Data-driven Methods for Industrial Alarm Flood Analysis

by

Shiqi Lai

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

The effectiveness of industrial process monitoring depends heavily on alarm systems. If alarm configurations are not rationally designed, the problem of excessive alarm messages would impact negatively the efficiency or even the safety of plant operations due to distracted information provided to operators. An alarm flood is an extreme case of this problem, during which the operator efficiency of handling important alarms is usually reduced significantly because of the overwhelming workload created by the numerous alarm messages. Consequently, techniques are needed to reduce the number and severity of alarm floods, as well as facilitate operators for proper operations during alarm floods. Motivated by this, this thesis focuses on the development of the data-driven techniques for alarm flood analysis.

Two research topics are considered. The first topic is pattern mining in multiple alarm flood sequences. Incorporating time information into the evaluation of an alignment and dealing with large scale datasets are the main challenges. Two methods have been developed. The first one conducts a traversal search based on dynamic programming to obtain the optimal alignment of multiple alarm flood sequences. The second one achieves significant improvement on computational efficiency by applying approximations, but at the cost of a small amount of alignment accuracy.

The second topic is online pattern matching and prediction of incoming alarm floods. The objective is to match the online alarm sequence with the patterns in the database and identify whether the online sequence is similar to any of the patterns in the database; predict the oncoming alarm flood if a matching is found. A method has been developed for this topic, which utilizes a proposed incremental dynamic programming procedure to break the whole computational burden of matching two sequences into small pieces that can be finished quickly in each individual step.

The effectiveness of the proposed methods and their parameter robustness are tested by case studies based on datasets from a real chemical plant. Furthermore, a causality analysis for alarm floods is conducted. Process data associated with the alarms raised during an alarm flood is acquired and causality analysis is applied on the process data to generate causal maps, which are useful for root cause analysis and early predictions of future similar alarm floods.

Preface

The research work in Chapters 3-4 of the thesis was part of an international research collaboration with Dr. Fan Yang at Tsinghua University. The ideas in Chapter 2 were my own ideas. The ideas in Chapter 3-4 were from discussions with Dr. Fan Yang. The ideas of the causality analysis in Chapter 5 were from discussions with Dr. Sirish L. Shah from the Department of Chemical and Materials Engineering at University of Alberta. The algorithms, mathematical derivations, and industrial case studies were my original work, as well as the introduction in Chapter 1.

- Chapter 2 has been published as: Shiqi Lai and Tongwen Chen, A method for pattern mining in multiple alarm flood sequences, *Chemical Engineering Research and Design*, 117: 831-839, 2017. A short version has been published as: Shiqi Lai and Tongwen Chen, Methodology and application of pattern mining in multiple alarm flood sequences, *Proceedings of 9th IFAC Symposium on Advanced Control of Chemical Processes* Whistler, Canada, pages 657-662, 2015.

- Chapter 3 has been published as: Shiqi Lai, Fan Yang, Tongwen Chen, Online pattern matching and prediction of oncoming alarm floods, *Journal of Process Control*, 56: 69-78, 2017 (in press).

- Chapter 4 has been submitted for publication as: Shiqi Lai, Fan Yang, Tongwen Chen, Accelerated multiple alarm flood sequence alignment for abnormality pattern mining, *Journal of Process Control*, 2017.

Dedicated to my parents, wife, and friends for being with me during my darkest moment.

Acknowledgements

First and foremost, I would like to give my sincere gratitude to Professor Tongwen Chen, for providing valuable suggestions and allowing me the freedom to pick my own interested research topics. I just got my Bachelor's degree when I started the PhD program, it was Dr. Chen who guided and inspired me on research from day one. I improved my presentations skills because of his comments and critiques, found my own research topics by following his guidance, and acquired academic writing ability through his efforts on proofreading my research articles. Working under his supervision was an unforgettable experience.

In addition, I would like to thank Dr. Fan Yang and Dr. Sirish L. Shah. I benefited greatly from the discussions with them. Their suggestions and comments facilitated my research work and helped form some of the important ideas on my research topics. Dr. Yang, especially, also gave me enormous help on proofreading my academic articles.

Furthermore, I am deeply thankful to the current and former group members. Dr. Dawei Shi, Dr. Yue Cheng, and Dr. Wenkai Hu shared with me their research work selflessly to help me find my own research topics. I also had collaborative research with Ahmad W. Al-Dabbagh, Dr. Hu, and Cen Guo (Tsinghua University); some of the collaborated research work has been published or submitted for publication in conference proceedings and peer-reviewed journals.

At last I am grateful to my parents, wife, and friends. They were with me the whole time when I was physically ill and had to go through a serious of tedious treatments. Their support lightened me the way through the darkness.

Contents

List of Tables

List of Figures

List of Symbols

μ	Mismatch Penalty
δ	Gap Penalty
σ	Variance of the Scaled Gaussian Function
Λ	Alphabet of Unique Alarms
$\mathbf{d}_m$	Time Distance Vector
$\mathbf{w}_m$	Time Weight Vector
$H_{m+1,n+1,o+1}$	Similarity Index
$S((e_a,t_a),(e_b,t_b))$	Similarity Score
$\mathscr{S}((e_a,t_a),(e_b,t_b))$	Modified Similarity Score
$\mathbb{S}(T_{Ap},T_{Bq})$	Generalized Similarity Score
$\mathcal{O}(kn)$	Asymptotic Notation
Σ	Covariance Matrix
$F_{X,Y}$	Interdependency
$F_{X\to Y}$	Granger Causality
$F_{Y\to X\|Z}$	Conditional Granger Causality
$f_{Y\to X\|Z}(w)$	Frequency Domain Conditional Granger Causality
$\mathbb{R}^+$	Domain of Real Numbers

List of Acronyms

A&E	Alarm & Event
AAD	Averaged Alarm Delay
ALM	Alarm Occurrences
CWF	Chattering Window Filter
BLAST	Basic Local Alignment Search Tool
BPCS	Basic Process Control System
DCS	Distributed Control System
DTE	Direct Transfer Entropy
DTW	Dynamic Time Warping
EEMUA	Engineering Equipment and Materials Users' Association
FAR	False Alarm Rate
FP-GROWTH	Frequent Pattern Growth
GSP	Generalized Sequential Patterns
HMI	Human-Machine Interface
HMM	Hidden Markov Models
IEC	International Electrotechnical Commission
ISA	International Society of Automation
LCS	Longest Common Subsequence
MAR	Missed Alarm Rate
P&ID	Piping and Instrumentation Diagram
ROC	Receiver Operating Characteristic

RTN	Return-To-Normal
SCADA	Supervisory Control and Data Acquisition System
SDG	Signed Directed Graph
SIS	Safety Instrumented System
STG	Steam Turbine Generator
SW	Smith-Waterman
SWF	Sequence Window Filter
TE	Transfer Entropy
T0E	Transfer Zero-Entropy
ZOH	Zero-Order Hold

Chapter 1

Introduction

1.1 Motivation and Background

As the scale of industrial plants grows, process monitoring becomes indispensable for ensuring the safety and efficiency of operation. Distributed Control Systems (DCS) and Supervisory Control and Data Acquisition (SCADA) systems have gained more popularity along with the occurrence of new technologies on sensors, software, and communication networks. Alarms can be easily configured on a DCS or SCADA system, which reduces the cost of alarm design and configuration, but on the other hand prolongs the life-cycle of alarm rationalization, targeted to avoid poor alarm designs. Consequently, standards and guidelines such as ISA-18.2 [46], EEMUA-191 [29], and IEC-62682 [20] have been developed and research work on fault diagnosis, process monitoring, and alarm management has received increasing attentions [22, 60, 61].

One consequence of poor alarm designs is an increase in both the number and intensity of alarm floods. Alarm floods can interfere with operators and may therefore cause or aggravate industrial accidents because even an experienced operator can be overwhelmed by tens or hundreds of alarms raised in a short period of time. Without enough time for analysis, an operator can only handle the abnormal event based on his/her experience or even take no actions, likely leading to improper executions for important abnormalities. As a result, based on operators' normal response time [21], both EEMUA-191 and ISA-18.2 standards [29, 46] recommend setting the upper limit of alarms announced to an operator to be 6 alarms per hour, and the alarm rate threshold for alarm floods to be 10 alarms per 10 min per operator. Both standards also suggest the time under alarm floods should be less than 1% of the reporting period.

1.1.1 Alarm System and Data

Figure 1.1 [46] shows a typical schematic of alarm system dataflow. The basic Process Control System (BPCS) and Safety Instrumented System (SIS) are the two important components that control the process and generate alarms based on sensor measurements and predefined logics. The panel and the Human Machine Interface (HMI) allow the operator to intervene the control of the process and view Alarm logs.

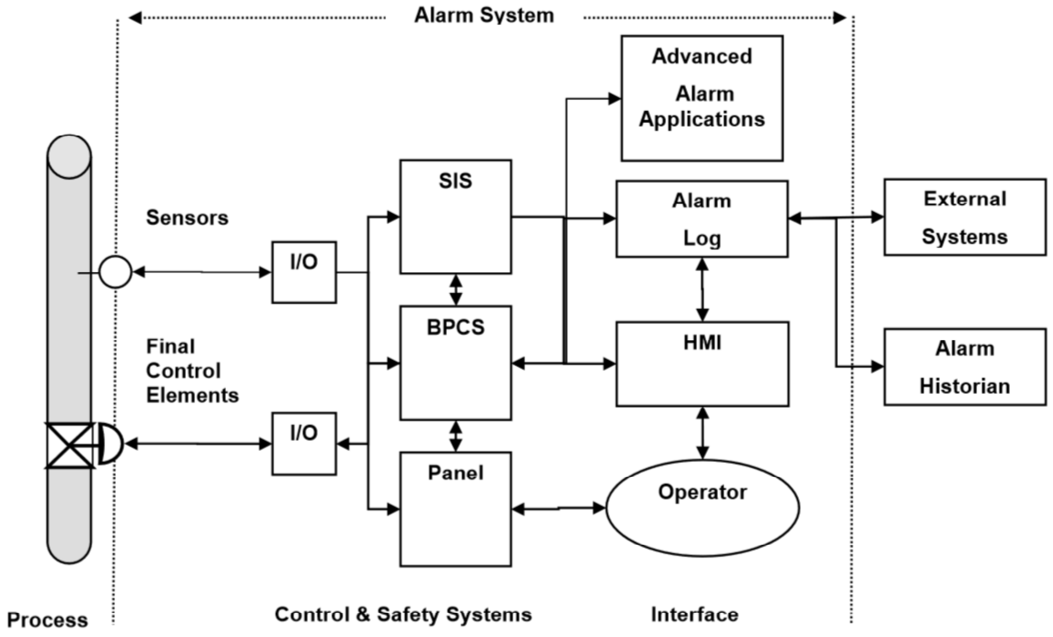


Figure 1.1: Alarm system dataflow [46].

In a DCS engaged plant, all monitored process variables and alarms are collected onto the DCS server, where alarms are formatted and stored in Alarm & Event (A&E) logs. Generally, in A&E logs, an alarm message contains important information such as time stamp, tag name, alarm type (identifier), priority, and acknowledgment state [47, 56]. An alarm message may also include descriptive information such as trip value, event, and description. “Time stamp” indicates the occurrence time of an alarm. “Tag name” is an unique code of a system variable that is being monitored. “Alarm type”, or “identifier”, is the alarm type related to the monitored variable (e.g., PVLO when the variable is under its low limit or BADPV when the variable is out of its nor-

Table 1.1: An example of a segment of an A&E log.

TIME	TAG	TRIP_VAL	TYPE	PRIO	DSCR	UNIT	EVENT	ACK
2013-09-01 0:13	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:13	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	134.994	RTN
2013-09-01 0:15	B	3.9	PVLO	JOURNAL	D3: ABS AREA SUMP LEVEL	Q1	3.899	ALM
2013-09-01 0:15	C		CMDDIS	HIGH	G-5B Abs Area Sump Pmp	Q1	CLOSED	ALM
2013-09-01 0:15	D		CMDDIS	HIGH	D1:ABS HOLD TNK FLSH WTR	Q1	TRANSIT	ALM
2013-09-01 0:15	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:15	D	2	CMDDIS	HIGH	D1:ABS HOLD TNK FLSH WTR	Q1	OPENED	RTN
2013-09-01 0:16	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	134.017	RTN
2013-09-01 0:16	C		CMDDIS	HIGH	G-5B Abs Area Sump Pmp	Q1	CLOSED	RTN
2013-09-01 0:18	E	37.8	PVHI	LOW	V2: HYDR CON-TROL UNIT	Q1		RTN
2013-09-01 0:19	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:20	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	136.02	RTN
2013-09-01 0:22	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:22	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	134.017	RTN
2013-09-01 0:25	B	3.9	PVLO	JOURNAL	D3: ABS AREA SUMP LEVEL	Q1		RTN
2013-09-01 0:26	F	1.341	PVHI	LOW	G4: PRMRY UF-PLT481/D1	Q1	1.341	ALM
2013-09-01 0:26	F	1.341	PVHI	LOW	G4: PRMRY UF-PLT481/D1	Q1	1.341	ALM
2013-09-01 0:28	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM

mal range). "Acknowledgment state" shows the status of an alarm, namely, Return to Normal (RTN) or Alarm (ALM). Table 1.1 gives an example of a segment of an A&E log. In the example, a PVLO alarm was raised for tag A at 2013-09-01 0:13 because its corresponding process variable went below the trip value, which was set as 131. Then, this PVLO alarm for tag A returned to normal at 2013-09-01 0:13 since the corresponding process variable went back to its normal range. Normally, we connect the tag name and the tag identifier of an alarm together with a dot in between while processing alarm messages, e.g., A.PVLO, and replace ALM and RTN messages with 1 and 0 for computing purposes.

Process data are the measurements of process variables and they are normally stored in the DCS database. Depending on the model of a DCS server, it could be possible to search on the DCS server for the corresponding process variable based on the tag name of an alarm. An alarm tag can be associated with one, multiple, or even none process variable. Table 1.2 shows an example of a segment of extracted process data, where A10, A11, A12 and A14A are the name of the process variables. One utility of process data for alarm flood

Table 1.2: An example of a segment of extracted process variables.

TIME	A10	A11	A12	A14A
14-08-2014 12:00:00 AM	95.99160004	83.85221863	0.136333764	3.821421623
14-08-2014 12:00:01 AM	95.99160004	83.85218811	0.136333764	3.818593025
14-08-2014 12:00:02 AM	95.99160004	83.85215759	0.136333764	3.815764427
14-08-2014 12:00:03 AM	95.99160004	83.85212708	0.136333764	3.812935829
14-08-2014 12:00:04 AM	95.99160004	83.85209656	0.136333764	3.810107231
14-08-2014 12:00:05 AM	95.99160004	83.85206604	0.136333764	3.807278633
14-08-2014 12:00:06 AM	95.99160004	83.85203552	0.136333764	3.804450274
14-08-2014 12:00:07 AM	95.99160004	83.852005	0.136333764	3.801621675
14-08-2014 12:00:08 AM	95.99160004	83.85196686	0.136333764	3.798793077

analysis is to be used in causality analysis for root cause identification.

1.1.2 Alarm Flood and Alarm Flood Analysis

In literature, there does not exist a strict quantitative definition for alarm floods. In the ISA-18.2 standard [46], the descriptive definition for an alarm flood is "a condition during which the alarm rate is greater than the operators can effectively manage". Normally, alarm floods are tracked based on the threshold suggested in the ISA-18.2 standard, 10 alarms per 10 min per operator. Alarms raised during the period when the alarm rate is higher than the suggested threshold are extracted as an alarm flood sequence.

Usually a large proportion of an alarm flood are univariate nuisance alarms, of which chattering alarms form an important part. Methods such as high density alarm plots, calculation of a chattering index, delay-timers, and dead-bands can be used to visualize, quantify, and reduce such chattering alarms [49, 55, 58]. However, by applying delay-timers and dead-bands, often one cannot totally suppress alarms during alarm floods; the remaining alarms are mainly consequence alarms (multivariate alarms), which can be caused by three reasons: (1) process state changes such as start-up and shutdown, (2) bad alarm configurations such as redundant measurements on a single process and (3) causal relationships among measured variables. In this case, alarm flood analysis is useful to reveal connection of alarm messages and discover possible patterns in alarm flood sequences.

The discovered patterns are helpful in alarm management. For example, the first alarm message in a pattern is usually more suspicion to be the root cause of the following alarms in the sequence. The discovered patterns can also help train the operators to handle corresponding series of alarm messages more efficiently in order to prevent the overwhelming situation during alarm floods. Some badly configured parts in alarm systems can be revealed by the alarm flood patterns as well. Moreover, if a pattern database can be set up, it would become possible to match online alarm messages with the existing patterns, providing operators an early warning of incoming floods and their corresponding management strategies. As suggested in the ISA standard [46], potential dynamic alarm management could be predictive alarming, online alarm attribute modification, and online alarm suppression. Causality analysis can also be applied once the patterns are obtained to help recover the connections between the corresponding tags in the pattern sequences and target the root cause.

Figure 1.2 shows the procedures of alarm flood analysis. The purposes of the offline part is to set up a pattern database for offline analysis and the use of online pattern matching. During the offline part, data preprocessing is

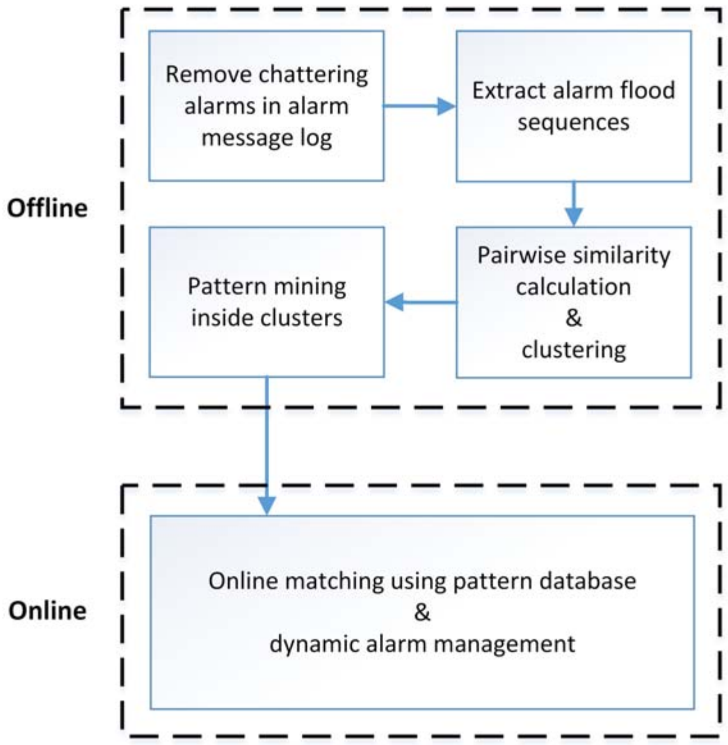

Offline
Remove chattering alarms in alarm message log
Extract alarm flood sequences
Pairwise similarity calculation & clustering
Pattern mining inside clusters
Online
Online matching using pattern database & dynamic alarm management

Figure 1.2: Flowchart of alarm flood analysis

carried out first to remove chattering alarms before the extraction of alarm floods. Usually an off-delay timer is used since it will not introduce detection delays when alarms are raised. After preprocessing, an alarm rate threshold of

Table 1.3: An example of a segment of an extracted alarm flood sequence.

