi > 0$, $\mu < 0$, $\delta < 0$, and $\phi > \max\{|\mu|, |\delta|\}$. To prefer gapped alignment instead of mismatches, it should set $\mu < 2\delta < 0$. Analogously, based on the NW algorithm, $H_{p+1,q+1}$ is calculated as

$$H_{p+1,q+1} = \max\{H_{p,q} + s(b_p, a_q^m), H_{p,q+1} + \delta, H_{p+1,q} + \delta\}. \tag{3.16}$$

where $p = 1, 2, ..., |B|$ and $q = 1, 2, ..., |A_m|$. The initial values of H are $H_{1,1} = 0$, $H_{p+1,1} = p\delta$ $(p = 1, 2, ..., |B|)$, and $H_{1,q+1} = q\delta$ $(q = 1, 2, ..., |A_m|)$. After the sequence alignment is completed, the best alignment result is obtained by a backtracking step [42, 52]. A sequence-based similarity score $S_{seq}(A_m, B)$ is obtained as

$$S_{seq}(A_m, B) = \frac{H_{\max}}{\min\{|B|\phi, |A_m|\phi\}}, \tag{3.17}$$

where $H_{\max}$ represents the highest value in H of size $(|B| + 1) \times (|A_m| + 1)$. The sequence-based similarity score $S_{seq}(A_m, B)$ ranges from 0 to 1; a larger value of $S_{seq}(A_m, B)$ indicates higher similarity between A_m and B in terms of alarm sequences.

Then, given $\tilde{B} =< b'_1, b'_2, \cdots, b'_l >$, namely, a sequence of newly appeared alarms at t_1 after t_0, the sequence alignment is conducted in an incremental way, i.e., the index matrix H of size $(|B| + 1) \times (|A_m| + 1)$ will be updated by index matrix H of of size $(|B| + |\tilde{B}| + 1) \times (|A_m| + 1)$, where the additional part is calculated by the SW algorithm as

$$\begin{aligned} H_{|B|+1+p,q+1} \;=\; & \max\{H_{|B|+p,q} + s(b'_p, a^m_q), H_{|B|+p,q+1} + \delta, \\ & H_{|B|+1+p,q} + \delta, 0\}, \end{aligned} \tag{3.18}$$

where $p = 1, 2, ..., |\tilde{B}|$ and $q = 1, 2, ..., |A_m|$. The initial values in the first column of H are $H_{|B|+1+p,1} = 0$, $p = 1, 2, ..., \tilde{B}$. Analogously, using the NW algorithm to update the matrix H, the additional part is calculated as

$$\begin{aligned} H_{|B|+1+p,q+1} \;=\; & \max\{H_{|B|+p,q} + s(b'_p, a^m_q), H_{|B|+p,q+1} + \delta, \\ & H_{|B|+1+p,q} + \delta\}, \end{aligned} \tag{3.19}$$

where the initial values in the first column of H are $H_{|B|+1+p,1} = (|B|+p)\delta$, $p = 1, 2, ..., \tilde{B}$. The sequence based similarity score S_{seq} is updated by substituting the highest value $H_{\max}$ in the updated H of size $(|B|+|\tilde{B}|+1)\times(|A_m|+1)$ into eqn. (3.17). The calculation proceeds to the next iteration when another series of new alarms appear. Real-time sequence alignment will continue to update H incrementally for the incoming group of alarms until the alarm flood situation terminates, i.e., when the alarm rate drops below 5 alarms in 10 minutes time period. It is noteworthy that the above incremental computation of H is conducted based on alignment results between $\tilde{B}$ and A_m other than between $B \oplus \tilde{B}$ and A_m; it avoids comparing the whole online sequence $B = B \oplus \tilde{B}$ with A_m in the iteration, and thus saves computational time.

In an alarm system, it is possible that alarms arise almost simultaneously but have different orders in alarm floods. To address the discrepancy of orders, a time ambiguity tolerance strategy in [10] is adopted and modified here. The basic similarity score $s(b_p, a_q^m)$ in eqns. (3.15) and (3.16) is reformulated as

$$s(b_p, a_q) = \max_{1 \le k \le K} [w_q^k](\phi - \mu) + \mu \tag{3.20}$$

where w_q^k is the time weight and K represents the number of unique alarm tags in A_m, namely, $K = |V_m|$. The time weight w_q^k is calculated as follows: A time distance vector for the qth alarm a_q in A_m is obtained as $[d_q^1, d_q^2, \cdots, d_q^K]^T$ with each element given by $d_q^k = \min_{1 \le k \le K}\{|t_q - t_j| : e_j^a = V_m^k\}$, where V_m^k represents the kth unique alarm tag in V_m. Then, a time weighting vector for a_q in A_m is obtained as $[w_q^1, w_q^2, \cdots, w_q^K]^T$ with each element given by $w_q^k = f(d_q^k)$, where $f(\cdot)$ is a weighting function; a suitable option is the scaled Gaussian function $f(x) = e^{-\frac{x^2}{2\sigma^2}}$ [10], where σ represents the kernel bandwidth in terms of the time difference. It should be noticed that the time distance vector is only calculated for alarms in A_m rather than B, i.e., it only considers the time information in A_m in the time ambiguity tolerance. This is reasonable since the online sequence B is increasing, and the time distance vector needs to be recalculated again with the addition of new alarms, thus increasing the computational burden. Therefore, this work proposes to calculate only the time distance vector for A_m and obtain a time weighting matrix W as

$$W = \begin{bmatrix} w_1^1 & w_2^1 & \cdots & w_{|A_m|}^1 \\ w_1^2 & w_2^2 & \cdots & w_{|A_m|}^2 \\ \vdots & \vdots & \ddots & \vdots \\ w_1^K & w_2^K & \cdots & w_{|A_m|}^K \end{bmatrix}, \tag{3.21}$$

which is calculated offline and historized with the alarm flood database $\mathbb{H}$. Accordingly, the calculation of eqn. (3.20) becomes straightforward and is much reduced.

3.2.4 Set Based Indexing and Extension

The incremental sequence alignment in Section 3.2.3 measures the sequence based similarity between the online flood and each historical sequence $A_m \in \mathbb{H}_{set}$. However, it is noteworthy that even though $A_m \in \mathbb{H}_{set}$ shares some common alarm tags with B, it may still be possible that most alarms between the two could be distinct, which leads to unnecessary computation in sequence alignment, especially in cases where the lengths of historical sequences are significantly larger than B. To overcome this problem, a set based indexing and extension strategy is proposed to exclude irrelevant subsequences from each historical alarm flood sequence $A_m \in \mathbb{H}_{set}$ in sequence alignment.

As presented in Section 3.2.2, a binary index J_j^m is obtained to indicate whether the jth alarm in A_m is a common tag also present in B, where $j = 1, 2, \cdots, |A_m|$. Then, sequence alignment is conducted to compare B with a subsequence A'_m of A_m rather than the complete A_m. Such a subsequence A'_m is represented by

$$A'_m =< a_1^m, a_2^m, \cdots, a_\Psi^m >, \tag{3.22}$$

where Ψ is an integer ranging between 1 and $|A_m|$; it represents the last position of common alarms in A_m, i.e.,

$$J_\Psi^m = 1 \text{ and } J_j^m = 0, \forall j > \Psi. \tag{3.23}$$

Therefore, at the triggering time instant t_0 of the online flood B, the initial index matrix H of dimension $(|B|+1) \times (\Psi+1)$ is obtained using eqn. (3.15) or (3.16). Then, with the increasing of B at t_1, the binary index J_j^m for alarms in A_m is updated by checking whether e_j^m belongs to $V'_b \cap V_m$, where V'_b denotes the set of the newly appeared sequence of alarms $\tilde{B} =< b'_1, b'_2, \cdots, b'_l >$. Accordingly, Ψ is then recalculated as Ψ' based on $J_j^m, j = 1, 2, \cdots, |A_m|$, and the subsequence A'_m is extended by including more alarms, i.e.,

$$A'_m =< a_1^m, a_2^m, \cdots, a_\Psi^m, a_{\Psi+1}^m, a_{\Psi+2}^m, \cdots, a_{\Psi'}^m > . \tag{3.24}$$

Accordingly, the index matrix H is extended in two directions and will be updated by H of size $(|B| + |\tilde{B}| + 1) \times (\Psi' + 1)$. To illustrate the online set-based indexing and extension strategy, an example is presented here.

Example 1. Denote a time stamped historical alarm sequence as $A_m =< (5,2), (6,3), (2,7), (5,7.8), (4,10), (6,13), (2,15.5), (8,17) >$ and an incoming alarm sequence as $B =< (7,2), (5,5) >$. Here, the first and second numbers in each bracket denotes the alarm tag and the time stamp, respectively. For simplicity, the unit information is not used here and thus is omitted. The match score, mismatch score, and gap penalty are set as 1, -0.5, and -0.2, respectively. Initially, A_m and B share one common alarm tag 5. At the current time instant t_0, the last position of the common alarm tag 5 in A_m is 4, namely, $\Psi = 4$. Thus, a sub-sequence A'_m is formed as $A'^m =< (5,2), (6,3), (2,7), (5,7.8) >$ from A_m. As shown in Fig 3.1-a, sequence alignment is performed between A^m_s and B where only block 1 is computed. At time instant t_1 when a new alarm $\tilde{B} =< (6,8) >$ appears, it is obtained that $\Psi' = 6$ and $A'^m =< (5,2), (6,3), (2,7), (5,7.8), (4,10), (6,13) >$. Then, as shown in Fig 3.1-b the sequence alignment updates the index matrix H by computing both block 2 and block 3 in two directions. Such a block-wise computation is effective in reducing unnecessary computation in sequence alignment.

3.2.5 Dynamic Ranking of Similar Alarm Floods

After obtaining the three similarity scores, namely, the plant unit-based similarity S_{unit}, the alarm set-based similarity S_{set}, and the sequence based similarity S_{seq}, all alarm floods in the historical database $\mathbb{H}$ are ranked, such that historical alarm floods most similar to the online flood B can be easily observed from the top of the ranking. Here, the ranking is achieved based on S_{unit}, S_{set}, and S_{seq}, according to the following criteria:

- The sequences are ranked in a descending order of the sequence based similarity S_{seq}.
- If two historical sequences hold the same S_{seq}, their ranking will be based

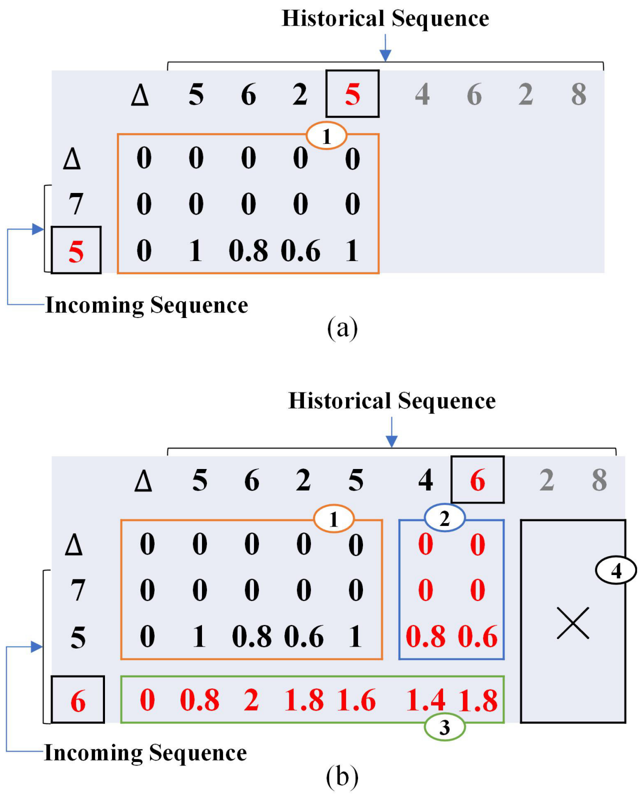


Figure 3.1: An example of set-based indexing and extension strategy: (a) the initial index matrix H at t_0; (b) the updated index matrix H at t_1 with an additional alarm included in the incoming sequence B.

on the alarm set-based similarity S_{set}.

- If two historical sequences hold the same S_{seq} and the same S_{set}, their ranking will be based on the plant unit-based similarity S_{unit}.

It should be noticed that two pre-matching steps are implemented first to screen out alarm floods prior to conducting sequence based similarity analysis; thus, the following conclusion is observed:

$$\begin{cases} S_{set}(A_m, B) = 0, S_{seq}(A_m, B) = 0, \forall A_m \notin \mathbb{H}_{unit}, \\ S_{seq}(A_m, B) = 0, \forall A_m \in \mathbb{H}_{unit}, A_m \notin \mathbb{H}_{set}. \end{cases} \tag{3.25}$$

A ranking list $\mathbb{R}$ of reordered historical alarm flood sequences is then generated as

$$\mathbb{R} = [A'_1, A'_2, \cdots, A'_M]^T, \tag{3.26}$$

Algorithm 1 Real-time pattern matching and dynamic ranking algorithm.

```
1: Input Arguments: $\mathbb{H}$, $W$
2: Output Argument: $\mathbb{R}$
3: Calculate $\eta(t)$ at $t$ after reducing chattering alarms
4: if $\eta(t) \geq \eta_1$ and $\psi(t-1) = 0$ then
5:     $t_0 = t$
6:     $\psi(t) = 1$
7:     Obtain $B$ in $[t_0 - 599, t_0]$
8:     $\mathbf{S} = h(\mathbb{H}, W, B)$
9:     Obtain $\mathbb{R}$ based on $S_{unit}$, $S_{set}$, and $S_{seq}$ in $\mathbf{S}$
10:    $\Upsilon(t_0) = t - t_0$
11: else if $\psi(t) = 1$ then
12:    $\psi(t) = \psi(t-1)$
13:    $t_1 = t$
14:    if $t_1 = t_0 + \max\{T, \Upsilon(t_0)\}$ then
15:        $t_0 = t_1$
16:        if $\Upsilon(t) \leq T$ then
17:            Obtain $\tilde{B}$ in $(t_1 - T, t_1]$
18:        else
19:            Obtain $\tilde{B}$ in $(t_1 - \Upsilon(t_0), t_1]$
20:        end if
21:        $B = B \oplus \tilde{B}$
22:        $\mathbf{S} = h(\mathbb{H}, W, B)$
23:        Update $\mathbb{R}$ based on $S_{unit}$, $S_{set}$, and $S_{seq}$ in $\mathbf{S}$
24:        $\Upsilon(t_0) = t - t_0$
25:    end if
26: else if $\eta(t) < \eta_2$ and $\psi(t-1) = 1$ then
27:    $\psi(t) = 0$
28: end if
```

where $A'_m \in \mathbb{H}$ ranks at the mth place, i.e., A'_m is the mth most similar alarm flood of B in the historical database $\mathbb{H}$. To present comprehensive information to the plant operator, an alternative form of the ranking list $\mathbb{R}$ is

$$\mathbb{R} = [R_1, R_2, \cdots, R_M]^T, \tag{3.27}$$

where R_m is a tuple containing all related information of A'_m, such as the start time τ^s_m, the time length τ^d_m, the set U_m of involved plant units, as well as the similarity scores ($S_{unit}(A_m, B)$, $S_{set}(A_m, B)$, and $S_{seq}(A_m, B)$). The produced ranking list can provide an important insight to facilitate prompt corrective

Algorithm 2 $\mathbf{S} = h(\mathbb{H}, W, B)$: Similarity calculation.

```
1: Input Arguments: $\mathbb{H}$, $W$, $B$
2: Output Argument: $\mathbf{S}$
3: $\mathbf{S} = [0]_{M\times 3}$
4: for $\forall A_m \in \mathbb{H}$ do
5:     Calculate $S_{unit}(A_m, B)$ using eqn. (3.10)
6:     $\mathbf{S}(m, 1) = S_{unit}(A_m, B)$
7: end for
8: Get $\mathbb{H}_{unit}$ using eqn. (3.11)
9: for $\forall A_m \in \mathbb{H}$ do
10:     if $A_m \in \mathbb{H}_{unit}$ then
11:         Calculate $S_{set}(A_m, B)$ using eqn. (4.12)
12:     else
13:         $S_{set}(A_m, B) = 0$
14:     end if
15:     $\mathbf{S}(m, 2) = S_{set}(A_m, B)$
16: end for
17: Get $\mathbb{H}_{set}$ using eqn. (3.14)
18: for $\forall A_m \in \mathbb{H}$ do
19:     if $A_m \in \mathbb{H}_{set}$ then
20:         Obtain $A'_m$ by set based indexing and extension
21:         Compute $H$ by comparing $A'_m$ and $B$
22:         Calculate $S_{seq}(A_m, B)$ based on $H$ using eqn. (3.17)
23:     else
24:         $S_{seq}(A_m, B) = 0$
25:     end if
26:     $\mathbf{S}(m, 3) = S_{seq}(A_m, B)$
27: end for
```

actions by referencing the information of similar alarm floods in the historical database.

It should be noticed that at the triggering time instant t_0, the online alarm flood B is short and incomplete. Thus, there may exist many historical alarm floods similar to the online one, making it difficult for plant operators to recognize similar situations. Thus, this work proposes to recalculate the similarity scores based on the incoming new alarms in the online alarm flood, and accordingly update the ranking list $\mathbb{R}$. With more alarms included, the pattern matching results become more reliable, making the ranking list more trustworthy compared to the initial one obtained at the triggering time instant.

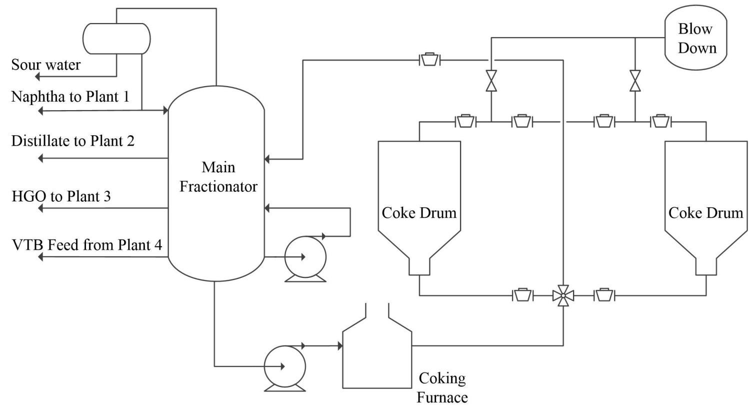


Figure 3.2: Schematic of a typical delayed coking plant.

Then, the operators can focus more on those top ranked alarm floods. Thus, updating the ranking list based on the growing of the online flood is necessary and also helpful for the operators to identify the truly matched situation.

Further, a time window T is used in the online calculation to prevent updating the ranking too frequently; the strategy is presented as follows: The initial list $\mathbb{R}$ is firstly obtained for B at time instant t_0 when it is detected $\eta(t_0) \geq \eta_1$, i.e., an alarm flood is found to arise. Theoretically, $\mathbb{R}$ is supposed to be updated at t_1 whenever a new alarm is added to B after t_0. However, it is possible that a series of new alarms may appear within a short period, making the real-time pattern matching hardly to respond to each new alarm. Accordingly, the updating of the ranking list $\mathbb{R}$ is presented as follows: Assume that the execution time of real-time pattern matching between $B = B \oplus \tilde{B}$ and $\forall A_m \in \mathbb{H}$ is $\Upsilon(t)$. Then, given $\mathbb{R}$ for B obtained at t,

- $\mathbb{R}$ is updated at time instant $t + T$ based on $\tilde{B} =< b'_1, b'_2, \cdots, b'_l >$, namely, a sequence of alarms that appears within $(t, t + T]$ if $\Upsilon(t) \leq T$;
- otherwise, $\mathbb{R}$ is updated at time instant $t+\Upsilon(t)$ based on $\tilde{B} =< b'_1, b'_2, \cdots, b'_l >$ that appears within $(t, t + \Upsilon(t)]$ if $\Upsilon(t) > T$.

The above iteration will continue until $\eta(t) < \eta_2$, i.e., the alarm flood is detected to be over. Here, T is a user-defined time window that decides how frequent to update the ranking list. Generally, one does not need to update the list in each second; instead, it should wait for a period T before the next round of real-time pattern matching is executed for new alarms to update the ranking list. It should be noticed that a large T may make the ranking list not updated timely and thus delay the operator's action to handle the alarm flood, while a small T can lead to the execution of the algorithm often incomplete within T. Thus, the value of T can be set based on how fast the algorithm is executed on the given alarm flood database, as well as the requirements from plant operators, namely, how frequent they want to get the ranking list updated.

The real-time pattern matching and dynamic ranking method is summarized in Algorithms 1 and 2. The required inputs of the algorithm are the historical database $\mathbb{H}$ and the weighting matrix W for all alarm floods in $\mathbb{H}$, which are prepared offline. For online analysis, whenever it detects $\eta(t) \geq \eta_1$, the real-time pattern matching is conducted by comparing the initial alarm flood sequence B with all historical sequences in $\mathbb{H}$. Plant unit and alarm set based pre-matching steps calculate S_{unit} and S_{set}, and exclude irrelevant historical sequences. Then, sequence alignment measures S_{seq}. Given the calculated similarity scores from $\mathbf{S} = h(\mathbb{H}, W, B)$ in line 8 of Algorithm 1, a ranking list $\mathbb{R}$ is obtained and presented to the user. While the online sequence B is increasing with the addition of a series of new alarms as $\tilde{B}$, the pattern matching obtains the similarity scores in line 22 of Algorithm 1 and the ranking list $\mathbb{R}$ is updated accordingly. The pattern matching in line 22 recalculates S_{unit} and S_{set}, and adjusts the two sets $\mathbb{H}_{unit}$ and $\mathbb{H}_{set}$; then, incremental sequence alignment is applied to compare $\tilde{B}$ rather than the complete sequence B with all historical sequences in $\mathbb{H}_{set}$. The ranking of similar historical sequences will be updated with the increasing of the online sequence B until $\eta(t) < \eta_2$, as indicated in line 27 of Algorithm 1.

Remark 1: This proposed method is different from the existing literature on alarm flood analysis in the following aspects: 1) This work provides a different perspective to handle alarm floods by real-time pattern matching and ranking of alarm floods, while the methods in [25, 43, 56, 72] aimed at mining interesting alarm patterns, and the methods in [12, 13, 38, 51] mainly addressed the problems on classification of alarm floods and fault diagnosis; the studied problems and objectives are different. 2) This work solves online similarity analysis by incremental sequence alignment and dynamic ranking, while the methods in [10, 27, 35, 71] belong to offline analysis. 3) The set-based indexing & extension strategy and the dynamic ranking of alarm floods in this chapter are new ideas that have not yet been studied in any other publications (to the best knowledge of the authors), making them unique compared to other online alarm flood analysis methods in [34, 37, 69].

3.3 Case Study

This section provides a case study to illustrate the proposed method. A historical data set was collected from a coking plant of an oil refinery. In this plant, heavy petroleum residues are transformed into lighter gaseous liquid products and solid coke through a thermal cracking process. As illustrated in Fig. 3.2, the coking plant consists of several units, including coke drums, coking furnace, fractionator unit, pumps, and controllers. These units were already defined and configured in the system based on their processing functionalities.

There were 2609 unique alarm tags from 9 units reported in the historical alarm data over the time period from November 1, 2019, to April 30, 2020. Off-delay timers of 300 seconds were implemented to remove the chattering alarms. Following the definition in [30], 103 alarm floods were extracted to form the database $\mathbb{H}$ of historical alarm flood sequences. The average length of alarm floods was 29.75. The longest and shortest alarm floods contained 609 and 10 alarms, respectively. Table 3.1 shows an example of an extracted alarm flood sequence, which is taken as the online sequence B in this case study. It included 24 alarms from 3 different plant units. The proposed method was applied to identify similar historical sequences from $\mathbb{H}$ by conducting real-time

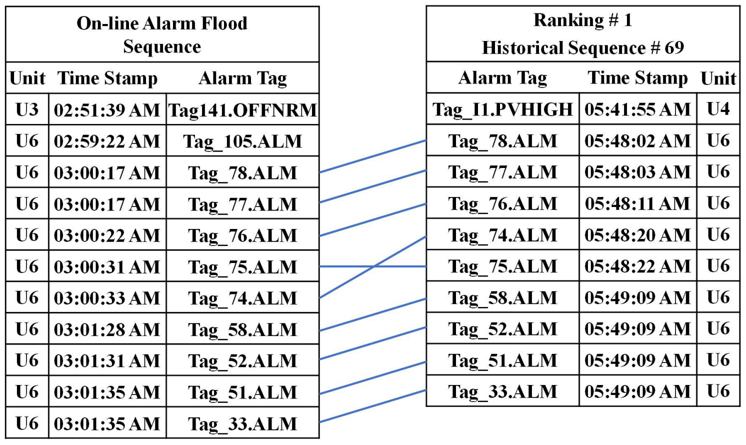
Table 3.1: An example of the alarm flood sequence, which was taken as the online sequence B.

Alarm Tag	Time Stamp	Unit
Tag_141.OFFNRM	2020/02/02 02:51:39 AM	Unit 3
Tag_105.ALM	2020/02/02 02:59:22 AM	Unit 6
Tag_78.ALM	2020/02/02 03:00:17 AM	Unit 6
Tag_77.ALM	2020/02/02 03:00:17 AM	Unit 6
Tag_76.ALM	2020/02/02 03:00:22 AM	Unit 6
Tag_75.ALM	2020/02/02 03:00:31 AM	Unit 6
Tag_74.ALM	2020/02/02 03:00:33 AM	Unit 6
Tag_58.ALM	2020/02/02 03:01:28 AM	Unit 6
Tag_52.ALM	2020/02/02 03:01:31 AM	Unit 6
Tag_51.ALM	2020/02/02 03:01:35 AM	Unit 6
Tag_33.ALM	2020/02/02 03:01:35 AM	Unit 6
Tag_141.OFFNRM	2020/02/02 03:05:26 AM	Unit 3
Tag_6A.PVHI	2020/02/02 03:07:19 AM	Unit 5
Tag_9A.PVLO	2020/02/02 03:08:22 AM	Unit 6
Tag_78.ALM	2020/02/02 03:10:08 AM	Unit 6
Tag_77.ALM	2020/02/02 03:10:09 AM	Unit 6
Tag_76.ALM	2020/02/02 03:10:11 AM	Unit 6
Tag_75.ALM	2020/02/02 03:10:25 AM	Unit 6
Tag_74.ALM	2020/02/02 03:10:26 AM	Unit 6
Tag_58.ALM	2020/02/02 03:11:03 AM	Unit 6
Tag_52.ALM	2020/02/02 03:11:07 AM	Unit 6
Tag_51.ALM	2020/02/02 03:11:11 AM	Unit 6
Tag_33.ALM	2020/02/02 03:11:15 AM	Unit 6
Tag_7B.PVLO	2020/02/02 03:16:28 AM	Unit 5

pattern matching between B and all sequences in $\mathbb{H}$. The results were exported as a ranking list that was updating with the presence of more incoming alarms in B.

At the triggering time instant "03:01:35" of B, the unit based screening and set based pre-matching were applied first to exclude irrelevant sequences from $\mathbb{H}$. Then, a refined database $\mathbb{H}_{set}$ containing 11 historical sequences nominated from $\mathbb{H}$ was obtained. Following pre-matching steps, set-based indexing and extension removed the sub-sequences from the 11 nominated sequences; these sub-sequences did not bear any similarity with the online sequence. Then, sequence alignment was implemented between B and each of the 11 sequences in $\mathbb{H}_{set}$. At the triggering time instant, ranking of 11 nominated sequences was presented according to the extent of similarity with the online flood sequence. Several attributes, such as the start time, units of annunciations, and three similarity scores were provided to facilitate industrial operators in decision making.

Table 3.2 shows that the ranking list $\mathbb{R}$ at the triggering time instant of



On-line Alarm Flood Sequence		
Unit	Time Stamp	Alarm Tag
U3	02:51:39 AM	Tag141.OFFNRM
U6	02:59:22 AM	Tag_105.ALM
U6	03:00:17 AM	Tag_78.ALM
U6	03:00:17 AM	Tag_77.ALM
U6	03:00:22 AM	Tag_76.ALM
U6	03:00:31 AM	Tag_75.ALM
U6	03:00:33 AM	Tag_74.ALM
U6	03:01:28 AM	Tag_58.ALM
U6	03:01:31 AM	Tag_52.ALM
U6	03:01:35 AM	Tag_51.ALM
U6	03:01:35 AM	Tag_33.ALM

Ranking # 1 Historical Sequence # 69		
Alarm Tag	Time Stamp	Unit
Tag_I1.PVHIGH	05:41:55 AM	U4
Tag_78.ALM	05:48:02 AM	U6
Tag_77.ALM	05:48:03 AM	U6
Tag_76.ALM	05:48:11 AM	U6
Tag_74.ALM	05:48:20 AM	U6
Tag_75.ALM	05:48:22 AM	U6
Tag_58.ALM	05:49:09 AM	U6
Tag_52.ALM	05:49:09 AM	U6
Tag_51.ALM	05:49:09 AM	U6
Tag_33.ALM	05:49:09 AM	U6

Figure 3.3: Sequence alignment result between the historical sequence # 69 and B at the triggering time instant "03:01:35"

.

the online alarm flood B. The historical alarm flood sequences were ranked by using S_{seq} as the first criterion, S_{set} as the second criterion, and S_{unit} as the third criterion. It can be observed that all historical sequences in $\mathbb{H}_{set}$ held significant sequence based similarity scores with B; among them, the historical sequence # 69 was found to be the most similar one in terms of S_{seq}. It is reasonable to see many alarm floods were similar to B at the early stage of B, since B was short and incomplete, and thus could be included in many historical sequences, causing difficulties in differentiating among these historical alarm floods. Fig. 3.3 shows the alignment between the online alarm flood sequence B and the historical sequence # 69 at the triggering time instant t_0. There were 9 pairs of matched alarms, Tag_78.ALM, Tag_77.ALM, $\cdots$, Tag_33.ALM from plant unit U6, annunciated sequentially in almost the same order.