TIME	ALARM	TRIP_VAL	PRIO	DSCR	UNIT	EVENT	ACK
2013-09-01 0:13	A.PVLO	131	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:15	B.PVLO	3.9	JOURNAL	D3: ABS AREA SUMP LEVEL	Q1	3.899	ALM
2013-09-01 0:15	C.CMDDIS		HIGH	G-5B Abs Area Sump Pmp	Q1	CLOSED	ALM
2013-09-01 0:15	D.CMDDIS		HIGH	D1:ABS HOLD TNK FLSH WTR	Q1	TRANSIT	ALM
2013-09-01 0:15	A.PVLO	131	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:19	A.PVLO	131	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:22	A.PVLO	131	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:26	F.PVHI	1.341	LOW	G4: PRMRY UF-PLT481/D1	Q1	1.341	ALM
2013-09-01 0:26	F.PVHI	1.341	LOW	G4: PRMRY UF-PLT481/D1	Q1	1.341	ALM
2013-09-01 0:28	A.PVLO	131	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM

10 alarms per 10 min per operator, as suggested by EEMUA and ISA standards [29, 46], is used to extract alarm floods. The periods during which the alarm rate is higher than the threshold are extracted. The extracted alarm flood sequences are then numbered and saved for further pattern analysis. Table 1.3 is an example of a segment of an extracted alarm flood sequence. The tag names and the tag identifiers are connected together with a dot in between, e.g., A.PVLO, and named as "ALARM". All the RTN messages have been removed because only the alarm annunciation information is used in alarm flood analysis. Figure 1.3 shows an example of data preprocessing and alarm flood extraction, in which an off-delay timer of 40 seconds has been applied to remove chattering alarms and a threshold of 10 alarms per 10 min per operator has been set for alarm flood extraction. The red and blue lines are the alarm burst plots (alarm rate) with and without off-delay timers.

Then the algorithm in [18] is applied to obtain the pairwise similarity scores of the 14 extracted alarm floods. Based on those scores, clustering can be carried out to group the closely related alarm flood sequences together, as shown in Figure 1.4. Inside the red box, alarm floods 6, 7, 8, 9, 10, and 13 have high similarity score with each other because they share a common alarm sequence segment. In the next step, pattern mining algorithms can be applied to automatically find the common pattern sequence for each of the clusters and save the patterns into a database for offline and online analysis.

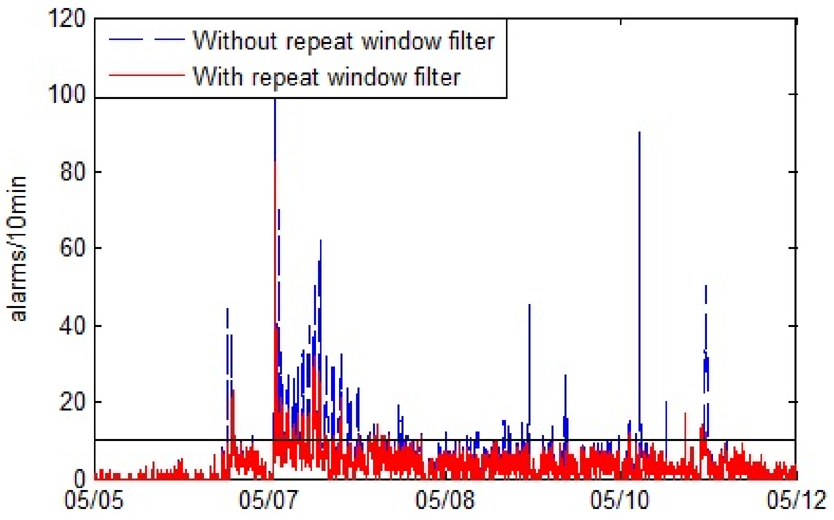


Figure 1.3: Example of alarm flood extraction

1.2 Literature Survey

This section contains a detailed literature survey on recent techniques for univariate alarm analysis, offline pattern mining of alarm floods, online alarm flood pattern matching, and causality analysis.

Before going into the survey, we would like to clarify the difference between pattern mining and pattern matching as they can be confusing. The biggest differences are the input and output. For pattern mining, the input is two or multiple sequences and the output is a pattern sequence that is shared by all or at least most of the input sequences. On the contrary, in terms of pattern matching, the input is a pair of sequences and the output is whether the pair of two sequences are similar with each other. In addition, if pattern matching is applied online, then the two inputs are (1) a full sequence to be matched with and (2) an online sequence that only contains the current and past alarm messages, as future alarms are unknown. The approaches for both pattern mining and pattern matching may generate a similarity score alongside their outputs. For pattern mining, the similarity score means how well the pattern sequence represents the input sequences. As for pattern matching, the similarity score shows how much the pair of sequences resemble each other.

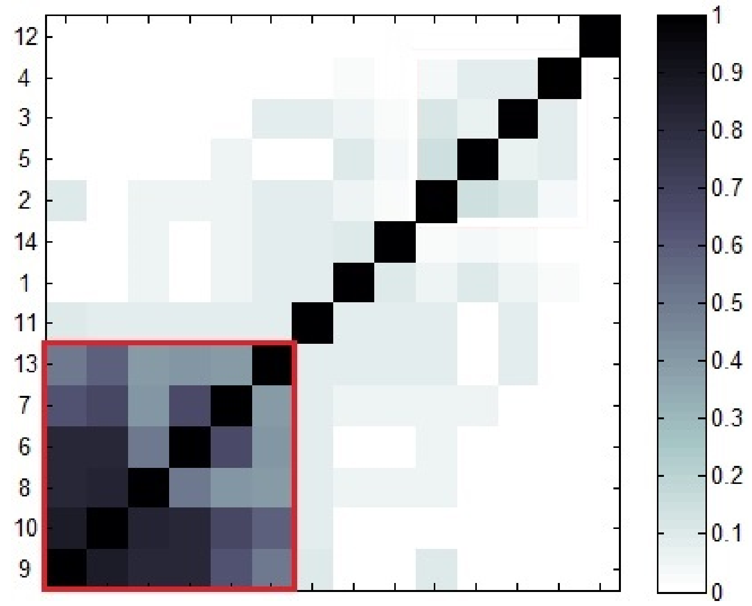


Figure 1.4: Example of clustering 14 alarm floods based on pairwise similarity scores

1.2.1 Methods for Univariate Alarm Analysis

A general framework for univariate alarm analysis based on the three performance metrics, false alarm rate (FAR), missed alarm rate (MAR), and average detection delay (ADD), was proposed in [48, 49, 85]. In the framework, Receiver Operating Characteristic (ROC) curves were generated to reveal the trade-off between FAR and MAR; the trip point was designed by selecting a desired point on the ROC curve while maintaining ADD within an acceptable range. The framework has been widely accepted to evaluate the methods for univariate alarm analysis.

Four categories of tools were proposed to analyze univariate alarms: trip point optimization techniques, delay timers, dead-bands, and filters. The authors in [39, 85] reconfigured alarm limits using historical process data. The design of delay timers based on performance metrics FAR, MAR, and ADD was studied in [3, 57]. A generalized delay timer that uses n_1 out of n consecutive samples was proposed and analyzed in [1, 2]. In [4], the authors proposed a multi-mode delay timer based on hidden Markov models (HMM). The performance of dead-bands was evaluated by authors in [3, 57]. A method to determine the optimal alarm dead-band by the measurement noise and the trajectory of the process data before the annunciation of an alarm was proposed in [43].

Chattering alarms are the mostly-encountered univariate alarms. The authors in [76] claimed that chattering alarm may account for 10-60% of alarm occurrences. This claim was backed up by the analysis carried out on 75 alarm systems [40], showing that chattering alarms account for more than 70% of the alarm occurrences. Causes of chattering alarms can be noise/disturbance, repeated control loop switches, and plant oscillations. The authors in [14] showed examples of chattering alarms caused by noise when the value of the process variables was close to their alarm limits. In [81], chattering alarms were found to be caused by control loop on-off actions and oscillatory disturbances.

Methods were proposed to detect and reduce chattering alarms. The authors in [90] proposed a way to detect chattering alarms based on alarm occurrences and operator actions. Rules based on chattering indices were proposed in [67, 81, 82], where chattering indices were obtained via the calculation of alarm run length distributions. To reduce the number of chattering alarms, [14, 15, 65] introduced shelving mechanisms to automatically suppress repeating alarms when detected. In [7], the authors developed filters based on the classification results of process variables to deal with chattering alarms more efficiently. The authors in [44] designed a pre-alarm mechanism to reduce chattering alarms. Delay timers, dead-bands, and filters were designed and evaluated in [3, 57] for the reduction of chattering alarms.

1.2.2 State of Art Concerning Pattern Mining of Alarm Floods

Expert consultation and operator's experience, by far, are still the two approaches that have been used by most industrial companies when dealing with consequence alarms in an alarm flood. Expert consultation provides good and accurate results; however, without doubts, its efficiency is extremely low because of the involvement of a relatively large amount of process knowledge. The approach based on the operators' experience is usually faster, but its accuracy is not guaranteed since it depends heavily on human judgments. Many pattern mining algorithms have been developed to facilitate the study of consequence alarms in alarm floods.

Sequence pattern mining finds relevant parts in data examples that appear repeatedly. In the commerce area, frequent transactions are highly useful information for retailers to learn what to stock and how to arrange the layout in their shops. The datasets are normally series of transactions made by customers. In [5], an apriori-like algorithm was proposed to mine frequent patterns by combining short sequences into longer ones. The authors in [38] developed an algorithm to achieve patterns based on tree structures. In [91], a vertical data format was utilized to generate patterns, which did not require multiple scans over the database. Algorithms based on projections were proposed in [37, 72], which had improvements on efficiency.

In the biology area, a very large number of pattern mining algorithms have been proposed, most of which were modifications of the pairwise sequence alignment methods [8, 69, 71, 79]. Mainly two types of modifications exist: the exhaustive search approach, as in [51], which can guarantee global optimality, and the progressive pairwise approach, such as [30] and [80], which can approximate global optimal solutions. There is another type of algorithms based on profile Hidden Markov Models (HMM), such as [28]. However, in [27], the authors pointed out that: "a suitable HMM architecture (the number of states, and how they are connected by state transactions) must usually be designed manually."

While a large amount of pattern mining algorithms exist in the literature, most of them cannot be directly applied to find patterns in alarm event logs, due to the special format of alarm data (every message comes with a time stamp). A variety of pattern mining methods have been developed to analyze alarm floods. The authors in [59] manually selected some alarm tags to be the target tags at first, then applied a context-based segmentation using the target tags to obtain the patterns. In [6], the authors proposed a way to capture the relations of alarm messages in an alarm flood using first-order Markov chains; then, the Euclidean distance between the transition probability matrices of two alarm floods was used to cluster the alarm floods into groups. The authors in [31] developed a pattern growth method to obtain patterns from alarm floods. However, the authors pointed out "the proposed method was sensitive to disturbances so that the pattern in sequences had to be exactly the same in order to be recognized." A dissimilarity based method was proposed in [17] to extract alarm sequence templates of given faults and a Needleman-Wansch based algorithm was developed to isolate alarm sequences caused by certain known faults in [16]. In [77], the authors proposed a method to re-order alarms during alarm floods by assigning their priorities values and designed an interface for a better operator support during flood scenarios.

Time information is also very important. Two sequences with the same alarms and their ordering but different time intervals in between the alarms can be caused by totally different faults. Because of this, methods have been proposed to take time intervals between alarms into consideration when studying the sequences. In [19], Generalized Sequential Patterns (GSP) were applied to search for pattern alarm sequences in alarm floods; the algorithm was able to blur the orders of alarms if they are raised in quick succession and thus their order becomes less important. Algorithms for pattern matching of two sequences were developed in [18] and [41], based on the Smith-Waterman [79] and BLAST [8] algorithms, respectively. In both papers, a weighted time distance and vector were used to take the relative time between alarms into account during the alignment. In [62], the authors proposed a method for pattern mining in multiple alarm flood sequences using a sequence alignment approach. However, the computational cost of the algorithm does not scale well with the number and length of sequences to be aligned.

1.2.3 Current Status of Online Alarm Flood Pattern Matching

The algorithms introduced in the previous subsection are all offline methods, as the full sequences need to be available before algorithms are carried out. However, for the online matching of alarm floods, the online sequence only contains the current and past alarm messages, as future alarms are unknown. Moreover, each time when there is a new alarm raised and added to the online alarm sequence, the matching between the online sequence and the patterns in the database need to be re-conducted, which significantly increases the requirement for computational efficiency. Segmentation of the online sequence is another obstacle for online alarm flood matching since as new alarms continue to be annunciated, the online sequence grows longer and longer; it should be segmented before matchings are carried out. These are the three main reasons why the offline pattern mining methods cannot be applied to online matching directly.

Some online algorithms have already been developed to match text sequences. In [64], the authors introduced a consecutive suffix alignment problem: given the matching result of the longest common subsequence (LCS) or the edit distance between two sequences A and B, incrementally compute the answer for A and bB, and the answer for A and Bb, where "b" is an added item. In the same paper, the authors proposed an algorithm to handle this problem that runs in $\mathcal{O}(kn)$ time and uses $\mathcal{O}(m+n+k^2)$ memory space, where m and n are the lengths of sequences A and B, k is the tolerated difference in edit distance, by taking advantage of the properties of dynamic programming matrices. The big O is the notation that characterizes the growth rates of the occupied space and computational complexity. In [53], the authors simplified the complicated properties of dynamic programming matrices and proposed an algorithm that runs in $\mathcal{O}((m+n)n)$ time and uses $\mathcal{O}(mn)$ memory space. The authors in [45] proposed another algorithm that improved space occupation to $\mathcal{O}(m+n)$. In [63], an algorithm with a preprocessing stage was introduced that runs in $\mathcal{O}(nL + n \log L)$ time and uses $\mathcal{O}(mn)$ space, where L was the length of the LCS between A and B.

Unfortunately, since text data does not contain time information and the LCS alignment problem is much simpler than the standard alignment problem (with gap and mismatch penalties), the dynamic programming matrix used for finding the LCS between text sequences has many essential monotonicity properties with respect to diagonals [64] that are not compatible to the problem of aligning alarm sequences. Thus, the algorithms introduced above cannot be used for the online alarm sequence matching problem directly.

1.2.4 Causality Analysis for Alarm Floods

When a fault propagates through the physical connections of a plant, a series of alarms may be generated and an alarm flood may occur due to mutual dependencies of the process elements. Causality analysis has been introduced in [35, 36, 78] and adapted for industrial processes in [11, 88] to localize root causes and investigate fault propagation pathways.

Statistic measurements such as cross-correlation and coherence were utilized in [10, 34] to measure correlation and causalities. In [25], a direct transfer entropy was proposed to not only quantify the causality between process tags but also tell if the causality is direct or indirect. A transfer zero-entropy based causality analysis method was proposed in [26] on the basis of 0-entropy and

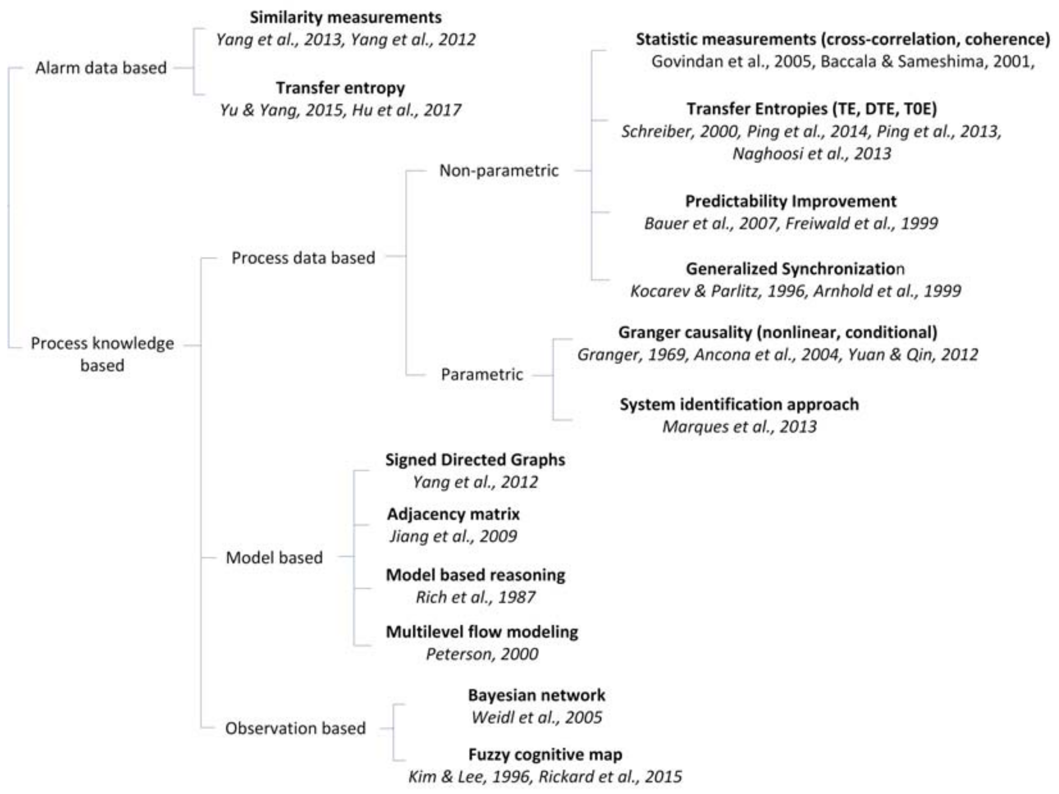
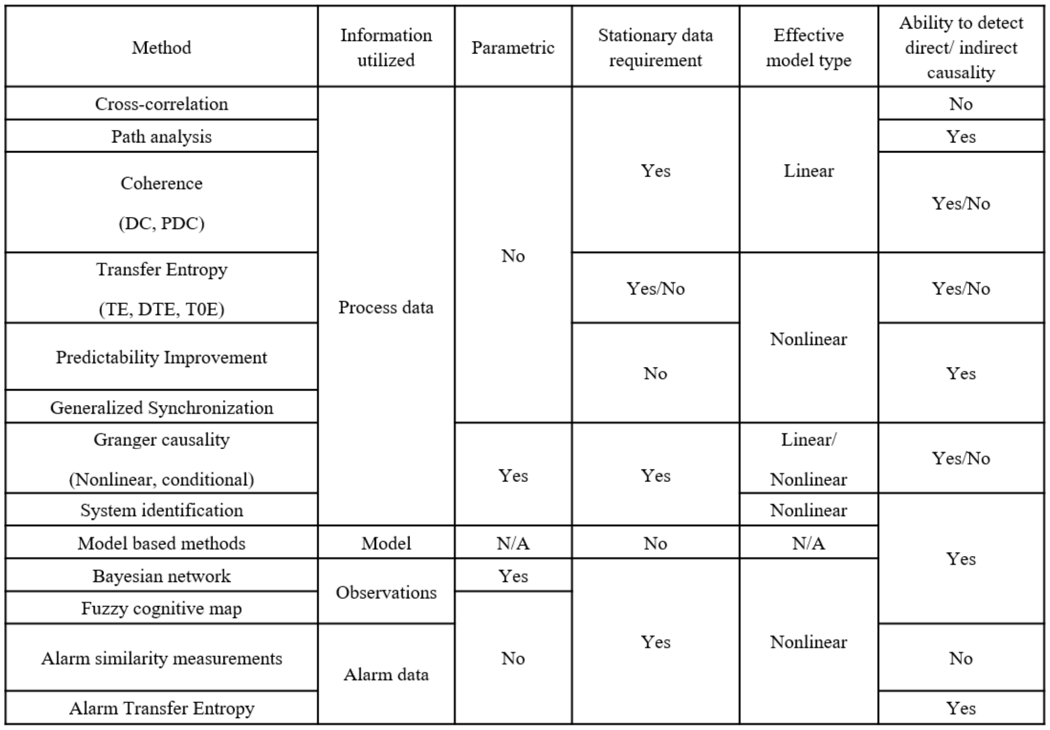

Alarm data based
Similarity measurements
Yang et al., 2013, Yang et al., 2012
Transfer entropy
Yu & Yang, 2015, Hu et al., 2017
Process knowledge based
Process data based
Non-parametric
Statistic measurements (cross-correlation, coherence)
Govindan et al., 2005, Baccala & Sameshima, 2001,
Transfer Entropies (TE, DTE, T0E)
Schreiber, 2000, Ping et al., 2014, Ping et al., 2013, Naghoosi et al., 2013
Predictability Improvement
Bauer et al., 2007, Freiwald et al., 1999
Generalized Synchronization
Kocarev & Parlitz, 1996, Arnhold et al., 1999
Parametric
Granger causality (nonlinear, conditional)
Granger, 1969, Ancona et al., 2004, Yuan & Qin, 2012
System identification approach
Marques et al., 2013
Model based
Signed Directed Graphs
Yang et al., 2012
Adjacency matrix
Jiang et al., 2009
Model based reasoning
Rich et al., 1987
Multilevel flow modeling
Peterson, 2000
Observation based
Bayesian network
Weidl et al., 2005
Fuzzy cognitive map
Kim & Lee, 1996, Rickard et al., 2015

Figure 1.5: Family tree of the causality analysis methods.

0-information, removing the assumption of data stationarity. In [24], methods for root cause diagnosis of plant-wide oscillations were summarized and compared, of which data-driven causality analysis as an important branch was reviewed. Approaches based on predictability improvement were proposed in [12, 32] for root cause analysis of plant-wide disturbances and processing neural interactions, respectively. In [54], authors proposed a generalized synchronization method to detect interdependencies; the method was then utilized in [9] for the interdependency detection of electroencephalography (EEG) signals. Granger causality and conditional Granger causality [35] were proposed by C.W.J Granger in 1969 for investigating causal relations of econometric models. Based on the framework of Granger causality, nonlinear Granger causality was developed using radial basis functions (RBF) for nonlinear models. The authors in [66] introduced a method for causality analysis from a system identification point of view. Model based causality analysis approaches based on signed directed graphs (SDG) [86], adjacency matrices [50], model based reasoning [74], and multilevel flow modeling [73] were also proposed. In [84], the authors built a Bayesian network, which was trained using the observations following certain actions, for root cause analysis and decision support of complex continuous processes. Fuzzy cognitive maps (FCM) based modeling was conducted towards causal relation analysis in [52, 75], where the authors investigated the implications from the FCM to find causal relationships.

Causality and correlation analysis based on alarm data were introduced as well. In [13], a time delay estimation method was proposed to analyze causality by taking the time delay between variables as an evidence of causality. Alarm correlation analysis methods were also proposed in [55, 70, 87, 89]; these methods could help restore the connections between alarm tags directly without using process data. A modified Transfer Entropy was introduced in [42], where the authors adapted the method in [25] to alarm data and used it to build causal maps between alarm variables. Figure 1.5 lists and categorizes the aforementioned causality analysis methods based on their characteristics. In addition, the table drawn in Figure 1.6 compares the methods based the utilized information, whether the method is parametric, whether the method requires stationary data as input, the effective model type, and the ability to detect direct/indirect causalities.



Method	Information utilized	Parametric	Stationary data requirement	Effective model type	Ability to detect direct/ indirect causality
Cross-correlation					No
Path analysis			Yes	Linear	Yes
Coherence (DC, PDC)					Yes/No
Transfer Entropy (TE, DTE, T0E)	Process data	No	Yes/No		Yes/No
Predictability Improvement			No	Nonlinear	Yes
Generalized Synchronization					
Granger causality (Nonlinear, conditional)		Yes	Yes	Linear/ Nonlinear	Yes/No
System identification				Nonlinear	
Model based methods	Model	N/A	No	N/A	Yes
Bayesian network	Observations	Yes			
Fuzzy cognitive map			Yes	Nonlinear	
Alarm similarity measurements	Alarm data	No			No
Alarm Transfer Entropy					Yes

Figure 1.6: Comparison of the methods listed in Figure 1.5.

1.3 Thesis Contributions

The major contributions in this thesis that distinguish it from other work are summarized as follows:

1. Provided guidelines and recommended procedures for conducting alarm flood analysis, which include: univariate alarm analysis, pattern mining of alarm floods, online pattern matching of alarm floods, and causality analysis for root cause detection in alarm floods.

2. Proposed an algorithm that extends the one in [18] to aligning multiple alarm flood sequences by introducing: (1) new scoring functions that are capable of describing the similarity of items in multiple time-stamped sequences, (2) a dynamic programming equation for the iterative calculation of similarity indices of multiple alarm flood sequences, and (3) back tracking and alignment generation procedures that can be used for multiple alarm flood sequences. With this proposed algorithm, we are able to find the optimal alignment of multiple alarm flood sequences and obtain the pattern for a selected alarm flood cluster, thus making up one of the missing steps in alarm flood analysis.

3. Proposed an online algorithm to provide early prediction of an incoming alarm flood by matching an online alarm sequence with a pattern database and conducting similarity calculation. It overcomes three main challenges in online time-stamped pattern matching: the partial information (future events are unknown in the online sequence), the high computational efficiency requirement, and the segmentation of the online alarm sequence. New elements introduced in this algorithm include chattering and sequence window filters, modified time distance and similarity measurements, a modified gap penalty, and an incremental dynamic programming strategy. Potentially, the proposed algorithm could serve as the state identification stage in applications of predictive alarming, online alarm attribute modification, and online alarm suppression.

4. Proposed an accelerated algorithm for pattern mining in multiple alarm floods. Unlike traditional methods which either cannot deal with multiple sequences with time stamps or suffer from high computational cost, the computational complexity of this proposed algorithm is reduced

significantly by introducing a generalized pairwise sequence alignment method and a progressive multiple sequence alignment approach. Two types of alignment refinement methods are developed to improve the alignment accuracy.

5. Conducted industrial case studies to show the effectiveness of causality analysis in terms of root cause analysis in alarm floods.

1.4 Thesis Outline

The remainder of the thesis is organized as follows.

We show the principal steps of the pattern mining algorithm for multiple alarm flood sequences using the case of three alarm flood sequences in Section 2.2; pseudo code of the algorithm for the three-sequence case will be shown in Section 2.2 as well. In Section 2.3, algorithms for aligning three and five alarm flood sequences will be tested on datasets of an actual petrochemical plant. Following that, some discussions concerning the algorithm and pattern selection are provided in Section 2.4. Finally, a summary is given in Section 2.5.

Problem formulation and the principles of the proposed online alarm flood pattern matching algorithm are introduced in Section 3.2. Efficiency and accuracy tests of the algorithm will be carried out on an industrial case study in Section 3.3. Then the computational complexity, advantages and drawbacks of the algorithm will be discussed in Section 3.4, followed by concluding remarks in Section 3.5.

Problem description and principle for the accelerated alarm flood pattern mining algorithm are given in Section 4.2. In Section 4.3, an industrial case study is provided to test both efficiency and accuracy of the proposed algorithm and comparisons are made with the exhaustive search approach in [62]. Detailed discussions on the accuracy, computational complexity, algorithm convergence, and some potential problems are carried out in Section 4.4, followed by the conclusions in Section 4.5.

Parameter robustness tests and an extended accuracy test are conducted using industrial data to reveal the insights on how the parameters affect the results of the proposed algorithms in Section 5.2. An industrial case study on the application of Granger causality towards the root cause analysis in alarm floods is carried out in Section 5.3.

Chapter 2

A Method for Pattern Mining in Multiple Alarm Flood Sequences*

2.1 Overview

As mentioned in Section 1.1, an alarm flood is a serious hazard for industrial processes; alarm management techniques such as delay timers and dead-bands often are incapable of suppressing alarm floods because of the existence of consequence alarms. However, this problem could be handled by alarm flood pattern analysis, which could help find the root cause, locate badly designed part in alarm systems and predict incoming alarm floods.

In this chapter, we propose a method for pattern mining in multiple alarm flood sequences by extending a time-stamp adapted Smith-Waterman algorithm to the case with multiple sequences, making up one of the missing steps in alarm flood analysis. The technique involves the following new elements: similarity scoring functions, a dynamic programming equation, a back-tracking procedure, and an alignment generation method. A dataset from an actual petrochemical plant has been used to test the effectiveness of the proposed algorithm.

The input data of the algorithm contains time stamp, tag name, and iden-

*A version of this chapter has been published as: Shiqi Lai and Tongwen Chen. A method for pattern mining in multiple alarm flood sequences. *Chemical Engineering Research and Design*, 117: 831–839, 2017, and a short version has been published as: Shiqi Lai and Tongwen Chen. Methodology and application of pattern mining in multiple alarm flood sequences. *Proceedings of 9th IFAC Symposium on Advanced Control of Chemical Processes* Whistler, Canada, pages 657–662, 2015.

tifier information. Table 2.1 shows one example of the simplified A&E log used in the algorithm. Since priority information is not considered in this study, such information has been eliminated.

Table 2.1: An alarm message log example

Time stamp	Tag name & identifier
2013-07-31 18:33	Tag1.PVLO
2013-07-31 18:33	Tag2.OFFNORM
2013-07-31 18:34	Tag8.BADPV
2013-07-31 18:38	Tag4.PVLO

2.2 Algorithm Principle

Our intended problem is to find the optimal alignment for a cluster of alarm floods so that based on this alignment an alarm sequence pattern can be easily found. However, the algorithm proposed in [18] is limited to pairwise alignment of flood sequences. In the following part, we will introduce an algorithm that extends the algorithm in [18] to the alignment of three alarm flood sequences. The idea for aligning more sequences will be similar, but more complex. Three new elements concerning similarity scoring functions, calculation of a similarity index cuboid and a back-tracking procedure will be introduced.

2.2.1 Problem Formulation

Consider the problem of searching for the optimal alignment of three sequences:

$$\begin{aligned}
A &= <(e_{11}, t_{11}), (e_{12}, t_{12}), ..., (e_{1m}, t_{1m}), ..., (e_{1M}, t_{1M})>, \\
B &= <(e_{21}, t_{21}), (e_{22}, t_{22}), ..., (e_{2n}, t_{2n}), ..., (e_{2N}, t_{2N})>, \\
C &= <(e_{31}, t_{31}), (e_{32}, t_{32}), ..., (e_{3o}, t_{3o}), ..., (e_{3O}, t_{3O})>,
\end{aligned}$$

where $e_{1m},\ e_{2n},\ e_{3o} \in \Lambda$, and $\Lambda = \{1, 2, ..., K\}$ is the set of different alarm types in the three sequences; t_{1m}, t_{2n} and t_{3o} are corresponding time stamps. The notation $< \cdot >$ is used to represent a sequence. The aim of the algorithm is to find the optimal local alignment (with deletions and inserted gaps '[ ]') of the three sequences that gives the highest alignment score, for example,

$$
\begin{aligned}
&< (e_{16}, t_{16}), (e_{17}, t_{17}), (e_{18}, t_{18}),\ [\ ]\ ,(e_{19}, t_{19}) >, \\
&< (e_{22}, t_{22}),\ [\ ]\ , (e_{23}, t_{23}), (e_{24}, t_{24})\ ,(e_{25}, t_{25}) >, \\
&<\ [\ ]\ , (e_{31}, t_{31}), (e_{32}, t_{32}),\ [\ ]\ ,(e_{33}, t_{33}) > .
\end{aligned}
$$

2.2.2 Time Distance and Weight Vectors

The original Simith-Waterman algorithm can help find optimal alignment between two sequences without time stamps. In order to adjust it to our problem, where time stamps are included in the sequences, a "time distance vector" and a "time weight vector" are defined, same as in [18]. A time distance vector for an alarm message (e_m, t_m) is defined as:

$$
\begin{aligned}
&\mathbf{d}_m = [d_m^1, d_m^2, ..., d_m^k, ..., d_m^K], \\
&d_m^k = \begin{cases} \min\limits_{1 \le i \le M} \{|t_m - t_i| : e_i = k\}, & \text{if the set is not empty} \\ \infty, & \text{otherwise.} \end{cases}
\end{aligned}
\tag{2.1}
$$

It carries the information of the shortest time distance from each different type of alarms in the sequence to alarm message (e_m, t_m). This distance will be set to infinity if the type of alarm does not appear in the sequence. A time weight vector for (e_m, e_m) is defined as:

$$
\begin{aligned}
\mathbf{w}_m = &\ [w_m^1, w_m^2, ..., w_m^k, ..., w_m^K] \\
= &\ [f(d_m^1), f(d_m^2), ..., f(d_m^k), ..., f(d_m^K)],
\end{aligned}
\tag{2.2}
$$

where $f(\cdot) : \mathbb{R} \to \mathbb{R}$ is a time weighting function. Two weighting functions are chosen as follows:

$$
f_1(x) = e^{-x^2/2\sigma^2}, \tag{2.3}
$$

$$
f_2(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{if } x \neq 0, \end{cases} \tag{2.4}
$$

where the first one is a scaled Gaussian function and the second is a simple binary selection function. $f_1(\cdot)$ helps convert the real-time distance to the value between 0 and 1. The longer the time distance is, the smaller the output value is. $f_2(\cdot)$ simply makes the time weight of the current alarm type $w_m^{e_m}$ to be 1, and set those of other alarm types to be 0. Both weighting functions work together to give the time weight vectors for the use of similarity score calculation. For example, we want to calculate the time distance and weight vectors of a sequence:

$$< (2,2),(1,3),(3,3.5),(1,5),(4,5.2) > .$$

In this case, $\Lambda = \{1,2,3,4\}$. For message $(2,2)$, time distance vector $\mathbf{d}_1$ is calculated as: $d_1^1 = 3-2 = 1$, $d_1^2 = 2-2 = 0$, $d_1^3 = 3.5-2 = 1.5$ and, $d_1^4 = 5.2-2 = 3.2$. All the time distance vectors for this sequence are:

$$[\mathbf{d}_1^T, \mathbf{d}_2^T, \mathbf{d}_3^T, \mathbf{d}_4^T, \mathbf{d}_5^T] = \begin{bmatrix} 1 & 0 & 0.5 & 0 & 2.2 \\ 0 & 1 & 1.5 & 3 & 3.2 \\ 1.5 & 0.5 & 0 & 1.5 & 1.7 \\ 3.2 & 2.2 & 1.7 & 0.2 & 0 \end{bmatrix}.$$

By applying the two weighting functions ($\sigma = 1$ for $f_1(\cdot)$) on time distance vectors, two groups of time weight vectors can be obtained:

$$[\mathbf{w}_1^T, \mathbf{w}_2^T, \mathbf{w}_3^T, \mathbf{w}_4^T, \mathbf{w}_5^T]_{f_1} = \begin{bmatrix} 0.61 & 1.00 & 0.88 & 1.00 & 0.09 \\ 1.00 & 0.61 & 0.32 & 0.01 & 0.01 \\ 0.32 & 0.88 & 1.00 & 0.32 & 0.24 \\ 0.01 & 0.09 & 0.24 & 0.98 & 1.00 \end{bmatrix}$$

$$[\mathbf{w}_1^T, \mathbf{w}_2^T, \mathbf{w}_3^T, \mathbf{w}_4^T, \mathbf{w}_5^T]_{f_2} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

2.2.3 Calculation of Similarity Scores

The time distance and weight vectors help adjust the Smith-Waterman algorithm to handle sequences with time stamps. In order to further extend it to multiple sequence alignment to address our problem, two kinds of scoring functions are defined (for the case of three sequences).