Later, the online sequence B was increasing with more alarms coming. In the next 9 minutes (which was set as the updating period), there were 8 more alarms annunciated, including Tag_141.OFFNRM, Tag_6A.PVHI, Tag_9A.PVLO, Tag_78.ALM, Tag_77.ALM, Tag_76.ALM, Tag_75.ALM, and Tag_74.ALM. Given the new alarms added to B, the three similarity scores were calculated to update the ranking list. Table 3.3 shows the updated ranking of top 20 most

Table 3.2: Ranking of similar historical alarm flood sequences at the triggering instant "03:01:35" of the online alarm flood. There were 11 sequences nominated from $\mathbb{H}$ using unit based screening and alarm set-based pre-matching. For the remaining historical sequences from $\mathbb{H}$, their sequence based similarity scores were 0's.

Ranking	Alarm flood ID	Start time	Unit of Annunciations	S_{unit}	S_{set}	S_{seq}
1	69	2020/02/02 05:41	U4, U6	0.9045	0.8581	0.9
2	68	2020/02/02 03:49	U3, U4, U6	0.9607	0.8770	0.8909
3	74	2020/02/02 10:22	U3, U5, U6	0.9534	0.8182	0.8182
4	75	2020/02/03 10:44	U5, U6	0.8624	0.8182	0.8182
5	76	2020/02/04 12:34	U5, U6	0.8624	0.8182	0.8182
6	72	2020/02/03 06:37	U6	0.9534	0.7833	0.8182
7	71	2020/02/03 04:33	U4, U6	0.8257	0.7833	0.8182
8	73	2020/02/03 09:53	U3, U4, U6	0.9636	0.7252	0.8182
9	70	2020/02/03 03:04	U3, U5, U6	0.8864	0.7252	0.8182
10	77	2020/02/04 04:07	U4, U6	0.8451	0.7252	0.8182
11	78	2020/02/03 06:37	U4, U6	0.8594	0.6784	0.8182
...	...	...	...	...	...	...

Table 3.3: Updated ranking of top 20 most similar historical alarm flood sequences following 8 new alarms added to the online sequence at "03:10:35".

Ranking	Alarm flood ID	Start time	Unit of Annunciations	S_{unit}	S_{set}	S_{seq}
1	69	2020/02/02 05:41	U4, U6	0.8429	0.8143	0.9
2	76	2020/02/04 12:34	U5, U6	0.9459	0.8471	0.8727
3	68	2020/02/02 03:49	U3, U4, U6	0.9087	0.8441	0.8307
4	74	2020/02/02 10:22	U3, U5, U6	1	0.8471	0.8181
5	75	2020/02/03 10:44	U5, U6	0.9459	0.7764	0.8181
6	72	2020/02/03 06:37	U6	0.8885	0.7433	0.75
7	71	2020/02/03 04:33	U4, U6	0.7694	0.7433	0.75
8	70	2020/02/03 03:04	U3, U5, U6	1	0.6882	0.6428
9	73	2020/02/03 09:53	U3, U4, U6	0.9114	0.6882	0.6428
10	77	2020/02/03 04:07	U4, U5	0.7875	0.6882	0.6428
11	78	2020/02/02 03:49	U4, U6	0.8009	0.6437	0.5625
12	89	2020/03/17 06:07	U3, U5	0.4588	0.1324	0.0833
13	4	2019/11/07 01:57	U4, U5, U6	0.9056	0.0936	0.0833
14	87	2020/03/06 07:10	U4, U5, U6	0.7870	0.0636	0.0769
15	2	2019/11/04 06:07	U4, U5	0.3003	0.0867	0.0714
16	10	2019/11/17 01:04	U3, U5, U6	1	0.0573	0.0625
17	48	2019/12/28 12:29	U3, U4, U5, U6	0.9607	0.1102	0.0526
18	47	2019/11/17 01:04	U3, U5	0.4588	0.0691	0.0526
19	28	2019/12/23 02:58	U3, U4, U5, U6	0.9888	0.0683	0.0526
20	9	2019/11/16 10:30	U4, U5, U6	0.8820	0.0676	0.0526
...	...	...	...	...	...	...

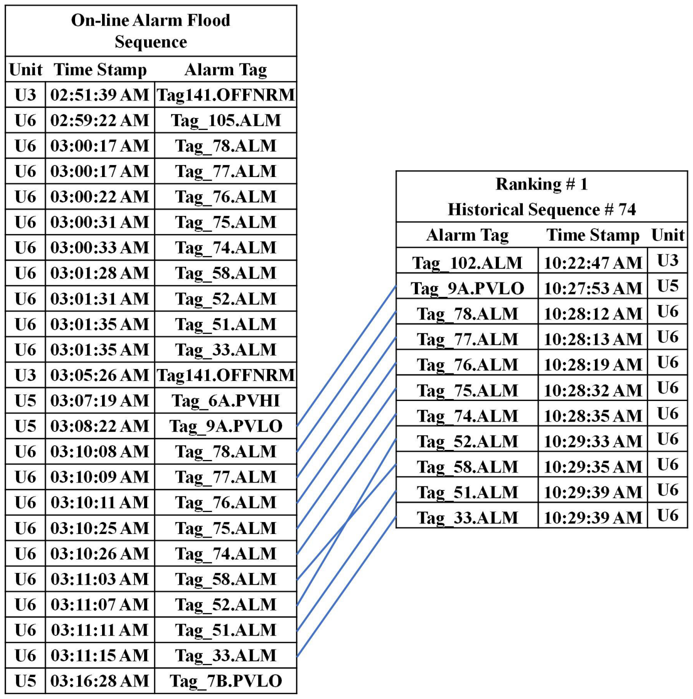
similar historical alarm flood sequences at "03:10:35" when 8 new alarms were added to B. It can be seen that the number of similar alarm floods holding non-zero sequence based similarity scores increased, but there were still only 11 historical alarm floods having significant similarity scores with the online sequence B. The most similar flood was still # 69.

Table 3.4: Updated ranking of top 20 most similar historical alarm flood sequences at the termination of the online alarm flood.

Ranking	Alarm flood ID	Start time	Unit of Annunciations	S_{unit}	S_{set}	S_{seq}
1	74	2020/02/02 10:22	U3, U5, U6	1	0.8483	0.9090
2	69	2020/02/02 05:41	U4, U6	0.8440	0.8215	0.9
3	75	2020/02/03 10:44	U5, U6	0.9574	0.8483	0.8909
4	76	2020/02/04 12:34	U5, U6	0.9574	0.8483	0.8727
5	68	2020/02/02 03:49	U3, U4, U6	0.8987	0.8397	0.8307
6	72	2020/02/03 06:37	U6	0.8897	0.75	0.75
7	71	2020/02/03 04:33	U4, U6	0.7705	0.75	0.75
8	70	2020/02/03 03:04	U3, U5, U6	1	0.6943	0.6428
9	73	2020/02/03 09:53	U3, U4, U6	0.9013	0.6943	0.6428
10	77	2020/02/03 04:07	U4, U5	0.7886	0.6943	0.6428
11	78	2020/02/02 03:49	U4, U6	0.8020	0.6495	0.5625
12	19	2019/11/21 08:18	U4, U5	0.3162	0.0645	0.1
13	101	2020/04/28 04:33	U4, U5, U6	0.8660	0.0615	0.0909
14	89	2020/03/17 06:07	U3, U5	0.4564	0.1178	0.0833
15	4	2019/11/07 01:57	U4, U5, U6	0.9166	0.0833	0.0833
16	87	2020/03/06 07:10	U4, U5, U6	0.7966	0.1132	0.0769
17	2	2019/11/04 06:07	U4, U5	0.3273	0.1336	0.0714
18	10	2019/11/17 01:04	U3, U5, U6	1	0.0510	0.0625
19	49	2019/12/28 02:56	U3, U5, U6	1	0.0468	0.0526
20	86	2020/02/25 12:29	U3, U5, U6	1	0.0445	0.0476
...	...	...	...	...	...	...

Table 3.4 shows the updated ranking of the top 20 most similar historical alarm flood sequences at the termination of the online alarm flood. Historical alarm flood # 74 was found to be most similar to the online sequence B and thus replaced historical sequence # 69 at the top of the ranking list. Fig. 3.4 shows the alignment result between the complete online alarm flood sequence B and the historical sequence # 74; there were 10 pairs of matched alarms. It can be observed in Fig. 3.4 that a pattern of alarms, starting from Tag_78.ALM to Tag_33.ALM was repeated in the online sequence B and also existed in historical sequence # 74. Compared to the sequence # 69 in Fig. 3.3, historical sequence # 74 held one more common alarm (Tag_9A.PVLO), leading to a higher sequence based similarity score.

In addition, it can also be observed that a few historical alarm floods ranking at the top of Table 3.4 held significant similarity scores with B. To explain this phenomenon, offline similarity analysis was conducted among the nominated sequences from $|\mathbb{H}_{set}|$ and results were presented in the similarity color map shown in Fig. 3.5. In Fig. 3.5, X and Y-axis represent the alarm flood IDs, and the color bar represents the extent of similarity. Flood sequence



On-line Alarm Flood Sequence		
Unit	Time Stamp	Alarm Tag
U3	02:51:39 AM	Tag141.OFFNRM
U6	02:59:22 AM	Tag_105.ALM
U6	03:00:17 AM	Tag_78.ALM
U6	03:00:17 AM	Tag_77.ALM
U6	03:00:22 AM	Tag_76.ALM
U6	03:00:31 AM	Tag_75.ALM
U6	03:00:33 AM	Tag_74.ALM
U6	03:01:28 AM	Tag_58.ALM
U6	03:01:31 AM	Tag_52.ALM
U6	03:01:35 AM	Tag_51.ALM
U6	03:01:35 AM	Tag_33.ALM
U3	03:05:26 AM	Tag141.OFFNRM
U5	03:07:19 AM	Tag_6A.PVHI
U5	03:08:22 AM	Tag_9A.PVLO
U6	03:10:08 AM	Tag_78.ALM
U6	03:10:09 AM	Tag_77.ALM
U6	03:10:11 AM	Tag_76.ALM
U6	03:10:25 AM	Tag_75.ALM
U6	03:10:26 AM	Tag_74.ALM
U6	03:11:03 AM	Tag_58.ALM
U6	03:11:07 AM	Tag_52.ALM
U6	03:11:11 AM	Tag_51.ALM
U6	03:11:15 AM	Tag_33.ALM
U5	03:16:28 AM	Tag_7B.PVLO

Ranking # 1 Historical Sequence # 74		
Alarm Tag	Time Stamp	Unit
Tag_102.ALM	10:22:47 AM	U3
Tag_9A.PVLO	10:27:53 AM	U5
Tag_78.ALM	10:28:12 AM	U6
Tag_77.ALM	10:28:13 AM	U6
Tag_76.ALM	10:28:19 AM	U6
Tag_75.ALM	10:28:32 AM	U6
Tag_74.ALM	10:28:35 AM	U6
Tag_52.ALM	10:29:33 AM	U6
Tag_58.ALM	10:29:35 AM	U6
Tag_51.ALM	10:29:39 AM	U6
Tag_33.ALM	10:29:39 AM	U6

Figure 3.4: Sequence alignment result between the historical sequence # 74 and B at the termination of online alarm flood.

# 67 was the online alarm flood sequence. It can be seen that there was a cluster of historical alarm floods holding high similarities with each other. According to the process knowledge, these alarm floods happened in the same plant units and had similar root causes. Thus, it is validated from Fig. 3.5 that the final ranking in Table 3.4 was reasonable given a few historical alarm floods ranking at the top. Accordingly, the operator would be more confident to judge the root cause and respond to the online alarm flood based on the information related to all top floods in this ranking list.

In Section 3.2, there are three key strategies used to reduce unnecessary computation, including I) the plant unit based screening, II) the alarm set based pre-matching, and III) the set based indexing and extension. To demonstrate the effectiveness of these strategies in computation reduction, Table 3.5 presents the computational times by the proposed method with full steps and those excluding some of the above three strategies. In the simulation, the computational times were recorded at three time instants of the online alarm

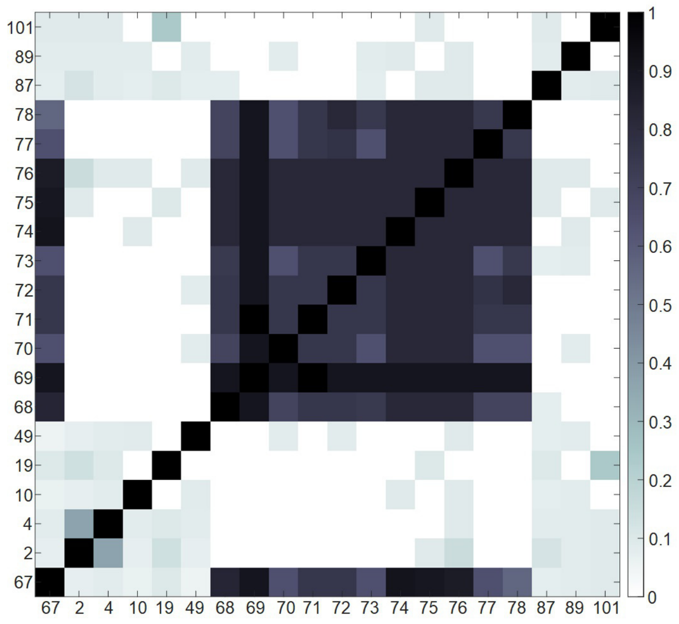


Figure 3.5: Validation based on similarity color map obtained for the complete online alarm flood and the historical alarm flood sequences ranking at the top.

Table 3.5: Comparison of computational times by the proposed method with full steps and those excluding some computation reduction strategies. These strategies include I) the plant unit based screening, II) the alarm set based pre-matching, and III) the set based indexing and extension. The computational times at three time instants of the online alarm flood were recorded, corresponding to when the ranking list was produced or updated as shown in Tables 3.2, 3.3, and 3.4.

Time instant	Proposed method with full steps	Excluding III	Excluding II & III	Excluding I, II, & III
03:01:35	1.35 s	3.52 s	25.83	27.44
03:10:35	7.54 s	15.63 s	23.17	24.85
End	5.09 s	7.68 s	13.91	14.87

flood, including “03:01:35”, “03:10:35”, and the end of the online alarm flood, which were corresponding to when the ranking list (shown in Tables 3.2, 3.3, and 3.4) was produced or updated. According to the result in Table 3.5, the computational times by the proposed method with full steps were much shorter compared to those excluding all the three computation reduction strategies. Specifically, comparing the last two columns, the computational times reduced by strategy I (namely, the plant unit based screening) were 1.61 s, 1.68 s, and 0.96 s at the three time instants, respectively. Comparing the 3rd and 4th columns, strategy II (namely, the alarm set based pre-matching) led to a large reduction of computational time with a reduction rate ranging from 32.5% to 86.3%. Comparing the 2nd and 3rd columns, the computational time reduced by strategy III (namely, the set based indexing and extension) was also considerable. Thus, it can be concluded that the reduction of computational time using the three strategies is significant. Such computation reduction strategies in the proposed method will be very helpful in applications, especially when the database of historical alarm floods is large.

3.4 Summary

In this chapter, we addressed the problem of predicting the incoming alarm flood sequence and providing decision support to plant operators during an ongoing alarm flood situation. Initially, we introduced the alarm flood problem and presented the concept of real-time pattern matching framework as a means to provide decision support. Furthermore, we discussed the detection criteria and representation of alarm flood sequences. To effectively manage alarm floods, we proposed a comprehensive method comprising several key steps. These steps included plant unit-based screening, alarm set-based pre-matching, incremental sequence alignment, online set-based indexing and extension, and dynamic ranking of similar alarm floods. By integrating these steps, we aimed to provide a robust solution to the alarm flood problem and assist plant operators to make informed decisions. Finally, we presented the case study to demonstrate the effectiveness of the proposed method, where we used the real alarm & event log from a coking plant of an oil refinery.

Chapter 4

An Association Rule Mining Approach to Predict Alarm Events in Industrial Alarm Floods

This chapter studies the problem of upcoming alarm events prediction in an alarm flood situation. Industrial operators often experience overwhelming situations during ongoing alarm floods due to high alarm rates. In such situations, real-time assistance in the form of prediction of upcoming alarm events can ease off the decision-making for industrial operators. The main contribution of this work lies in a novel association rule mining approach for real-time prediction of alarm events and their corresponding times of annunciation during an ongoing alarm flood. The proposed method is capable of performing predictions at the triggering instant and modifying the predictions with the increasing of the ongoing alarm flood. The proposed method is implemented mainly in the following steps: 1) A Compact Prediction Tree (CPT) model is modified with new features, namely, the time table and co-occurrence matrix, and constructed based on historical alarm sequences; 2) An alarm relevancy detection strategy is designed to detect and eliminate irrelevant alarms in alarm floods; 3) An online alarm prediction algorithm is designed to predict upcoming alarms at the early stage of the ongoing alarm flood; 4) The confidence intervals of the time differences between the annunciations of subsequent predicted alarm events are calculated for time prediction. To demonstrate the effectiveness of the proposed method, an industrial case study based on real alarm & event logs from an oil refinery is provided.

4.1 Problem Description

A historical alarm flood sequence A_m is a unique representation of an underlying abnormality which causes a specific set of alarms $\{a_k^m\}_{k=1}^{|A_m|}$ to appear chronologically in the sequence, which is denoted as

$$A_m =< a_1^m, a_2^m, a_3^m, \cdots, a_{|A_m|}^m >, \tag{4.1}$$

where $m = 1, 2, \cdots, |\mathbb{T}|$; $|\cdot|$ means the length of a sequence or the cardinality of a set; $|\mathbb{T}|$ represents the set of all the historical alarm flood sequences and is denoted by $\mathbb{T} = \{A_m : m = 1, 2, \cdots, |\mathbb{T}|\}$. In Eq. (4.1), $a_i^m = (e_i^{a^m}, t_i^{a^m})$, $i = 1, 2, 3, \cdots, |A_m|$. $e_i^{a^m}$ and $t_i^{a^m}$ are the alarm tag and annunciation time of a_i^m. $e_i^m \in \mathbb{A}$; $\mathbb{A}$ indicates the set of all unique alarms configured in the plant. The binary signal presenting the alarm annunciation of $e_i^{a^m}$ is given by

$$x_i^m(t) = \begin{cases} 1 & \text{if } \chi_{a_i^m}(t) \notin \tau_s, \\ 0 & \text{o.w.}, \end{cases} \tag{4.2}$$

where $\chi_{a_i^m}$ indicates the corresponding process variable of a_i^m and τ_s indicates the steady state operating condition for $\chi_{a_i^m}$.

Alarm floods can be triggered from fault propagation, change of operating modes, or switching of system conditions. The benchmark thresholds for alarm flood detection are $\zeta_1 = 10$ and $\zeta_2 = 5$ alarms per 10 minutes for each operator as the start and termination, respectively. An online alarm flood sequence at the triggering instant t is denoted as

$$B =< b_1, b_2, \cdots, b_k >, \tag{4.3}$$

where $b_i = (e_i^b, t_i^b)$ and $e_i^b \in \mathbb{A}$, $i = 1, 2, \cdots, k$. When an alarm flood is detected, i.e., $\zeta(t) \geq \zeta_1$, the online sequence should have 10 or more alarms appearing within $[t - \beta + 1, t]$, where $\beta = 600$s and $\zeta(t)$ denotes the current alarm rate. Such early captured alarms form an integral part of a pattern associated with a certain root cause and can be exploited to identify similar sequences from $\mathbb{T}$ for meaningful information on upcoming abnormalities. At any instant of B, a set of historical alarm flood sequences that are closely similar to B is denoted as $\mathbb{S} = \{A_n | s = 1, 2, \cdots, |\mathbb{S}|, A_n \in \mathbb{T}\}$. Thus, the prediction problem is formulated as: Given the existing online sequence $B_{|:t-1|}$ with alarms annunciated until time instant $t-1$ and the set of similar sequences $\mathbb{S} \subseteq \mathbb{T}$, the objective is to obtain $\tilde{B}_t = \{b_{|B|+1}, b_{|B|+2}, \cdots, b_{|\tilde{B}_t|}\}$, namely, the set of predicted alarms at time instant t.

While predicting the upcoming alarm events in an ongoing alarm flood, the difficulties and corresponding solutions in this study are as follows:

1) Online prediction is sensitive to computational time. Here, a Compact Prediction Tree model is integrated to address the spatial complexity and modified with new features to conduct most of the computation offline.

2) In an ongoing alarm flood, some alarms might be irrelevant to a true pattern, such as nuisance alarms caused by disturbances. Thus, this work designs an alarm relevancy detection strategy to detect and eliminate irrelevant alarms based on co-occurrence frequency and confidence.

3) With more alarms occurring in the ongoing alarm flood, the prediction needs to update along with time so as to ensure the prediction accuracy. Motivated by this, an online alarm prediction method is proposed to predict upcoming alarms at the triggering instant and update the predictions at subsequent instants of an ongoing flood.

4) Predicting the time of annunciation for a new alarm would be helpful to the decision-making to handle an ongoing alarm flood. Therefore, this work calculates the time intervals for subsequent predicted upcoming alarms.

4.2 Methodology

This section presents the systematic alarm prediction method, including the offline training of a compact prediction tree, real-time exclusion of irrelevant alarms from the online sequence, evaluation criteria of the candidate alarms, and prediction of time for the upcoming alarm events.

4.2.1 Training of Compact Prediction Tree

The Compact Prediction Tree (CPT) provides a lossless compression of the training set in data mining [18]. Frequent subsequence compression (FSC) and Simple branches compression (SBC) strategies in [17] are included to improve the computational efficiency. CPT generates three distinct features in the training, including a Prediction Tree, an Inverted Index, and a Lookup Table [18]. This subsection designs a new feature named Time Table, which consists of unique alarms and their corresponding time stamps of annunciations in different historical sequences. To illustrate the training of the CPT, the following set of time-stamped alarm sequences is used:

$$S_1 = <(\text{Tag\_A}, 02\colon 21\colon 31), (\text{Tag\_B}, 02\colon 21\colon 38), (\text{Tag\_D}, 02\colon 24\colon 26), (\text{Tag\_A}, 02\colon 37\colon 11)>$$
$$S_2 = <(\text{Tag\_A}, 04\colon 00\colon 37), (\text{Tag\_B}, 04\colon 00\colon 58), (\text{Tag\_C}, 04\colon 02\colon 51), (\text{Tag\_E}, 04\colon 15\colon 05)>$$
$$S_3 = <(\text{Tag\_E}, 05\colon 02\colon 42), (\text{Tag\_C}, 05\colon 03\colon 21), (\text{Tag\_D}, 05\colon 19\colon 26), (\text{Tag\_E}, 05\colon 19\colon 57)>$$

Here, the set of unique alarms makes up an alphabet $\sum = \{\text{Tag\_A}, \text{Tag\_B}, \text{Tag\_C}, \text{Tag\_D}, \text{Tag\_E}\}$, and each element in a sequence is denoted as (e, t), where e and t indicate the alarm tag and the corresponding time of annunciation, respectively. In the training, sequences are inserted chronologically to build a prediction tree [18]. Fig. 4.1 shows the initial prediction tree, Inverted Index, and Lookup Table following the insertion of sequences S_1, S_2, and S_3. The training consists of the following steps:

1) The first element of a training sequence is connected to the root node of the prediction tree if the root node has no child nodes [18]. Then, the remaining elements from the sequence are connected to the immediately

preceding elements to form a path representing the whole sequence. If any other training sequence has a prefix common to the first sequence, the following elements of this training sequence form a path starting from the last element of the prefix that already exists in the first training sequence and is connected to the root node. For instance, in Fig. 4.1, sequences S_2 and S_1 have a common prefix $< \text{Tag\_A}, \text{Tag\_B} >$. Then, the following elements of S_2, namely, $< \text{Tag\_C}, \text{Tag\_E} >$ are connected to the prefix $< \text{Tag\_A}, \text{Tag\_B} >$ of S_1 that is already connected to the root node. Analogously, each training sequence from $\mathbb{T}$ is inserted one by one to construct the CPT.

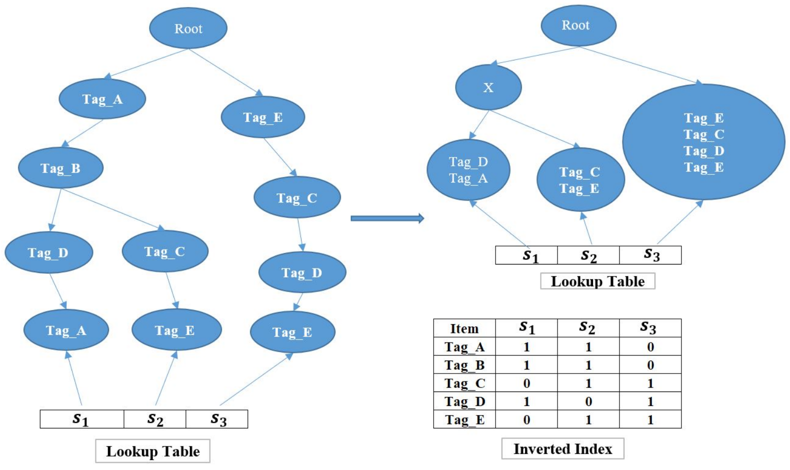
2) The Inverted Index includes each unique element as a key that exists in a training sequence [18]. Following the training process, the sequences containing an unique element can be identified using the bitset from the Inverted Index, which is a combination of 1 and 0. For instance, in Fig. 4.1, the bitset for Tag_A with respect to sequences S_1, S_2, and S_3 is 110, indicating that Tag_A only exists in S_1 and S_2.

3) A Lookup Table is introduced to locate a sequence within the prediction tree using the sequence identifications (IDs) [18]. As shown in Fig. 4.1, the Lookup Table is pointed to the last node of each training sequence.

Table 4.1: Time Table (TT).

Alarm	Time stamps of occurrences		
Tag_A	[02:21:31]	[02:37:11]	[04:00:37]
Tag_B	[02:21:38]	[04:00:58]	
Tag_C	[04:02:51]	[05:03:21]	
Tag_D	[02:24:26]	[05:19:26]	
Tag_E	[04:15:05]	[05:19:57]	

As shown in Fig. 4.1, the prediction tree results in a compressed version after integrating FSC and SBC strategies. Table 4.1 shows the Time Table, where the time stamps of alarm occurrences are inserted in the training and assigned with a unique key, namely, an alarm tag. Based on the Time Table,



Item	s_1	s_2	s_3
Tag_A	1	1	0
Tag_B	1	1	0
Tag_C	0	1	1
Tag_D	1	0	1
Tag_E	0	1	1

Figure 4.1: Example of constructing a Compact Prediction Tree.

a co-occurrence matrix C_M is obtained as

$$\begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{array} \begin{array}{c} \begin{array}{ccccc} c_1 & c_2 & c_3 & c_4 & c_5 \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}, \tag{4.4}$$

where each element in C_M is given by

$$C_M = \sum_{k=1}^{|\mathbb{T}|} \sum_{i=1}^{l} \sum_{j=1}^{l} \begin{cases} f_{ij} + 1 & \text{if } |\Delta t_{kij}| \leq \sigma, \\ f_{ij} & \text{o.w.} \end{cases} \tag{4.5}$$

Here, σ is 600 seconds and l denotes the number of unique alarms in the training set $\mathbb{T}$. According to Eq. (4.5), c_i and r_i denote the the column and row indexes, respectively. In C_M, f_{ij} indicates the co-occurrence frequency of c_i and r_j within 600 seconds in the training sequences, and Δt_{kij} indicates the time difference between the annunciations of c_i and r_j in the training sequence k. The dimension of the co-occurrence matrix is $l \times l$ and all the elements are initialized as 0's. The co-occurrence matrix is computed offline during the construction of the CPT. In an ongoing alarm flood, the co-occurrence frequency of any two alarms can be known from the co-occurrence matrix, and coupled with the information from the Time Table, where different association rule parameters can be easily obtained.