The *2-way similarity scoring function* is

$$\begin{aligned} & S((e_a,t_a),(e_b,t_b)) \\ = \ & \max\{S_0((e_a,t_a),(e_b,t_b)), \\ & \quad S_0((e_b,t_b),(e_a,t_a))\} \times (1-\mu) + \mu, \end{aligned} \tag{2.5}$$

where,

$$S_0((e_a,t_a),(e_b,t_b)) = \max_{1 \le k \le K} [w_a^k \times w_b^k]. \tag{2.6}$$

(e_a,t_a) and (e_b,t_b) are alarm messages from sequences A and B. Respectively, $f_1(\cdot)$ and $f_2(\cdot)$ are used to calculate the time weight vectors of the alarm messages (e_a, t_a) and (e_b, t_b). Thus, the commutative laws do not hold for the calculations of S_0, which means $S_0((e_a, t_a), (e_b, t_b))$ and $S_0((e_b, t_b), (e_a, t_a))$ are not guaranteed to be equal. For this reason, we choose the larger one of S_0 during the calculation of $S((e_a, t_a), (e_b, t_b))$. Because the range of elements in a time weight vector is $[0, 1]$, thus $S_0((e_a, t_a), (e_b, t_b)) \in [0, 1]$, and $S((e_a, t_a), (e_b, t_b)) \in [\mu, 1]$. The negative parameter μ is the miss-match penalty.

The *3-way similarity scoring function* is

$$\begin{aligned} &S((e_a, t_a), (e_b, t_b), (e_c, t_c)) \\ &= S_0((e_a, t_a), (e_b, t_b), (e_c, t_c)) \times (1 - 2\mu) + 2\mu, \end{aligned} \tag{2.7}$$

where

$$\begin{aligned} &S_0((e_a, t_a), (e_b, t_b), (e_c, t_c)) \\ &= \max \left\{ \frac{S_0((e_b, t_b), (e_a, t_a)) + S_0((e_c, t_c), (e_a, t_a))}{2}, \right. \\ &\qquad \frac{S_0((e_a, t_a), (e_b, t_b)) + S_0((e_c, t_c), (e_b, t_b))}{2}, \\ &\qquad \left. \frac{S_0((e_a, t_a), (e_c, t_c)) + S_0((e_b, t_b), (e_c, t_c))}{2} \right\}. \end{aligned} \tag{2.8}$$

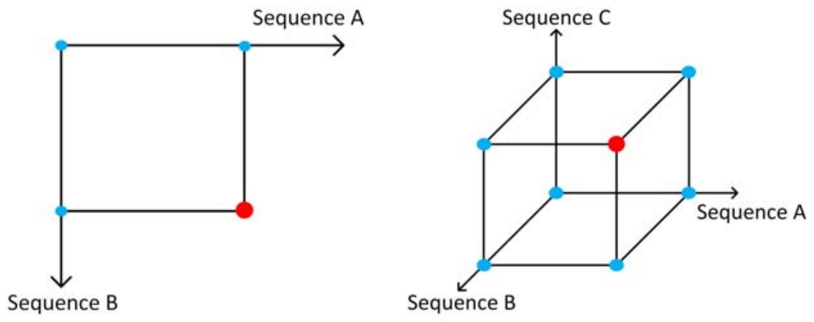
(e_a, t_a), (e_b, t_b) and (e_c, t_c) are alarm messages from sequences A, B, and C. The 3-way similarity score is approximated by the averages of 2-way similarity scores. Commutative laws applies to $S_0((e_a, t_a), (e_b, t_b), (e_c, t_c))$ since only the maximum value of the averages of the 2-way scores is used for its calculation. Note that $S_0((e_a, t_a), (e_b, t_b), (e_c, t_c)) \in [0, 1]$; thus $S((e_a, t_a), (e_b, t_b), (e_c, t_c)) \in [2\mu, 1]$.

2.2.4 Dynamic Programming

In the case of two alarm flood sequences, shown in Figure 2.1a, the score (red dot) in each cell of the similarity index matrix is obtained from three candidates (small blue dots). As each sequence holds one dimension in the alignment space, there is a similarity index cuboid instead of a matrix if the number of sequences to be aligned grows from two to three. Thus, for the case of three alarm flood sequences, as shown in Figure 2.1b, in each step of dynamic programming, the score (big red dot) is obtained from seven candidates (small blue dots). Equation (2.9) gives the way to calculate similarity index during each step:

$$
\begin{aligned}
& H_{m+1,n+1,o+1} \\
&= \max_{1\leq i\leq M, 1\leq j\leq N, 1\leq g\leq O} (I(A_{i:m}, B_{j:n}, C_{g:o}), 0) \\
&= \max\{H_{m+1,n+1,o} + 2\delta, H_{m+1,n,o+1} + 2\delta, H_{m,n+1,o+1} + 2\delta, \\
&\qquad H_{m,n,o+1} + \delta + S((e_{m+1}, t_{m+1}), (e_{n+1}, t_{n+1})), \\
&\qquad H_{m,n+1,o} + \delta + S((e_{m+1}, t_{m+1}), (e_{o+1}, t_{o+1})), \\
&\qquad H_{m+1,n,o} + \delta + S((e_{n+1}, t_{n+1}), (e_{o+1}, t_{o+1})), \\
&\qquad H_{m,n,o} + S((e_{m+1}, t_{m+1}), (e_{n+1}, t_{n+1}), (e_{o+1}, t_{o+1})), \\
&\qquad 0\},
\end{aligned}
\tag{2.9}
$$

where $I(A_{i:m}, B_{j:n}, C_{g:o})$ is the similarity index for the ternary $(A_{i:m}, B_{j:n}, C_{g:o})$, and δ is a negative parameter for gap penalty. Initial values such as $H_{0,n,o}$, $H_{m,0,0}$ and $H_{0,0,0}$ are all set to be 0. The aim of the algorithm can also be interpreted as to find the segment ternary that has the highest similarity index.



(a) Two sequences case (b) Three sequences case

Figure 2.1: Illustration of similarity score calculation for two and three sequences cases

Back tracking is carried out on the three-dimensional space as well, as shown in Figure 2.2. First, the position of the largest similarity index is found (noted by big red dot on Figure 2.2). Then it follows the arrow which points to the ancestor from whom the current score is obtained, until the path reaches the place with a zero similarity index.

Finally, one forward pass of the tracking path gives us the optimal alignment. Start from the beginning point, on each step, the dimension on which there is an increase in subscript will keep the corresponding alarm message. If the subscript on one dimension doesn't grow, then fill this position with a gap.

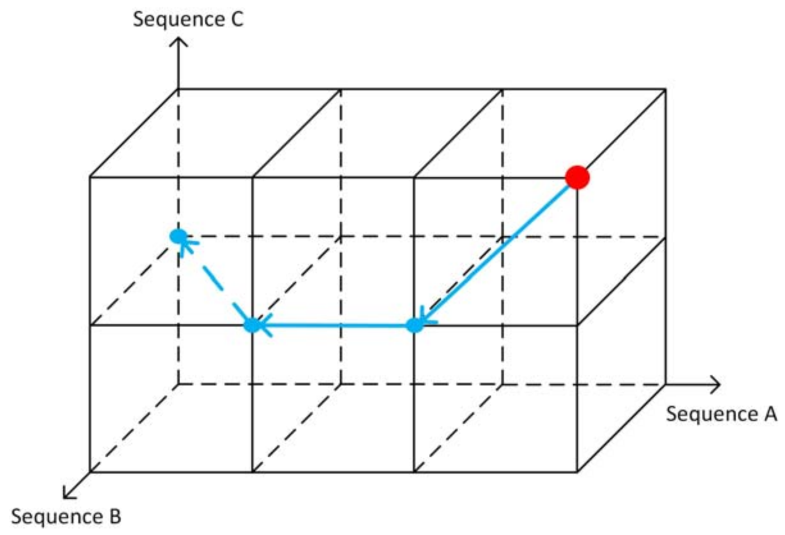


Figure 2.2: An example of back tracking of a case with three alarm flood sequences

2.2.5 An Illustration Example

Here we provide a simple example of aligning three alarm flood sequences to demonstrate the calculation of similarity scores and the dynamic programming procedures. Pseudo codes are presented in Algorithm 1. For cases with more than three sequences, the idea will be the same. Note that the back tracking and alignment generation procedures have been combined together, so the description of alignment generation part is somewhat different from the one that has been introduced.

Choose parameters of the algorithm to be $\sigma = 0.5$, $\delta = -0.4$ and $\mu = -1$. Consider three alarm flood sequences

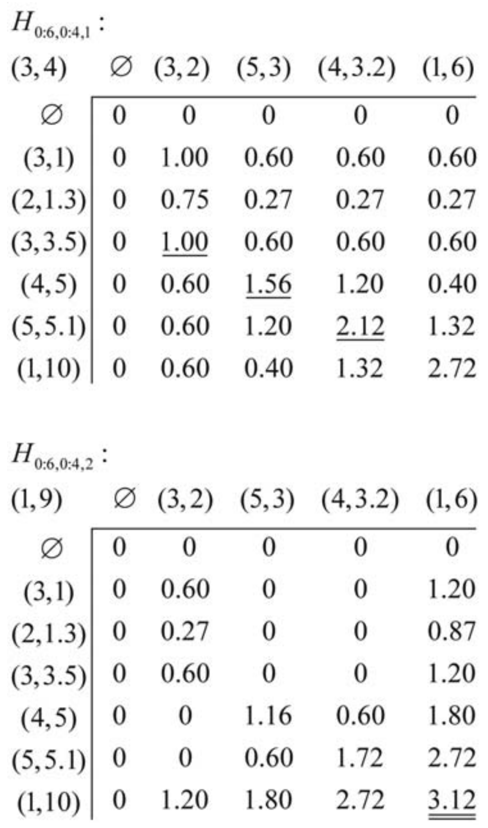
$$\begin{aligned} A =& < (3,1),(2,1.3),(3,3.5),(4,5),(5,5.1),(1,10) > \\ B =& < (3,2),(5,3),(4,3.2),(1,6) > \\ C =& < (3,4),(1,9) > \end{aligned}$$

with $\Sigma = \{1,2,3,4,5\}$. We first show the calculation of 2-way and 3-way similarity scores for the ternary made up of the first messages in the three sequences, namely, $(3,1)$, $(3,2)$ and $(3,4)$. Time weight vectors for $(3,1)$ in sequence A are $[0,0.84,1,0,0]_{f_1}$ and $[0,0,1,0,0]_{f_2}$; similarly, those for $(3,2)$ in sequence B are $[0,0,1,0.05,0.14]_{f_1}$ and $[0,0,1,0,0]_{f_2}$, and those for $(3,4)$ in sequence C are $[0,0,1,0,0]_{f_1}$ and $[0,0,1,0,0]_{f_2}$. $S_0((3,1),(3,2)) = \max\{0 \times$

$0, 0.84 \times 0, 1 \times 1, 0, 0 \times 0\} = 1$ and $S_0((3,2),(3,1)) = \max\{0 \times 0, 0 \times 0, 1 \times 1, 0.05, 0.14 \times 0\} = 1$, thus $S((3,1),(3,2)) = \max\{1,1\} \times (1-\mu) + \mu = 1$. Similarly we can obtain $S((3,1),(3,4)) = 1$ and $S((3,2),(3,4)) = 1$. With all the 2-way scores, the value of the 3-way score $S((3,1),(3,2),(3,4))$ can be calculated to be 1. We can also manually check this result. Since the alarm types of the three messages are the same, they should get a matching score. After similarity scores are obtained, dynamic programming can be carried out based on equation (2.9). Two slices of the obtained similarity cuboid is shown in Figure 2.3. Since the values in the slice $H_{0:6,0:4,0}$ are all 0, they are not shown in the figure in order to save space.

Algorithm 1: Aligning three alarm flood sequences

```
input  : 3 alarm flood sequences, variance of Gaussian function $\sigma$, gap
         penalty $\delta$ and miss-match penalty $\mu$
output: Optimal local alignment of three alarm flood sequences
1  begin
2      Obtain a time distance vector $\mathbf{d}_m$ for each alarm message in every
         sequence.
3      Then apply time weighting functions $f_1(\cdot)$ and $f_2(\cdot)$ on each time
         distance vector to get the corresponding time weight vectors $\mathbf{w}_m$.
4      for $m \leftarrow 1$ to $M$ do
5          for $n \leftarrow 1$ to $N$ do
6              for $o \leftarrow 1$ to $O$ do
7                  Calculate all the 2-way and 3-way similarity scores.
8                  Then obtain the similarity index $H_{m,n,o}$.
9                  Record the corresponding ancestor's location into
                     $Ptr_{m,n,o}$.
10     Find the maximum value in the cuboid of similarity index $H_{max}$ and
         its indices $m_{max}$, $n_{max}$ and $o_{max}$.
11     $m \longleftarrow m_{max}$;
12     $n \longleftarrow n_{max}$;
13     $o \longleftarrow o_{max}$;
14     repeat
15         $m, n, o \longleftarrow Ptr_{m,n,o}$;
16         The dimension on which there is a decrease in subscript will
             keep the corresponding alarm message on that alignment
             sequence. If the subscript for that dimension remains the same,
             then fill this position with a gap on that alignment sequence.
17     until $H_{m,n,o} = 0$;
18     return alignment result of the three sequences
```



$H_{0:6,0:4,1}$:

(3,4)	∅	(3,2)	(5,3)	(4,3.2)	(1,6)
∅	0	0	0	0	0
(3,1)	0	1.00	0.60	0.60	0.60
(2,1.3)	0	0.75	0.27	0.27	0.27
(3,3.5)	0	$\underline{1.00}$	0.60	0.60	0.60
(4,5)	0	0.60	$\underline{1.56}$	1.20	0.40
(5,5.1)	0	0.60	1.20	$\underline{2.12}$	1.32
(1,10)	0	0.60	0.40	1.32	2.72

$H_{0:6,0:4,2}$:

(1,9)	∅	(3,2)	(5,3)	(4,3.2)	(1,6)
∅	0	0	0	0	0
(3,1)	0	0.60	0	0	1.20
(2,1.3)	0	0.27	0	0	0.87
(3,3.5)	0	0.60	0	0	1.20
(4,5)	0	0	1.16	0.60	1.80
(5,5.1)	0	0	0.60	1.72	2.72
(1,10)	0	1.20	1.80	2.72	$\underline{\underline{3.12}}$

Figure 2.3: Similarity index cuboid and back tracking of the example

Next, we search for the maximum score and its position (doubly underlined in Figure 2.3); then start backing tracking from there. Since $H_{6,4,2}$, with score 3.12, is generated from $H_{5,3,1} + S((1,10),(1,6),(1,9))$, score 2.12 in the first table of Figure 2.3 is selected into the back tracking path. The rest of the path (underlined in Figure 2.3) can be selected in the same way. Finally, we obtain the full path: $\{H_{2,0,0}, H_{3,1,1}, H_{4,2,1}, H_{5,3,1}, H_{6,4,2}\}$.

Based on the tracking path, start from the first transition from $H_{2,0,0}$ to $H_{3,1,1}$, since there are increases on all three subscripts, we keep the corresponding messages: $(3,3.5)$, $(3,2)$ and $(3,4)$ on the three sequences. During the transition from $H_{3,1,1}$ to $H_{4,2,1}$: since the subscript for dimension 3 doesn't grow, we leave a gap on sequence C and keep alarm messages: (4,5) and (5,3) on sequence A and B. Keeping on doing so, we obtain the alignment output:

$$< (3,3.5),(4,5),(5,5.1),(1,10) >$$
$$< \ (3,2),\ (5,3),(4,3.2),\ (1,6) >$$
$$< \ (3,4),\ [],\ [],\ (1,9) > .$$

Even though the order of alarm message $(4, 5)$ and $(5, 5.1)$ in sequence A is different from $(5, 3)$ and $(4, 3.2)$ in sequence B, they are still aligned together because the two types of alarms are both raised closely in the two sequences.

2.3 Industrial Case Study

A dataset from an actual chemical process has been used to test the effectiveness of the proposed algorithm. Equipment used in the process include pumps, compressors, furnaces, and filters. 300 seconds' off-delay timers were applied to remove chattering alarms. The reason for using off-delay rather than on-delay is because it does not introduce any delay when raising an alarm. Also, such a long time was chosen for off-delay-timers in order to prevent obtaining the patterns formed by repeating alarms. 359 alarm flood sequences were extracted based on the ISA standard, which is 10 alarms per 10 minutes. General descriptions of the dataset with off-delay timers applied and the extracted alarm floods can be found in Table 2.2.

Table 2.2: General descriptions of the dataset

Description	Number
Total time period	336 days
Total number of tags	1502
Total number of alarms	109393
Average alarm rate	14/h
Highest peak alarm rate	334/10 min
Number of alarm floods	359
Average length of alarm floods	39

The pairwise pattern matching algorithm proposed in [18] was applied to calculate pairwise similarity scores of the extracted alarm floods. Clustering based on the obtained similarity scores was carried out thereafter; result is shown in Figure 2.4. Both horizontal and vertical axes of the chessboard in the figure are formed by the indices of alarm floods. Each dot in the figure represents the similarity score between a corresponding pair of alarm flood sequences; the darker the color of a dot is, the higher the similarity score is. Based on the clustering result, pattern mining could be carried out manually by comparing the corresponding sequences; but it would be time consuming and the accuracy would not be guaranteed. In the following part, we will carry out the proposed algorithm respectively on two clusters of three and five alarm flood sequences.

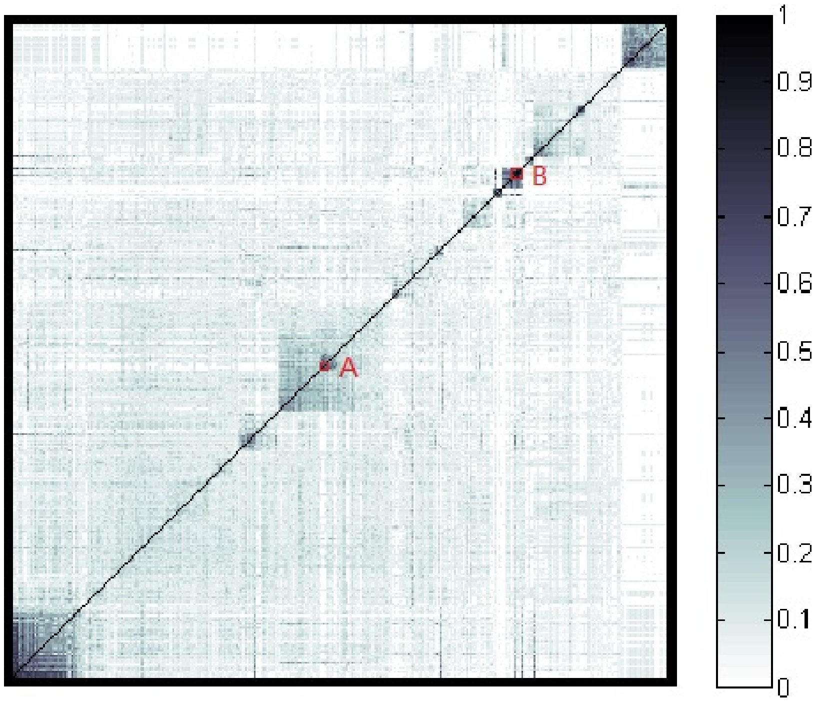


Figure 2.4: Clustering result of extracted alarm floods

2.3.1 Pattern Mining in Three Alarm Flood Sequences

Three alarm floods, of lengths 13, 16, and 12 respectively, in cluster A, shown in Figure 4.8, were selected to be the testing sequences. The proposed algorithm for pattern mining in three sequences was applied with parameters set as: $\sigma = 0.2$, $\mu = -1$, and $\delta = -1$. On a 64-bit Windows PC with Intel(R) Core(TM) i7-4770 3.40GHz CPU and 24.0 GB memory, the algorithm took only 0.9 seconds to finish and the result is shown in Table 2.3.