In this work, the modified CPT has two new features, namely, the Time Table and co-occurrence matrix, to conduct most of the computation offline, making it different from that in [18] and [17]. In the construction of CPT, the Time Table stores the time of annunciations for each unique alarm in the training sequences, and the co-occurrence matrix gives the frequency of co-occurrences between all possible pairs of unique alarms with reference to the training sequences. Such information is directly exploited at different steps of the proposed method without the need to compute in real time during an ongoing alarm flood.

4.2.2 Eliminating Irrelevant Alarms

In an ongoing alarm flood, some alarms might be caused by disturbances or noises in the process and thus are irrelevant. An online sequence with such alarms may not represent the true alarm pattern of the underlying root cause and thus confuse the operator. In such scenarios, those less relevant alarms can be identified by incorporating the co-occurrence frequency and the confidence between two alarms.

Table 4.2: An online alarm flood sequence at the triggering time instant.

Alarm Tag	Time Stamp
Tag_A	06:23:01 AM
Tag_B	06:23:44 AM
Tag_C	06:24:12 AM
Tag_D	06:25:37 AM
Tag_E	06:26:12 AM
Tag_F	06:27:25 AM
Tag_G	06:27:51 AM
Tag_H	06:28:47 AM
Tag_I	06:29:32 AM
Tag_J	06:30:44 AM

Table 4.2 shows an online alarm flood sequence at the triggering time instant, i.e., when $\zeta \geq \zeta_1$. The set of alarms in this online alarm flood sequence is given by

$$\mathbb{B} = \{b_i |\ e_i^b \in \mathbb{A}, \exists\ x_i^b(l) = 1 \text{ s.t. } l \in [t-\beta+1, t]\}_{i=1}^{|\mathbb{B}|} \tag{4.6}$$

where t indicates the triggering time instant. Each alarm $b_i \in \mathbb{B}$ is paired with each remaining alarm in $\mathbb{B}$ and the set of pairs with respect to $b_i \in \mathbb{B}$ is represented as $\omega_{b_i} = \{(b_i, b_j)|b_i \neq b_j, \{b_i, b_j\} \subseteq \mathbb{B}, j = 1, 2, \cdots, |\mathbb{B}|\}$. Table 4.3 shows that Tag_F is paired with each remaining alarm from $\mathbb{B}$, represented as co-occurring alarms. Analogously, for each alarm $b_i \in \mathbb{B}$, ω_{b_i} is formulated with $(|\mathbb{B}| - 1)$ pairs at the triggering instant.

Table 4.3: Co-occurrences frequency and confidence.

Alarm Tag	Co-occurring Alarm	Frequency of Co-occurrence, $\mathbb{F}_i$	Confidence, $\mathbb{C}_i$
Tag_F	Tag_A	8	0.36
	Tag_B	4	0.18
	Tag_C	7	0.32
	Tag_D	**9**	**0.41**
	Tag_E	2	0.11
	Tag_G	7	0.32
	Tag_H	8	0.36
	Tag_I	6	0.27
	Tag_J	6	0.27

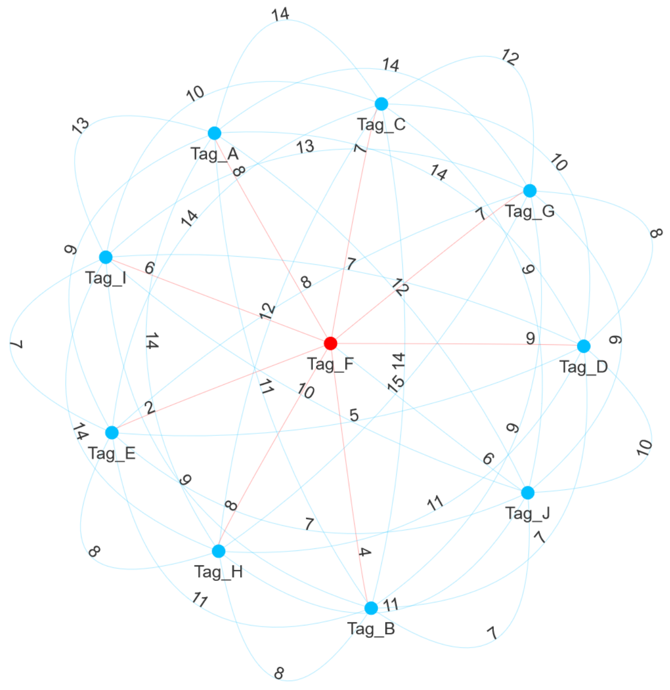


Figure 4.2: Co-occurrence network at the triggering time instant of the online alarm flood sequence.

For each pair in ω_{b_i}, the co-occurrence frequency is computed in the co-occurrence matrix C_M and given by

$$\mathbb{F}_i = \{f_{ij} | i \neq j, c_i = b_i(b_j) \ \& \ r_j = b_j(b_i)\}_{j=1}^{|\mathbb{B}|} \tag{4.7}$$

The confidence is also calculated for each alarm pair in ω_{b_i} based on the co-occurrence matrix C_M and the Time Table. In Table 4.3, $\mathbb{C}_i$ denotes the set of confidence between the elements of each pair in ω_{b_i}. In the Time Table, the number of time instants implies the total number of occurrences for an alarm in the training set. For instance, Table 4.1 shows that alarm Tag_A occurs 3 times in S_1, S_2, and S_3 as it includes only 3 time stamps. The set of time instants of alarm Tag_A is considered as $t_A = \{(02:21:31), (02:37:11), (04:00:37)\}$. Then, the confidence $\mathbb{C}_{ij}$ for the alarm pair (b_i, b_j) is

$$\mathbb{C}_{ij} = \text{Supp}(b_i, b_j)/\text{Supp}(b_i) = \frac{f_{ij}}{|t_i|}. \tag{4.8}$$

where Supp indicates the support of an item, i.e., the probability of an item occurring in the training sequences; f_{ij} indicates the co-occurrence frequency of alarms b_i and b_j; $|t_i|$ denotes the frequency that alarm b_i occurs in the training sequences. The purpose is to identify the pairs (b_i, b_j) from ω_{b_i} with either $f_{ij} = \max(\mathbb{F}_i)$ or $\mathbb{C}_{ij} = \max(\mathbb{C}_i)$ for $i, j \in [1, |\mathbb{B}|]$.

Table 4.3 shows $\mathbb{F}_i$ and $\mathbb{C}_i$, namely, the co-occurrence frequency and the confidence for each pair of alarms formulated with alarm Tag_F from $\mathbb{B}$, respectively. The alarm pair (Tag_F, Tag_D) shows the maximum co-occurrence frequency and confidence among all the pairs. Analogously, alarm pairs corresponding to each alarm in $\mathbb{B}$ with the maximum co-occurrence frequency and confidence are identified and shown in Table 4.4. Further, Fig. 4.2 displays a co-occurrence network formulated with the existing alarms in $\mathbb{B}$ at the triggering instant. Each node represents an alarm in $\mathbb{B}$; each edge connecting each pair of alarms in ω_{b_i}, $i = 1, 2, \cdots, |\mathbb{B}|$, represents the co-occurrence frequency. In Fig. 4.2, the relationship between alarm Tag_F and all the co-occurring alarms is highlighted in red in Table 4.3.

The user-defined thresholds on the frequency of co-occurrence and the confidence, denoted by γ_f and γ_p, are set to 10 and 0.5, respectively. Such values

Table 4.4: Co-occurrence frequency and confidence at the triggering instant.

Target Alarm	Co-occurring Alarm	Frequency of Co-occurrence	Confidence
Tag_ A	Tag_ H	14	0.32
Tag_ B	Tag_ C	14	0.58
Tag_ C	Tag_ A	14	0.48
Tag_ D	Tag_ A	14	0.35
Tag_ E	Tag_ C	14	0.47
Tag_F	**Tag_D**	**9**	**0.41**
Tag_G	Tag_H	15	0.83
Tag_H	Tag_G	15	0.68
Tag_I	Tag_H	14	0.93
Tag_J	Tag_A	12	0.55

are determined by observing the trend of the irrelevant alarms in the training sequences. Additionally, the performance of the proposed method is evaluated in different scenarios by assigning various combinations of values to both γ_f and γ_p within specific ranges. The optimal values for γ_f and γ_p can be chosen based on the accuracy in applications. An investigation of the sensitivity of both user-defined thresholds on the performance of the proposed method is provided in Section 4.1. If any of the alarm pairs in Table 4.4 fails to meet the threshold condition, the corresponding target alarm is identified as irrelevant. For instance, in Table 4.4, Tag_F fails to meet either of the thresholds and is eliminated from subsequent analysis. Following the triggering instant, the co-occurrence frequency and confidence are computed between the upcoming alarm and the existing relevant alarms in the online sequence that appear within the time interval $[t - \beta + 1, t]$, where t is the annunciation time of the upcoming alarm. Fig. 4.3 shows the co-occurrence frequency between the upcoming alarm Tag_K and the existing relevant alarms from the online sequence annunciated within 600 seconds. It can be observed that the upcoming alarm Tag_K has the highest co-occurrence frequency (greater than γ_f) with the existing alarm Tag_A and thus is included in the online sequence. To better illustrate the idea, a sliding window of $\beta = 600$ s is used in Table 4.5, where the existing alarms Tag_A and Tag_B appear beyond the time window β since the annunciation of an upcoming alarm Tag_L; thus, they are not considered and the remaining alarms in the online sequence will be used to check the relevancy of the upcoming alarm Tag_L.

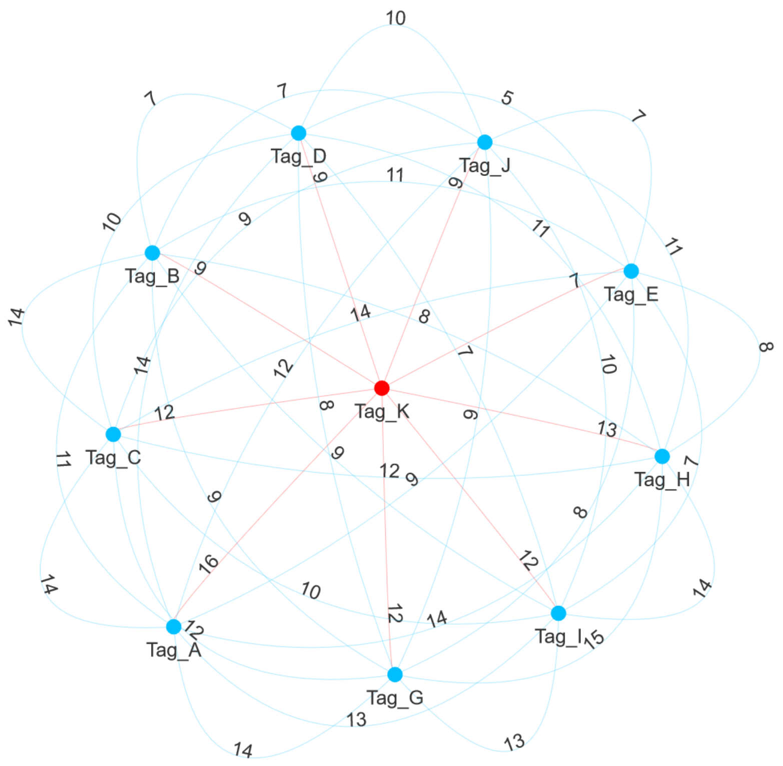


Figure 4.3: Co-occurrence network with an upcoming alarm following the triggering instant.

Table 4.5: Online alarm flood sequence at the triggering instant.

Alarm Tag	Time Stamp
Tag_A	06:23:01 AM
Tag_B	06:23:44 AM
Tag_C	06:24:12 AM
Tag_D	06:25:37 AM
Tag_E	06:26:12 AM
Tag_F	06:27:25 AM
Tag_G	06:27:51 AM
Tag_H	06:28:47 AM
Tag_I	06:29:32 AM
Tag_J	06:30:44 AM
Tag_K	06:31:23 AM
↑	↑
Tag_L	**06:33:52 AM**

4.2.3 Identifying Similar Sequences

In an ongoing alarm flood, alarms appear one by one and are added to the online sequence. The upcoming alarms are predicted by analyzing the existing alarm set in the online sequence B and the closely similar training sequences from $\mathbb{T}$. To identify similar training sequences, the algorithm executes the following two steps:

1) It sets the prefix of the online sequence as $X = \{b_i^x\}_{i=|B|-\aleph}^{|B|} = \{b_{|B|-\aleph}^x, b_{|B|-\aleph+1}^x, \cdots, b_{|B|}^x\}$, where $\aleph$ is the prefix length. Using the Inverted Index, a set containing the IDs of the similar training sequences $\mathbb{T}_x$ from CPT is identified based on the presence of at least one element from the prefix X. The choice of prefix length $\aleph$ is important for the desired performance of the proposed method. Clearly, if $\aleph$ is set too high, the method will identify too many training sequences from CPT, which will increase the computation and response time. Conversely, if $\aleph$ is set too short, the method may miss truly identical similar training sequences, leading to reduced accuracy. An ideal value of $\aleph$ achieves optimal accuracy for the proposed method while maintaining a satisfactory response time.

2) The size of $\mathbb{T}_x$ depends on the criteria of the similarity analysis. A modified set-based similarity mechanism is implemented to further reduce the number of sequences and identify the sequences most similar to B.

The Inverted Index includes a set of keys, namely, unique alarm tags, which are denoted as $\mathbb{I} = \{e_i | e_i \in \mathbb{A}, i = 1, 2, \cdots, |\mathbb{A}|\}$. In the Inverted Index, each unique alarm tag contains a bitset of length $|\mathbb{T}|$, namely, the total number of sequences in the training set $\mathbb{T}$ [18]. The indexes corresponding to each element in the bitset of a unique alarm tag indicate the IDs of the sequences in $\mathbb{T}$ and are denoted as $\mathbb{M} = \{\psi | \psi \in \mathbb{N}^+, \psi = 1, 2, \cdots |\mathbb{T}|\}$. For an alarm of the prefix X, the indexes of the bits, namely, the IDs of the sequences containing that alarm, are included in $\mathbb{M}_i^x = \{k | k \in \mathbb{N}^+, k \in \mathbb{M}, b_i^x \in A_k\}, i =$

$1, 2, \cdots, |X|$. For instance, the bitset of alarm Tag_A in the Inverted Index is 01011, indicating that the total number of training sequences is 5; the 2nd, 4th, and 5th sequences include alarm Tag_A, which also belongs to the prefix X of the online sequence B. Thus, the set containing the IDs of the training sequences that include any one of the alarms from the prefix X, is denoted as $\mathbb{T}_x = \{\mathbb{M}_i^x : i = 1, 2, \cdots, |X|\}$. The training sequences corresponding to the IDs in T_x can be identified using the Lookup Table of the CPT and denoted as $\mathbb{S}_x = \{A_q | q \in \mathbb{N}^+, q \in \mathbb{T}_x, \exists i \in [1, |X|] \ s.t. \ b_i^x \in A_q, q = 1, 2, \cdots, |\mathbb{S}_x|\}$. To implement the modified set-based similarity mechanism, two vectors of binary indexes are formulated; each element is determined based on the presence of the adjacent element in A_m and B as

$$J_i^b = \begin{cases} 1 & \text{if } \exists j \in (1, 2, \cdots, |A_m|) \ s.t. \ e_i^b = e_j^m, \ \& \ e_{i+1}^b = e_{j+1}^m, \\ 0 & \text{o.w.,} \end{cases} \tag{4.9}$$

$$J_j^m = \begin{cases} 1 & \text{if } \exists i \in (1, 2, \cdots, |B|) \ s.t. \ e_j^m = e_i^b, \ \& \ e_{j+1}^m = e_{i+1}^b, \\ 0 & \text{o.w.,} \end{cases} \tag{4.10}$$

where e_i^b (e_j^m) is the alarm tag of the ith (jth) alarm event in B (A_m); $i = 1, 2, \cdots, |B|$ and $j = 1, 2, \cdots, |A_m|$. Two criteria should be considered while formulating J_i^b and J_j^m:

1) If multiple alarms are present in both sequences and annunciated simultaneously, they can be considered to appear in a similar order in both sequences.

2) Due to the detection delay or disturbances, the orders of closely annunciated alarms or adjacent alarms may not be identical in different sequences. Accordingly, a time interval τ_d is used here, such that if multiple alarms exist in both sequences, they can be considered adjacent to each other and appear in identical order. If τ_d is set too high, it may lead to false identical sequences from the training set, whereas a too small value may lead to many similar training sequences being neglected. A reasonable choice can be of any value that can nullify the effect of detection delay or disturbances in the training sequences.

Then, the set-based similarity score S_{set} between B and $A_m \in \mathbb{T}$ is determined by

$$S_{set}(A_m, B) = 0, \text{ if } U_B \cap U_m = \emptyset, \tag{4.11}$$

else if $U_B \cap U_m \neq \emptyset$,

$$S_{set}(A_m, B) = \sqrt{\frac{\sum_{i=1}^{|B|} J_i^b \cdot \sum_{j=1}^{|A_m|} J_j^m}{(|B|-1)\cdot(|A_m|-1)}}, \tag{4.12}$$

where U_B and U_m denote the alarm sets in B and A_m, respectively. A threshold μ is introduced on S_{set} to identify most similar sequences from $\mathbb{S}_x$. These similar sequences form a new set $\mathbb{S}$. To illustrate the method, an example is presented below.

Example 1. There are two time-stamped alarm sequences in Table 4.6. The binary indexing vectors J^x and J^y are given as

$$J^x = [0\ 1\ 1\ 1\ 1\ 0\ 0];\ J^y = [0\ 1\ 1\ 1\ 1\ 0]. \tag{4.13}$$

The set-based similarity is calculated to be $S_{set}(X, Y) = \sqrt{16/30} = 0.73$. When the algorithm attempts to recalculate, the prefix X changes with new upcoming alarms from B, which will lead to the change in $\mathbb{T}_x$, and then $\mathbb{S}$ is updated accordingly based on $S_{set}(A_n, B) \geq \mu$.

Table 4.6: Time stamped alarm sequences X and Y.

Sequence X		**Sequence Y**	
Alarm Tag	Time Stamp	Alarm Tag	Time Stamp
Tag_A	00:00:01 AM	Tag_K	10:30:00 AM
Tag_B	00:01:18 AM	Tag_C	10:31:21 AM
Tag_C	00:01:18 AM	Tag_B	10:31:21 AM
Tag_D	00:01:18 AM	Tag_D	10:31:21 AM
Tag_E	00:02:36 AM	Tag_E	10:32:36 AM
Tag_F	00:02:38 AM	Tag_G	10:33:12 AM
Tag_G	00:02:40 AM		

4.2.4 Identifying Pattern Index

At any instant of an ongoing alarm flood, the objective is to identify the candidate alarms for the prediction of subsequent upcoming alarms. After identifying the most similar sequences A_n from $\mathbb{S}_x$, the consequent A_y^n from each similar sequence is identified, where $n = 1, 2, \cdots, |\mathbb{S}|$. The consequent is defined as the subsequence following a specific index, which is denoted as a pattern index $\Psi_n \in \mathbb{N}^+$. Given a similar sequence $A_n =< a_1^n, a_2^n, \cdots, a_{|A_n|}^n >\in \mathbb{S}$, the consequent A_y^n is identified from it. A binary vector A_Ψ^n is formulated from A_n with each element $j_i^n \in A_\Psi^n$ given by

$$j_i^n = \begin{cases} 1 & \text{if } e_i^n \in \{a_k^n\}_{k=1}^{|A_n|} \cap X \ \text{ and } \ \nexists e_i^n \in \{a_k^n\}_{k=1}^{i-1}, \\ 0 & \text{o.w.}, \end{cases} \tag{4.14}$$

where $e_i^n \in a_i^n$ is the ith alarm in A_n; X is the prefix, namely, $X = \{b_i^x\}_{i=|B|-\aleph}^{|B|}$, $\aleph = 1, 2, \cdots, |X|$; $n = 1, 2, \cdots, |\mathbb{S}|$. In A_Ψ^n, the pattern index Ψ_n indicates the position of an element $J_{\Psi_n}^n \in A_\Psi^n$ such that

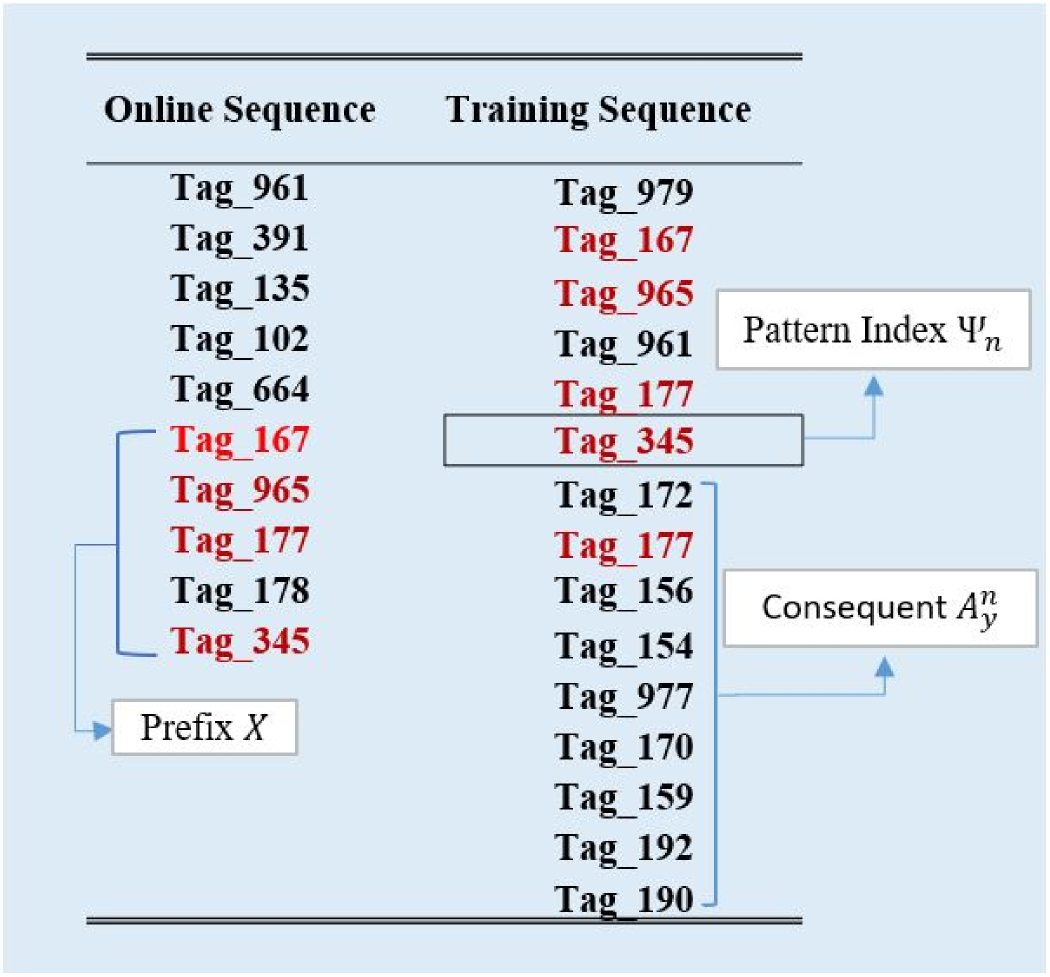
$$j_{\Psi_n}^n = 1 \ \text{ and } \ \forall i > \Psi_n, \ j_i^n = 0. \tag{4.15}$$

Then, the consequent from each similar sequence is extracted if $|A_y^n| > 0$, and is given by

$$A_y^n =< a_{\Psi_n+1}^n, a_{\Psi_n+2}^n, \cdots, a_{|A_n|}^n > . \tag{4.16}$$

The set of pattern indexes for each similar sequence $A_n \in \mathbb{S}$ is denoted as $\Psi = \{\Psi_n | \Psi_n \in \mathbb{N}^+, n = 1, 2, \cdots, |\mathbb{S}|\}$. Following the triggering instant, whenever the algorithm attempts to recompute, the prefix X changes as new alarms are added to B one by one. With a change in prefix X, the algorithm recalculates the new pattern index $\tilde{\Psi}_n$; if $\tilde{\Psi}_n > \Psi_n$, the consequent A_y^n in the similar sequence $A_n \in \mathbb{S}, n = 1, 2, \cdots, |\mathbb{S}|$ changes accordingly. Thus, the pattern index Ψ_n and the consequent A_y^n may change based on the new alarms in the prefix X of the online sequence B. To illustrate the method, an example is presented below.

Example 2. Fig. 4.4 illustrates the procedures of identifying the pattern indexes and consequents from the training sequences. The prefix length $\aleph$ is set to 5, i.e., the prefix X is formulated with the last 5 alarms from the online sequence B. The indexes of each common alarm from the prefix X are identified in the training sequence. Then, the index of alarm Tag_345 is identified as the pattern index Ψ_n. Following the pattern index, the subsequence consisting of the remaining alarms is represented as the consequent of the training sequence. Following this procedure, the pattern indexes and consequents are identified from each similar sequence at any instant of the online alarm flood.



Online Sequence	Training Sequence
Tag_961	Tag_979
Tag_391	Tag_167
Tag_135	Tag_965
Tag_102	Tag_961
Tag_664	Tag_177
Tag_167	Tag_345
Tag_965	Tag_172
Tag_177	Tag_177
Tag_178	Tag_156
Tag_345	Tag_154
	Tag_977
	Tag_170
	Tag_159
	Tag_192
	Tag_190

Figure 4.4: Identification of pattern indexes and consequents from a training sequence.

4.2.5 Evaluation Criteria and Strategy

At the triggering time instant of the online alarm flood B, irrelevant alarms are removed from B, and then the prefix X is identified from B. The pattern index $\Psi_n, n = 1, 2, \cdots, |\mathbb{S}|$, is determined based on the existing alarms in the prefix X. The consequents from each similar sequence A_n are identified.

In each consequent A_y^n, the alarms that appear following the pattern index Ψ_n, are the candidate upcoming alarms and will be evaluated. Following the triggering instant, the proposed algorithm continues to update its prediction at different instants of the ongoing alarm flood. To evaluate each candidate alarm, a varying alarm set B_λ is formulated from B. At the time instant t, the varying alarm set B_λ is denoted as

$$B_\lambda = \{b_k^\lambda | b_k^\lambda \in B, \ \& \ \exists \ x_k^\lambda(l) = 1, \quad \text{s.t.} \quad l \in [t - \beta + 1, t]\}_{k=1}^{|B_\lambda|}. \tag{4.17}$$

The criteria to obtain the varying alarm set B_λ is $|B_\lambda| \ \leq 10$, which indicates that B_λ can contain no more than 10 alarms at the instant when the computation begins provided that the alarms are annunciated within $[t - \beta + 1]$. Each existing alarm from B_λ is selected as a target alarm, and the relationships between each target alarm and the candidate alarms from the consequents $A_y^n, n = 1, 2, \cdots, |\mathbb{S}|$, are analyzed using the following measures to identify the potential upcoming alarms:

1. **Confidence** ($\mathbb{C}$): Confidence [66] measures how frequent the candidate alarm from the consequents occurs given that the target alarm from B_λ is present in the sequence where both alarms appear within the time interval $\beta = 600$ s. The confidence of an association rule $(A \rightarrow B)$ is calculated as

$$\mathbb{C}(A \rightarrow B) = \frac{\text{supp}(A \cup B)}{\text{supp}(A)}, \tag{4.18}$$

   where $\text{supp}(A \cup B)$ and $\text{supp}(A)$ indicate the probabilities of cooccurrence of two alarms and the occurrence of alarm A in the training set, respectively.

2. **Interest** ($\mathbb{I}$): Interest [24] identifies those infrequent association rules that show high confidences due to the correlation between $\text{supp}(A \cap B)$ and $\text{supp}(A)$. The interest of an association rule $(A \rightarrow B)$ is computed as

$$\mathbb{I}(A \rightarrow B) = \frac{\text{supp}(A \cup B)}{\text{supp}(A) \times \text{supp}(B)}. \tag{4.19}$$

An interest ratio close to 1 implies that the occurrences of alarms A and B are independent and there exists no true association. If it is greater than 1, the association is true, and the occurrences of alarms A and B are dependent.

3. **Mean occurrence delay (τ):** Mean occurrence delay is an important criterion to indicate the strength of the relationship. It indicates, on average, how closely the target alarm from B_λ and the candidate alarm from the consequent occur together in the training sequences.

If the confidence and interest are higher than 1, and the mean occurrence delay is low, it implies that the candidate alarm is a true potential upcoming alarm. Given the pattern index Ψ_k of the kth sequence in $\mathbb{S}$, the modification of B_λ with 10 alarms after confirming a candidate alarm at index $\Psi_k + 1$ of the kth sequence is represented as

$$\tilde{B}_\lambda = \{b_q^\lambda\}_{q=2}^{|B_\lambda|} \cup a_{\Psi_k+1}^k. \tag{4.20}$$

An example is shown in Fig. 4.5. Upon confirming Tag_159 as a potential upcoming alarm based on the criteria mentioned above, B_λ includes Tag_159 and eliminates Tag_961. An empty set $B_t = \emptyset$ is defined at the triggering instant of B that only includes the potential upcoming alarms nominated by B_λ. Thus, analogous to Eq. (4.20), the modification of B_t after nominating the candidate alarm $a_{\Psi_k+1}^k$ is represented as

$$\tilde{B}_t = B_t \cup a_{\Psi_k+1}^k. \tag{4.21}$$

The objective of this strategic formulation of B_λ is attributed to the causal relationship between the process variables. During an alarm flood, the annunciation of an alarm would trigger the annunciation of other alarms, and they co-exist in the training sequences with a low occurrence delay. Due to the causal relationship, such alarm variables would show strong associations in terms of the criteria above. Thus, it should pick up those alarms from the consequents that have strong associations with any alarms in B_λ.

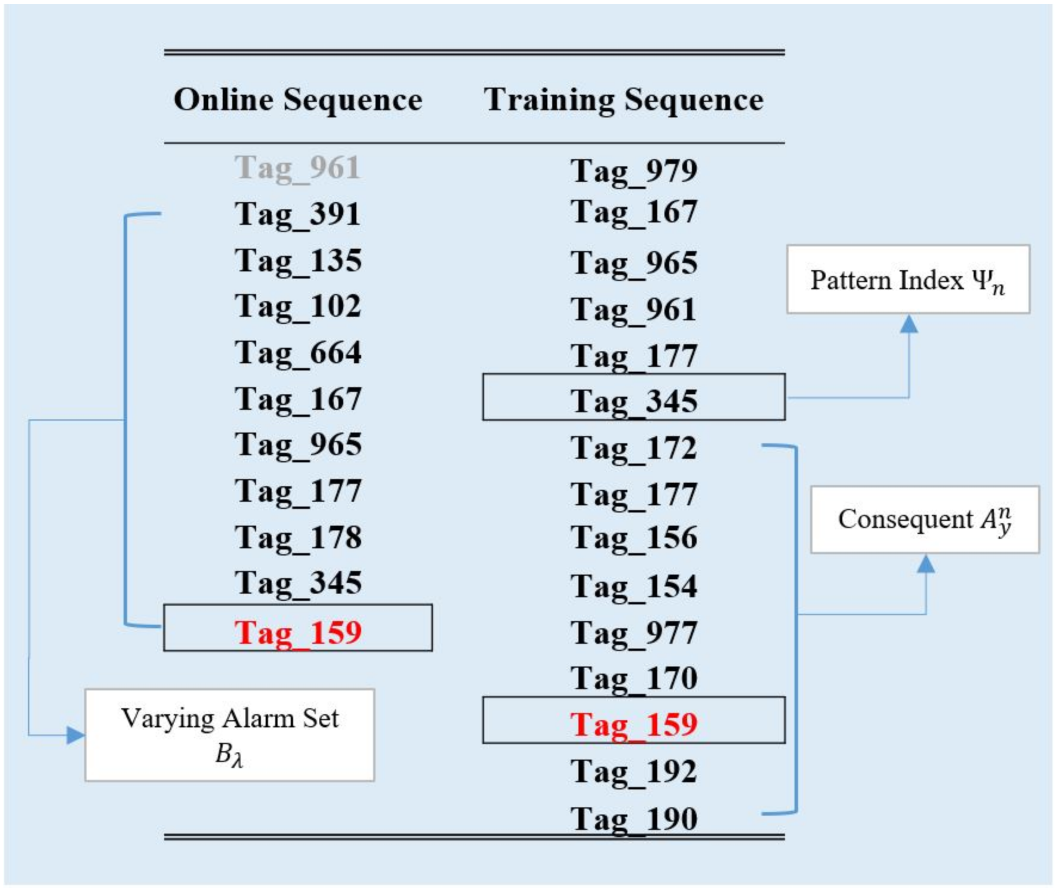


Figure 4.5: Varying alarm set.

However, the alarms in B_λ will continuously be replaced by the candidate alarms upon confirming as potential upcoming alarms. In such cases, if the alarms from a consequent are subsequently evaluated and added to B_λ, the alarm set B_λ may be completely changed before evaluating the next consequent. The alarms in the next consequent may not hold a strong association with the new alarm set in B_λ, and as a result, some potential upcoming alarms could be missed. Thus, it should evaluate and confirm as many candidate alarms as possible from the consequents with a specific set of alarms in B_λ to avoid missing any potential alarms. A strategy for maintaining the orders for evaluating the candidate alarms from each consequent needs to be implemented. The algorithm evaluates a batch of candidate alarms formed as

$$\Phi_w = \{a^k_{\Psi_k+w} | \exists A_n \in \mathbb{S},\ s.t.\ a^k_{\Psi_k+w} \in A^n_y, |A^n_y| \geq w\}_{k=1}^{|\Phi_w|}, \tag{4.22}$$

where $w \in \mathbb{N}^+$ and $n = \{1, 2, \cdots, |\mathbb{S}|\}\}$. After identifying the pattern indexes and consequents from each similar sequence in $\mathbb{S}$, the first batch of candidate alarms is

$$\Phi_1 = \{a^1_{\Psi_1+1}, a^2_{\Psi_2+1}, \cdots, a^{|\mathbb{S}|}_{\Psi_{|\mathbb{S}|}+1}\}, \text{ if } \forall |A^i_y| > 0, \tag{4.23}$$

where $i = 1, 2, \cdots, |\mathbb{S}|$; $\Psi_1, \Psi_2, \cdots, \Psi_{|\Phi_1|}$ are the pattern indexes; A^i_y represents the consequents of the corresponding ith sequence in $\mathbb{S}$. At any instant of B, a set of batches consisting of the candidate alarms selected iteratively from each consequent is

$$\Phi = \bigcup_{w=1}^{|\Phi|} \{\Phi_w\}, \tag{4.24}$$

where $|\Phi|$ indicates the total number of batches. Each batch may not have an equal number of candidate alarms due to the unequal lengths of the consequents. The criteria for formulating Φ_w is denoted as $\exists \Phi_w$ if $\exists i \in 1, 2, \cdots, |\mathbb{S}|$ s.t. $|A^i_y| \geq w$, where w and $|A^i_y|$ indicate the batch number and the total number of alarms in the consequent A^i_y, respectively. The algorithm will formulate a sequence of $|\Phi|$ batches and evaluate according to $< \Phi_1, \Phi_2, \cdots, \Phi_{|\Phi|} >$. The batch-wise evaluation of candidate alarms from the consequents is effective in identifying the true potential upcoming alarms.

4.2.6 Prediction of Annunciation Time

When an upcoming alarm is predicted, it is more interesting to know the potential time when it may occur. Predicting the time of annunciations of the upcoming alarms can provide better real-time assistance. Here, the confidence interval (CI) around the mean of time differences between the annunciation of two subsequent predicted alarms is determined. At any instant t, the prediction of upcoming alarms in B is represented as $\tilde{B}_t = \{\tilde{b}_{|B|+1}, \tilde{b}_{|B|+2}, \cdots, \tilde{b}_{|\tilde{B}_t|}\}$. The time differences between the annunciations of each pair of alarms $\tilde{b}_{|B|+i}$ and $\tilde{b}_{|B|+i+1}$ from $\tilde{B}_t$ are analyzed to obtain the confidence interval, which is computed as follows:

Given two alarms A and B , their time stamps in the Time Table are denoted as $t_A = \{t^A_i | i = 1, 2, \cdots, |t_A|\}$ and $t_B = \{t^B_j | j = 1, 2, \cdots, |t_B|\}$.

Then, a set of tuples is obtained and represented as $s_d = \{s_i^d = (t_k^A, t_l^B), i = 1, 2, \cdots |s_d|\}$, where each tuple in s_d holds the following properties:

1. For $s_i^d = (t_k^A, t_l^B)$, $|t_k^A - t_l^B| \leq \beta$, where $k \in [1, 2, \cdots, |t_A|]$ and $l \in [1, 2, \cdots, |t_B|]$.
2. For $s_i^d = (t_k^A, t_l^B)$, $\forall j \neq i$, $t_k^A \notin s_j^d$ and $t_l^B \notin s_j^d$, i.e, each element only exists in one tuple.

Each tuple in s_d represents one instance of co-occurrence for alarms A and B in $\mathbb{T}$, where the time difference is within 600 seconds. From the sample data s_d, the set of time differences is $t_d = \{t_i^d = |t_k^A - t_l^B|, i = 1, 2, \cdots, |t_d| : \exists k \in [1, 2, \cdots, |t_A|] \ s.t. \ t_k^A \in s_i^d, \exists l \in [1, 2, \cdots, |t_B|] \ s.t. \ t_l^B \in s_i^d\}$. The mean of the time differences can be calculated as

$$\bar{t_d} = \frac{\sum_{i=1}^{|t_i^d|} t_i^d}{|t_d|}. \tag{4.25}$$

The 95% confidence interval estimates is determined for $\bar{t_d}$. For a larger sample size ($\geq$ 30), sample means are considered approximately normally distributed [47]. The confidence intervals can be computed as

$$\text{CI} = \bar{t_d} \pm z \frac{s}{\sqrt{|t_d|}}, \tag{4.26}$$

where $\bar{t_d}$, s and $|t_d|$ are the sample mean, sample standard deviation, and number of samples, respectively. For 95% confidence, the z value is 1.96. For a smaller sample size (< 30, the t distribution can be used [47]. The confidence interval is computed as

$$\text{CI} = \bar{t_d} \pm t \frac{s}{\sqrt{|t_d|}}. \tag{4.27}$$

Analogously, the confidence interval on the time gap between the subsequent predicted alarms is determined and denoted as

$$\text{CI}_{\tilde{B_t}} = \left\{ \text{CI}_{|B|+1}, \text{CI}_{|B|+2}, \cdots, \text{CI}_{|B|+|\tilde{B_t}|} \right\}, \tag{4.28}$$

where each element $\text{CI}_{|B|+i}, i = 1, 2, \cdots, |\tilde{B_t}|$ has lower and upper endpoints. In the context of time, it is reasonable to replace the negative endpoints with

0's. Then, each element in Eq. (4.28) is denoted as

$$\mathrm{CI}_{|B|+i} = \begin{cases} \left[\max\left\{0, \bar{t_d} - z\frac{s}{\sqrt{|t_d|}}\right\}, \left\{\bar{t_d} + z\frac{s}{\sqrt{|t_d|}}\right\}\right], & \text{if sample size} \geq 30, \\ \left[\max\left\{0, \bar{t_d} - t\frac{s}{\sqrt{|t_d|}}\right\}, \left\{\bar{t_d} + t\frac{s}{\sqrt{|t_d|}}\right\}\right], & \text{if sample size} < 30. \end{cases} \tag{4.29}$$

4.2.7 Online Prediction Algorithm

The online method to predict upcoming alarms during an ongoing alarm flood is summarized in Algorithm 5. The input arguments include the compact prediction tree (CPT) and co-occurrence matrix C_M; the output argument is the set of predicted alarm events $\tilde{B}_t$. At the triggering time instant, i.e., when $\zeta(t) \geq \zeta_1$, the set of co-occurrence frequency $\mathbb{F}_i$ and the set of confidences $\mathbb{C}_i$ are determined for each existing alarm $b_i \in B, i = 1, 2, \cdots, |B|$. Using γ_f and γ_p as thresholds of $\mathbb{F}_i$ and $\mathbb{C}_i$, respectively, irrelevant alarms are eliminated from B. Initially, the set of similar training sequences $\mathbb{S}_x$ is formulated from CPT using $\mathbb{T}_x$ that contains the IDs of the corresponding similar sequences. Each sequence in $\mathbb{S}_x$ contains at least one alarm from the prefix X. To further eliminate the irrelevant training sequences, a set of closely similar training sequences $\mathbb{S}$ is identified from $\mathbb{S}_x$ based on S_{set}. The pattern indexes Ψ_j and consequents A_y^j are identified from each similar sequence $A_j \in \mathbb{S}, j = 1, 2, \cdots, |\mathbb{S}|$. Then, the varying alarm set B_λ is formulated from B to evaluate the set of batches Φ, formed with the candidate alarms iteratively selected from each consequent $A_y^j, j = 1, 2, \cdots, |\mathbb{S}|$. The algorithm determines the confidence $\mathbb{C}$, interest $\mathbb{I}$, and mean occurrence delay τ between each alarm in B_λ and each candidate alarm; then, modify the existing alarm set in B_λ and $\tilde{B}_t$ if strong association is observed in terms of $\mathbb{C}$, $\mathbb{I}$, and τ. The set of co-occurrence frequency $\mathbb{F}_{|B|+1}$ and confidence $\mathbb{C}_{|B|+1}$ are determined between each upcoming alarm $\tilde{b}_{|B|+1}$ and the existing alarms from B within $[t - \beta + 1, t]$. Based on γ_f and γ_p, the upcoming alarm $\tilde{b}_{|B|+1}$ is eliminated or added to B.

The procedures to provide the confidence interval on the time gap between the subsequent annunciations of the predicted alarm events are summarized in Algorithm 6. The predicted alarm events in $\tilde{B}_t$ are analyzed in pairs to

Algorithm 3 Online prediction of upcoming alarms during an ongoing alarm flood.

```
1: Input Arguments: CPT, $C_M$
2: Output Argument: $\tilde{B}_t$
3: Define $\gamma_f, \gamma_p, \mu, \aleph$
4: Eliminate chattering alarms and calculate $\zeta(t)$
5: if $\zeta(t) \geq \zeta_1$ then
6:     Formulate $B$ and Obtain $\mathbb{B}$ by Eq. (4.6)
7:     for $i \in [1, |\mathbb{B}|]$ do
8:         Obtain $\mathbb{F}_i$ and $\mathbb{C}_i$ for $b_i \in \mathbb{B}$ by Eqs. (4.7) and (4.8)
9:         Eliminate $b_i$ if $max(\mathbb{F}_i) < \gamma_f$ and $max(\mathbb{P}_i) < \gamma_p$
10:    end for
11:    Identify prefix $X$ and obtain $\mathbb{T}_x$
12:    Formulate $\mathbb{S}_x$ and Obtain $\mathbb{S}$ based on $S_{set}$ and $\mu$
13:    for $j \in [1, |\mathbb{S}|]$ do
14:        Identify $\Psi_j$ and $A_y^j$ for $A_j \in \mathbb{S}$
15:    end for
16:    Obtain $\Phi$ by Eq. (4.24)
17:    Formulate $B_\lambda$ by Eqs. (4.17)
18:    for $k \in [1, |\Phi|]$ do
19:        for $l \in [1, |\Phi_k|]$ do
20:            Calculate $\mathbb{C}$, $\mathbb{I}$ and $\tau$ between $B_\lambda$ and $a^l_{\Psi_l+k} \in \Phi_k$
21:            Update $B_\lambda$ and $\tilde{B}_t$ by Eqs. (4.20) and (4.21)
22:        end for
23:    end for
24:    if $\zeta(t) > \zeta_2$ then
25:        Obtain $\mathbb{F}_{|B+1|}$ and $\mathbb{C}_{|B+1|}$ between $B$ in $[t-\beta+1, t]$ and $\tilde{b}_{|B|+1}$
26:        if $max(\mathbb{F}_{|B+1|}) < \gamma_f$ and $max(\mathbb{P}_{|B+1|}) < \gamma_p$ then
27:            Go to 24
28:        else
29:            $B \oplus \tilde{b}_{|B|+1}$
30:            Go to 11
31:        end if
32:    end if
33: end if
```

calculate the confidence interval. Initially, the time stamps of annunciations $t_{\tilde{b}_{|B|+i}}$ and $t_{\tilde{b}_{|B|+i+1}}$ for the predicted alarms $\tilde{b}_{|B|+i}$ and $\tilde{b}_{|B|+i+1}$ are obtained from the Time Table of CPT. Then, s_d is obtained from $t_{\tilde{b}_{|B|+i}}$ and $t_{\tilde{b}_{|B|+i+1}}$. Based on s_d, the time differences for each instance t_d and the mean of the time differences $\bar{t_d}$ between the predicted alarms are obtained. Then, depending on the number of samples, z or t distributions are used to compute the confidence intervals.

Algorithm 4 Confidence interval on the time gap between the subsequent annunciations of predicted alarms

```
1: Input Arguments: CPT, $\tilde{B}_t$
2: Output Argument: $CI_{\tilde{B}_t}$
3: Define $z$ for a 95% confidence level
4: for $i \in [1, |\tilde{B}_t| - 1]$ do
5:     Obtain $t_{\tilde{b}_{|B|+i}}$ and $t_{\tilde{b}_{|B|+i+1}}$ from Time Table
6:     Formulate $s_d$ and calculate $t_d$
7:     Calculate $\bar{t_d}$ by Eq. (4.25)
8:     if $|t_d| \geq 30$ then
9:         Obtain $CI_i$ by Eq. (4.26)
10:    else
11:        Obtain $CI_i$ by Eq. (4.27)
12:    end if
13: end for
```

It is noteworthy that the proposed method performs most of the computation offline and thus reduces the online computation. For instance, one of the criteria used to eliminate the irrelevant alarms in B is the co-occurrence frequency, which is determined offline during the construction of CPT and is available in the co-occurrence matrix C_M. In Eq. (4.8), the confidence is obtained using the co-occurrence frequency and the number of annunciations of the corresponding target alarm, both of which are also available in the co-occurrence matrix and the Time Table from the offline computation. Analogously, the evaluation criteria of each candidate alarm, namely, the confidence and interest, can be easily obtained using the information available in the co-occurrence matrix and the Time Table. Thus, much of the computation is delegated to offline analysis, reducing the online computational burden and the response time of the algorithm.

Remark 2: This proposed approach is different from the existing literature on real-time alarm flood analysis in the following aspects: (1) In this study, an association rule mining approach is proposed to predict upcoming alarms during industrial alarm floods, while the methods in [72], [71], [43], and [25] addressed alarm floods by pattern matching and identifying interesting abnormality patterns, which are mainly used offline; thus, the study objectives are different. (2) The methods in [71], [35], [27], and [10] are aimed at identifying similar alarm flood sequences to facilitate early prediction of alarm floods in real-time applications, while the proposed method targets at predicting incoming alarms directly during an alarm flood; the prediction objects are different. (3) This work proposes an online strategy for detecting and eliminating alarms in real time that is irrelevant to the true alarm pattern of an alarm flood sequence. Existing methods in [71], [34], [27], and [10] mainly addressed the removal of chattering alarms as preprocessing steps. However, alarm flood sequences can still be vulnerable to irrelevant alarms, which can conceal the true alarm pattern and lead to inaccurate results from similarity analysis. (4) The proposed approach introduces varying alarm set and batch-wise evaluation strategies to accurately identify potential upcoming alarms, while the method in citelai2017online primarily exploit real-time similarity analysis to identify similar alarm flood sequences. Additionally, the proposed method predicts the time of annunciation for each predicted alarm, while the method [34] ignore the time information.

4.3 Industrial Case Study

This section presents industrial case studies to demonstrate the effectiveness of the proposed method based on a real historical data set collected from a coking plant of an oil refinery. In this plant, the delayed coking process upgrades the heavy petroleum residues into lighter products and solid coke. The coking plant consists of a fractionator unit, controllers, a coking furnace, and the coke drums. More details of this industrial process can be found in [45]. In the historical data set, there were 2609 unique alarm tags from 9 different units over the time period from November 1, 2019, to April 30, 2020. Chattering alarms were removed using off-delay timers. In total, 103 alarm floods were extracted. The average length of historical alarm flood sequences was around 30. The maximum and minimum lengths were 609 and 10, respectively.

4.3.1 Overall Results

To evaluate the efficiency of the proposed method in various scenarios, a validation set consisting of 10 sequences was selected from the complete set of 103 sequences. Based on the "leave-one-out" strategy, the accuracy of the proposed method was evaluated for each of the sequences in the validation set. While the proposed method can eliminate irrelevant alarms in real time, the sequences in the validation set may have numerous irrelevant alarms. In this scenario, evaluating the accuracy based on the complete validation sequences, which are prone to irrelevant alarms, may not be ideal. Thus, two performance criteria were adopted. In the first criterion, the accuracy was calculated as

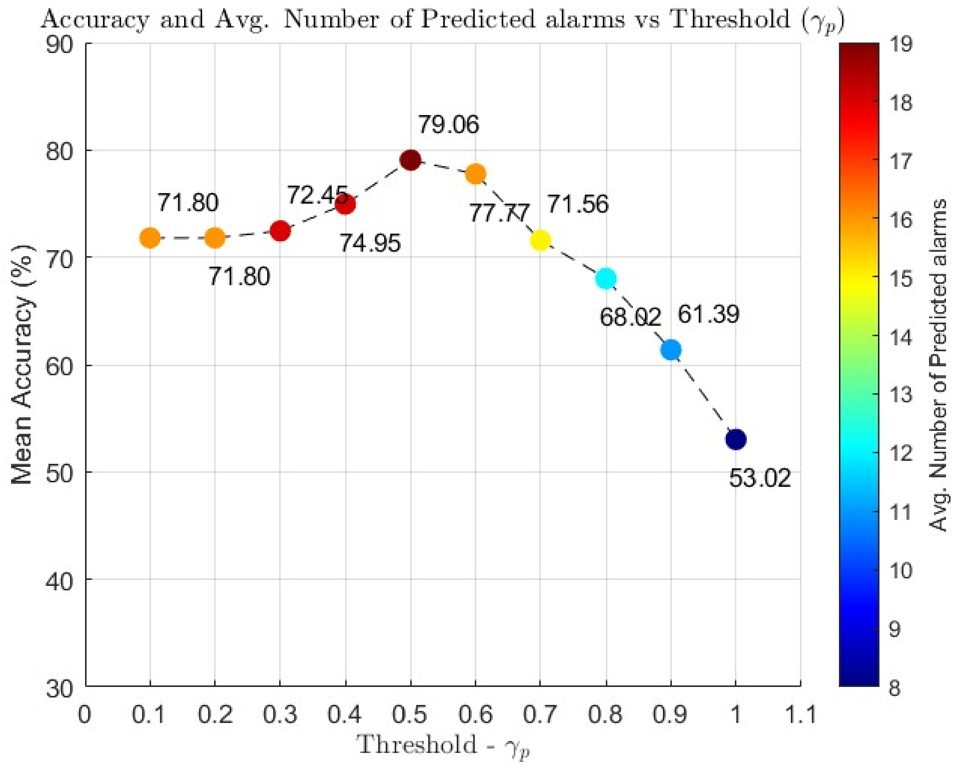


Figure 4.6: Effect of γ_p on the accuracy and the avgerage number of predicted alarms.

the percentage of alarms that existed in the corresponding validation sequence following the triggering instant and were matched with the predicted alarms, without excluding the irrelevant alarms. The second criterion only considered the relevant alarms of the validation sequence while determining the accuracy. In this scenario, the relevancy criteria were relaxed compared to Section 3.2. Specifically, in Section 3.2, an alarm was only considered relevant if it was associated with any of the alarms annunciated within the prior time interval β, based on the frequency of co-occurrence and the confidence, where $\beta = 600$ seconds. In contrast, for the validation set, an alarm was considered relevant if it was associated with any of the alarms within the validation sequence, irrespective of the time of annunciation.

Further, the proposed method used several user-defined thresholds that needed to be set to optimal values for the desired performance. Thus, the sensitivity of these user-defined thresholds to the performance of the proposed method was thoroughly analyzed. First, the user-defined thresholds γ_f and γ_p corresponding to the frequency of co-occurrence and the confidence were considered. The value of γ_f was kept constant at 10, while γ_p was gradually increased from 0.1 to 1 incrementally. The mean accuracy and the average number of predicted alarms of the validation set at each increment were observed and shown in Fig. 4.6. The results indicated that as γ_p increased, both the mean accuracy and the average number of predicted alarms gradually increased until γ_p surpassed 0.5. Beyond $\gamma_p = 0.5$, the mean accuracy begun to decrease, suggesting that setting γ_p to a high value may exclude some truly relevant alarms. Furthermore, a small increase in the average number of predicted alarms led to an improvement in mean accuracy from 71.80% to 79.06%, highlighting the effectiveness of the proposed method in recommending a limited number of potentially incoming alarms that match the existing alarms in validation sequences. Additionally, the average number of predicted alarms remained at a manageable level, providing industrial operators with the advantage of planning corrective actions without being overwhelmed.

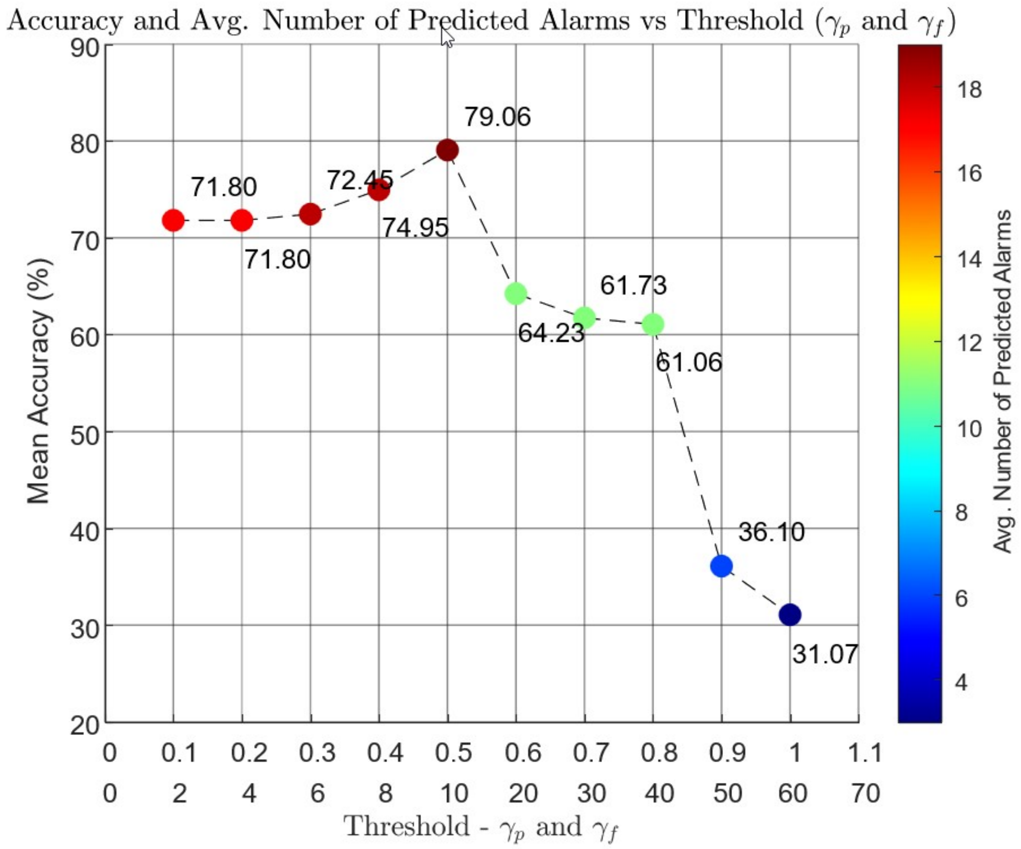


Figure 4.7: Effect of γ_p and γ_f on the accuracy and the avgerage number of predicted alarms.

However, to explore the possibility of achieving better performance for the proposed method, an additional investigation was conducted by simultaneously increasing both γ_f and γ_p incrementally. This approach aimed to assess the impact of different combinations of γ_f and γ_p values on the overall performance of the method. As shown in Fig. 4.7, the mean accuracy decreased significantly at each step of the increment beyond $\gamma_p > 0.5$, compared to the scenario presented in Fig. 4.6. Simultaneously, the average number of predicted alarms increased initially but started to decrease beyond $\gamma_p > 0.5$ and $\gamma_f > 10$. Thus, considering the cases discussed above, setting γ_f and γ_p to 10 and 0.5, respectively, is appropriate for the desired performance of the proposed method.