Even though the cluster is formed by short alarm floods, manually comparing the three sequences would still be time consuming. However, using the proposed algorithm, the alignments can be obtained accurately and almost immediately. Moreover, by taking a closer look at the time stamps of the alignments, one can notice that these three alarm floods were raised during Oct, Dec, and Feb respectively, which means the pattern found from those alignments could be a regular one. Moreover, priorities (not listed here in order to save space) of many alarms in the three sequences had been configured as "High". Thus, the pattern obtained from the alignments could be valuable for predictive alarming and operator training.

Table 2.3: Alignment result of the three sequences in cluster A

Flood 142	Flood 143	Flood 320
Tag593.PVHH 27-Oct-2013 16:58:28	Tag662.PVHH 26-Feb-2014 00:18:12	Tag593.PVHH 16-Dec-2013 21:30:16
Tag662.PVHH 27-Oct-2013 16:58:29	Tag593.PVHH 26-Feb-2014 00:18:14	Tag662.PVHH 16-Dec-2013 21:30:18
Tag598.PVHH 27-Oct-2013 16:58:31	Tag598.PVHH 26-Feb-2014 00:18:18	Tag598.PVHH 16-Dec-2013 21:30:21
[] []	[] []	Tag163.OFFLINE 16-Dec-2013 21:30:27
Tag71.PVLO 27-Oct-2013 16:58:45	Tag71.PVLO 26-Feb-2014 00:18:30	Tag71.PVLO 16-Dec-2013 21:30:33
Tag403.NORM 27-Oct-2013 16:58:56	[] []	Tag403.NORM 16-Dec-2013 21:30:36
Tag407.NORM 27-Oct-2013 16:59:00	Tag407.NORM 26-Feb-2014 00:19:08	Tag407.NORM 16-Dec-2013 21:30:42
Tag408.NORM 27-Oct-2013 16:59:02	[] []	Tag408.NORM 16-Dec-2013 21:30:49
Tag427.OFFLINE 27-Oct-2013 16:59:09	Tag427.OFFLINE 26-Feb-2014 00:19:17	Tag427.OFFLINE 16-Dec-2013 21:30:51
Tag1457.OFFNORM 27-Oct-2013 17:01:09	Tag1457.OFFNORM 26-Feb-2014 00:21:48	[] []

2.3.2 Pattern Mining in Five Alarm Flood Sequences

Setting the parameters to be: $\sigma = 0.2$, $\mu = -1$, and $\delta = -1$, the proposed algorithm for pattern mining in five sequences was applied on cluster B, which was formed by five alarm flood sequences. Lengths of the five sequences are 20, 49, 122, 25, and 56 respectively. On a 64-bit Windows PC with Intel(R) Core(TM) i7-4770 3.40GHz CPU and 24.0 GB memory, the algorithm took 982 seconds to complete and the result is shown in Table 2.4.

Table 2.4: Alignment result of the five sequences in cluster B

Flood 66	Flood 69	Flood 65	Flood 60	Flood 53
Tag255.OFFLINE 01-Aug-2013 16:49:28	Tag267.OFFLINE 03-Aug-2013 22:28:26	Tag123.PVLO 01-Aug-2013 16:24:33	Tag240.OFFLINE 31-Jul-2013 20:23:35	Tag348.OFFLINE 30-Jul-2013 21:08:34
Tag240.OFFLINE 01-Aug-2013 16:49:28	Tag236.OFFLINE 03-Aug-2013 22:28:26	Tag264.OFFLINE 01-Aug-2013 16:24:33	Tag122.PVLO 31-Jul-2013 20:23:35	Tag271.OFFLINE 30-Jul-2013 21:08:34
Tag236.OFFLINE 01-Aug-2013 16:49:28	Tag240.OFFLINE 03-Aug-2013 22:28:26	Tag240.OFFLINE 01-Aug-2013 16:24:33	Tag267.OFFLINE 31-Jul-2013 20:23:35	Tag251.OFFLINE 30-Jul-2013 21:08:34
Tag271.OFFLINE 01-Aug-2013 16:49:28	Tag262.OFFLINE 03-Aug-2013 22:28:26	Tag255.OFFLINE 01-Aug-2013 16:24:33	Tag271.OFFLINE 31-Jul-2013 20:23:35	Tag262.OFFLINE 30-Jul-2013 21:08:34
Tag267.OFFLINE 01-Aug-2013 16:49:28	Tag264.OFFLINE 03-Aug-2013 22:28:26	Tag348.OFFLINE 01-Aug-2013 16:24:33	Tag348.OFFLINE 31-Jul-2013 20:23:35	Tag264.OFFLINE 30-Jul-2013 21:08:34
Tag264.OFFLINE 01-Aug-2013 16:49:28	Tag271.OFFLINE 03-Aug-2013 22:28:26	Tag267.OFFLINE 01-Aug-2013 16:24:33	Tag262.OFFLINE 31-Jul-2013 20:23:35	Tag236.OFFLINE 30-Jul-2013 21:08:34
Tag262.OFFLINE 01-Aug-2013 16:49:28	Tag348.OFFLINE 03-Aug-2013 22:28:26	Tag236.OFFLINE 01-Aug-2013 16:24:33	Tag264.OFFLINE 31-Jul-2013 20:23:35	Tag240.OFFLINE 30-Jul-2013 21:08:34
Tag251.OFFLINE 01-Aug-2013 16:49:28	Tag255.OFFLINE 03-Aug-2013 22:28:26	Tag262.OFFLINE 01-Aug-2013 16:24:33	Tag251.OFFLINE 31-Jul-2013 20:23:35	Tag267.OFFLINE 30-Jul-2013 21:08:34
Tag348.OFFLINE 01-Aug-2013 16:49:28	Tag251.OFFLINE 03-Aug-2013 22:28:26	Tag251.OFFLINE 01-Aug-2013 16:24:33	Tag255.OFFLINE 31-Jul-2013 20:23:35	Tag255.OFFLINE 30-Jul-2013 21:08:34

When the number of alarm floods increases, manual pattern mining becomes almost impossible. However, the proposed algorithm can still output the alignment result in an acceptable short period of time, with guaranteed accuracy. Moreover, notice the orders of those alarms in the three sequences are not the same; this is because those alarms were raised closely, so the algorithm made their orders vague and allowed swaps in alignments, in order to reduce the influence of process delays on the alignment result. This kind of alignment can hardly be achieved even by an expert. In addition, time stamps of the five sequences reveal these alarm floods were raised only within a few days. The priorities of the alarms contained in the alignments had been configured as "High"; thus, the pattern could be caused by some ill-functional parts occurred during those days, and the operators could be overwhelmed during these periods since all the alarms in the floods were raised in almost the same time. If this pattern could be detected online, more specific and in-time operations regarding this type of ill-functionality could be carried out.

2.4 Discussions

The proposed algorithm has a computational complexity of $O(2\sum_{k=1}^{n} L_k \times \prod_{k=1}^{n} L_k)$, where L_k are the lengths of the flood sequences to be aligned; the second part $\prod_{k=1}^{n} L_k$ comes from the total steps of dynamic programming; and the first part $2\sum_{k=1}^{n} L_k$ is the complexity of similarity score calculation during each step. Thus, the computational time of the algorithm increases quickly with the number and lengths of the flood sequences to be aligned. Fortunately, this part is to be done offline; the low computational efficiency only influences user experiences (waiting time), but it does present a limitation of the method.

One potential way to reduce computational burden is to first build a dendrogram based on the pairwise similarity scores of the alarm flood sequences; then, follow the dendrogram from the leaves to the root, align the alarm flood sequences progressively. In this way, since not all the alignment combinations are checked, the computational intensity could be reduced. However, one possible drawback brought by this pruned alignment procedure could be the relatively low alignment accuracy.

Another problem comes from implementation. As the number of sequences to be aligned grows, there are be more types of similarity scores; and more cases to be considered during each step in dynamic programming as well.

Moreover, the three parameters used in the algorithm, the variance of the scaled Gaussian function, the gap and miss-match penalties need to be tuned. As mentioned in [18], users need to adjust those parameters based on their requirements. The variance of the scaled Gaussian function will influence the time span within which the algorithm treats the orders of the alarms to be less important. The bigger this value is, the broader the time span is. For example, if this value would go to infinity then the algorithm would treat all the alarms to be raised simultaneously and ignore the time stamp information. On the contrary, if it was set to be zero, the algorithm would require the orders of alarm messages to be exactly the same for each aligned sequence. Gap and miss-match penalties determine the tolerance of the algorithm to gaps and mismatch terms. Usually the alignment would contain more alarm messages if we increase these two parameters; but we would expect more gaps and miss-match terms in the alignment. The flexibilities make the algorithm more adjustable to meet different requirements, but at the same time make it hard for users to tune the algorithm.

One advantage of the proposed algorithm is that it can provide exact optimal solutions through dynamic programming, since there are not any approximate steps, which have been adopted in progressively pairwise approaches in the bioinformatics area, as described in the introduction section. This advantage is quite valuable for our problem because alarms play a critical role in industrial process monitoring and we want to keep this information as accurate as possible. Moreover, since this part is done offline, accuracy should be given the first priority.

Regarding to the pattern selection, our way is somewhat different from the one used in the bioinformatics area, where the consensus part of the alignment is usually selected as the pattern. However, in our case, we tolerant some extent of order changes in alignment: some valuable patterns could be left out if we only select the consensus part. For this reason, we usually choose the alignment sequence that contains the least number of gaps as the pattern for the whole cluster. Experts may be involved in this part to help the pattern selection. Another difference is that for our case, time stamps should be included in the selected patterns as well, since they carry the information that describes relative positions of alarm messages in the time domain.

2.5 Summary

Univariate and consequence alarms are two big components of industrial alarm floods. While delay timers and dead-bands are effective for removing univariate alarms such as chattering alarms and fleeting alarms, they are not as powerful in front of consequence alarms, which are usually caused by correlations and fault propagations in processes. However, consequence alarms can be identified by applying pattern analysis on alarm floods.

In this chapter, an algorithm to find the optimal alignment of multiple alarm flood sequences and obtained a common pattern of them thereafter is proposed. The algorithm is an extension of the algorithm in [18] to the case of multiple sequences. It makes up one of the missing steps in alarm flood analysis. A lot of analysis and management such as root cause analysis, dynamic alarm suppression, and analysis on potential problems in alarm systems can be carried out based on the results of this proposed algorithm.

Chapter 3

Online Pattern Matching and Prediction of Oncoming Alarm Floods*

3.1 Overview

There are two steps to reduce alarm floods. The first is to remove univariate alarms by applying techniques such as delay timers and dead-bands. Design techniques for delay timers and dead-bands have been introduced in [57], [58], and [68]. The second step to reduce alarm floods is to study consequence alarms (alarms that have correlation with each other) because they form another important part of an alarm flood. For example, when the compressor trips, alarms for low speed, low oil pressure, high suction pressure, low discharge pressure, and low amps are usually raised successively as a pattern. This type of consequence alarms are triggered by one or more faults and appear frequently in the alarm and event logs. Since the consequence alarms are annunciated successively, they can easily increase the alarm rate and trigger alarm floods. Pattern mining algorithms are commonly used among the offline methods to find pattern sequences in alarm floods. Then the patterns can be used in root cause analysis (the compressor trip as the root cause in the example), locating bad configurations in alarm systems (e.g., redundant measurement on the same process), and operator trainings (train operators to deal with the consequence alarms triggered by frequent faults). Advanced methods

*A version of this chapter has been published as: Shiqi Lai, Fan Yang, Tongwen Chen. Online pattern matching and prediction of oncoming alarm floods. *Journal of Process Control*, 56: 69-78, 2017 (in press).

include predictive alarming to warn the operator of oncoming alarms, online alarm attribute modification, and online alarm suppression, as mentioned but not fully developed in the EEMUA standard [29]. These techniques, however, all depend on matching the online alarm sequence with patterns previously found from historical alarm data.

In this chapter, we propose an algorithm that can monitor the online alarm systems for certain alarm sequence patterns, which potentially can be used to provide operator warnings of oncoming alarms and their root causes to give operators more time to take actions (predictive alarming), to automatically suppress oncoming alarms in the pattern sequence that do not provide valuable information for operator decision making (online alarm suppression), and to automatically modify alarm attributes when the pattern is operating mode related (online alarm attributes modification). Our main contributions are:

1. proposed a chattering window filter to eliminate chattering alarms online;

2. introduced a sequence window filter to segment the online alarm sequence, avoiding unnecessary matches between the online sequence and the patterns;

3. modified the time distance measurements proposed in [18], enabling them to be calculated incrementally;

4. modified the gap penalty calculation method proposed in [18] by relating it with time distance measurements in order to reduce the influences of disturbance alarms on the matching results;

5. introduced an incremental strategy to compute the time distance and dynamic programming matrices while matching the sequences, thus dividing the whole computational burden into small pieces that can be treated quickly on each individual step when a new alarm is raised in the online sequence.

These techniques allow the computational cost of the algorithm to increase linearly with the number of patterns in the database, the lengths of patterns, and the length of the online sequence, making the algorithm applicable to large scale plants.

3.2 Algorithm Principle

Our intended problem is to match the online alarm sequence with the patterns in the database and obtain their similarity scores; then, based on these scores, identify whether the online sequence is similar to any of the patterns in the database and thereafter predict the incoming alarm flood if a matching can be found. Moreover, since the algorithm works online, efficiency becomes a critical requirement. In the following part, we will introduce the proposed algorithm that deals with this problem. The flowchart and the pseudo code of the proposed algorithm will be given later.

3.2.1 Problem Formulation

Consider the pattern database with the following sequences:

$$\begin{aligned} P_1 &= \ <(e_{11}, t_{11}), (e_{12}, t_{12}), ..., (e_{1m}, t_{1m}), ..., (e_{1M}, t_{1M})>, \\ P_2 &= \ <(e_{21}, t_{21}), (e_{22}, t_{22}), ..., (e_{2n}, t_{2n}), ..., (e_{2N}, t_{2N})>, \\ &\ \ . \\ &\ \ . \\ P_J &= \ <(e_{J1}, t_{J1}), (e_{J2}, t_{J2}), ..., (e_{Jo}, t_{Jo}), ..., (e_{JO}, t_{JO})>, \end{aligned}$$

where e_{1m}, e_{2n},..., $e_{Jo} \in \Lambda$, the set of all alarm types; t_{1m}, t_{2n},..., t_{Jo} are the time stamps. Given an online sequence formed by a series of raised alarm messages, the aim of the algorithm includes:

1. remove chattering alarms from the online sequence;
2. find the optimal alignment scores of the online sequence with each of the patterns;
3. identify whether the online sequence is similar to any of the patterns based on the alignment scores;
4. make the prediction of incoming alarms if the online sequence is identified as similar to a pattern.

The first functionality removes chattering alarms, which usually act as disturbances in an alignment, to increase the accuracy of the measured similarities between the online sequence and the patterns. The similarities are measured by aligning the sequences and finding their optimal alignment scores, as the second functionality does. The third and fourth functionalities work with the results obtained from the first two.

3.2.2 Chattering Window Filter (CWF)

In order to eliminate chattering alarms from an online sequence, a chattering window filter is applied. It is a filter based on the record of alarm types that have been raised within the past time span γ. Whenever a new alarm is raised, the record in the window will be updated first before it is used to identify chattering alarms. During the update, old alarm types raised beyond the time span γ will be deleted from the record. Then, the algorithm checks whether the newly raised alarm is a chattering one by comparing its alarm type with the record. If it is contained in the record, the new alarm will be identified as a chattering alarm; if not, it will be included into the record for the next iteration.

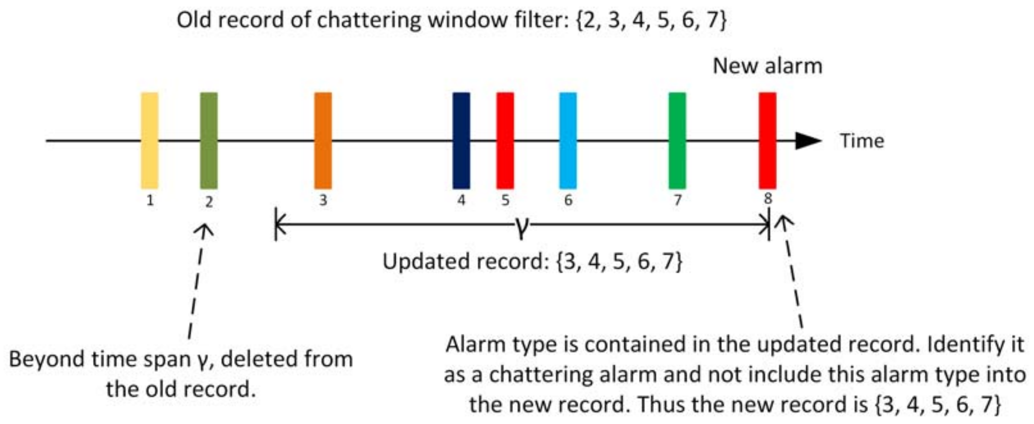

Old record of chattering window filter: {2, 3, 4, 5, 6, 7}
New alarm
Time
1
2
3
4
5
6
7
8
γ
Updated record: {3, 4, 5, 6, 7}
Beyond time span γ, deleted from the old record.
Alarm type is contained in the updated record. Identify it as a chattering alarm and not include this alarm type into the new record. Thus the new record is {3, 4, 5, 6, 7}

Figure 3.1: An illustrative example of chattering window filter

Figure 3.1 gives an illustrative example of how the chattering window filter works. Different colors represent different alarm types. In this example, alarms 1-8 are raised in the online sequence. Alarm 8 is the latest one and is of the same type as alarm 5. The previous record is $\{2, 3, 4, 5, 6, 7\}$. When alarm 8 is raised, the record is firstly updated by deleting alarm 2 since it is beyond the time span γ. Then, because the type of alarm 8 is contained in the updated record, $\{3, 4, 5, 6, 7\}$, the algorithm identifies alarm 8 as a chattering alarm and does not include its alarm type into the new record.

3.2.3 Sequence Window Filter (SWF)

The sequence window filter is proposed to reduce the computational burden by specifically segmenting the online sequence for the matching with each pattern. The number of sequence window filters equals the number of patterns in the database. Respectively, the start and end points of each segmentation are based on the pre-matching and the dynamic programming matrix (details are explained in the following paragraphs of this subsection). The matching of the online sequence with each pattern will be approximated by the matching between each segmentation (the record of corresponding sequence window filter) and the corresponding pattern.

To determine the start point of a segmentation, pre-matching is carried out. During the pre-matching, the algorithm checks whether the type of the newly raised alarm is contained in the corresponding pattern sequence; if it is (the new alarm passes pre-matching), then the segmentation begins (the corresponding sequence window filter starts to include incoming alarms into its record); if it is not (the new alarm fails pre-matching), no segmentation will be carried out (the record of the corresponding sequence window filter stays empty).

The end point of the segmentation is determined by the alignment result. When the last row of the dynamic programming matrix, to be introduced in the following subsection, are all 0, the algorithm clears the corresponding record of the sequence window filter.

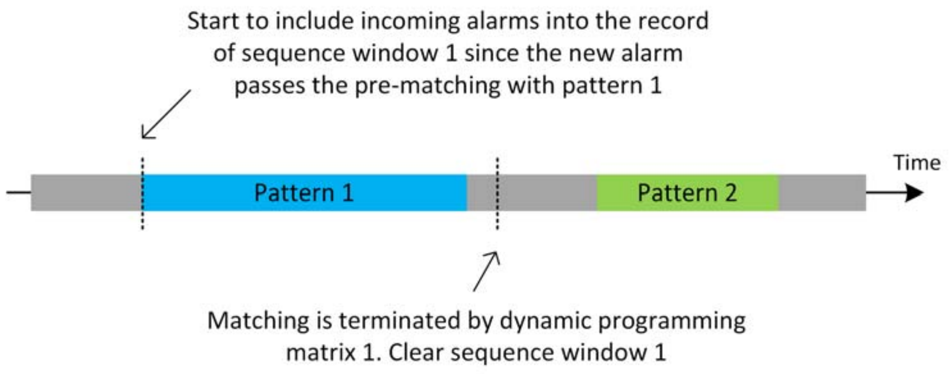

Start to include incoming alarms into the record of sequence window 1 since the new alarm passes the pre-matching with pattern 1
Time
Pattern 1
Pattern 2
Matching is terminated by dynamic programming matrix 1. Clear sequence window 1

Figure 3.2: An illustration example of the sequence window filter

An example of sequence window filter is shown in Figure 3.2. In the example, there are two patterns in the database, and the online sequence contains these two patterns. The gray part of the online sequence is formed by alarms irrelevant to the two patterns. When the first alarm of pattern 1 is raised in the online sequence, pre-matchings of this alarm with both pattern 1 and pattern 2 are carried out. Since this alarm is contained in pattern 1 but not in pattern 2, it passes the pre-matching with pattern 1 but fails to match with pattern 2; so the sequence window 1 starts to include incoming alarms into its record but the record of sequence window 2 stays empty. Matchings of the online sequence with the two patterns are approximated by the matchings between the two records of sequence window filters and the patterns. The segmentation of the online sequence for the matching with pattern 1 will be terminated and the record of sequence window 1 will be cleared when the elements of the last row of the corresponding dynamic programming matrix are all 0.

3.2.4 An Incremental Dynamic Programming Strategy

The algorithm proposed in [18] makes use of dynamic programming to search for the optimal alignment of two alarm sequences. However, the whole dynamic programming matrix need to be calculated every time when a matching is carried out, which is highly time consuming. Here we propose a new strategy to calculate the dynamic programming matrix incrementally each time when there is a new alarm annunciated and added to the online sequence. This way, the total computational burden is divided into small pieces that can be finished quickly in each individual step. In this subsection, we first introduce the modified time distance and weight vectors and the modified similarity score calculation, which are essential for the strategy of incremental dynamic programming; then the ordinary dynamic programming will be introduced, followed by the incremental strategy.

Modified Time Distance and Weight Vectors

In the proposed algorithm, the time distance vector for an alarm message (e_m, t_m) is defined as:

$$\begin{aligned} \mathbf{d}_m &= [d_m^1, d_m^2, ..., d_m^k, ..., d_m^K] \\ d_m^k &= |t_m - t_k|, \end{aligned} \tag{3.1}$$

where K is the total number of alarm messages in the sequence. It extracts the absolute time distances from (e_m, t_m) to all the other alarm messages in the sequence. The time weight vector for (e_m, t_m) is defined as:

$$\begin{aligned} \mathbf{w}_m = & \ [w_m^1, w_m^2, ..., w_m^k, ..., w_m^K] \\ = & \ [f(d_m^1), f(d_m^2), ..., f(d_m^k), ..., f(d_m^K)], \end{aligned} \tag{3.2}$$

where $f(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is the time weighting function. We choose it to be a scaled Gaussian function:

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

where σ is the variance. The function normalizes the time distances to $[0, 1]$. The larger the output value is, the shorter the time distance is. Here is an example of calculating the time distance and weight vectors. Consider sequence

$$< (2, 2), (1, 3), (3, 3.5), (1, 5), (4, 5.2) >,$$

where $K = 5$. The time distance vector $\mathbf{d}_1$ for message $(2, 2)$ is calculated as: $d_1^1 = 2 - 2 = 0$, $d_1^2 = 3 - 2 = 1$, $d_1^3 = 3.5 - 2 = 1.5$, $d_1^4 = 5 - 2 = 3$, and $d_1^5 = 5.2 - 2 = 3.2$. The time distance matrix formed by vectors $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{d}_3$, $\mathbf{d}_4$, and $\mathbf{d}_5$ is

$$[\mathbf{d}_1^T, \mathbf{d}_2^T, \mathbf{d}_3^T, \mathbf{d}_4^T, \mathbf{d}_5^T] = \begin{bmatrix} 0.0 & 1.0 & 1.5 & 3.0 & 3.2 \\ 1.0 & 0.0 & 0.5 & 2.0 & 2.2 \\ 1.5 & 0.5 & 0.0 & 1.5 & 1.7 \\ 3.0 & 2.0 & 1.5 & 0.0 & 0.2 \\ 3.2 & 2.2 & 1.7 & 0.2 & 0.0 \end{bmatrix}.$$

By applying the weighting function (set σ as 1) on the time distance matrix, the time weight matrix is obtained:

$$[\mathbf{w}_1^T, \mathbf{w}_2^T, \mathbf{w}_3^T, \mathbf{w}_4^T, \mathbf{w}_5^T] = \begin{bmatrix} 1.00 & 0.61 & 0.32 & 0.01 & 0.01 \\ 0.61 & 1.00 & 0.88 & 0.14 & 0.09 \\ 0.32 & 0.88 & 1.00 & 0.32 & 0.24 \\ 0.01 & 0.14 & 0.32 & 1.00 & 0.98 \\ 0.01 & 0.09 & 0.24 & 0.98 & 1.00 \end{bmatrix}.$$

Modified Similarity Scores

In the proposed algorithm, a similarity score is defined as:

$$\mathscr{S}((e_a, t_a), (e_b, t_b)) = \max\{\mathscr{S}_0((e_a, t_a), (e_b, t_b)), \mathscr{S}_0((e_b, t_b), (e_a, t_a))\} \times (1 - \mu) + \mu, \tag{3.4}$$

where (e_a, t_a) and (e_b, t_b) are alarm messages from two sequences of length K_1 and K_2, respectively. The negative parameter μ is the miss-match penalty. $\mathscr{S}_0((e_a, t_a), (e_b, t_b))$ and $\mathscr{S}_0((e_b, t_b), (e_a, t_a))$ are calculated in:

$$\mathscr{S}_0((e_a, t_a), (e_b, t_b)) = \begin{cases} \max\limits_{1 \le i \le K_1} \{w_a^i : e_i = e_b\}, & \text{if the set isn't empty} \\ 0, & \text{otherwise,} \end{cases} \tag{3.5}$$

$$\mathscr{S}_0((e_b, t_b), (e_a, t_a)) = \begin{cases} \max\limits_{1 \le i \le K_2} \{w_b^i : e_i = e_a\}, & \text{if the set isn't empty} \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

The value of $\mathscr{S}_0((e_a, t_a), (e_b, t_b))$ and $\mathscr{S}_0((e_b, t_b), (e_a, t_a))$ may not be the same since commutative law does not hold for $\mathscr{S}_0$. For this reason, we choose the larger value of $\mathscr{S}_0$ while calculating $\mathscr{S}((e_a, t_a), (e_b, t_b))$. For $\mathscr{S}_0((e_a, t_a), (e_b, t_b))$, we first filter out the values in $\mathbf{w}_a$ whose corresponding alarm type is not he same as e_b; then choose the maximum value in the remaining vector if the vector is not empty, otherwise $\mathscr{S}_0((e_a, t_a), (e_b, t_b)) = 0$. Since $\mathscr{S}_0((e_a, t_a), (e_b, t_b)) \in [0, 1]$, $\mathscr{S}((e_a, t_a), (e_b, t_b)) \in [\mu, 1]$. The highest value 1 can be reached when the two alarm messages (e_a, t_a) and (e_b, t_b) are of the same alarm type, indicating a match; the negative value μ is obtained if alarm type e_a or e_b does not appear in the other sequence or the time distance between the alarm messages that contain the same alarm type is too large.

Dynamic Programming with Modified Gap Penalty

Dynamic programming matrices are calculated for aligning the sequences. Each cell of the matrix $H_{m,n}$ is obtained in:

$$H_{m+1,n+1} = \max_{1 \le i \le m, 1 \le j \le n} (I(A_{i:m}, B_{j:n}), 0), \tag{3.7}$$

where $I(A_{i:m}, B_{j:n})$ is the alignment score of the segment pair $(A_{i:m}, B_{j:n})$. $H_{1:K_1,1}$ and $H_{1,1:K_2}$ are all set to be 0 during initialization. The other cells of the matrix are iteratively filled in by

$$\begin{aligned} H_{m+1,n+1} = & \max_{1 \le i \le m, 1 \le j \le n} (I(A_{i:m}, B_{j:n}), 0) \\ = \ & \max\{H_{m,n} + S((e_{m+1}, t_{m+1}), (e_{n+1}, t_{n+1})), \\ & H_{m,n+1} + \delta_1,\ H_{m+1,n} + \delta_2,\ 0\}, \end{aligned} \tag{3.8}$$

where

$$\delta_1 = (1 - w_{B,n}^{n+1}) * \delta, \tag{3.9}$$

$$\delta_2 = (1 - w_{A,m}^{m+1}) * \delta \tag{3.10}$$

are the gap penalties. The closer the time stamps of the two alarm messages (e_n, t_n) and (e_{n+1}, t_{n+1}) in a sequence are, the higher the value of $w_{B,n}^{n+1}$ is, and the lower the value of $1 - w_{B,n}^{n+1}$ is; thus a smaller penalty will be put on the gap caused by inserting alarm (e_{n+1}, t_{n+1}) into the alignment. In this way, the irrelevant alarms (disturbance alarms) that are raised closely in a sequence will cause smaller reduction on the alignment score. The values of δ_1 and δ_2 are between $[\delta, 0]$ based on the range of time weight distances.

Once the dynamic programming matrix is obtained, the optimal alignment score of the two sequences can be obtained easily by finding the maximum value in the matrix.

Incremental Dynamic Programming

In this subsection, we will show how to incrementally obtained the optimal alignment score when new alarms are raised and added into the online sequence.

We first provide an example of incrementally calculating the time weight matrix. Consider a new alarm $(4, 5.2)$ is raised and added into the online sequence:

$$< (2,2), (1,3), (3,3.5), (1,5) > .$$

The time weight matrix can be updated by augmenting the old one with a new row and column related to this alarm, as shown in Figure 3.3. Moreover, notice the time weight matrix is symmetric, half of the computation can be saved.

$$\left[\mathbf{w}_1^T, \mathbf{w}_2^T, \mathbf{w}_3^T, \mathbf{w}_4^T, \underline{\mathbf{w}_5^T}\right] = \begin{bmatrix} 1.00 & 0.61 & 0.32 & 0.01 & \underline{0.01} \\ 0.61 & 1.00 & 0.88 & 0.14 & \underline{0.09} \\ 0.32 & 0.88 & 1.00 & 0.32 & \underline{0.24} \\ 0.01 & 0.14 & 0.32 & 1.00 & \underline{0.98} \\ \underline{0.01} & \underline{0.09} & \underline{0.24} & \underline{0.98} & \underline{1.00} \end{bmatrix}$$

Figure 3.3: An illustrative example of updating the time weight matrix

Similar tricks can be used on updating the dynamic programming matrix as well. Given the pattern sequence

$$< (1,3), (2,26), (3,28), (4,293) >$$

and the online sequence

$$< (1,4),(8,26),(9,105) >,$$

when a new alarm $(2,122)$ is raised and included into the online sequence, the dynamic programming matrix can be updated by augmenting the old one (Figure 4.5) with a new row and column, as shown in Figure 4.6. Parameters were chosen as: $\delta = -0.2,\ \sigma = 2$, and $\mu = -0.6$ in this example.

	∅	(1,3)	(2,26)	(3,28)	(4,293)
∅	0	0	0	0	0
(1,4)	0	1	0.8	0.72	0.52
(8,26)	0	0.8	0.6	0.52	0.32
(9,105)	0	0.6	0.4	0.32	0.12

(a) Old dynamic programming matrix

	∅	(1,3)	(2,26)	(3,28)	(4,293)
∅	0	0	0	0	0
(1,4)	0	1	0.8	0.72	0.52
(8,26)	0	0.8	0.6	0.52	0.32
(9,105)	0	0.6	0.4	0.32	0.12
(2,122)	0	0.4	1.6	1.52	1.32

(b) Updated dynamic programming matrix

Figure 3.4: An illustrative example of updating the dynamic programming matrix

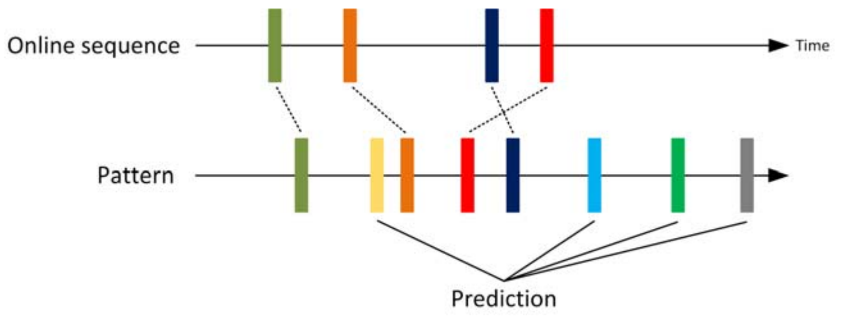


Figure 3.5: An illustrative example of predicting incoming alarms

3.2.5 Prediction of Incoming Alarms

Whenever the online sequence is identified as similar to a pattern, the prediction of incoming alarms will be provided based on the record of sequence window filter and the pattern in the database: the alarm messages contained in the pattern but not included in the record of sequence window filter will be given as predictions. Figure 3.5 shows an example of a prediction procedure.

3.2.6 Algorithm Flowchart and Pseudo Code

Figure 3.6 shows the flowchart of the algorithm. There are three main parts: chattering alarm elimination, sequence window record updating, and sequence matching based on incremental dynamic programming. Algorithm 2 is the corresponding pseudo code.

3.3 Industrial Case Study

A dataset from a real chemical process was used to test the proposed algorithm. Equipment used in the process include pumps, compressors, furnaces, and filters. General descriptions of the dataset can be found in Table 3.1. Off-delay timers of 300 seconds were applied uniformly to all the tags to remove chattering alarms. 359 alarm flood sequences were extracted based on the ISA standard, which is more than 10 alarms per 10 minutes. The extracted alarm floods were then clustered based on pairwise similarity scores calculated using the method in [18], given parameter values: $\sigma = 2$, $\mu = -0.6$, and $\delta = -0.2$; the result is shown in Figure 3.7. The following tests were done using MATLAB on a 64-bit Windows PC with Intel(R) Core(TM) i7-4770 3.40GHz CPU and 24.0 GB memory.