Additionally, the user-defined threshold τ_d was considered to eliminate

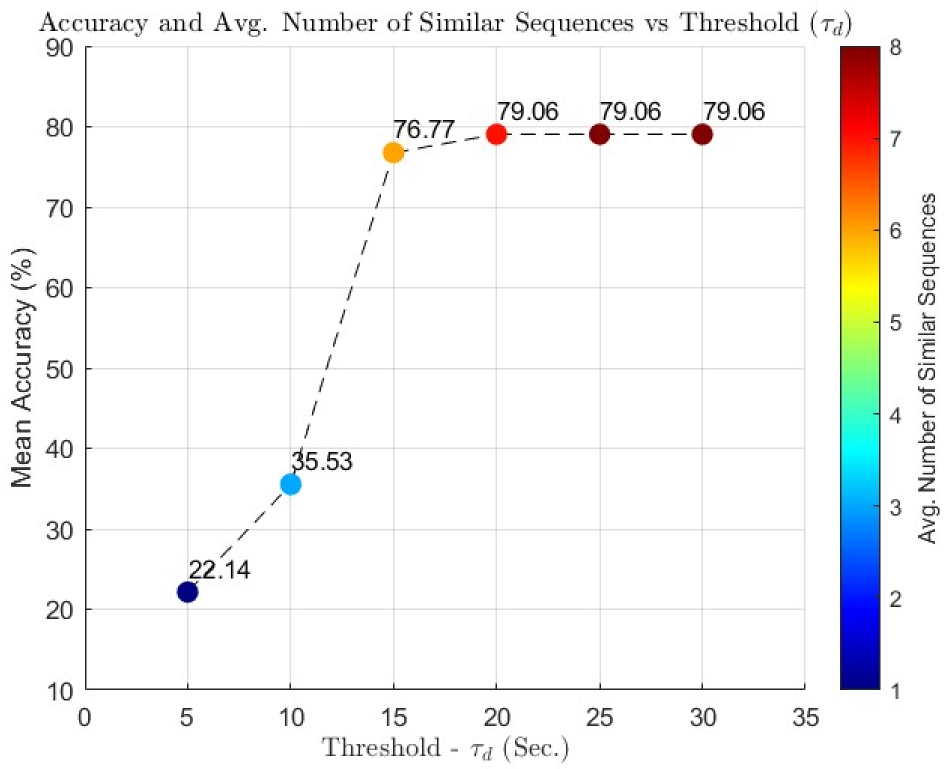


Figure 4.8: Effect of τ_d on the accuracy and the avgerage number of predicted alarms.

the effect of disturbances and detection delays when determining set-based similarity scores, S_{set}. The threshold τ_d was incrementally adjusted, ranging from 5s to 30s. The effects on both the mean accuracy and the average number of identified similar sequences with $\mu = 0.15$ were closely examined at each increment. The corresponding results are shown in Fig. 4.8. It can be observed that as τ_d increases, the mean accuracy gradually improves until reaching 79.06% at $\tau_d = 20$s. The mean accuracy remains constant at 79.06% even when τ_d exceeds 20s. However, the average number of similar sequences continues to increase as τ_d exceeds $20s$, which may increase the computation time for the proposed method. Thus, $\tau_d = 20$s is a reasonable choice.

Furthermore, the set-based similarity scores S_{set} are subjected to a user-defined threshold μ to eliminate irrelevant sequences. To analyze the effect of μ on the accuracy, the value of μ was increased from 0.05 to 0.3; the mean

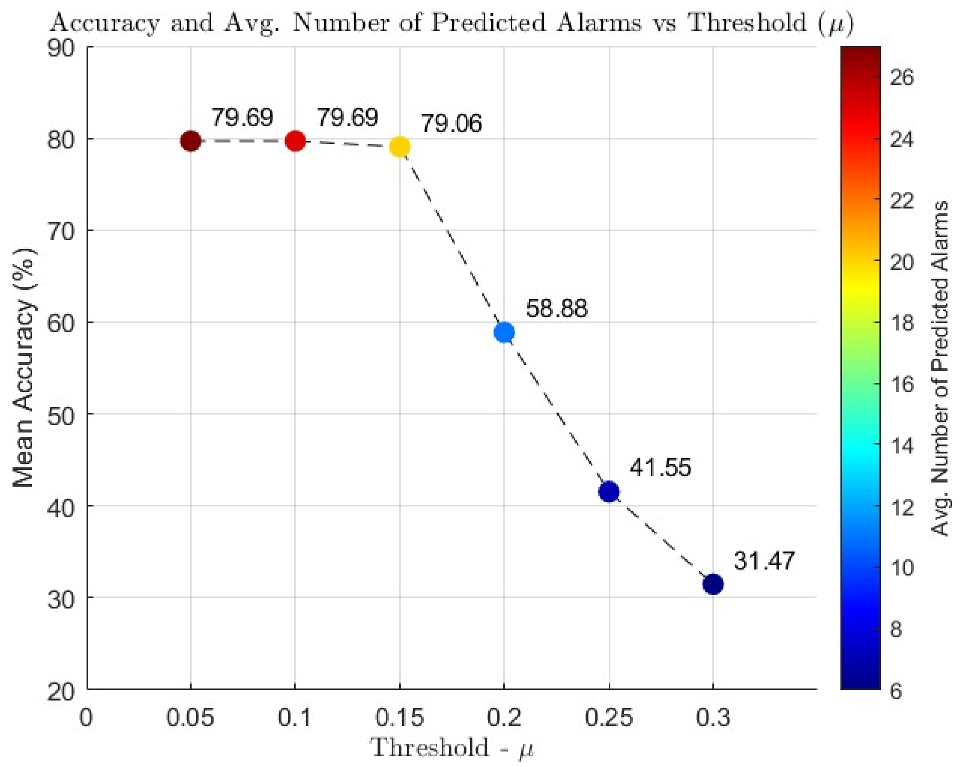


Figure 4.9: Effect of μ on the accuracy and the avgerage number of predicted alarms.

accuracy, average number of predicted alarms, and average computation time were observed, as shown in Figs. 4.9 and 4.10. The results showed that reducing μ below 0.15 significantly increased the number of predicted alarms. At $\mu = 0.15$, the accuracy was close to the maximum accuracy achieved at $\mu = 0.05$ and $\mu = 0.1$, but the average number of predicted alarms remained relatively low. The average computation time was also observed at various incremental steps of μ, as shown in Fig. 4.10. Initially, when μ is small, the average computation time was high due to that a high number of historical sequences were identified to be similar. As μ increased, the average computation time gradually decreased. At $\mu = 0.15$, the mean accuracy was slightly below the maximum accuracy obtained at $\mu = 0.05$ but the average computation time decreased significantly. This ultimately improves the response time of the algorithm. Overall, by analyzing both Figs. 4.9 and 4.10, setting $\mu = 0.15$

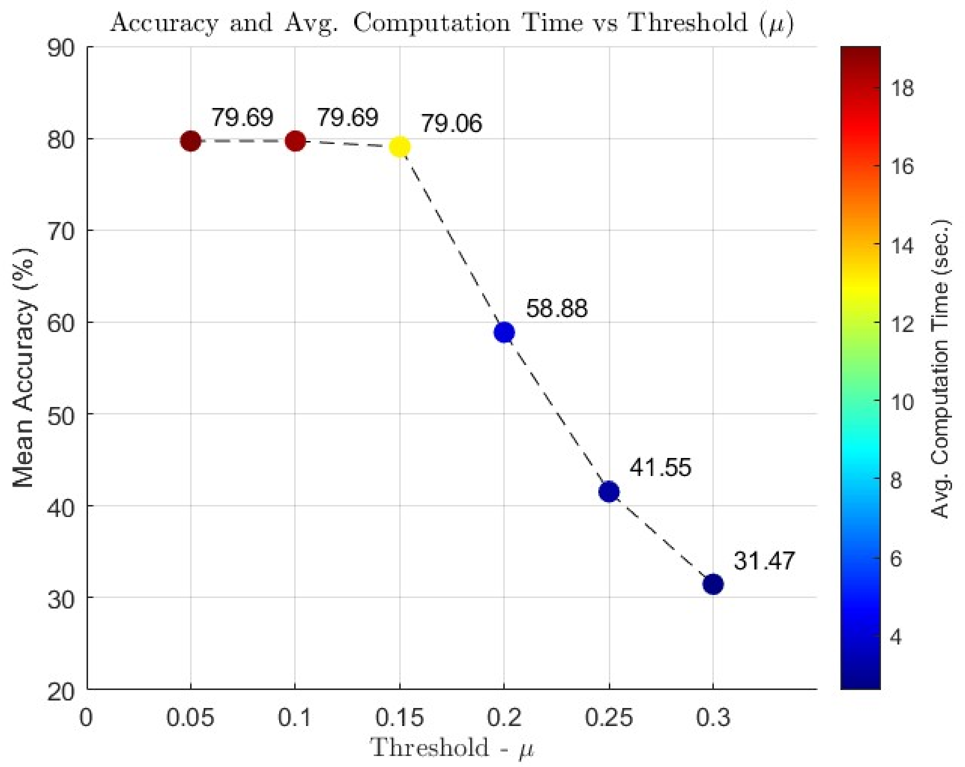


Figure 4.10: Effect of μ on the accuracy and the avgerage computation time.

is appropriate for the optimal performance of the proposed method. Additionally, the prefix length $\aleph$ is set at 5 and the sensitivity of this user-defined threshold is explained in Section 3.3.

With such optimal settings for the user-defined thresholds, the accuracy of the proposed method for the validation set under various performance criteria is shown in Table 4.7. The first column of Table 4.7 indicates the sequence number; the second and third columns show the overall accuracies for each sequence, including and excluding irrelevant alarms. Taking sequence # 6 from the validation set as an example, the proposed method correctly predicted 12 out of 15 upcoming alarms. However, two of the three missed alarms, namely, Tag_800 and Tag_1200, were deemed irrelevant in the sequence based on the relevancy criteria for the validation set. The accuracy was improved from 80% to 92.3% considering only the relevant alarms in the sequence. The mean accuracies of the proposed method, including and excluding irrelevant

Table 4.7: Accuracy of the proposed method for the validation set, including and excluding the irrelevant alarms

Seq.#	Accuracy		Position
	Including irrelevant alarms	Excluding irrelevant alarms	(max. accuracy obtained)
1	66.66%	100%	Triggering instant
2	75%	100%	Triggering instant
3	66.66%	66.66%	Triggering instant
4	83.33%	83.33%	Triggering instant
5	85.7%	100%	Triggering instant
6	80%	92.3%	11
7	75%	85.7%	11
8	83.33%	90.9%	11
9	75%	75%	12
10	100%	100%	15

alarms, were 79.06% and 89.3%, respectively. The fourth column of Table 4.7 shows the position of the incoming alarm within the sequence at which the maximum accuracy is achieved. In sequence # 1 of the validation set, the accuracy including the irrelevant alarms was 66.66%, which was obtained at the triggering instant when the alarm flood was detected initially. For the subsequent instants of incoming alarms, the accuracy remains the same. Table 4.7 shows that in most cases, the accuracy improves significantly after excluding the irrelevant alarms. Additionally, the proposed method predicted most of the upcoming alarms at the initial stage of each sequence, which is particularly beneficial for the plant operators to take early alarm responses.

4.3.2 Predictions on Upcoming Alarms

To further demonstrate the effectiveness of the proposed method, an individual case study is presented with detailed explanation of the results. Table 4.8 shows an extracted alarm flood sequence, which was used as the online sequence in this case study. The 25 alarms in this online sequence were raised one by one based on their time stamps. At any instant during the online se-

Table 4.8: Online alarm flood sequence in this case study.

Alarm Tag	Time Stamp
Tag_979	2019-12-24 04:51:21 AM
Tag_167	2019-12-24 04:51:28 AM
Tag_175	2019-12-24 04:51:34 AM
Tag_977	2019-12-24 04:51:34 AM
Tag_961	2019-12-24 04:54:00 AM
Tag_159	2019-12-24 04:55:31 AM
Tag_179	2019-12-24 04:58:48 AM
Tag_178	2019-12-24 04:59:01 AM
Tag_177	2019-12-24 04:59:14 AM
Tag_186	2019-12-24 04:59:56 AM
Tag_185	2019-12-24 05:00:09 AM
Tag_184	2019-12-24 05:00:22 AM
Tag_135	2019-12-24 05:00:37 AM
Tag_145	2019-12-24 05:00:56 AM
Tag_192	2019-12-24 05:01:11 AM
Tag_190	2019-12-24 05:01:33 AM
Tag_800	2019-12-24 05:02:04 AM
Tag_189	2019-12-24 05:02:08 AM
Tag_152	2019-12-24 05:02:27 AM
Tag_151	2019-12-24 05:02:42 AM
Tag_198	2019-12-24 05:02:58 AM
Tag_148	2019-12-24 05:03:13 AM
Tag_792	2019-12-24 05:08:26 AM
Tag_961	2019-12-24 05:09:00 AM
Tag_1200	2019-12-24 05:10:51 AM

quence, the proposed method was performed to predict the upcoming alarms and the confidence intervals of the annunciation time. At the triggering instant, the online sequence B in Table 4.8 has at least 10 alarms, and the co-occurring alarms with the highest co-occurrence frequency and confidence for each existing alarm in B are shown in Table 4.9. Tag_159 was eliminated from B as both the co-occurrence frequency and confidence of the alarm pair (Tag_159, Tag_979) are below the user-defined thresholds γ_f and γ_p. Then, according to Section 3.3, a prefix X was formulated with the most recent $\aleph$ alarms from B which are {Tag_961, Tag_179, Tag_178, Tag_177, Tag_186}. Using prefix X, there were 52 training sequences identified from CPT, where the set-based similarity scores were determined between each of the identified training sequences and the online sequence B. Based on user-defined thresholds μ, there were 8 training sequences identified to be closely similar to B.

Table 4.9: Co-occurrence frequency and confidence at the triggering instant.

Target Alarm	Co-occurring Alarm	Frequency of Co-occurrence	Confidence
Tag_979	Tag_178	13	0.35
Tag_167	Tag_175	14	0.58
Tag_175	Tag_167	14	0.56
Tag_977	Tag_979	12	0.4
Tag_961	Tag_ 175	14	0.5
Tag_159	**Tag_979**	**7**	**0.47**
Tag_179	Tag_178	13	0.87
Tag_178	Tag_979	13	0.81
Tag_177	Tag_979	13	0.93
Tag_186	Tag_979	11	0.61

According to Section 3.4, the pattern indexes identified from each of the similar sequences were $\{14, 13, 9, 9, 13, 14, 5, 5\}$. Using these pattern indexes, consequents A_y^n were identified from $\mathbb{S}$, $n = 1, 2, \cdots, |\mathbb{S}|$. In these consequents, each candidate alarm was evaluated by B_λ in terms of the criteria in Section 3.5. At the triggering instant, the varying alarm set B_λ consisted of 9 existing alarms from B and was used to analyze the first batch of candidate alarms. Table 4.10 shows the evaluation of the first candidate alarm Tag_185 by B_λ, where the satisfactory association is observed with the alarms Tag_179, Tag_178, and Tag_177 in terms of all three criteria. Thus, Tag_185 was confirmed to be a potential upcoming alarm and included in B_λ.

Table 4.10: Association between the candidate alarm Tag_185 and the varying alarm set B_λ.

Candidate Alarm	Varying Alarm Set B_λ	Confidence	Interest	Mean Occurrence Delay (Sec.)
	Tag_ 979	0.43	1.04	186
	Tag_167	0.37	0.93	294
	Tag_175	0.48	1.23	355
	Tag_977	0.30	0.86	280
Tag_185	Tag_961	0.25	0.69	420
	Tag_179	0.80	1.55	75
	Tag_178	0.75	1.50	49
	Tag_177	0.86	1.61	76
	Tag_186	0.50	1.06	71

Table 4.11 shows some potential candidate alarms and their association

rules with each existing alarm in B_λ with respect to the association rule evaluation metrics. For instance, the confidence is high, the interest is higher than 1, and the mean occurrence delay is low for the association rules between Tag_184 and each of the alarms #5, #6, #7, #9, and #10 in B_λ; thus, Tag_184 was confirmed to be a potential upcoming alarm. A significant association can also be observed in Table 4.11 between the remaining predicted alarms and B_λ. Table 4.11 shows that for the first candidate alarm Tag_185, the varying alarm set B_λ only has 9 alarms; Tag_159 was identified to be an irrelevant alarm and eliminated at the triggering instant.

Table 4.11: Prediction of upcoming alarm events at the triggering instant.

Predictions	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Confidence										
Tag_185	0.43	0.37	0.48	0.30	0.25	0.80	0.75	0.86	0.5	-
Tag_184	0.33	0.28	0.20	0.21	0.67	0.62	0.71	0.50	0.63	0.67
Tag_979	0.48	0.40	0.32	0.80	0.81	0.93	0.61	1.00	0.78	0.77
Tag_145	0.44	0.50	0.33	0.44	0.56	0.38	0.19	0.71	0.80	0.50
Tag_190	0.40	0.31	0.35	0.33	0.31	0.56	0.46	0.14	0.71	0.5
Tag_961	0.80	0.33	0.55	0.71	0.33	0.31	0.36	0.17	0.50	0.42
Interest										
Tag_185	1.04	0.93	1.2	0.86	0.69	1.55	1.50	1.61	1.06	-
Tag_184	0.95	0.82	0.66	0.67	1.43	1.39	1.48	1.19	1.39	1.13
Tag_979	0.80	0.72	0.56	1.12	1.16	1.27	0.90	1.43	0.97	1.04
Tag_145	1.13	1.20	0.93	1.14	1.06	0.88	0.89	1.21	1.60	1.00
Tag_190	1.00	0.81	0.86	0.93	0.81	1.06	1.06	0.64	1.21	1.00
Tag_961	1.08	5.12	0.73	0.89	0.55	0.46	0.61	0.34	0.71	0.53
Mean Occurrence Delay (Sec.)										
Tag_185	186	294	355	280	420	75	49	76	71	-
Tag_184	230	413	262	474	84	87	80	79	79	55
Tag_979	173	211	269	189	185	219	240	186	142	210
Tag_145	209	177	154	101	219	123	333	72	89	156
Tag_190	131	94	104	68	106	94	149	67	113	150
Tag_961	402	321	329	396	397	258	475	126	411	270

Table 4.12 shows the complete list of alarms predicted by the algorithm at the triggering instant. In Table 4.8, alarm flood is detected at the annunciation time of Tag_186 and the objective is to predict the alarms that are annunciated after alarm Tag_186. Table 4.12 shows that there are 18 alarms predicted at the triggering instant and 13 out of 18 alarms were matched with the existing alarms in the online sequence. Following the triggering instant, the online

sequence has 15 alarms, and the algorithm predicted 11 out of 15 alarms successfully at the earliest possible time of the ongoing alarm flood.

Table 4.12: Validations at the triggering instant.

Online Sequence	Predicted Alarm Events
Tag_979	**Tag_185**
Tag_167	**Tag_189**
Tag_175	**Tag_184**
Tag_977	**Tag_979**
Tag_961	**Tag_148**
Tag_159	**Tag_135**
Tag_179	**Tag_190**
Tag_178	**Tag_145**
Tag_177	Tag_160
Tag_186	**Tag_192**
Tag_185	**Tag_152**
Tag_184	Tag_176
Tag_135	**Tag_151**
Tag_145	Tag_154
Tag_192	**Tag_977**
Tag_190	Tag_174
Tag_800	Tag_191
Tag_189	**Tag_961**
Tag_152	
Tag_151	
Tag_198	
Tag_148	
Tag_792	
Tag_961	
Tag_1200	

The algorithm recomputes and attempts to modify its predictions based on the newly added alarms in B. Table 4.13 shows the predicted potential alarms when the 16th alarm in B was annunciated. Each predicted alarm holds strong association with at least one alarm in B_λ. High confidences are observed between the candidate alarm Tag_167 and each of the alarms #1, #2, #3, #5, #7, and #10 in B_λ. Only the association rules with #2 and #5 have interest ratios higher than 1. The mean occurrence delays for the association rules are much lower than 10 minutes and thus imply that true association exists between these alarms. Strong association rules are also observed between multiple alarms in B_λ and the candidate alarms Tag_198 and Tag_175.

Table 4.13: Predictions of upcoming alarms at the 16th annunciation of the online sequence.

Predictions	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Confidence										
Tag_198	0.33	0.62	0.62	0.50	0.50	0.44	0.60	0.28	0.25	0.56
Tag_167	0.60	0.70	0.67	0.32	0.86	0.39	0.67	0.32	0.29	0.71
Tag_175	0.32	0.57	0.58	0.62	0.20	0.47	0.71	0.33	0.19	0.15
Interest										
Tag_198	0.64	1.15	1.04	0.76	0.76	0.65	0.92	0.39	0.62	0.82
Tag_167	0.85	1.09	0.92	0.42	1.11	0.85	0.92	0.56	0.76	0.92
Tag_175	0.80	0.73	1.14	0.82	0.44	0.75	0.91	0.41	0.35	0.23
Mean Occurrence Delay (Sec.)										
Tag_198	58	105	116	244	180	33	219	116	475	142
Tag_167	448	392	383	330	193	174	228	285	262	342
Tag_175	173	138	141	68	210	282	338	124	262	229

The efficiency of the algorithm depends on how early and accurately it can predict the upcoming alarms and provide suggestions to the industrial operators. Table 4.14 shows the list of predicted alarms when the 16th alarm in B is annunciated. It can be seen that most of the predicted alarms are the same as the predictions at the triggering instant, which demonstrates that the algorithm can predict the potential upcoming alarms at the earliest possible time. Meanwhile, there are 3 newly predicted alarms Tag_198, Tag_167, and Tag_175.

The sensitivity of the proposed method to the training set was extensively investigated, and an illustration is shown in Fig. 10 with reference to the individual case study presented above. This research proposed an unsupervised approach to predict upcoming alarms by exploiting the information available in the training set. Given its unsupervised nature, the primary requirement of building the training set is to maximize the number of historical sequences in the training set, thus enabling the identification of highly potential upcoming alarms owing to the availability of more similar historical sequences. For instance, 103 historical sequences were used to formulate the training set for the individual case study in the revised manuscript. To examine how the accuracy of the proposed method is affected by the number of historical sequences in

Table 4.14: Validations at the 16th annunciation of the online sequence.

Online Sequence	Predicted Alarm Events
Tag_979	Tag_191
Tag_167	**Tag_961**
Tag_175	**Tag_189**
Tag_977	**Tag_198**
Tag_961	**Tag_979**
Tag_159	**Tag_148**
Tag_179	**Tag_167**
Tag_178	**Tag_151**
Tag_177	**Tag_977**
Tag_186	Tag_160
Tag_185	Tag_188
Tag_184	Tag_870
Tag_135	**Tag_175**
Tag_145	Tag_174
Tag_192	Tag_166
Tag_190	Tag_176
Tag_800	
Tag_189	
Tag_152	
Tag_151	
Tag_198	
Tag_148	
Tag_792	
Tag_961	
Tag_1200	

the training set, the number of sequences was increased from 20 to 100, by adding 20 sequences each time. Fig. 4.11 shows the complete results, which indicate that the accuracy is expected to improve as the number of historical sequences in the training set increases. However, when the number of historical sequences becomes large, the improvement in accuracy is deemed insignificant.

4.3.3 Predictions on Time of Annunciations

Further simulations were conducted to test the prediction of annunciation time using the proposed method. Table 4.15 shows prediction as the confidence intervals on the time gaps for the annunciations of subsequent predicted alarm events. The first alarm Tag_186 in Table 4.15 is the alarm that

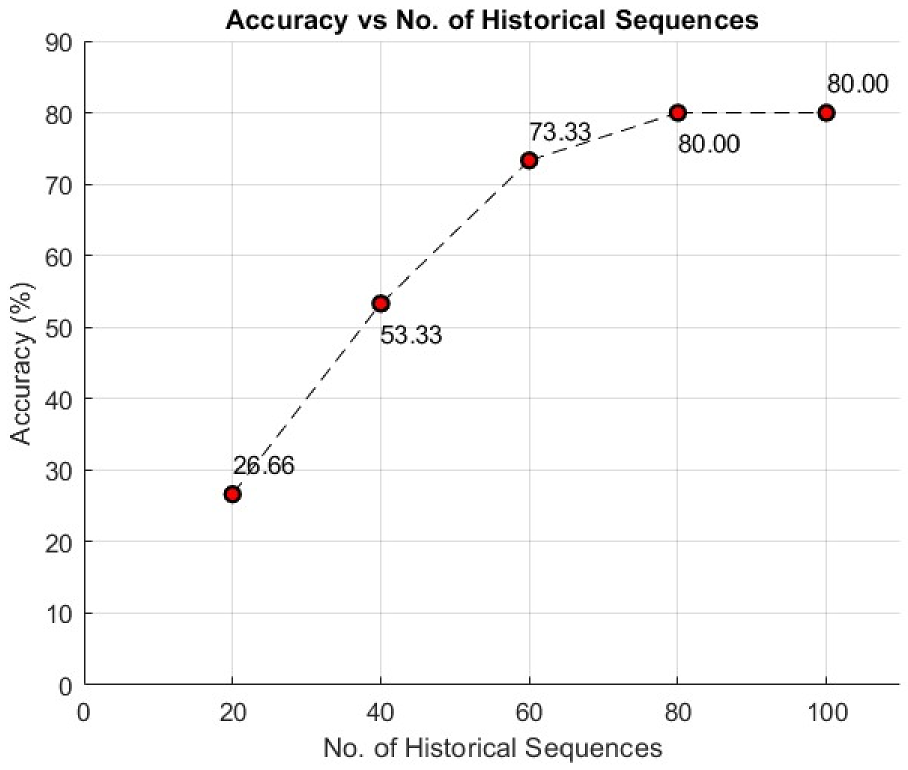


Figure 4.11: Effect of increasing the number of historical sequences on the accuracy of the proposed method.

occurred at the triggering instant of B. The confidence intervals are shown only for those predicted alarms that also exist in B following the triggering instant. While calculating the confidence interval, the order of the predicted alarms was kept similar to the online sequence. For instance, the confidence interval for Tag_185 is $[0, 137]$, which indicates that the time gap between the annunciations of Tag_185 and Tag_186 is between 0 and 137 seconds during the alarm flood. In addition, as shown in Table 4.16, the confidence intervals can also be obtained between the annunciations of the predicted alarm Tag_185 and each of the remaining predicted alarms. Such information will ease off the overwhelming situation for the industrial operators and assist in planning the corrective actions during the online flood.

Table 4.15: Confidence interval on the time gap between the subsequent predicted alarms.

Predicted Alarms	95% Confidence Interval (seconds)
Tag_186	—
Tag_185	[0, 137]
Tag_184	[0, 69]
Tag_135	[0, 364]
Tag_145	[25, 153]
Tag_192	[0, 208]
Tag_190	[14, 128]
Tag_189	[0, 131]
Tag_152	[0, 397]
Tag_151	[0, 74]
Tag_148	[19, 166]

Table 4.16: Confidence interval on the time gap between Tag_185 and each predicted alarm.

Target Alarm	Predicted Alarms	95% Confidence Interval (seconds)
Tag_ 185	Tag_ 184	[0, 69]
	Tag_ 135	[0, 236]
	Tag_145	[28, 200]
	Tag_192	[0, 142]
	Tag_190	[0, 195]
	Tag_189	[23, 90]
	Tag_152	[0, 436]
	Tag_151	[26, 485]
	Tag_148	[72, 255]

4.4 Summary

In this chapter, we addressed the problem of predicting the upcoming alarm events during an ongoing alarm flood by providing a systematic real-time decision support framework. Initially, we introduced the problem of predicting an upcoming alarm events, and then, we presented the systematic alarm prediction method. The method involved several key steps, including the offline training of a compact prediction tree, real-time exclusion of irrelevant alarms from the online sequence, evaluation criteria for selecting candidate alarms, and prediction of the time for upcoming alarm events. These steps were designed to provide effective and timely assistance to plant operators in dealing with alarm floods. Finally, we demonstrated the effectiveness of the proposed method based on a real historical data set collected from a coking plant of an oil refinery. The case study showcased how the method could enable plant operators to proactively identify and address issues associated with alarm floods, allowing for more efficient problem mitigation.

Chapter 5

A Reinforcement Learning Approach for Early Prediction of Industrial Alarm Floods

In this chapter, our primary focus is on addressing some additional challenges related to the early prediction of industrial alarm floods. Industrial alarm floods signify the occurrence of a major problem that demands immediate and effective measures to mitigate the situation. Without crucial information about current and upcoming alarms, responding efficiently becomes challenging, especially when the alarm rate is high. Therefore, the early prediction of alarm floods is of utmost importance as it provides real-time guidance to plant operators, enabling them to promptly take necessary actions to mitigate such situations. However, predicting alarm floods comes with its own set of challenges. One major challenge is the presence of irrelevant alarms, which can obscure the true alarm pattern caused by an underlying fault. Moreover, the ongoing situation may deviate from previously observed historical sequences that are considered similar, incorporating a different set of alarms. This lack of similarity can leave plant operators without the necessary information to respond effectively, leading to a significant deterioration of the situation. Additionally, the accuracy and timeliness of corrective actions play a vital role in ensuring operational efficiency for plant operators during alarm floods.

To tackle these challenges, we propose an approach based on reinforcement learning (RL). We formulate the early prediction problem as a partially observable Markov decision process (POMDP) and employ the DDQN (Double Deep Q-Network) algorithm with improved learning processes to optimize accuracy and effectiveness. Additionally, we introduce a sequence reconstruction strategy that leverages association rule mining. This strategy helps eliminate irrelevant alarms and generates potential online scenarios by utilizing the alarm relations found in historical sequences. Furthermore, the training set is reformulated based on several novel criteria for effective learning of the algorithm. To demonstrate the effectiveness of the algorithm, an industrial case study based on the real alarm & event log is presented.