Table 3.1: Statistics of the dataset

Description	Number
Total time period	336 days
Total number of tags	1502
Total number of alarms	109393
Average alarm rate	14/h
Highest peak alarm rate	334/10 min
Number of alarm floods	359
Average length of alarm floods	39

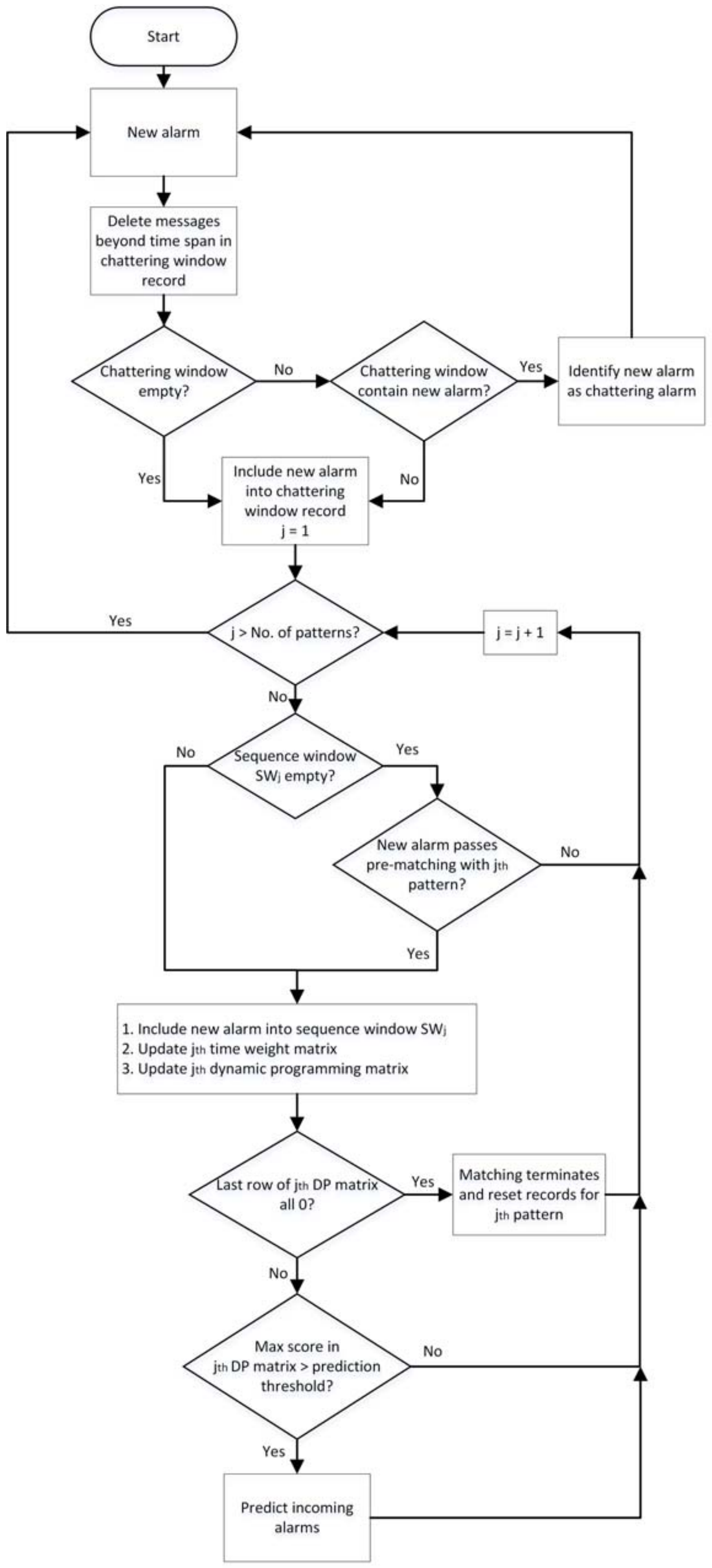

Start
New alarm
Delete messages beyond time span in chattering window record
Chattering window empty?
No
Chattering window contain new alarm?
Yes
Identify new alarm as chattering alarm
Yes
No
Include new alarm into chattering window record j = 1
Yes
j > No. of patterns?
j = j + 1
No
No
Sequence window SWj empty?
Yes
New alarm passes pre-matching with jth pattern?
No
Yes
1. Include new alarm into sequence window SWj
2. Update jth time weight matrix
3. Update jth dynamic programming matrix
Last row of jth DP matrix all 0?
Yes
Matching terminates and reset records for jth pattern
No
Max score in jth DP matrix > prediction threshold?
No
Yes
Predict incoming alarms

Figure 3.6: Flowchart of the proposed algorithm

Algorithm 2: Online pattern matching and prediction of incoming alarms

```
input  : Latest raised alarm message, pattern database $D$, variance of
         Gaussian function $\sigma$, scalar of gap penalty $\delta$, miss-match
         penalty $\mu$, prediction threshold $\alpha$, delay-timer length $\gamma$
output: Identified pattern index and prediction of upcoming alarms

1  begin
2      Eliminate alarm messages raised beyond time span $\gamma$ from
         chattering window record;
3      if Chattering window $CW$ is empty then
4          Include new alarm message to chattering window $CW$;
5          Go to 11;
6      else
7          if Chattering window $CW$ contains this new alarm message
             then
8              Identify this alarm as a chattering alarm
9          else
10             Include new alarm message to chattering window $CW$;
11             for $j \leftarrow 1$ to the number of patterns do
12                 if Sequence window $SW_j$ is empty then
13                     if New alarm passes the pre-matching with the
                         pattern then
14                         Go to 16
15                 else
16                     Include new alarm into $SW_j$. Augment $j^{th}$ time
                         distance and weight matrices and $j^{th}$ DP matrix
                         through incremental computation;
17                     if The last row of dynamic programming matrix $DP_j$
                         are all 0 then
18                         Terminate matching of pattern $j$;
19                         Reset matching records with pattern $j$;
20                     else
21                         if Maximum value in dynamic programming
                             matrix $DP_j$ larger than prediction threshold $\alpha$
                             then
22                             Identify temporal online sequence similar to
                                 pattern $j$;
23                             Predict incoming alarms by eliminating
                                 identical alarm types in $SW_j$ from pattern $j$;
24     return Identified pattern number and prediction of upcoming alarms
```

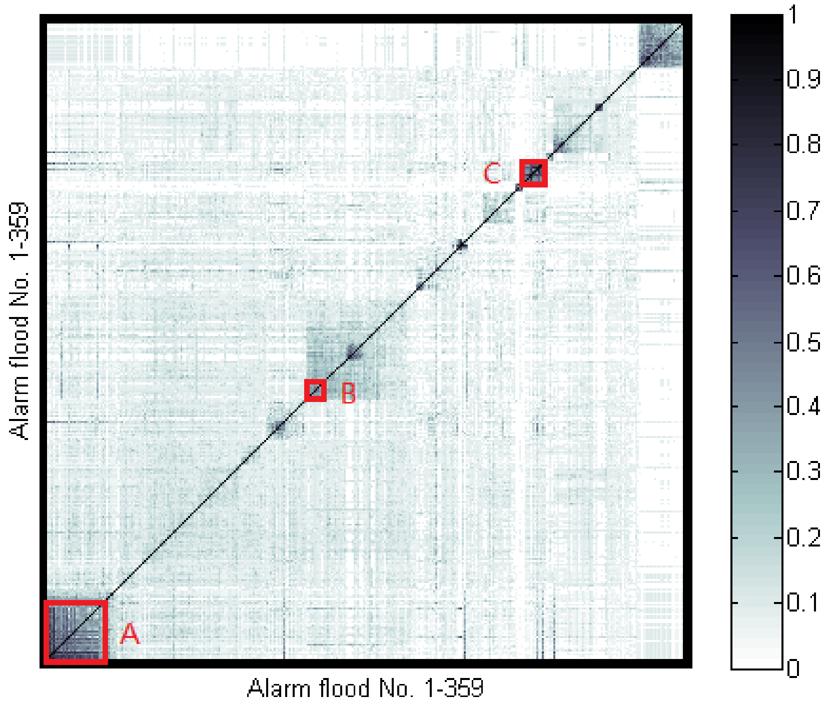


Figure 3.7: Clustering result of the alarm floods and the selected alarm floods for the accuracy tests

3.3.1 Efficiency Tests

Efficiency of the algorithm can be evaluated by the response time, which is defined as the time interval between the point when a new alarm is raised in the online sequence and the point when the algorithm makes the judgment (whether the temporal online sequence is similar to any of the patterns in the database). In order to test the efficiency of the proposed algorithm, comparison with the modified Smith-Waterman algorithm [18] has been conducted.

Response Time with Respect to The Number of Patterns in The Database

All the 359 alarm floods (chattering alarms included) were used as the online sequences, and the pattern database for each test was formed by selecting different number of sequences in the 359 alarm floods (chattering alarms excluded). The selection of alarm flood sequences was random, except for the restriction that the average sequence length should be about the same in the pattern database for each test.

Five tests were carried out using the same batch of online sequences but different pattern databases (with similar average pattern lengths but different number of patterns). Table 3.2 gives the detailed information for the pattern database in each test.

Table 3.2: Information of the pattern databases used in the tests to study the efficiency of the algorithm with respect to the number of patterns in the database

Test No.	Number of patterns in database	Average pattern length
1	20	14.1
2	40	14.8
3	60	13.9
4	80	14.3
5	100	14.5

During the tests, alarms in the online alarm sequence were raised in order; response times of the two algorithms were recorded when each of the alarms in the online sequence was raised. Figure 3.8 shows the comparisons of the response times of the two algorithms in the tests. Apparently, the worst runtime (longest response time) of the modified Smith-Waterman algorithm was almost 6 seconds when there were 100 patterns in the database, which is not acceptable for online application. On the contrary, the proposed algorithm always finished fast in all the tests, even when the pattern number reached 100. The tests also confirmed the algorithm's computational complexity analyzed in the discussion section; the response time of the proposed algorithm increased almost linearly with the number of patterns in the database.

Response Time with Respect to The Average Pattern Length

Similarly, all the 359 alarm floods (chattering alarms included) were used as the online sequences to study the relationship between the algorithm response time and the average pattern length. Each pattern database was formed by a selection of 10 alarm flood sequences (chattering alarms excluded). The selection of alarm floods was random, except for the restriction that the average pattern length should be different for each database. Five tests were carried out and the detailed information of the pattern database used in each test can be found in Table 3.3.

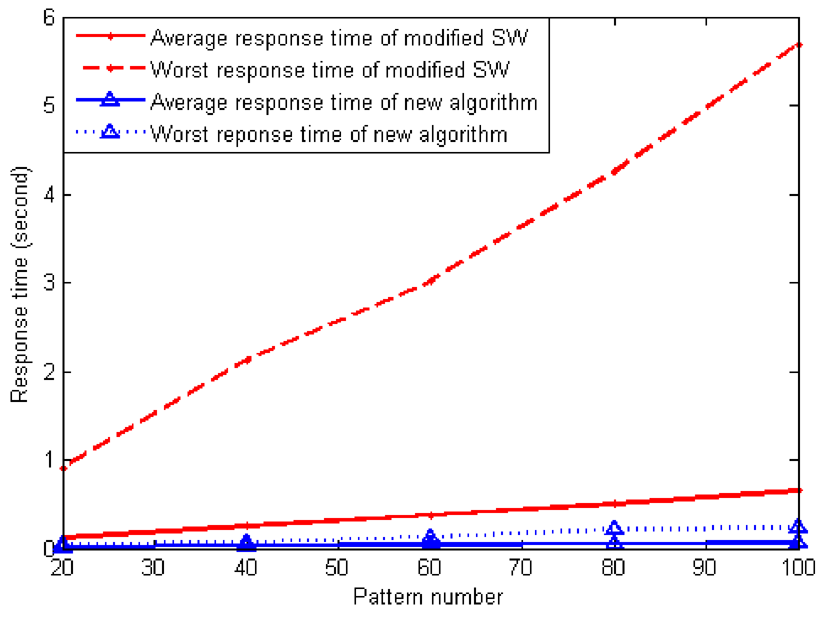


Figure 3.8: Comparison of the response time of the two algorithms when the number of patterns in the database increases

Table 3.3: Information of the pattern databases used in the tests to study the efficiency of the algorithm with respect to the average pattern length in the database

Test No.	Number of patterns in database	Average pattern length
1	10	10.8
2	10	20.4
3	10	29.6
4	10	39.7
5	10	50.1

Figure 3.9 shows the results of both the average and worst response times for the two tested algorithms. Notice that both the worst and average response times of the proposed algorithm increased almost linearly with the average pattern length, and that the worst response time was far under 0.1 second even when the average pattern length reached 50. This encouraging result, together with the previous one about the response time with respect to the number of patterns, allows the proposed algorithm to be applicable to very large scale plants.

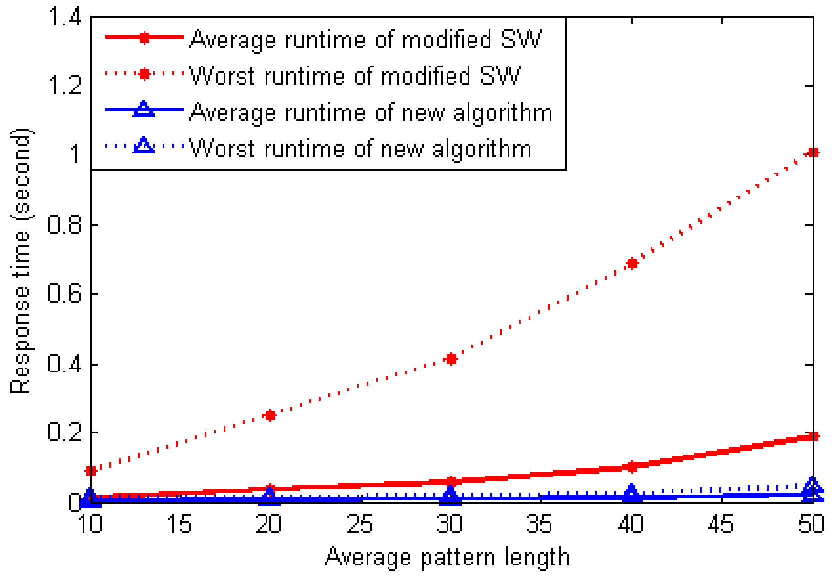


Figure 3.9: Comparison of the response times of the two algorithms when the average pattern length increases

3.3.2 Accuracy Tests

The detection accuracy of the proposed algorithm has been tested from the aspects of missed and false detection rates since there is a trade-off between the two; by lowing the prediction threshold α, the algorithm becomes less conservative on asserting the online sequence to be similar to the patterns in the database.

In preparation of the accuracy tests, 3 clusters of similar alarm floods (chattering alarms excluded) have been selected as the testing dataset, as shown by the red boxes in Figure 3.7. Each dot on the figure represents the similarity score between two alarm flood sequences. A darker color means a stronger similarity. The reason for selecting these 3 clusters of alarm floods is because they held a relatively long common pattern within each cluster and if any other alarm flood is included into the cluster, the length of the common pattern would drop significantly. In this way, we assume that the selected alarm floods represent a common abnormality that is different from the ones represented by the other alarm floods. Table 3.4 shows the information of the alarm flood sequences in each selected cluster.

Table 3.4: Information of the alarm sequences in each selected cluster

Cluster	A	B	C
Number of alarm floods	19	12	9
Average sequence length	26.6	14.6	31.6
Longest sequence length	46	23	64
Shortest sequence length	15	10	14
Pattern length	14	4	11

Missed detection (missing to detect the online sequence to be similar to a pattern when it indeed is) rates with respect to different prediction thresholds α were obtained by applying cross validations. Given one prediction threshold α, the procedures of cross validation are as follows:

1. Randomly select half of the sequences from each of the 3 clusters and find their common patterns using the method in [62]. Form the pattern database with the 3 obtained patterns;

2. Treat the other half of the alarm flood sequences in each cluster as the online sequences and match them with the 3 pattern sequences in the database; and

3. Record a missed detection if the algorithm fails to identify the alarm flood to be similar to the corresponding pattern sequence.

False detection (detecting the online sequence to be similar to a pattern when it is actually not) rates versus the prediction threshold were also obtained. Given one prediction threshold α, the procedures of the tests for the false detection rates are as follows:

1. Apply the method in [62] to all the alarm flood sequences in each of the 3 selected clusters to find their common patterns. Form the pattern database with the 3 obtained patterns;

2. Treat the rest of the 359 alarm flood sequences that are not included in any of the 3 chosen clusters as the online sequences and match them with the 3 pattern sequences in the database; and

3. Record a false detection if the algorithm identifies the alarm flood to be similar to any of the pattern sequences.

Table 3.5: Results of accuracy tests

Prediction threshold	False detection rate	missed detection rate
2	8.5%	2.5%
3	6.6%	2.5%
4	3.8%	5.0%
5	2.5%	32.5%
6	1.6%	37.5%
7	1.6%	42.5%

Table 3.5 shows the accuracy test results. Both the False Detection Rate (FDR) and the Missed Detection Rate (MDR) are listed for different values of prediction threshold α. The table reveals clearly a trade-off between the missed and false detection rates while changing the prediction threshold. When the prediction threshold was set to be 4, both the missed and false detection rate were below 5%. There is a significant increase on MDR when the prediction threshold α becomes bigger than 4. The reason is because the pattern length in cluster B is 4; when α is larger than 4, even though there exists a match between the online sequence and the pattern from cluster B, their matching score will still be smaller than the prediction threshold. The changes of false and missed detection rates with respect to the different values of prediction threshold are shown in the Receiver Operating Characteristic (ROC) curve in Figure 3.10.

By further inspecting the cases where the algorithm made the false detections, we found the reason was because there was a short common sequence between the pattern and the online sequence. Thus, when this subsequence of alarms were raised, the algorithm identified the online alarm sequence to be similar to the pattern. One way to reduce the false detection rate is to increase the prediction threshold. However, it may deteriorate the missed detection rate, due to the trade-off between the FDR and MDR. Moreover, the missed detection rate is also related to the length of the pattern of the alarm floods in the selected cluster since the longer the pattern is, the higher the similarity score between the pattern and a similar alarm flood would be, and the larger margin there would be between the obtained similarity score and the prediction threshold.

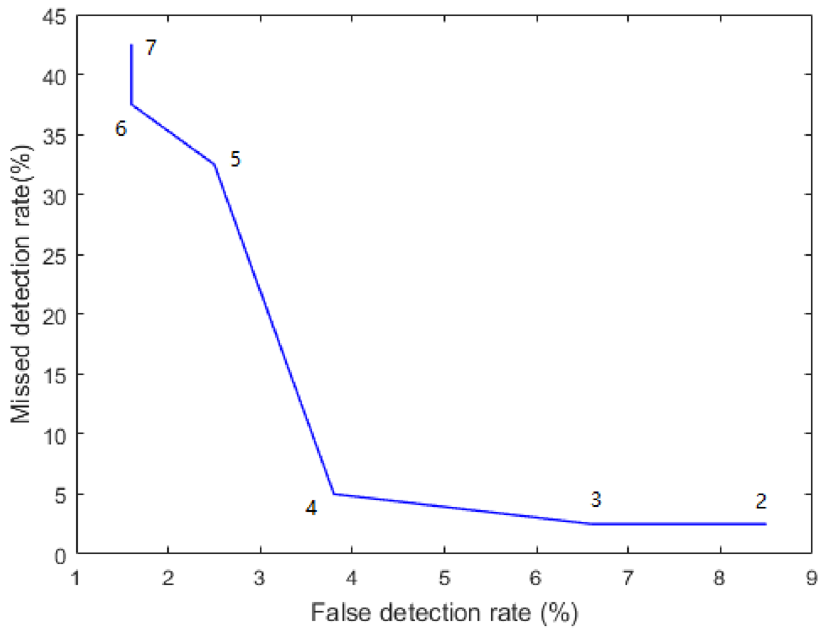


Figure 3.10: The ROC plot of the accuracy test results with different prediction thresholds

3.4 Discussions

The computational complexity of the proposed algorithm is $\mathcal{O}(CW+\sum_{j=1}^{N}(P_j+SW_j))$, where CW is the length of the alarm record in the chattering window filter, N is the number of non-empty sequence window filters, SW_j is the length of the alarm record in sequence window filter j, and P_j is the length of the corresponding pattern. The first part CW is the time occupied to check whether the new alarm is a chattering alarm, and the second part $\sum_{j=1}^{N}(P_j + SW_j)$ is to update every sequence window filter and match their records with all the patterns. Based on this analysis, the worst case computational complexity $\mathcal{O}(CW + \sum_{j=1}^{M}(P_j) + M \times L)$ occurs when the online sequence is similar to every pattern in the database, where M is the number of patterns in the database and L is the length of the online sequence. Notice even for the worst case, the computational complexity still increases linearly with the number of patterns and the lengths of the patterns and the online sequence.