5.1 Background

5.1.1 Industrial Alarm Floods

An alarm flood refers to an abnormal situation where alarm rate gradually increases within a short time interval, exceeding the ability of an industrial operator to efficiently handle the situation. According to the industrial standard [30], the threshold for detecting an alarm flood is 10 alarms in 10 minutes, denoted as λ_1. An alarm flood situation continues to develop until the current alarm rate λ_t falls below 5 alarms in 10 minutes, denoted by λ_2. In such situations, it can be challenging for industrial operators to efficiently respond to all the critical alarms. To assist them, recommending similar scenarios from the historical database can be an effective real-time aid, particularly at the early stages of the ongoing alarm flood. A historical alarm flood sequence is denoted as

$$X_m = [x_1^m, x_2^m, \cdots, x_{|X_m|}^m] \tag{5.1}$$

where $|.|$ indicates the cardinality of a set; $x_i^m = (e_i^m, t_i^m)$ where $e_i^m \in \sum$; $\sum = \bigcup_{k=1}^{|\sum|}\{e_k\}$, is the set of all the unique alarm tags set up in a plant; t_i^m is the corresponding time stamp; $i = 1, 2, \cdots, |X_m|$; $m = 1, 2, \cdots, |\mathbb{Z}|$; $\mathbb{Z}$ indicates the set of all the historical alarm flood sequences X_m associated with the corresponding alarm flood identification (ID) I_m and is denoted as

$$\mathbb{Z} = \bigcup_{m=1}^{|\mathbb{Z}|} \{(X_m, I_m)\} \tag{5.2}$$

where $m = I_m \in \mathbb{N}^+$. RL problems are widely represented as Markov decision processes (MDP) [54] which are characterized by the tuple (S, A, T, R), where S represents the state space, A represents the action space, T represents the transition model, and R represents the reward function. A state $s \in S$ can be defined as $(X_m, I_m) \in \mathbb{Z}$ where X_m indicates the complete alarm flood sequence, which is only available when the alarm flood terminates, i.e., when $\lambda_t < \lambda_2$.

At the triggering instant t, an online alarm flood is denoted as

$$Y = [y_1, y_2, \cdots, y_k] \tag{5.3}$$

Following the triggering instant, if an alarm y_{k+1} or a group of n alarms $\{y_{k+i}\}_{i=1}^{n}$ annunciate simultaneously, the alarm set in Y is modified as $\{y_l\}_{l=1}^{k} \cup y_{k+1} \vee \{y_l\}_{l=1}^{k} \cup \{y_{k+i}\}_{i=1}^{n}$ given that $\lambda_t < \lambda_2$. At any instant of an ongoing alarm flood, complete information of the state, i.e., the entire sequence is not available. Thus, the RL problem can be framed as a partially observable Markov decision process (POMDP) which we can represent by the tuple $\{S, A, T, \Omega, R, \gamma\}$ where Ω indicates the set of observations and γ denotes the discount factor. In POMDP, instead of the complete information, the RL agent will receive an observation of the state, which is a partial view of the entire online alarm flood sequence. Based on the current and past observations of the online alarm flood sequence, the RL agent will attempt to infer the complete information of the entire sequence and identify the most similar scenario from the set of historical sequences. Thus, an observation $\Phi \in \Omega$ of an online sequence Y is a partial alarm flood sequence and is denoted as

$$\Phi = [y_1, y_2, \cdots, y_{|\varphi|}] \tag{5.4}$$

Action space A consists of the set of actions that the agent can choose in each observation. The agent can perform either an action of prediction

$a_p \in A_p$ or an action of deferral a_d in each observation, where A_p is the set of action of predictions denoted as $A_p = \{a_{p_i} = I_i | i = 1, 2, \cdots, |\mathbb{Z}|\}$.

5.1.2 Reinforcement Learning

Reinforcement Learning is the science of solving a decision-making task, depicted by an agent that learns to act optimally in an interactive environment. In RL, an environment is framed as Markov Decision Processes (MDPs), where an agent performs an action $a \in A$ in each state $s \in S$, receives reward or punishment as feedback, and makes a transition to the next possible state. Agent's actions and transitions through states are guided by a reward function R and transition model T which determine the best course of action for the agent. By experimenting and observing the results, the agent learns the optimal policy that maximizes the state values and leads to the highest overall reward.

The value of a state $V_\pi(s)$ indicates how rewarding for an agent in terms of expected return [54] to start from state s at time step t and then following policy π:

$$V_\pi(s) = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | S_t = s] \tag{5.5}$$

where γ is the discounting factor that encourages immediate reward and r_{t+k} is the reward obtained at time step $t + k$. The action value (Q-value) [54] depends on the action performed and indicates how rewarding for an agent to choose such an action in state s and then following policy π:

$$Q_\pi(s, a) = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | S_t = s, A_t = a] \tag{5.6}$$

where $Q_\pi(s, a)$ represents the action value of state s with a particular action a. The objective is to solve a decision making problem by setting up an optimal policy, denoted as π^*, that yields maximum accumulated reward from the MDP. The optimal state value $V^*(s)$ for all $s \in S$ and action value $Q^*(s)$ for all $s \in S$ and $a \in A$ are denoted as [54]:

$$V^*(s) = \max_\pi V_\pi(s) \tag{5.7}$$

$$Q^*(s,a) = \max_{\pi} Q_\pi(s,a) \tag{5.8}$$

Eqn 5.7 and Eqn 5.8 indicate that the state value $V^*(s)$ and the action value $Q^*(s)$ corresponding to the optimal policy yield the maximum expected return over the finite MDPs. Optimal policy maps the states to the best actions by searching greedily using the action value function such that $\pi^*(s) = \arg\max_a Q^*(s,a)$. To find the optimal policy, the Bellman equation [54] can be used, which iteratively updates the action value of each state-action pair until it converges to the optimal solution. As shown in Eqn 5.9, Bellman equation decomposes the action value of a particular state into two components: the immediate reward gained from the current state, and the expected cumulative reward of following the optimal policy, π^*, thereafter:

$$Q_\pi(s,a) = E_\pi[r_t + \gamma Q_\pi(s_{t+1,}a_{t+1})|s_t = s, a_t = a] \tag{5.9}$$

To ensure stable learning and approximate the optimal action value function $Q(s,a,\theta_t)$ with optimal parameters θ_t, a Deep Q-network (DQN) algorithm [39] can be exploited, which uses multiple neural networks with identical configurations: target network and prediction network. With a single network, the predicted and target Q-values are the same. This may lead to instability as the Q-value for a particular action may get nominated repeatedly, leading to overshooting and bias towards that action. Therefore, other viable actions will not be explored, resulting in suboptimal learning. In the DQN algorithm, the prediction network aims to approximate the action Q values estimated by the target network, which guide the learning process. For optimal learning, the target network's parameters θ_t are periodically updated from the prediction network, which updates its parameters θ_p in each training episode.

For the training process, all possible interactions between the agent and the environment in the form of (s,a,r,s') are used to form a replay buffer. The purpose of using the replay buffer and randomly sampling the past interactions from it in the form of mini batches during each training epoch is to ensure learning from diverse set of experiences. Without the replay buffer, the RL agent may learn too well from recent experiences and may get biased toward certain actions in specific situations. This may lead to poor performance for the agent in diverse situations. The DQN algorithm uses the Bellman equation and applies stochastic gradient descent on a minibatch of past experiences to minimize the loss function in the following equation:

$$L(\theta) = (r + \gamma \arg\max_{a_{t+1}} Q(s', a_{t+1}, \theta_t) - Q(s, a, \theta_p))^2 \tag{5.10}$$

In DQN, the reward from the current state is combined with the maximum action Q-value of the next state in the target network. However, a problem arises when the Q-value for a state becomes overestimated each time due to the biasedness for a specific action. In such cases, the scope for exploring other viable actions is reduced, and thus, the learning process becomes biased. To resolve such issues, a DDQN algorithm is implemented, where the prediction network chooses the best action for the next state, and the corresponding action Q-value of the target network is used in the Bellman equation. Therefore, the loss function in eqn 5.10 is modified as:

$$L(\theta) = (r + \gamma Q(s', \arg\max_{a_{t+1}} Q(s', a_{t+1}, \theta_p), \theta_t) - Q(s, a, \theta_p))^2 \tag{5.11}$$

A reward function, denoted as $R(s, a)$, maps a state s and an action a performed in that state to a scalar reward value. The reward function is used to evaluate the quality of the agent's actions and guide its learning process [54]. The ultimate goal of the agent in reinforcement learning is to maximize the cumulative reward it receives over time. Thus, the reward function plays a crucial role in defining the objectives of the agent and shaping its behavior. In RL, the transition model T represents how the environment will evolve based on the actions taken by the agent [54]. It specifies the probability of transitioning from one state to another, given an action taken by the agent. For example, for a discrete state space with states S and a discrete action space with actions A, T can be defined as a function $T : S \times A \rightarrow P(S')$. In this case, for a given state-action pair (s, a), the function T maps it to a probability distribution over next states s', represented by $P(S')$. This distribution tells us the probability of transitioning to each of the possible next states, given the current state and action. For an early classification problem, T is deterministic and defined as $T(s, a) = s'$ with probability 1, which means that the next state is deterministically determined by the current state and action.

5.2 Problem Description

During an alarm flood, an industrial operator can be proactive with context-specific guidance obtained from past experience and industry best practices. This proactive approach should involve prioritizing and evaluating critical alarms based on their potential impact on the process at the earliest. Thus, an operator's efficiency during an alarm flood greatly relies on promptly identifying accurate corrective actions. A real-time decision support framework is an effective solution to such kind of problem. The objective of such a framework is to provide real time assistance in the form of identifying similar scenarios from the training set at the earliest. To ensure the efficiency and reliability of such a framework, several criteria need to be considered:

Issue 1: In an alarm flood sequence, it is crucial to identify the true alarm pattern that represents the underlying fault. This enables accurate recommendations for plant operators. Existing methods only eliminate chattering alarms prior to perform alarm flood analysis, However, irrelevant alarms that are rarely associated with the true alarm pattern may also exist and bury the true alarm pattern. In such a scenario, results from alarm flood analysis will be inaccurate and may confuse the operators.

Issue 2: The efficiency of such a framework depends on how early the alarm flood situation can be resolved to prevent major process interruptions. Thus, early prediction is a priority and needs to fulfill the following conditions:

$$I_k = \arg\max_{I_k \in A_p} \mathbb{S}(X_{I_k}, Y) \tag{5.12}$$

$$|\Phi| = \arg\min_{|\Phi| \in [1, |Y|]} t_p \tag{5.13}$$

where $k \in [1, 2, \cdots, |\mathbb{Z}|]$, and $\mathbb{S}(X_{I_k}, Y)$ represents the similarity index between X_{I_k} and Y that can be determined using the Smith-Waterman algorithm (SWA) [45]. Eqn 5.12 states that the historical sequence X_{I_k} corresponding to the alarm flood ID I_k, recommended by the algorithm, should have the maximum similarity index with the online sequence Y. In that case, maximum similarity index also indicates the accuracy of the prediction action. In Eqn 5.13, Φ is considered as the current observation of the online sequence when the RL agent predicts the incoming sequence and chooses action of prediction over action of deferral; t_p represents the time required for the RL agent to perform such action; $|\Phi|$ indicates the length of the observation. Eqn 5.13 is complementary to Eqn 5.12 and states that the objective is to predict the incoming sequence at the earliest possible time while maintaining the optimum accuracy in terms of similarity index.

Issue 3: During an alarm flood, an online alarm flood sequence may initially appear similar to a particular historical sequence, but may significantly deviate later and resemble a different historical sequence. In such situations, early recommendations provided may not be accurate and mislead the operators.

Motivated by the issues outlined above, this work proposes a methodology that includes the following steps:

Step 1: To address the issues 1 and 3, this work proposes a sequence reconstruction strategy. In this strategy, an association rule approach is used on the existing historical sequences in $\mathbb{Z}$ to remove irrelevant alarms and reconstruct sequences. Such reconstructed sequences are not available in $\mathbb{Z}$ but may occur during an alarm flood. Each of the reconstructed sequences will be assigned the alarm flood ID of the most similar historical sequence in $\mathbb{Z}$. The purpose is to train the algorithm on all possible variations of the online sequence that may occur while repeating a historical alarm pattern during an alarm flood. By exploiting the information available in $\mathbb{Z}$, a modified training set $\mathbb{D}$ is formulated with only the existing and reconstructed alarm patterns for the training of RL agent. To provide more than one recommendation, historical sequences in $\mathbb{Z}$, which are similar to each other will be identified and grouped using SWA. During an alarm flood, the algorithm will perform the following action:

$$\mathbb{C} : Y_{:t} \rightarrow I_k \in G_{I_k} \tag{5.14}$$

where $Y_{:t}$ indicates the online sequence with alarms annunciated until time instant t, I_k is the alarm flood ID where $k \in [1, 2, \cdots, |\mathbb{Z}|]$ and $G_{I_k} = \{I_i | \exists j \in [1, 2, \cdots, |\mathbb{Z}|] \; s.t. \; \mathbb{S}(X_{I_i}, X_{I_j}) \geq \rho_s, I_i \neq I_j, I_i, I_j \in \mathbb{N}^+\}_{i=1}^{|G_{I_k}|}$. $\mathbb{S}(X_{I_i}, X_{I_j})$ indicates the similarity index between the historical sequences, and ρ_s denotes a user defined threshold. Eqn 5.14 states that the algorithm, denoted as a mathematical function $\mathbb{C}$, will map an ongoing alarm flood sequence $Y_{:t}$ to the alarm flood id I_k corresponding to the most similar historical sequence $X_{I_k} \in \mathbb{Z}$. The recommended alarm flood ID I_k is associated with a group of similar historical sequences, denoted as G_{I_k}. Therefore, the algorithm will provide multiple scenarios, all of which have high similarity with each other and with the online sequence, allowing the plant operators to review and plan accurate corrective actions.

Step 2: To handle the second issue, the DDQN algorithm is adopted, where the neural networks are trained over all possible observations from the historical sequences to approximate the optimal action Q-value function. With the optimal parameters, the DDQN algorithm learns to predict at the earliest stage of an ongoing alarm flood while maintaining optimal accuracy. Thus, the main objective of this method can be described as follows:

$$\begin{gathered} \underset{a \in A_p}{\arg\max}\, Q(s(t), a, \theta_t) = I_k \in G_{I_k} \\ s.t. \; \mathbb{S}(X_{I_k}, Y_{:t}) = max(\mathbb{S}_{SWA}) \text{ and } |\Phi| = \underset{|\Phi| \in [1, |Y|]}{\arg\min}\, t_p \end{gathered} \tag{5.15}$$

where $\mathbb{S}_{SWA} = \{\mathbb{S}(X_{I_i}, Y_{:t}) | \mathbb{S}(X_{I_i}, Y_{:t}) \geq \rho_s, I_i \in \mathbb{N}^+\}_{i=1}^{|\mathbb{S}_{SWA}|}$. In Eqn 5.15, performing an action of prediction $a \in A_p$ interprets as recommending an alarm flood ID on an ongoing alarm flood sequence such that the corresponding historical sequence will have maximum similarity provided that the time elapsed between the detection of alarm flood and the availability of such recommendation will be minimum. The steps proposed above are intended to allow industrial operators to plan and execute corrective actions in a prompt and efficient manner.

5.3 Methodology

This section proposes strategies based on association rule mining to eliminate irrelevant alarms from online sequence and reconstruct sequences from historical sequences. Next, a comprehensive explanation of the DDQN algorithm is presented.

5.3.1 Irrelevant Alarms in Online Sequence

Due to random variations in the process or disturbances, some alarms may trigger and add to the online sequence, which often obscures the true alarm pattern that represents the underlying fault. Such irrelevant alarms may lead to false prediction actions and confuse the plant operators. Thus, it is necessary to remove such irrelevant alarms from an online sequence. At the triggering time instant t, i.e., when $\lambda_t \geq \lambda_1$, an alarm flood sequence consists of the set of alarms annunciated within $[t-\Lambda+1, t]$, which is denoted as $\mathbb{Y}_{:t} = \{y_i | e_i^y \in \sum, \exists\ x_i^y(l) = 1 \text{ s.t. } l \in [t-\Lambda+1, t]\}_{i=1}^{|\mathbb{Y}_{:t}|}$ where $x_i^y(l)$ indicates a binary signal [46] of an alarm e_i^y and $\Lambda = 600s$. The relevance between each alarm and the remaining alarms in $\mathbb{Y}_{:t}$ at triggering instant can be analyzed based on the following association rule metrics:

1) Confidence, denoted by (η_{conf}), is the measure of the likelihood of the association rule [23] between two alarms $(A \rightarrow B)$ in the dataset. In this study, confidence provides the probability of an alarm B occurs given that an alarm A is present within the time interval 600s and denoted as follows:

$$\eta_{conf}(A \rightarrow B) = \frac{\text{supp}(A \cup B)}{\text{supp}(A)} = \frac{|A \cup B|}{|A|}$$

where ($|A \cup B|$) represents the number of times alarms A and B occur together, while $|A|$ represents the number of occurrences of alarm A separately. Confidence values range from 0 to 1, with a higher value indicating a stronger association between the two alarms. The time interval is set at 600s to ensure consistency with the alarm flood detection criteria, which is at least 10 alarms [30] in a 10-minute time period. This is based on the assumption that the alarms that are strongly associated and have an occurrence delay of less than 600s are highly likely to be present in alarm floods.

2) Interest, denoted by (η_{int}), quantifies the correlation between two alarms in relation to what would be expected if the alarms were not related. It evaluates the extent to which the likelihood of the occurrence of A is affected by the occurrence of B.

$$\eta_{int}(A \rightarrow B) = \frac{\eta_{conf}(A \rightarrow B)}{\mathrm{supp}(B)}$$

An interest value greater than 1 signifies that the association between the alarms is significant. Conversely, an interest value less than 1 suggests that there is no real association between the alarms A and B, and their occurrences are independent of each other.

Confidence and interest between each alarm $y_i \in \mathbb{Y}_{:t}$ and the remaining alarms in $\mathbb{Y}_{:t}$ are computed and are included in $\mathbb{C}_i^{conf}$ and $\mathbb{C}_i^{int}$, which are the set of confidence and interest for alarm $y_i \in \mathbb{Y}_{:t}$ where $i = 1, 2, \cdots, |\mathbb{Y}_{:t}|$. For example, the set of confidence for alarm $y_i \in \mathbb{Y}_{:t}$ is denoted as $\mathbb{C}_i^{conf} = \{\eta_{conf_{ij}} | \{y_i, y_j\} \in \mathbb{Y}_{:t}, i \neq j\}_{j=1}^{|\mathbb{C}_i^{conf}|}$. Analogously, $\mathbb{C}_i^{int}$ is defined for each alarm $y_i \in \mathbb{Y}_{:t}$ as $\mathbb{C}_i^{int} = \{\eta_{int_{ij}} | \{y_i, y_j\} \in \mathbb{Y}_{:t}, i \neq j\}_{j=1}^{|\mathbb{C}_i^{int}|}$. Then, based on user-defined thresholds ρ_{conf} and ρ_{int}, irrelevant alarms are eliminated from $\mathbb{Y}_{:t}$ at the triggering instant. Thus, an alarm $y_i \in \mathbb{Y}_{:t}$ is considered a relevant alarm if it complies with the following criterion:

$$\begin{aligned} \exists j \in [1, 2, \cdots, |\mathbb{Y}_{:t}|] \; s.t. \; \eta_{conf_{ij}} \in \mathbb{C}_i^{conf} &\geq \rho_{conf} \\ \text{and } \eta_{int_{ij}} \in \mathbb{C}_i^{int} &\geq \rho_{int} \end{aligned} \tag{5.16}$$

After eliminating the irrelevant alarms, the set of alarms in online sequence

$Y_{:t}$ at triggering instant is denoted as

$$\begin{aligned}\mathbb{Y}_{:t} = \{y_i | e_i^y \in \sum, \exists e_j^y \in \mathbb{Y}_{:t} \ s.t. \ \eta_{conf_{ij}} \geq \rho_{conf}, \\ \eta_{int_{ij}} \geq \rho_{int}, i \neq j\}_{i=1}^{|\mathbb{Y}_{:t}|}\end{aligned} \tag{5.17}$$

Following triggering instant, an incoming alarm will only be added to the existing alarms in $\mathbb{Y}_{:t}$ if the following criterion is satisfied:

$$\tilde{\mathbb{Y}}_{:t} = \{y_i\}_{i=1}^{|\mathbb{Y}_{:t}|} \cup \ y_{|\tilde{\mathbb{Y}}_{:t}|}, \tag{5.18}$$

where $\exists j \in [1, \cdots, |\mathbb{Y}_{:t}|] \ s.t. \ \eta_{conf_{(|\tilde{\mathbb{Y}}_{:t}|)j}} \geq \rho_{conf}$, $\eta_{int_{(|\tilde{\mathbb{Y}}_{:t}|)j}} \geq \rho_{int}$. This step ensures that the alarms that exist in the online sequence have strong association with each other and form a pattern that represents an underlying fault.

5.3.2 Sequence Reconstruction

After the pre-processing phase, the online alarm flood sequence will consist of only the relevant alarms at the triggering instant, and alarms will annunciate one by one and only add to the online sequence if it complies with Eqn 5.18, until $\lambda_t < \lambda_2$. During an alarm flood, the RL agent will experience the same phenomenon and sequentially make decisions in each observation of the online sequence. To train the RL agent on early prediction actions while maintaining optimal accuracy, it is imperative to create an identical environment for the training process. To create such an environment, a training set $\mathbb{D}$ is formulated with all possible observations that are most likely to repeat online during an alarm flood. Some of these observations are not available in $\mathbb{Z}$ but may appear during an ongoing alarm flood. Such observations are formulated by using an association rule approach on the existing historical sequences in $\mathbb{Z}$. In DDQN algorithm, neural network ought to be trained with the interactions between the RL agent and the observations in $\mathbb{D}$ to approximate the optimal action Q-value function. To create such observations, following criteria will be considered:

Criteria 1: Eqn 5.1 shows a historical sequence that includes the time of annunciation of each alarm. Using the temporal information and the industrial standard [30], the triggering instant of the corresponding historical sequence can be identified. At triggering instant, the set of alarms in a historical sequence X_m is denoted as

$$\begin{aligned}\tilde{\mathbb{X}}_m = \{x_i^m | e_i^m \in \sum, \exists\ x_i^m(l) = 1, \\ \text{s.t. } l \in [t^m - \Lambda + 1, t^m]\}_{i=1}^{|\tilde{\mathbb{X}}_m|}\end{aligned} \tag{5.19}$$

where $m = 1, 2, \cdots, |\mathbb{Z}|$, $x_i^m(l)$ indicates a binary signal [46] of alarm e_i^m and t^m is the triggering instant of X_m. Afterward, alarms were annunciated one by one and added to $\tilde{\mathbb{X}}_m$. Similar to online sequences, irrelevant alarms may also exist in historical sequences that are not consistently associated with the true alarm pattern and need to be eliminated. Afterward, a set of observations Ω_1^m from the same historical sequence X_m can be formulated for effective training of the RL agent. Following the criterion in Eqn 5.17, the set of alarms in an observation Φ_1^{m,Ω_1} at the triggering instant t^m is denoted as

$$\begin{aligned}\varphi_1^{m,\Omega_1} = \{x_i^m | e_i^m \in \sum, \exists e_j^m \in \tilde{\mathbb{X}}_m\ s.t.\ \eta_{conf_{ij}} \geq \rho_{conf}, \\ \eta_{int_{ij}} \geq \rho_{int}\}_{i=1}^{|\varphi_1^{m,\Omega_1}|}\end{aligned} \tag{5.20}$$

Then, an alarm $x_k^m \in X_m$ will only be added in Φ_1^{m,Ω_1} to initialize a new observation Φ_2^{m,Ω_1} if the following criterion is satisfied:

$$\varphi_2^{m,\Omega_1} = \varphi_1^{m,\Omega_1} \cup\ x_k^m, \tag{5.21}$$

where $\exists j \in [1, \cdots, |\varphi_1^{m,\Omega_1}|]\ s.t.\ \eta_{conf_{jk}} \geq \rho_{conf}, \eta_{int_{jk}} \geq \rho_{int},\ k \in [|\tilde{\mathbb{X}}_m| + 1, \cdots, |X_m|]$, and φ_2^{m,Ω_1} indicates the set of alarms in observation Φ_2^{m,Ω_1}. Therefore, Ω_1^m is initially formulated as follows:

$$\begin{aligned}\Omega_1^m = \varphi_1^{m,\Omega_1} \cup \{\varphi_i^{m,\Omega_1} = \varphi_{i-1}^{m,\Omega_1} \cup x_k^m | \varphi_i^{m,\Omega_1} \subseteq \{x_l^m\}_{l=1}^{|X_m|}, \\ \varphi_i^{m,\Omega_1} \notin \{\varphi_n^{m,\Omega_1}\}_{n=1}^{|i-1|}, k \in [|\tilde{\mathbb{X}}_m| + 1, \cdots, |X_m|]\}_{i=2}^{|\Omega_1^m|}\end{aligned} \tag{5.22}$$

Criteria 2: An online sequence may show resemblance to multiple historical sequences at different stages during an alarm flood. The sequential alarm pattern that exists in a historical sequence may change significantly while repeating during an alarm flood and may show resemblance to a different historical sequence at a later stage. Such phenomenon may lead to false corrective actions at the early stage of an alarm flood. An RL agent should also be trained to predict accurately in such situations and assist operators with unwavering decisions. To create such identical situations from X_m, which are not available in $\mathbb{Z}$ but may occur during an alarm flood, the association rule approach introduced in section 5.3.1 can be systematically exploited. Initially, $\varphi_1^{m,\Omega_1} \in \Omega_1^m$, obtained from X_m at the triggering instant t^m, is used to identify most similar historical sequences from $\mathbb{Z}$ using Sorensen-Dice similarity co-efficient, denoted as

$$D_i = 2 * \frac{|\varphi_1^{m,\Omega_1} \cap \mathbb{X}_i|}{|\varphi_1^{m,\Omega_1}| + |\mathbb{X}_i|} \tag{5.23}$$

where $\mathbb{X}_i$ represents the set of alarms that exist in the historical sequence X_i and $i = 1, 2, \cdots, |\mathbb{Z}|$. Based on a user-defined threshold ρ_d on $\{D_i\}_{i=1}^{|\mathbb{Z}|}$, a set of similar historical sequences is identified and denoted as $S_1^{m,\Omega_1} = \{X_s | D_s \geq \rho_d, X_s \in \mathbb{Z}\}_{s=1}^{|S_1^{m,\Omega_1}|}$. Then, the relevance between each alarm $x_j^s \in X_s$ and the alarms in φ_1^{m,Ω_1} is evaluated based on confidence η_{conf}, and interest η_{int} where $j = 1, \cdots, |X_s|$. Based on the existing alarms in φ_1^{m,Ω_1}, a new set of alarms can be nominated using the following criteria:

$$\begin{aligned} \mathbb{X}_p = \{x_k | x_k \in \textstyle\sum, x_k \notin X_m, \exists l \in [1, \cdots, |S_1^{m,\Omega_1}|] \\ s.t.\ x_k \in X_l \text{ and } \exists j \in [1, \cdots, |\varphi_1^{m,\Omega_1}|] \\ s.t.\ \eta_{conf_{jk}} \geq \rho_{conf}, \eta_{int_{jk}} \geq \rho_{int}\}_{k=1}^{|\mathbb{X}_p|} \end{aligned} \tag{5.24}$$

A new observation will only be initialized when a single or multiple alarms in $\mathbb{X}_p$ are added to Φ_1^{m,Ω_1}. For example, a new observation Φ_1^{m,Ω_1^1} is formulated with an alarm $x_k \in \mathbb{X}_p$ based on the following criterion:

$$\varphi_1^{m,\Omega_1^1} = \varphi_1^{m,\Omega_1} \cup\ x_k, \tag{5.25}$$

where $k \in [1, 2, \cdots, |\mathbb{X}_p|]$, φ_1^{m,Ω_1^1} represents the set of alarms in observation Φ_1^{m,Ω_1^1}, $\varphi_1^{m,\Omega_1^1} \notin \Omega_1$, and $\Phi_1^{m,\Omega_1^1} \notin \mathbb{Z}$. Previously, Ω_1^m only included alarm sets from a particular historical sequence X_m and thus, all the alarms in each observation were on the same timeline and maintained a chronological order based on the time of annunciations, where $m = 1, 2, \cdots, |\mathbb{Z}|$. However, the alarm set φ_1^{m,Ω_1^1} formulated based on Eqn 5.25 may include alarms from different historical sequences, and the corresponding times of annunciations may not match the timeline of the existing alarms. For efficient learning of RL agent, the time of annunciations of all alarms in each observation ought to be on the same timeline to maintain a chronological order. To resolve such an issue, the 95% confidence interval estimates around the mean of the time differences [46] between the annunciation of the nominated alarm e_k and the strongly associated alarm $e_j^{m,\Omega_1} \in \varphi_1^{m,\Omega_1}$ can be determined as follows:

$$\mathrm{CI}_{kj} = \begin{cases} \left[\max\left\{0, \bar{t_d} - z\frac{\sigma}{\sqrt{|t_d|}}\right\}, \left\{\bar{t_d} + z\frac{\sigma}{\sqrt{|t_d|}}\right\}\right], & \text{if sample size} \geq 30, \\ \left[\max\left\{0, \bar{t_d} - t\frac{\sigma}{\sqrt{|t_d|}}\right\}, \left\{\bar{t_d} + t\frac{\sigma}{\sqrt{|t_d|}}\right\}\right], & \text{if sample size} < 30. \end{cases} \tag{5.26}$$

where $\bar{t_d}$, σ and $|t_d|$ are the sample mean, sample standard deviation, and number of samples, respectively [46]. The time of annunciation of the nominated alarm e_k, matching the same timeline of the existing alarms in φ_1^{m,Ω_1}, is determined as $(t_j^{m,\Omega_1} + \bar{t_d} + x\frac{\sigma}{\sqrt{|t_d|}})$ where t_j^{m,Ω_1} is the time of annunciation of $e_j^{m,\Omega_1} \in \varphi_1^{m,\Omega_1}$; x indicates z or t distributions depending on the sample size as shown in Eqn 5.26.

Therefore, given φ_1^{m,Ω_1} and $\mathbb{X}_p$, all possible new observations can be generated following the criterion in Eqn 5.25. Such observations are generated by selecting different combinations of alarms from $\mathbb{X}_p$. Analogously, for each alarm set in Ω_1^m, a new set of observations will be initialized based on the following criterion:

$$\Omega_i^{m,1} = \Bigg\{ \varphi_j^{m,\Omega_i^1} | \exists \varphi_n^{m,\Omega_1} \in \Omega_1^m \ s.t.\ \varphi_j^{m,\Omega_i^1} = \varphi_n^{m,\Omega_1} \cup \{x_l\}_{l=r}^q \subseteq \mathbb{X}_p, \\ \varphi_j^{m,\Omega_i^1} \notin \Omega_1^m,\ \Phi_j^{m,\Omega_i^1} \notin \mathbb{Z} \} \Bigg\}_{j=1}^{|\Omega_i^{m,1}|} \tag{5.27}$$

where $i = 1, \cdots, |\Omega_1^m|$, $n \in [1, \cdots, |\Omega_1^m|]$, $1 \leq r \leq q \leq |\mathbb{X}_p|$, and φ_j^{m,Ω_i^1} represents the corresponding alarm set of the observation Φ_j^{m,Ω_i^1}. Therefore, following criteria 1 and 2 as outlined above, several new sets of observations can be generated from each historical sequence in $\mathbb{Z}$ to train the algorithm on all possible situations that may appear during an alarm flood. For the learning process, a training set $\mathbb{D}$ is formulated as $\mathbb{D} = \{\Omega_1^m, \{\Omega_i^{m,1}\}_{i=1}^{|\Omega_1^m|}\}_{m=1}^{|\mathbb{Z}|}$. Additionally, to minimize the learning time and improve the RL agent's early prediction performance, following criteria are considered when formulating the training set $\mathbb{D}$:

$$\forall i, j \in [1, 2, \cdots, |\mathbb{D}|],\ i \neq j \implies (\varphi_i \neq \varphi_j) \text{ and } (\Phi_i \neq \Phi_{j:|\Phi_i|}) \tag{5.28}$$

$$\forall i, j \in [1, 2, \cdots, |\mathbb{D}|],\ i \neq j \implies \varphi_i \nsubseteq \varphi_j \text{ where } \Phi_{i:\alpha} = \Phi_{j:\alpha} \tag{5.29}$$

In Eqn 5.28, $\Phi_{j:|\Phi_i|}$ indicates a subsequence of Φ_j, consists of the initial $|\Phi_i|$ alarms from Φ_j. Eqn 5.28 states that all observations corresponding to the alarm sets in $\mathbb{D}$ should be unique, and no observation should exist as a prefix of any of the remaining observations. Otherwise, training on both observations would be redundant, and the RL agent may require more time to make predictions if both observations are assigned different alarm flood ids. In Eqn 5.29, $\Phi_{i:\alpha}(\Phi_{j:\alpha})$ represents the subset containing the initial α alarms from $\Phi_i(\Phi_j)$. According to Eqn 5.29, no such alarm sets should exist in $\mathbb{D}$ such that the initial α alarms of the corresponding observations are common and none should be a subset of another. α is a user-defined threshold, and it is recommended to set it to 10 or lower. According to the benchmark [30], an alarm flood will have at least 10 alarms at the triggering instant, and thus, if such an alarm pattern repeats online, the RL agent will be able to predict at the earliest, i.e., at the triggering instant, if $\varphi_i \nsubseteq \varphi_j$ and $\alpha \leq 10$.

Each historical sequence in $\mathbb{Z}$ is consists of an alarm pattern and is labeled with an alarm flood ID. Also, the alarm sets in $\{\Omega_1^m, \Omega_i^{m,1}\}$ are either a subset or different versions of X_m by including new sets of strongly associated alarms where $m = 1, 2, \cdots, |\mathbb{Z}|$. For optimal learning of RL agent, the observations corresponding to these alarm sets will be associated with the alarm flood IDs of the most similar historical sequences that exist in $\mathbb{Z}$. During the training phase, the RL agent learns to recognize the alarm patterns in such observations and label them with the corresponding alarm flood IDs through repeated interactions. The rewards or penalties assigned to the interactions between the RL agent and these observations are based on the ground truth, and as the agent learns and improves its performance, it should be able to accurately map such observations to the appropriate alarm flood IDs. For the accuracy of early prediction actions, the assignment of alarm flood IDs to such observations should be done systematically. The observations corresponding to the alarm sets in $\{\Omega_1^m, \Omega_i^{m,1}\}$ are generated based on the alarm pattern in X_m, but are modified to include a broader range of strongly associated alarms and exclude the irrelevant ones. These modified observations intend to account for most of the deviations of X_m that may occur when the sequence is repeated in real-time. Due to modifications, these observations may resemble other sequences in $\mathbb{Z}$ more closely than X_m. To address this issue, the Modified Smith-Waterman algorithm can be exploited as follows:

$$I_i = \underset{I_i \in A_p}{\arg\max}\, \mathbb{S}(\Phi, X_{I_i}), \tag{5.30}$$

where $I_i \in \mathbb{N}^+$ and $\mathbb{S}$ represest the similarity index. Eqn 5.30 states that the alarm flood ID I_i will be assigned to the observation Φ due to the corresponding historical sequence X_{I_i} has the maximum similarity index [45] with Φ. Analogously, each observation corresponding to the alarm sets in $\{\Omega_1^m, \Omega_i^{m,1}\}_{i=1}^{|\Omega_1^m|}$ will be paired with the appropriate alarm flood ID for optimal learning in the training process. For the efficient learning of the RL agent, each alarm set in $\{\Omega_1^m, \Omega_i^{m,1}\}_{i=1}^{|\Omega_1^m|} \in \mathbb{D}$ is converted to an exponentially attenuated component (EAC) vector [4]. For instance, the alarm sets $\{\varphi_i^{m,\Omega_1}\}_{i=1}^{|\Omega_1^m|} \in \Omega_1^m$ are converted into corresponding EAC vectors as follows:

$$\chi_i^{m,\Omega_1} = \zeta_i^{m,\Omega_1} \circ exp(-\Gamma \tau_i^{m,\Omega_1}) \tag{5.31}$$

where ζ_i^{m,Ω_1} and τ_i^{m,Ω_1} represents the binary and relative time vectors of φ_i^{m,Ω_1}, $\circ$ denotes the Hadamard product, $exp(.)$ represents elementwise exponential function, and Γ denotes the attenuation coefficient; $m = 1, 2, \cdots, |\mathbb{Z}|$; $i = 1, 2, \cdots, |\Omega_1^m|$. Then, each EAC vector will be associated with the alarm flood

ID of the corresponding observation. For instance, the alarm sets in $\Omega_1^m \in \mathbb{D}$ are modified as follows:

$$\Omega_1^m = \{(\chi_i^{m,\Omega_1}, I_i^{m,\Omega_1}) | I_i^{m,\Omega_1} \in \mathbb{N}^+, i = 1, 2, \cdots, |\Omega_1^m|\} \tag{5.32}$$

where χ_i^{m,Ω_1} is the EAC vector of the alarm set φ_i^{m,Ω_1}; I_i^{m,Ω_1} is the associated alarm flood ID of the corresponding observation Φ_i^{m,Ω_1}. Analogously, all new alarm sets from each historical sequence will be modified according to Eqn 5.32 by converting each alarm set into the corresponding EAC vectors. Therefore, $(\chi_i^{m,\Omega_1}, I_i^{m,\Omega_1}) \in \mathbb{D}$ represents the corresponding observation Φ_i^{m,Ω_1} during the training of the RL agent.

The objective of this algorithm is to warn industrial operators about upcoming abnormalities by extracting similar scenarios from the historical database $\mathbb{Z}$. Sometimes, several historical sequences are closely similar to one other, and it may be beneficial for plant operators to take action if such sequences are grouped and presented as recommendations when the current situation matches any of them. In such cases, the modified Smith-Waterman algorithm can be further utilized as follows:

$$\begin{aligned} G_{I_m} = \{X_n | m \neq n, \mathbb{S}(X_m, X_n) \geq \rho_s, n = 1, 2, \cdots, |G_{I_m}|, \\ m = 1, 2, \cdots, |\mathbb{Z}|\} \end{aligned} \tag{5.33}$$

Eqn 5.33 states that to identify a group of closely similar historical sequences G_{I_m} for each historical sequence $X_m \in \mathbb{Z}$, similarity index between X_m and the other sequences in $\mathbb{Z}$ can be determined where $m = 1, 2, \cdots, |\mathbb{Z}|$. Then, based on a user-defined threshold ρ_s, closely similar historical sequences can be identified and grouped together to form G_{I_m}. Therefore, when an ongoing situation is mapped to the alarm flood ID I_m i.e., to its associated historical sequence X_m, the algorithm will also provide a group of closely similar scenarios G_{I_m} related to I_m as recommendation to the industrial operators. The detailed procedure for formulating $\mathbb{D}$ and identifying G_{I_m} is outlined in Algorithm 5.

Algorithm 5 Reconstruction of sequences and formulation of $\mathbb{D}$

```
1: Input Arguments: $\mathbb{Z}$
2: Output Argument: $\mathbb{D}$
3: Define $\rho_{conf}, \rho_{int}, \rho_d$
4: for $i \in [1, |\mathbb{Z}|]$ do
5:     $k = 1$
6:     if $\lambda(t^i) \geq \lambda_1$ then
7:         Formulate $\tilde{X}_i$ by Eqn 5.19
8:         for $j \in [|\tilde{X}_i|, |X_i|]$ do
9:             if $j == |\tilde{X}_i|$ then
10:                Obtain $\varphi_k^{i,\Omega_1}$ by Eqn 5.20
11:                Obtain $I_k^{i,\Omega_1}$ by Eqn 5.30
12:                Go to 16
13:            else
14:                if $x_j^i$ satisfy Eqn 5.21 then
15:                    k=k+1
16:                    Obtain $\varphi_k^{i,\Omega_1}$ by Eqn 5.21
17:                    Obtain $I_k^{i,\Omega_1}$ by Eqn 5.30
18:                    Calculate $\{D_r\}_{r=1}^{|\mathbb{Z}|}$ by Eqn 5.23 and obtain
19: $S_k^{i,\Omega_1}$ based on $\rho_d$
20:                    Obtain $\mathbb{X}_p$ by Eqn 5.24
21:                    $l = 1$
22:                    for $q \in [1, |\mathbb{X}_p|]$ do
23:                        if $x_q$ satisfy Eqn 5.25 then
24:                            Obtain $\varphi_l^{i,\Omega_1^k}$ by Eqn 5.25
25:                            Obtain $I_l^{i,\Omega_1^k}$ by Eqn 5.30
26:                            $l = l + 1$
27:                        end if
28:                    end for
29:                    Obtain $\Omega_k^{i,1}$ by Eqn 5.27
30:                end if
31:            end if
32:        end for
33:    end if
34:    Obtain $\Omega_1^i$ by Eqn 5.22
35:    Obtain $G_{I_i}$ by Eqn 5.33
36: end for
37: Formualte $\mathbb{D} = \{\Omega_1^m, \{\Omega_i^{m,1}\}_{i=1}^{|\Omega_1^m|}\}_{m=1}^{|\mathbb{Z}|}$ based on Eqn 5.28 and Eqn 5.29
```

5.3.3 Double Deep-Q-Network Algorithm

During an alarm flood, alarms are triggered one by one and sequentially added to the online sequence. Therefore, only a partial sequence is available at any instant during an alarm flood. With a new incoming alarm, the objective is to predict the incoming sequence accurately or wait until more information becomes available. Allowing the online sequence to grow sequentially may lead to major disasters such as process shutdowns, etc. Therefore, the problem of early prediction can be framed as sequentially nominating the best action based on the information available at any instant of an alarm flood.

The core of our framework is an RL agent that is trained with all possible observations in $\mathbb{D}$ to perform prediction at the earliest order of the online sequence. To decorrelate the observations in $\mathbb{D}$ and improve the stability of the learning algorithm, a replay buffer is used and initialized with a collection of experience tuples. An experience tuple, denoted as $< \Phi, a, r, \Phi_n >$, is used to modify the agent's policy that guides its actions in various observations. The current observation of the state is Φ which is represented as (χ, I) during the training process, the action taken by the agent at Φ is represented by a, the reward received for the action is indicated by r, and Φ_n, i.e., (χ_n, I_n) represents the consequent observation, which is determined by the transition model T. The reward function $R(s, a)$ is typically designed to ensure that it accurately reflects the desired objectives of the RL agent. Thus, to align with the goal of the RL agent, the reward function is defined as follows:

$$R((\Phi, I_i), a) = \begin{cases} 1 & \text{if } a = I_i \in A_p \\ -\Gamma & \text{if } a \neq I_i \in A_p \\ r_d & \text{if } a = a_d \end{cases} \tag{5.34}$$

$$\text{where } r_d = \begin{cases} 1 - (|\Phi|/\delta) & \text{if } |\Phi| \leq \delta \\ -1 & \text{if} |\Phi| > \delta \end{cases} \tag{5.35}$$

In Eqn 5.34, $\Gamma = \max\{|X_i| : X_i \in \mathbb{Z}, i = 1, 2, \cdots, |\mathbb{Z}|\}$, and δ is a hyperparameter that needs to be adjusted together with other hyperparameters for optimal performance. The process of identifying the optimal settings for such hyperparameters is briefly discussed in section 5.4. The RL agent is rewarded for an action $a \in A_p$ if it correctly predicts the alarm flood ID for a given observation in the training set $\mathbb{D}$. However, a severe penalty is imposed if the prediction is incorrect. To avoid such high penalty, the RL agent is encouraged with a much smaller penalty to defer its action of prediction if it is uncertain, as shown in Eqn 5.35. The penalty for deferral, however, increases with the addition of more alarms to the online sequence, which motivates early prediction. With this strategy, the RL agent can avoid incorrect prediction actions and make a balance between action of deferral and action of prediction to find the optimal policy resulting in the highest cumulative reward.

In this work, transition model T is deterministic and denoted as follows:

$$T((\Phi, I_i), a) = \begin{cases} (\Phi, I_i) & \text{if } (a \in A_p) \vee (|\Phi| = |X|) \vee (\lambda_t < \lambda_2) \\ (\Phi_n, I_i) & \text{if } a = a_d \text{ and } |\Phi| \neq |X| \end{cases} \tag{5.36}$$

Eqn 5.36 states that the RL agent remains in its current state if any of the following conditions are met:

1. $a \in A_p$.
2. The current observation Φ is the full observation of the state, i.e., the complete historical sequence X.
3. During an alarm flood, the current alarm rate λ_t falls below λ_2.

However, if the RL agent decides to defer the action of prediction, it will transition to the next observation Φ_n. For each observation $\Phi_i \in \mathbb{D}$, a set of experience tuples is formulated as $\Phi_i^e = \{< \Phi_i, a_j, R(\Phi_i, a_j), T(\Phi_i, a_j) > |a_j \in A, j = 1, \cdots, |A|\}$ where $A = \{A_p, a_d\}$, Φ_i is represented as (χ_i, I_i) and $i = 1, 2, \cdots, |\mathbb{D}|$. Φ_i^e is formulated by combining all possible actions $a \in A$ with their corresponding rewards and transitions to the next states. Each experience tuple captures the essential details of a single interaction between the agent and a particular observation. The replay buffer stores all such interactions to allow the agent to learn from these past experiences.

To learn from a diverse set of experiences, the minibatch sampling technique is exploited to update the agent's learning algorithm based on a small, randomly chosen subset of experiences from the replay buffer. The RL agent uses such experiences to train a deep neural network, which serves as the function approximator for the expected future rewards for each action in each state, i.e., the Q-values.

The RL agent updates its policy by using stochastic gradient descent (SGD), which works by iteratively updating the weights θ_p in the direction that reduces the loss. To ensure optimal learning without any biasedness, the DDQN algorithm is exploited in this work. Initially, the target Q-values are determined as follows:

$$Q_T = r + \gamma Q(s', \arg\max_{a_{t+1}} Q(s', a_{t+1}, \theta_p), \theta_t) \tag{5.37}$$

where the Q-value from the target network corresponding to the action nominated by the prediction network is used to compute Q_T. Then, the loss is computed by comparing Q_T with the current Q-values from the prediction network with weights θ_p, which is used to update the weights of the prediction network θ_p using SGD. After certain iterations, the weights of the prediction network θ_p are transferred to the target network to stabilize the learning process and improve convergence.

The replay buffer stores all possible experiences of interactions between the RL agent and the environment, which is denoted as

$$\beta = \bigcup_{m=1}^{|\mathbb{Z}|} \{E_m\} \tag{5.38}$$

where $E_m = \{\Phi_i^{e,m} | i = 1, 2, \cdots, |E_m|\}$, which indicates the set of experiences generated from each historical sequence in $\mathbb{Z}$. E_m is composed of experiences with prediction and deferral actions, denoted as E_p^m and E_d^m where $m = 1, 2, \cdots, |\mathbb{Z}|$. E_p^m is further divided into two categories: experiences with correct and incorrect prediction actions, and denoted as $E_p^{r,m}$ and $E_p^{w,m}$. Thus, E_m can be expressed as $E_m = E_p^{r,m} \cup E_p^{w,m} \cup E_d^m$. All these past experiences are based on observations from historical sequences with different alarm flood IDs, and the Minibatch sampling technique decorrelates these experiences from different categories. However, uniform sampling can overemphasize a certain category of experiences, which may cause biasedness in the learning algorithm. Also, if experiences with action of deferrals E_d^m are oversampled, the RL agent will more likely to defer the prediction of the incoming alarm flood sequence.

Additionally, the performance may degrade if a particular historical sequence is emphasized and sampled for experiences. To overcome this issue, each mini-batch should contain an equal number of experiences from each historical sequence E_m where $m = 1, 2, \cdots, |\mathbb{Z}|$. Furthermore, when sampling experiences from each historical sequence, a ratio of $\rho > 1$ will be maintained while taking samples from E_p^m and E_d^m, respectively. This allocation gives greater weight to the experiences from E_p^m as compared to E_d^m. This sampling strategy ensures diverse learning and emphasizes early prediction of incoming sequences. The details of the training procedure are outlined in Algorithm 6.

The DDQN algorithm involves tuning several hyperparameters, including the learning rate ς, discount factor γ, exploration rate ϵ, number of episodes $\mathbb{E}$, number of iterations n etc., to optimize the convergence of $L(\theta)$ and improve its accuracy. For this particular early prediction problem, initially a range of values is identified for each hyperparameter through a trial and error process. Subsequently, all possible combinations of these values are evaluated. The best performing set of values is identified that provides the most accurate approximation of the optimal action Q values with optimal parameters θ_t. Therefore, during an alarm flood, the DDQN algorithm uses the learned parameters θ_t from the training phase to estimate the Q-values against the incrementing online alarm flood sequence. The action corresponding to the highest Q-values will be nominated, determining whether to make a prediction or wait for more information. If the DDQN algorithm suggests a prediction action ($a \in A_p$), it will provide the alarm flood ID and the group of scenarios that are most similar to a historical sequence from $\mathbb{Z}$ and resemble the current online sequence the closest. Conversely, the algorithm will wait for the online sequence to include more incoming alarms until it can predict accurately. The steps for predicting the incoming alarm flood sequence are outlined in Algorithm 7.

Algorithm 6 Training with DDQN algorithm

```
1: Input Arguments: $\mathbb{D}$
2: Output Argument: Deep neural network with optimal parameters $\theta_t$
3: for $i \in [1, |\mathbb{D}|]$ do
4:     for $j \in [1, |\Phi_i|]$ do
5:         Formulate $Q = \Phi_{i,j}$
6:         Convert $Q$ to EAC vector $\chi_Q$ by Eqn 5.31
7:         for $a \in [1, |\mathbb{A}|]$ do
8:             Obtain reward $R((\chi_Q, I_i), a)$ by Eqn 5.34
9:             Obtain next observation $T((\chi_Q, I_i), a)$ by Eqn 5.36
10:            Store $< (\chi_Q, a, R((\chi_Q, I_i), a)), T((\chi_Q, I_i), a)) >$
11: in $\beta$
12:         end for
13:     end for
14: end for
15: Define $\gamma$, $\epsilon$, $\mathbb{E}$, $\mu$ and $n$
16: Initialize Target and Prediction network with random weights $\theta_t$ and $\theta_p$,
    and set $\theta_t = \theta_p$
17: for $k \in [1, \mathbb{E}]$ do
18:     Sample a mini-batch of past experiences with a ratio of $\rho$
19:     Calculate $Q_T$ by Eqn 5.37
20:     Calculate $L(\theta)$ by Eqn 5.11 and update $\theta_p$ using SGD
21:     if $k == n$ then
22:         Set $\theta_t = \theta_p$
23:     end if
24: end for
```

5.4 Case Study

An industrial case study is presented in this section to illustrate the proposed method. A historical alarm & log over the time period from November 1, 2019, to April 30, 2020 was obtained from a coking plant[45] of an oil refinery. Total 103 alarm floods were extracted after removing the chattering alarms using off-delay timers. The average length of alarm floods was 29.75. The alarm floods with the highest and least number of alarms were 609 and 10, respectively.

From each historical alarm flood in $\mathbb{Z}$, all possible observations were generated according to criteria 1 and 2 as outlined in Section 5.3.2. Table 5.1

Algorithm 7 Online prediction of incoming alarm flood sequence

```
1: Input Arguments: $Y_{:t}$, Deep neural network with optimal parameters $\theta_t$
2: Output Argument: $I_i$, $G_{I_i}$
3: Determine $\lambda_t$ after eliminating chattering alarms
4: if $\lambda_t \geq \lambda_1$ then
5:     Obtain $Y_{:t}$ by Eqn 5.17
6:     Convert $Y_{:t}$ to EAC vector $\chi_y$ by Eqn 5.31
7:     if $a \in A_p$ then
8:         Obtain $max(Q(\chi_y, a, R(\chi_y, a), T(\chi_y, a)) = I_i \in A_p$
9:         Identify $G_{I_i}$
10:    else
11:        if $y_{|\tilde{Y}_{:t}|}$ satisfy Eqn 5.18 then
12:            Obtain $\tilde{Y}$ by Eqn 5.18
13:            Go to 6
14:        end if
15:    end if
16: end if
```

illustrates the observations generated from the historical sequence # 34 in $\mathbb{Z}$. Table 5.1-A shows the historical sequence # 34 where the triggering instant was identified at the annunciation of Tag_185. Tag_158 and Tag_694 were identified as irrelevant alarms according to the criteria shown in Table 5.2. The user-defined thresholds ρ_{conf} and ρ_{int} were set to 0.5 and 1, respectively. In Table 5.2, evaluation of some of the alarms in historical sequence # 34 is shown where Tag_158 could not satisfy ρ_{conf} and thus identified as irrelevant alarms. After excluding Tag_158, observation # 1 was generated by including all the alarms that were annunciated up until the triggering instant. Instead of the time of annunciations, observation # 1 shows the relative time differences between each alarm and the first alarm. Observation # 2 was formulated by including the next relevant alarm Tag_184. As Tag_694 was also identified as an irrelevant alarm, no more observations could be formulated according to the criteria 1.

As outlined in criteria 2 of Section 5.3.2, new observations were formulated by predicting upcoming alarms based on the existing alarms within each observation formulated according to criteria 1. Table 5.3 shows some of the

Table 5.1: Observations from historical sequence using criteria 1: (A) Historical sequence # 34; (B) Observation # 1; (C) Observation # 2.

(A)

Historical Sequence # 34	
Alarm Tag	**Time (sec)**
TAG_836	3:21:21 AM
TAG_166	3:21:28 AM
TAG_174	3:21:28 AM
TAG_834	3:21:28 AM
TAG_828	3:21:28 AM
TAG_158	3:21:28 AM
TAG_178	3:21:28 AM
TAG_177	3:21:28 AM
TAG_176	3:21:28 AM
TAG_185	3:21:28 AM
TAG_184	3:22:45 AM
TAG_694	3:23:13 AM

(B)

Observation # 1	
Alarm Tag	**Time (sec)**
TAG_836	0.0
TAG_166	7.0
TAG_174	7.0
TAG_834	7.0
TAG_828	7.0
TAG_178	7.0
TAG_177	7.0
TAG_176	7.0
TAG_185	7.0

(C)

Observation # 2	
Alarm Tag	**Time (sec)**
TAG_836	0.0
TAG_166	7.0
TAG_174	7.0
TAG_834	7.0
TAG_828	7.0
TAG_178	7.0
TAG_177	7.0
TAG_176	7.0
TAG_185	7.0
TAG_184	8.2

Table 5.2: Evaluation criteria for eleminating the irrelevant alarms.

Alarm Tag	Associated Alarm	Confidence	Interest
TAG_836	TAG_176	0.87	1.16
TAG_174	TAG_178	0.67	1.08
TAG_158	TAG_834	0.47	1.21
TAG_178	TAG_836	0.78	1.1
⋮	⋮	⋮	⋮

alarms nominated as potential upcoming alarms based on the existing alarms in observation # 1 as shown in Table 5.1-B. The predicted alarms show high confidence and an interest value greater than 1, indicating a strong association between the predicted alarms and the existing alarm pattern in observation # 1. These predicted alarms and the existing alarms in observation # 1 were then used to form a new observation, as shown in Table 5.4. The time information of Tag_184, Tag_183 and Tag_188 in observation # 3 are shown as $[0, 137]$, $[0, 69]$ and $[23, 90]$, respectively. These values correspond to the 95% confidence interval estimates that specify the time interval between the annunciation of Tag_185 and each predicted alarm in observation # 3. As stated in Section 5.3.2, only the highest endpoints of the confidence intervals for each nominated alarm were used to convert the alarm set corresponding to each observation into feature vectors. This process of generating such new observations from historical sequence # 34 and then, converting them into the

Table 5.3: Prediction of upcoming alarms for observation # 1.

Predicted Alarms	Associated Alarms	Confidence	Interest
TAG_184	TAG_176	0.8	1.51
TAG_183	TAG_177	0.67	1.41
TAG_188	TAG_185	0.6	1.6
⋮	⋮	⋮	⋮

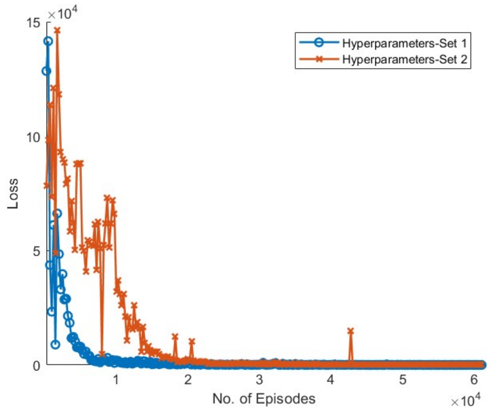


Figure 5.1: Loss function with different sets of hyperparameters

corresponding feature vectors was applied to all the historical sequences in $\mathbb{Z}$. Total 115 observations were generated after complying with the requirements specified in Eqn 5.28 and Eqn 5.29, and converted into feature vectors that are used to formulate the training set $\mathbb{D}$.

Table 5.4: Observations from historical sequence: (A) Observation # 1 formulated using criteria 1; (B) Observation # 3 formulated using criteria 2.

(A)

Observation # 1	
Alarm Tag	**Time (sec)**
TAG_836	0.0
TAG_166	7.0
TAG_174	7.0
TAG_834	7.0
TAG_828	7.0
TAG_178	7.0
TAG_177	7.0
TAG_176	7.0
TAG_185	7.0

(B)

Observation # 3	
Alarm Tag	**Time (sec)**
TAG_836	0.0
TAG_166	7.0
TAG_174	7.0
TAG_834	7.0
TAG_828	7.0
TAG_178	7.0
TAG_177	7.0
TAG_176	7.0
TAG_185	7.0
TAG_184	[0, 137]
TAG_183	[0, 69]
TAG_188	[23, 90]

The DDQN algorithm was trained until the loss function stopped decreas-

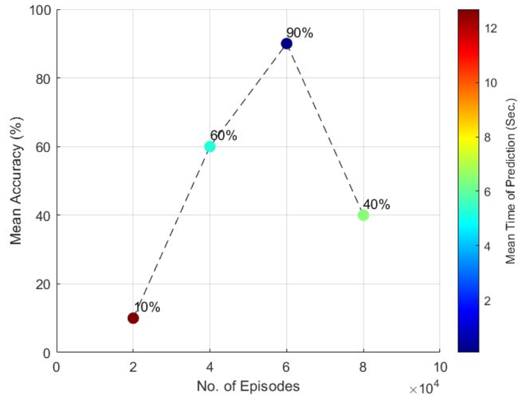


Figure 5.2: Mean accuracy and mean time of prediction of the validation set

ing. Multiple sets of hyperparameters were employed, and the loss function was monitored. Fig 5.1 shows the loss function with two sets of hyperparameters that were selected based on their relative stability in minimizing the loss function during the training. However, the second set of hyperparameters exhibited high fluctuations in the loss function, indicating that the training was less stable with this set of hyperparameters. Such fluctuations may be caused by various factors, such as inappropriate setting for the learning rate. On the other hand, the loss function with the first set of hyperparameters showed a gradual decrease with low fluctuations, suggesting that this set of hyperparameters was better suited for the given problem. The set of hyperparameters includes the learning rate ς, discount factor γ, exploration rate ϵ, number of episodes $\mathbb{E}$ and δ. The values assigned to these hyperparameters were 0.000025, 0.99, 0.1, 60000 and 20, respectively. Additionally, Fig 5.2 illustrates the training accuracy and mean time of prediction on a validation set of 10 observations at various numbers of episodes while keeping the remaining hyperparameters in set 1 constant. Fig 5.2 shows that the DDQN algorithm converged to optimal policy when the number of episodes was set to 60000, with the highest training accuracy and lowest mean prediction time. However, as the number of episodes increased beyond 60000, the training accuracy decreased and the mean time of prediction increased, which may be attributed to overfitting of the algorithm to the training data. The mean time of prediction increases due to delaying the action of prediction for certain observations in the validation set.

A historical sequence was nominated as an online sequence where alarms were raised one by one and added to the online sequence during simulation. Each time the online sequence was modified by including new alarms, it was presented as input to the DDQN network to evaluate the prediction accuracy in real-time. Table 5.5 shows an alarm flood sequence at the triggering instant where Tag_836 was identified as irrelevant alarms according to the criteria shown in Table 5.2. After excluding Tag_836, the online sequence with only relevant alarms is shown in Table 5.5-B which was the input to the DDQN network after converting to the feature vector according to Eqn 5.31. As shown in Table 5.5-B, with only relevant alarms at triggering instant, the algorithm prefered action of deferral over action of prediction as it required more information. With Tag_880 added to the online sequence as shown in Table 5.5-C, the algorithm nominated historical sequence # 31 as the most similar to the online sequence. Table 5.6 illustrates the accuracy of the early prediction action. Table 5.6-A shows the complete online sequence, and Table 5.6-B shows the historical sequence # 31 with highlighted alarms that also exist in the online sequence. As shown in Table 5.6, the proposed method also provided a group of similar historical sequences to the nominated sequence # 31 as recommendation to the operators. Table 5.6-C and Table 5.6-D shows some of the closely similar historical sequences with similarity scores 0.9 and 0.81 respectively. With Tag_836 being identified as irrelevant alarm, it is clearly observed that an alarm pattern exists both in the online and historical sequence # 31. Also, most of the alarms in the historical sequences # 31, #32, and #51 are also present in the complete online sequence, which validates the early prediction action.

To verify the accuracy of the early prediction action, similarity analysis using modified SWA [45] was conducted between the complete online sequence and the historical sequences in $\mathbb{Z}$. The outcome of the analysis is shown

Table 5.5: Early prediction of an online alarm flood sequence: (A) At triggering instant; (B) Online alarm flood sequence with only relevant alarms; (C) Online alarm flood sequence after including TAG_880 following triggering instant.

(A)

Online Sequence (at triggering instant)	
Alarm Tag	**Time (sec)**
TAG_880	02:51:39 AM
TAG_836	03:00:22 AM
TAG_417	03:01:17 AM
TAG_416	03:01:17 AM
TAG_415	03:01:17 AM
TAG_414	03:01:17 AM
TAG_413	03:01:17 AM
TAG_412	03:01:17 AM
TAG_410	03:01:17 AM
TAG_409	03:01:17 AM
TAG_407	03:01:17 AM

(B)

Observation # 1 (online Sequence)	
$Y \rightarrow a_d$	
Alarm Tag	**Time (sec)**
Tag_880	02:51:39 AM
Tag_417	03:01:17 AM
Tag_416	03:01:17 AM
Tag_415	03:01:17 AM
Tag_414	03:01:17 AM
Tag_413	03:01:17 AM
Tag_412	03:01:17 AM
Tag_410	03:01:17 AM
Tag_409	03:01:17 AM
Tag_407	03:01:17 AM

(C)

Observation # 2 (Online Sequence)	
$Y \rightarrow a \in A_p = 31$	
Alarm Tag	**Time (sec)**
TAG_880	02:51:39 AM
TAG_417	03:01:17 AM
TAG_416	03:01:17 AM
TAG_415	03:01:17 AM
TAG_414	03:01:17 AM
TAG_413	03:01:17 AM
TAG_412	03:01:17 AM
TAG_410	03:01:17 AM
TAG_409	03:01:17 AM
TAG_407	03:01:17 AM
TAG_880	03:05:26 AM

in Fig 5.3 using a color map including only the historical sequences from $\mathbb{Z}$ that exhibited significant similarity with the online sequence. During the simulation, historical sequence # 26 was chosen as the online sequence, and the color map shows the most similar sequences to the online sequence, which are marked as C_2 in Fig 5.3, with historical sequence # 31 being the most similar. Furthermore, the historical sequences which demonstrated significant similarity to historical sequence # 31, marked as C_1 in Fig 5.3, were also identified and can be provided to the plant operators as a recommendation. Additionally, the average running time of the proposed method was compared with the methods in [51] and [15], and showed in Table 5.7. As indicated by the table, the proposed method exhibits significantly lower running time compared to the methods outlined in [51] and [15].

5.5 Summary

In this chapter, we addressed the problem of early prediction of industrial alarm floods and presented a comprehensive solution to overcome the associated challenges. Initially, we provided necessary background on industrial alarm floods and reinforcement learning, and introduced the problem.

Table 5.6: Validation of early prediction action: (A) Online sequence; (B) Historical sequence # 68; (C) Historical sequence # 71 with similarity score 0.9; (D) Historical sequence # 76 with similarity score 0.81.

(A)

Online Sequence	
Alarm Tag	**Time (sec)**
TAG_880	02:51:39 AM
TAG_836	03:00:22 AM
TAG_417	03:01:17 AM
TAG_416	03:01:17 AM
TAG_415	03:01:17 AM
TAG_414	03:01:17 AM
TAG_413	03:01:17 AM
TAG_412	03:01:17 AM
TAG_410	03:01:17 AM
TAG_409	03:01:17 AM
TAG_407	03:01:17 AM
TAG_880	03:05:26 AM
TAG_819	03:10:19 AM
TAG_839	03:11:22 AM
TAG_417	03:18:38 AM
TAG_416	03:18:38 AM
TAG_415	03:18:38 AM
TAG_414	03:18:38 AM
TAG_413	03:18:38 AM
TAG_412	03:18:38 AM
TAG_410	03:18:38 AM
TAG_409	03:18:38 AM
TAG_407	03:18:38 AM
TAG_1205	03:28:03 AM

(B)

Historical Sequence # 31	
Alarm Tag	**Time (sec)**
TAG_880	03:49:19 AM
TAG_417	03:59:00 AM
TAG_416	03:59:00 AM
TAG_415	03:59:00 AM
TAG_414	03:59:00 AM
TAG_413	03:59:00 AM
TAG_412	03:59:00 AM
TAG_410	03:59:00 AM
TAG_409	03:59:00 AM
TAG_407	03:59:00 AM
TAG_880	04:02:18 AM
TAG_1113	04:02:37 AM
TAG_1375	04:05:16 AM

(C)

Historical Sequence # 32	
Alarm Tag	**Time (sec)**
Similarity Score : 0.9	
TAG_788	05:41:55 AM
TAG_417	05:42:26 AM
TAG_416	05:42:26 AM
TAG_415	05:42:26 AM
TAG_414	05:42:26 AM
TAG_413	05:42:26 AM
TAG_412	05:42:26 AM
TAG_410	05:42:26 AM
TAG_409	05:42:26 AM
TAG_407	05:42:26 AM

(D)

Historical Sequence # 51	
Alarm Tag	**Time (sec)**
Similarity Score : 0.81	
TAG_417	12:34:40 PM
TAG_416	12:34:40 PM
TAG_415	12:34:40 PM
TAG_414	12:34:40 PM
TAG_413	12:34:40 PM
TAG_412	12:34:40 PM
TAG_410	12:34:40 PM
TAG_409	12:34:40 PM
TAG_407	12:34:40 PM
TAG_1337	12:36:28 PM
TAG_819	12:36:56 PM

Table 5.7: Running times of different methods

Method	Proposed method	Method in [51]	Method in [15]
Avg. run time (ms)	**0.076**	0.118	0.141

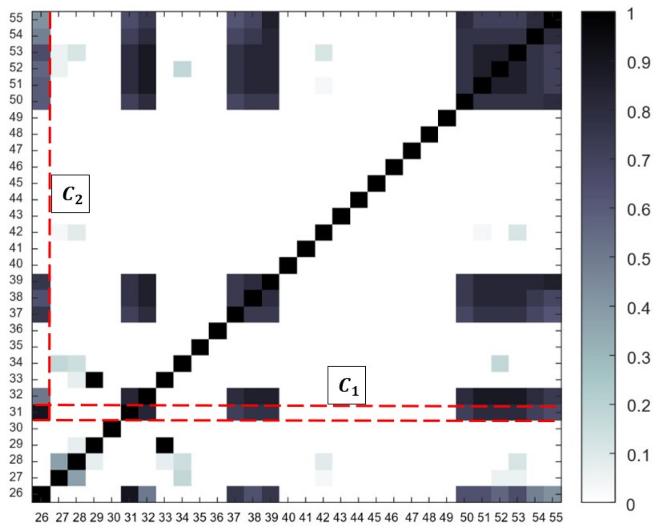


Figure 5.3: Validation of early prediction action by demonstrating the similarity between the complete online and the most similar historical sequences in $\mathbb{Z}$ based on the similarity color map.

Then, we proposed strategies based on an association rule mining approach to eliminate irrelevant alarms from online sequences and reconstruct sequences from historical sequences. Next, a comprehensive explanation of the DDQN algorithm is presented. Finally, an industrial case study is presented to illustrate the proposed method. Overall, the proposed method provided a comprehensive approach to early prediction of industrial alarm floods, combining association rule mining and reinforcement learning to overcome the challenges associated with this problem. The presented case study effectively showcased the capabilities and benefits of our method in real-world industrial scenarios.

Chapter 6

Conclusions and Future Work

This chapter concludes the thesis by providing concluding remarks and suggesting potential research directions for future work.

6.1 Conclusions

This thesis focuses on developing data-driven methods aimed at effectively managing alarm flood situations and minimizing their impacts. The main conclusions drawn from the studies in this thesis are as follows:

1. We proposed a real-time pattern matching and dynamic ranking approach to conduct similarity analysis in an online alarm flood situation and to export a ranking list of historical alarm floods ranked by the similarity scores. The proposed method can provide real-time decision support to plant operators by finding and ranking closely similar scenarios for the online alarm flood from the database of historical alarm flood sequences. Unit-based screening and set-based pre-matching, along with the online set-based indexing and extension strategy, greatly reduce the computational burden by excluding unnecessary computation. Incremental sequence alignment was used to measure sequence-based similarity. Ranking of the most similar alarm flood sequences provides the opportunity to explore all the similar sequences and acquire valuable information on the incoming alarm flood sequence.

2. We introduced a new approach for predicting alarm events in real time during alarm flood situations. The proposed method, based on association rule mining, aimed to assist industrial operators in promptly responding to alarm floods by providing timely predictions of upcoming alarms. To achieve this, the Compact Prediction Tree (CPT) model was modified with additional features and constructed using historical alarm flood sequences. An online strategy was proposed, utilizing co-occurrence frequency and confidence, to eliminate irrelevant alarms and maintain prediction accuracy. Furthermore, a real-time algorithm was developed that effectively predicts upcoming alarms by leveraging true alarm patterns and updates predictions as the online alarm flood progresses. The method also provides the annunciation times for subsequent alarms, thereby supporting industrial operators in effectively managing alarm flood situations.

3. We proposed a reinforcement learning approach aimed at predicting alarm floods in real time and providing assistance to address the situation at the earliest. Various association rule metrics were employed to assess and identify irrelevant alarms within the online sequence. A strategy was introduced to reconstruct sequences by exploiting the alarm relations in the existing set of historical sequences and to modify the training set for effective learning. By employing this strategy, the proposed method can address potential online scenarios that are not present in the set of historical sequences, thus enabling it to provide recommendations when similar situations reoccur in real time. The proposed method adopted the DDQN algorithm with a modified learning process, leading to improved accuracy and promptness in predicting industrial alarm floods.

The effectiveness and practicality of the proposed methods are validated through case studies conducted on alarm data obtained from complex industrial facilities.

6.2 Future Work

The future research directions on effective management of alarm flood situations are summarized as follows:

1. Initially, we introduced a real-time pattern matching and dynamic ranking approach to perform similarity analysis during online alarm floods. An interesting future aspect of this method is identifying novel situations, which relies on determining a dynamic similarity threshold. This threshold can vary due to several influencing factors, such as the root cause type, the presence of nuisance alarms, and other related considerations. Using a constant threshold may lead to misclassifications and mislead industrial operators. Thus, investigating and dedicating further efforts to establish a dynamic threshold based on these factors is crucial. Furthermore, to improve accuracy and relevance, critical contextual information like operational conditions can be integrated into the dynamic threshold determination process. This ensures that the threshold aligns with the current situation and becomes more adaptable. Additionally, refining the proposed method by incorporating feedback from operators will be an interesting addition to this method. Operators' insights can contribute to a more accurate determination of dynamic threshold, making the approach more practical and effective.

2. Secondly, we proposed a new approach for predicting alarm events in real time during alarm flood situations. One of the limitations of the proposed method is the inadequacy of similar situations as the ongoing alarm flood in the training set. To address this, a potential future direction is to explore the application of transfer learning and pre-trained models with extensive datasets. By leveraging knowledge from related domains or previous datasets, the model's ability to handle scenarios with limited similar training instances can be improved. Moreover, the training of CPT is currently performed offline, but a new feature can

be included that enables real-time training only when a new example is detected. In such scenarios, the current features of CPT can be incrementally updated. Such an issue is not investigated in this thesis and is considered as a promising future work.

3. Finally, we proposed a reinforcement learning approach to address some specific issues related to predicting alarm floods in real time and providing assistance to handle the situation at the earliest. One interesting future research involves identifying the root cause associated with each alarm flood in real time and providing immediate recommendations for corrective actions. While the proposed method effectively suggests similar scenarios to the ongoing alarm flood with optimal accuracy, selecting appropriate corrective actions based on such recommendations may require prior experience or context-specific training, which can be time-consuming and resource-intensive. In addition, the recommendations for corrective actions should be explainable and interpretable to operators. This will help operators understand the underlying reasoning behind the suggested corrective actions, ultimately building trust in the capabilities of the system.

Bibliography

[1] N. A. Adnan, Y. Cheng, I. Izadi, and T. Chen. Study of generalized delay-timers in alarm configuration. *Journal of Process Control*, 23(3):382–395, 2013.

[2] M. S. Afzal, T. Chen, A. Bandehkhoda, and I. Izadi. Analysis and design of time-deadbands for univariate alarm systems. *Control Engineering Practice*, 71:96–107, 2018.

[3] H. S. Alinezhad, M. H. Roohi, and T. Chen. A review of alarm root cause analysis in process industries: Common methods, recent research status and challenges. *Chemical Engineering Research and Design*, 2022.

[4] H. S. Alinezhad, J. Shang, and T. Chen. Early classification of industrial alarm floods based on semisupervised learning. *IEEE Transactions on Industrial Informatics*, 18(3):1845–1853, 2021.

[5] H. S. Alinezhad, J. Shang, and T. Chen. Open set online classification of industrial alarm floods with alarm ranking. *IEEE Transactions on Instrumentation and Measurement*, 72:1–11, 2022.

[6] M. Bransby and J. Jenkinson. The management of alarm systems: a review of best practice in the procurement, design and management of alarm systems in the chemical and power industries. *Technical Report CRR 166, Health and Safety Executive*, 1998.

[7] S. Cai, L. Zhang, A. Palazoglu, and J. Hu. Clustering analysis of process

alarms using word embedding. *Journal of Process Control*, 83:11–19, 2019.

[8] S. Charbonnier, N. Bouchair, and P. Gayet. A weighted dissimilarity index to isolate faults during alarm floods. *Control Engineering Practice*, 45:110–122, 2015.

[9] S. Charbonnier, N. Bouchair, and P. Gayet. Fault template extraction to assist operators during industrial alarm floods. *Engineering Applications of Artificial Intelligence*, 50:32–44, 2016.

[10] Y. Cheng, I. Izadi, and T. Chen. Pattern matching of alarm flood sequences by a modified smith-waterman algorithm. *Chemical engineering research and design*, 91(6):1085–1094, 2013.

[11] G. Dorgo and J. Abonyi. Sequence mining based alarm suppression. *IEEE Access*, 6:15365–15379, 2018.

[12] G. Dorgo, A. Palazoglu, and J. Abonyi. Decision trees for informative process alarm definition and alarm-based fault classification. *Process Safety and Environmental Protection*, 149:312–324, 2021.

[13] G. Dorgo, P. Pigler, and J. Abonyi. Understanding the importance of process alarms based on the analysis of deep recurrent neural networks trained for fault isolation. *Journal of Chemometrics*, vol. 32, no. 4, art. no. e3006, 2018.

[14] EEMUA (Engineering Equipment and Materials Users' Association), London, UK. *Alarm Systems: A Guide to Design, Management and Procurement*, 2014.

[15] M. Fullen, P. Schüller, and O. Niggemann. Semi-supervised case-based reasoning approach to alarm flood analysis. In *Machine Learning for Cyber Physical Systems*, pages 53–61. Springer, 2020.

[16] H. Gharahbagheri, S. Imtiaz, and F. Khan. Root cause diagnosis of process fault using KPCA and Bayesian network. *Industrial & Engineering Chemistry Research*, 56(8):2054–2070, 2017.

[17] T. Gueniche, P. Fournier-Viger, R. Raman, and V. S. Tseng. Cpt+: Decreasing the time/space complexity of the compact prediction tree. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 625–636. Springer, 2015.

[18] T. Gueniche, P. Fournier-Viger, and V. S. Tseng. Compact prediction tree: A lossless model for accurate sequence prediction. In *International Conference on Advanced Data Mining and Applications*, pages 177–188. Springer, 2013.

[19] C. Guo, W. Hu, S. Lai, F. Yang, and T. Chen. An accelerated alignment method for analyzing time sequences of industrial alarm floods. *Journal of Process Control*, 57:102–115, 2017.

[20] P. Gyasi and J. Wang. Design of serial alarm systems based on deadbands and delay timers for removing false alarms. *Process Safety and Environmental Protection*, 162:1033–1041, 2022.

[21] P. Gyasi and J. Wang. Design of alarm thresholds and delay timers for non-IID process variables based on alarm durations. *Process Safety and Environmental Protection*, 170:1173–1187, 2023.

[22] B. R. Hollifield and E. Habibi. *Alarm Management: A Comprehensive Guide: Practical and Proven Methods to Optimize the Performance of Alarm Management Systems.* ISA, 2011.

[23] J. Hong, R. Tamakloe, and D. Park. Application of association rules mining algorithm for hazardous materials transportation crashes on expressway. *Accident Analysis & Prevention*, vol. 142, art. no. 105497, 2020.

[24] K. Hornik, B. Grün, and M. Hahsler. arules-a computational environment for mining association rules and frequent item sets. *Journal of statistical software*, 14(15):1–25, 2005.

[25] W. Hu, T. Chen, and S. L. Shah. Detection of frequent alarm patterns in industrial alarm floods using itemset mining methods. *IEEE Transactions on Industrial Electronics*, 65(9):7290–7300, 2018.

[26] W. Hu, J. Wang, and T. Chen. A new method to detect and quantify correlated alarms with occurrence delays. *Computers & Chemical Engineering*, 80:189–198, 2015.

[27] W. Hu, J. Wang, and T. Chen. A local alignment approach to similarity analysis of industrial alarm flood sequences. *Control Engineering Practice*, 55:13–25, 2016.

[28] W. Hu, J. Wang, T. Chen, and S. L. Shah. Cause-effect analysis of industrial alarm variables using transfer entropies. *Control Engineering Practice*, 64:205–214, 2017.

[29] W. Hu, G. Yang, Y. Li, W. Cao, and M. Wu. Root cause identification of industrial alarm floods using word embedding and few-shot learning. *IEEE Transactions on Industrial Informatics*, 2023.

[30] ISA (International Society of Automation), Durham, NC USA. *ANSI/ISA-18.2: Management of Alarm Systems for the Process Industries*, 2016.

[31] R. Kaced, A. Kouadri, and K. Baiche. Designing alarm system using modified generalized delay-timer. *Journal of Loss Prevention in the Process Industries*, 61:40–48, 2019.

[32] S. R. Kondaveeti, I. Izadi, S. L. Shah, and T. Chen. On the use of delay timers and latches for efficient alarm design. In *2011 19th Mediter-*

ranean Conference on Control & Automation (MED), pages 970–975. IEEE, 2011.

[33] S. R. Kondaveeti, I. Izadi, S. L. Shah, D. S. Shook, R. Kadali, and T. Chen. Quantification of alarm chatter based on run length distributions. *Chemical Engineering Research and Design*, 91(12):2550–2558, 2013.

[34] S. Lai, F. Yang, and T. Chen. Online pattern matching and prediction of incoming alarm floods. *Journal of Process Control*, 56:69–78, 2017.

[35] S. Lai, F. Yang, T. Chen, and L. Cao. Accelerated multiple alarm flood sequence alignment for abnormality pattern mining. *Journal of Process Control*, 82:44–57, 2019.

[36] Y. Li, W. Cao, R. B. Gopaluni, W. Hu, L. Cao, and M. Wu. False alarm reduction in drilling process monitoring using virtual sample generation and qualitative trend analysis. *Control Engineering Practice*, 133:105457, 2023.

[37] M. Lucke, M. Chioua, C. Grimholt, M. Hollender, and N. F. Thornhill. Advances in alarm data analysis with a practical application to online alarm flood classification. *Journal of Process Control*, 79:56–71, 2019.

[38] M. Lucke, A. Stief, M. Chioua, J. R. Ottewill, and N. F. Thornhill. Fault detection and identification combining process measurements and statistical alarms. *Control Engineering Practice*, 94:104195, 2020.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[40] E. Naghoosi, I. Izadi, and T. Chen. Estimation of alarm chattering. *Journal of Process Control*, 21(9):1243–1249, 2011.

[41] E. Naghoosi, I. Izadi, and T. Chen. A study on the relation between alarm deadbands and optimal alarm limits. In *Proceedings of the 2011 American Control Conference*, pages 3627–3632. IEEE, 2011.

[42] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[43] T. Niyazmand and I. Izadi. Pattern mining in alarm flood sequences using a modified prefixspan algorithm. *ISA Transactions*, 90:287–293, 2019.

[44] A. Pariyani, W. D. Seider, U. G. Oktem, and M. Soroush. Incidents investigation and dynamic analysis of large alarm databases in chemical plants: A fluidized-catalytic-cracking unit case study. *Industrial & Engineering Chemistry Research*, 49(17):8062–8079, 2010.

[45] M. R. Parvez, W. Hu, and T. Chen. Real-time pattern matching and ranking for early prediction of industrial alarm floods. *Control Engineering Practice*, 120:105004, 2022.

[46] M. R. Parvez, W. Hu, and T. Chen. An association rule mining approach to predict alarm events in industrial alarm floods. *Control Engineering Practice*, 138:105617, 2023.

[47] D. G. Rees. *Essential Statistics*. Chapman & Hall/CRC, Philadelphia, PA, 4th edition, Dec. 2000.

[48] V. Rodrigo, M. Chioua, T. Hagglund, and M. Hollender. Causal analysis for alarm flood reduction. *IFAC-PapersOnLine*, 49(7):723–728, 2016.

[49] M. H. Roohi, P. Ramazi, and T. Chen. Towards accurate root-alarm identification: The causal Bayesian network approach. In *5th International*

Conference on Control and Fault-Tolerant Systems, pages 169–174. IEEE, 2021.

[50] D. H. Rothenberg. *Alarm Management for Process Control: A Best-Practice Guide for Design, Implementation, and Use of Industrial Alarm Systems*. Momentum Press, 2009.

[51] J. Shang and T. Chen. Early classification of alarm floods via exponentially attenuated component analysis. *IEEE Transactions on Industrial Electronics*, 67(10):8702–8712, 2019.

[52] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[53] T. F. Smith, M. S. Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[54] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[55] N. Tamascelli, N. Paltrinieri, and V. Cozzani. Predicting chattering alarms: A machine learning approach. *Computers & Chemical Engineering*, vol. 143, art. no. 107122, 2020.

[56] B. Vogel-Heuser, D. Schütz, and J. Folmer. Criteria-based alarm flood pattern recognition using historical data from automated production systems (aPS). *Mechatronics*, 31:89–100, 2015.

[57] J. Wang and T. Chen. An online method for detection and reduction of chattering alarms due to oscillation. *Computers & chemical engineering*, 54:140–150, 2013.

[58] J. Wang and T. Chen. An online method to remove chattering and repeating alarms based on alarm durations and intervals. *Computers & Chemical Engineering*, 67:43–52, 2014.

[59] J. Wang and T. Chen. Main causes of long-standing alarms and their removal by dynamic state-based alarm systems. *Journal of Loss Prevention in the Process Industries*, 43:106–119, 2016.

[60] J. Wang, H. Li, J. Huang, and C. Su. Association rules mining based analysis of consequential alarm sequences in chemical processes. *Journal of Loss Prevention in the Process Industries*, 41:178–185, 2016.

[61] J. Wang, S. Sun, Z. Wang, X. Zhou, and P. Gyasi. Alarm deadband design based on maximum amplitude deviations and bayesian estimation. *IEEE Transactions on Control Systems Technology*, 31(4):1941–1948, 2023.

[62] J. Wang, Z. Wang, X. Zhou, and F. Yang. Design of delay timers based on estimated probability mass functions of alarm durations. *Journal of Process Control*, 110:154–165, 2022.

[63] J. Wang, F. Yang, T. Chen, and S. L. Shah. An overview of industrial alarm systems: Main causes for alarm overloading, research status, and open problems. *IEEE Transactions on Automation Science and Engineering*, 13(2):1045–1061, 2016.

[64] J. Wang, Y. Zhao, and Z. Bi. Criteria and algorithms for online and offline detections of industrial alarm floods. *IEEE Transactions on Control Systems Technology*, 26(5):1722–1731, 2017.

[65] Z. Wang, X. Bai, J. Wang, and Z. Yang. Indexing and designing deadbands for industrial alarm signals. *IEEE Transactions on Industrial Electronics*, 66(10):8093–8103, 2018.

[66] P. C. Wong, P. Whitney, and J. Thomas. Visualizing association rules for text mining. In *Proceedings 1999 IEEE Symposium on Information Visualization*, pages 120–123. IEEE, 1999.

[67] J. Xu, J. Wang, I. Izadi, and T. Chen. Performance assessment and design for univariate alarm systems based on FAR, MAR, and AAD.

IEEE Transactions on Automation Science and Engineering, 9(2):296–307, 2011.

[68] Y. Xu and J. Wang. A maximum-entropy-based method for alarm flood prediction. *Journal of Process Control*, 107:58–69, 2021.

[69] Y. Xu, J. Wang, and Y. Yu. Alarm event prediction from historical alarm flood sequences based on bayesian estimators. *IEEE Transactions on Automation Science and Engineering*, 17(2):1070–1075, 2019.

[70] B. Yang, H. Wang, H. Li, and Y. He. A novel detection of correlated alarms with delays based on improved block matching similarities. *ISA Transactions*, 98:393–402, 2020.

[71] B. Zhou, W. Hu, K. Brown, and T. 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.

[72] B. Zhou, W. Hu, and T. Chen. Pattern extraction from industrial alarm flood sequences by a modified CloFAST algorithm. *IEEE Transactions on Industrial Informatics*, 18(1):288–296, 2021.