Regarding the algorithm's detection accuracy, there is a trade-off between the missed and false detection rates. Both of them depend on the value of the prediction threshold and there exist a trade-off. Raising the prediction threshold will reduce the false detection rate but it will worsen the missed detection rate. On the contrary, the missed detection rate could decrease if we loosen the prediction threshold, but the false detection rate would increase on the other hand. Notice the discussion is based on the condition when other parameters: δ, μ, and σ are unchanged. It would be more complicated if we consider the influence of all the parameters on the detection accuracy.

Compared with the method in [18], the proposed algorithm is much more efficient in terms of online application, as shown clearly in the industrial case study. The proposed incremental dynamic programming strategy allows the matching between the online alarm sequence and the patterns to be updated incrementally when every new alarm is raised instead of re-matching the whole sequences repeatedly. Regarding the alignment accuracy, when applied offline, the proposed method achieves exactly the same alignment result that the method in [18] gives. Since the method in [18] guarantees the optimal alignment result given parameters δ, μ, and σ, the proposed method can guarantee the optimal alignment result as well when applied offline. Here we need to clarify the difference between alignment accuracy and detection accuracy. Alignment accuracy measures how accurately an algorithm is able to align a pair of sequences and find their best matching score given a set of parameters δ, μ, and σ. It is an offline measurement because the inputs are two complete sequences. While the detection accuracy measures the ability of an algorithm to detect a matching between an online sequence and a pattern sequence. It is an online measurement since the online sequence is not complete and keeps updating. Thus, even though the proposed algorithm guarantees the optimal alignment result, it cannot guarantee the optimal detection accuracy.

As mentioned in [18], the variance of Gaussian function σ, the gap penalty δ, and the mismatch penalty μ all influence sequence alignments, so they have to be adjusted based on users' specifications. The variance of the Gaussian function σ controls the size of the time span within which the algorithm blurs the occurrence orders between the alarm messages. When the value of σ goes to infinity, the algorithm totally ignores the orders of alarms in the alignment and simply counts the alarm occurrences. On the contrary, if $\sigma = 0$, the orders of alarms have to be exactly the same in the two sequences in order to get a match. The gap and mismatch penalties, δ and μ, determine the algorithm's tolerance to including gaps and mismatch terms in alignments. When these two parameters get larger, the algorithm places more tolerance on the irrelevant alarms raised within a pattern sequence when determining a match, leading to a rise in FDR and a drop in MDR. γ is the length of the delay timer used in the Chattering Window filter. It acts as a preprocessing stage to remove chattering alarms from the online alarm sequence and does not have big influence on the performance of the algorithm.

Apart from parameter tuning, which requires some process knowledge and experience, another limitation of the algorithm is that it depends on the pattern sequences to repeat in the online alarm sequence in order to be effective. In other words, the proposed method can be effective in recognizing frequent faults such as compressor trips and stuck valves, but not as effective for non-repeatable problem such as an emergency plant shut-down.

3.5 Summary

In this chapter, we proposed an algorithm for online pattern matching and prediction of incoming alarm floods. A chattering window filter was introduced to eliminate chattering alarms online; a sequence window filter, a modified calculation of time weight matrix, and an incremental dynamic programming strategy were introduced to improve the efficiency of the algorithm. The computational complexity analysis has shown that the proposed algorithm is applicable to large scale plants since the computational cost increases linearly with the number of patterns and the lengths of patterns and the online sequence. Accuracy tests revealed a trade-off between the missed and false detection rates.

Chapter 4

Accelerated Multiple Alarm Flood Sequence Alignment for Abnormality Pattern Mining*

4.1 Overview

The authors in [83] summarized and categorized the existing studies on the alarm overloading problem and formulated nine fundamental research problems to be solved. In this chapter, we propose an accelerated multiple sequence alignment algorithm for pattern mining in multiple alarm floods, which potentially can be used in solving the 5th and 6th listed problems, namely, "whether there exist any nuisance alarms in the historical data, worthy of redesigning alarm generation mechanisms" and "how to design mechanisms to generate predictive alarms in order to predict upcoming critical abnormal events." Unlike traditional methods which either cannot deal with multiple sequences with time stamps or suffer from high computational cost, the computational complexity of this proposed algorithm is reduced significantly by introducing a generalized pairwise sequence alignment method and a progressive multiple sequence alignment approach. Two types of alignment refinement methods are developed to improve the alignment accuracy. The effectiveness of the proposed algorithm is tested using a dataset from a real chemical plant.

Low computational cost is one of the advantages of the proposed algorithm, allowing it to find patterns effectively even in large scale datasets. The com-

*A version of this chapter has been submitted for publication as: Shiqi Lai, Fan Yang, Tongwen Chen. Accelerated multiple alarm flood sequence alignment for abnormality pattern mining. *Journal of Process Control*, 2017.

putational cost function is of a quadratic form in terms of sequence lengths, not like the exhaustive search approach as in [62], where the computational cost would increase exponentially with the numbers of sequences and their lengths. Being able to take time information (time intervals between alarms) into consideration when aligning multiple sequences is another advantage of the algorithm, allowing it to have some robustness on process delays that may result in order shifts among successively raised alarms. Time information also allows the algorithm to distinguish sequences with the same alarm occurrences but difference time intervals between alarms. The third advantage of the algorithm is the capability to find patterns in multiple sequences. New elements introduced in the proposed algorithm are as follows:

1. A generalized pairwise sequence alignment method, which is composed of a generalized similarity score calculation and a generalized dynamic programming procedure;

2. A progressive multiple sequence alignment approach, which iteratively applies the generalized pairwise sequence alignment method to approximate the optimal alignment of alarm floods;

3. A leave-one-out and a random-division refinement method, to improve the alignment result in an iterative manner.

There is a trade-off between efficiency and accuracy of the algorithm. The proposed algorithm does not guarantee to find the optimal solution like the method in [62]. However, based on our tests, the accuracy of the proposed algorithm is still suitable for applications, averaging 5% difference when compared to the results given by the method in [62]. The efficiency of the proposed algorithm is higher than the method in [62] by several orders of magnitude. Thus, given a large scale dataset, the proposed algorithm is able to complete fast with usable results while the method in [62] may stuck for hours or even days.

4.2 Algorithm

4.2.1 Problem Formulation

The problem formulation is the same as the one described in Section 2.2.1. Consider the following alarm sequences:

$$\begin{aligned}
S_1 = &< (e_{11}, t_{11}), (e_{12}, t_{12}), ..., (e_{1m}, t_{1m}), ..., (e_{1M}, t_{1M}) >,\\
S_2 = &< (e_{21}, t_{21}), (e_{22}, t_{22}), ..., (e_{2n}, t_{2n}), ..., (e_{2N}, t_{2N}) >,\\
&...\\
S_J = &< (e_{J1}, t_{J1}), (e_{J2}, t_{J2}), ..., (e_{Jo}, t_{Jo}), ..., (e_{JO}, t_{JO}) >,
\end{aligned}$$

where e_{1m}, e_{2n},..., $e_{Jo} \in \Lambda$ (the set of alarm types) and t_{1m}, t_{2n},..., t_{Jo} are the corresponding time stamps, respectively. S_1, S_2,..., S_J are similar sequences detected by running pairwise sequence matching algorithms such as [18]. The objective is to obtain the optimal alignment of these sequences by adding gaps (represented by "[]") and deleting unrelated alarm messages, for example:

$$\begin{aligned}
&< (e_{16}, t_{16}), (e_{17}, t_{17}), (e_{18}, t_{18}),\ [\ ]\ ,(e_{19}, t_{19}) >,\\
&< (e_{22}, t_{22}),\ [\ ]\ , (e_{23}, t_{23}), (e_{24}, t_{24})\ ,(e_{25}, t_{25}) >,\\
&...\\
&<\ [\ ]\ , (e_{J1}, t_{J1}), (e_{J2}, t_{J2}),\ [\ ]\ ,(e_{J3}, t_{J3}) > .
\end{aligned}$$

4.2.2 Generalized Pairwise Sequence Alignment

Time Distance and Weight Vectors

Time stamps are important components of alarm messages; two alarm messages with different time intervals could indicate totally different problems in the process. Thus, in order to incorporate time information into the evaluation of an alarm sequence alignment, the time distance and weight vectors, as defined in Section 3.2.4, are applied. The time distance vector for an alarm message (e_m, t_m) is defined as:

$$\begin{aligned}
\mathbf{d}_m &= [d_m^1, d_m^2, ..., d_m^k, ..., d_m^K]\\
d_m^k &= |t_m - t_k|,
\end{aligned} \tag{4.1}$$

where K is the total number of alarm messages in the sequence. Thus, d_m^k gives the absolute time distance between (e_m, t_m) and (e_k, t_k) in the sequence.

The time weight vector for (e_m, t_m) is defined as:

$$\begin{aligned} \mathbf{w}_m = & \ [w_m^1, w_m^2, ..., w_m^k, ..., w_m^K] \\ = & \ [f(d_m^1), f(d_m^2), ..., f(d_m^k), ..., f(d_m^K)], \end{aligned} \tag{4.2}$$

where $f(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is the weighting function. A scaled Gaussian function is selected in this paper:

$$f(x) = \ e^{-x^2/2\sigma^2}, \tag{4.3}$$

where σ is the standard deviation, controlling how much weight to be put on the close-by alarm messages to blur their orders in the alignment. The function also normalizes the real-time distance to $[0, 1]$; a large output indicates a short time distance.

Generalized Similarity Score

In [18], the authors proposed a way to calculate the similarity score between two alarm messages in two alarm sequences. Here we generalize it to calculate the similarity score between two tuples in two alarm sequence alignments. The definition of a tuple is illustrated by the example in Figure 4.1. In the figure, there are two alarm sequence alignments ($A_1, A_2, \ldots, A_M$ and $B_1, B_2, \ldots, B_N$); the tuples (T_{A1}, T_{A2},. . . , T_{AP} and T_{B1}, T_{B2},. . . , T_{BQ}) are formed by alarm messages or gaps located at the same position of the corresponding alignment. The lengths of the alignments are P and Q, respectively. Notice the lengths of $A_1, A_2, \ldots, A_M$ are always the same because they are from the same alignment, while P and Q may not be the same value.

The formula for calculating the similarity score $\mathbb{S}(T_{Ap}, T_{Bq})$ between the tuples T_{Ap} and T_{Bq} is as follows:

$$\mathbb{S}(T_{Ap}, T_{Bq}) = \ \mu + \frac{1-\mu}{MN} \times \sum_{i=1}^{M} \sum_{j=1}^{N} \mathcal{S}((e_{Ai}^p, t_{Ai}^p), (e_{Bj}^q, t_{Bj}^q)), \tag{4.4}$$

where $\mathcal{S}((e_{Ai}^p, t_{Ai}^p), (e_{Bj}^q, t_{Bj}^q))$ is the similarity score between two alarm messages and is obtained by

$$\begin{aligned} & \mathcal{S}((e_{Ai}^p, t_{Ai}^p), (e_{Bj}^q, t_{Bj}^q)) \\ & = \ \max\{s((e_{Ai}^p, t_{Ai}^p), (e_{Bj}^q, t_{Bj}^q)), \\ & \qquad s((e_{Bj}^q, t_{Bj}^q), (e_{Ai}^p, t_{Ai}^p))\}, \end{aligned} \tag{4.5}$$

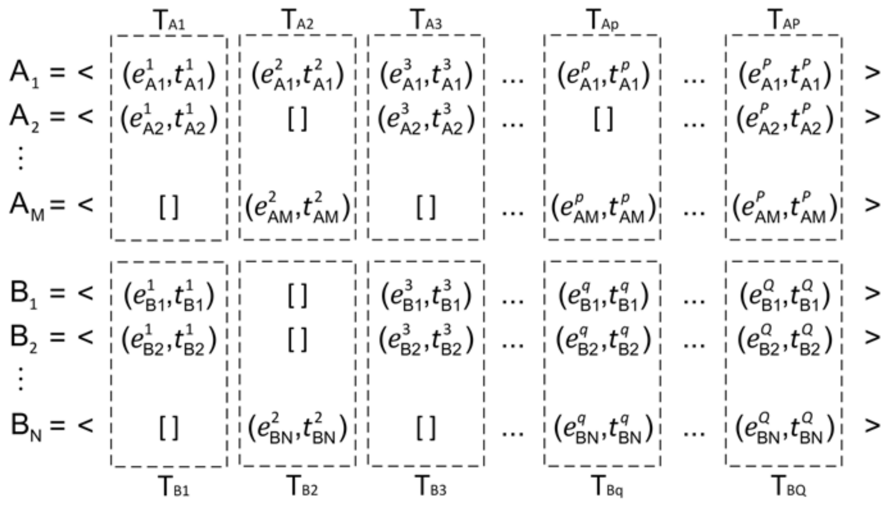

T_{A1}
T_{A2}
T_{A3}
T_{Ap}
T_{AP}
$A_1 = <$ (e^1_{A1},t^1_{A1}) (e^2_{A1},t^2_{A1}) (e^3_{A1},t^3_{A1}) ... (e^p_{A1},t^p_{A1}) ... (e^P_{A1},t^P_{A1}) $>$
$A_2 = <$ (e^1_{A2},t^1_{A2}) [] (e^3_{A2},t^3_{A2}) ... [] ... (e^P_{A2},t^P_{A2}) $>$
$A_M = <$ [] (e^2_{AM},t^2_{AM}) [] ... (e^p_{AM},t^p_{AM}) ... (e^P_{AM},t^P_{AM}) $>$
$B_1 = <$ (e^1_{B1},t^1_{B1}) [] (e^3_{B1},t^3_{B1}) ... (e^q_{B1},t^q_{B1}) ... (e^Q_{B1},t^Q_{B1}) $>$
$B_2 = <$ (e^1_{B2},t^1_{B2}) [] (e^3_{B2},t^3_{B2}) ... (e^q_{B2},t^q_{B2}) ... (e^Q_{B2},t^Q_{B2}) $>$
$B_N = <$ [] (e^2_{BN},t^2_{BN}) [] ... (e^q_{BN},t^q_{BN}) ... (e^Q_{BN},t^Q_{BN}) $>$
T_{B1}
T_{B2}
T_{B3}
T_{Bq}
T_{BQ}

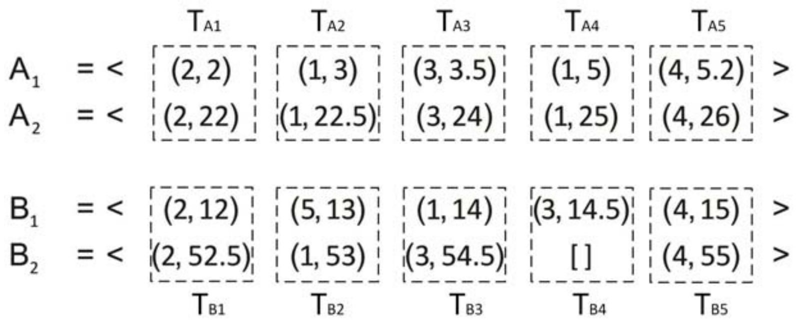
Figure 4.1: An example of tuples in two alarm sequence alignments

where

$$s((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj})) = \begin{cases} \max\limits_{1\le k\le P}\{w^k_{Ai,p} : e^k_{Ai} = e^q_{Bj}\}, & \text{if the set isn't empty} \\ & \text{and } (e^q_{Bj}, t^q_{Bj}) \text{ is not a gap} \\ 0, & \text{otherwise,} \end{cases} \tag{4.6}$$

$$s((e^q_{Bj}, t^q_{Bj}), (e^p_{Ai}, t^p_{Ai})) = \begin{cases} \max\limits_{1\le k\le Q}\{w^k_{Bj,q} : e^k_{Bj} = e^p_{Ai}\}, & \text{if the set isn't empty} \\ & \text{and } (e^p_{Ai}, t^p_{Ai}) \text{ is not a gap} \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

The negative parameter μ in equation (4.4) is the mismatch penalty. The similarity score $\mathbb{S}(T_{Ap}, T_{Bq})$ between the two tuples T_{Ap} and T_{Bq} is obtained by averaging all the pairwise similarity scores of the alarms in the two tuples. The value of $\mathbb{S}(T_{Ap}, T_{Bq})$ is always within $[\mu,\ 1]$; a larger value means a stronger similarity. The similarity score $\mathcal{S}((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj}))$ between two alarm messages (e^p_{Ai}, t^p_{Ai}) and (e^q_{Bj}, t^q_{Bj}) is achieved by selecting the larger value between $s((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj}))$ and $s((e^q_{Bj}, t^q_{Bj}), (e^p_{Ai}, t^p_{Ai}))$, as the commutative law does not hold. When both (e^p_{Ai}, t^p_{Ai}) and (e^q_{Bj}, t^q_{Bj}) are gaps, their score will be assigned as 0. Note that the pairwise similarity score formula in [18] is a special case of the generalized similarity score calculation proposed in this paper; when the two alignments are reduced to two alarm sequences (e.g., $M = N = 1$), the two formulae will give exactly the same output.



TA1 TA2 TA3 TA4 TA5
A1 = < (2, 2) (1, 3) (3, 3.5) (1, 5) (4, 5.2) >
A2 = < (2, 22) (1, 22.5) (3, 24) (1, 25) (4, 26) >
B1 = < (2, 12) (5, 13) (1, 14) (3, 14.5) (4, 15) >
B2 = < (2, 52.5) (1, 53) (3, 54.5) [] (4, 55) >
TB1 TB2 TB3 TB4 TB5

Figure 4.2: A numerical example for the calculation of the generalized similarity scores

Consider the example of two alarm sequence alignments in Figure 4.2, where $M = N = 2$ and the lengths of the two sequence alignments are both 5. The time weight matrices of A_1, A_2, B_1, and B_2 are obtained as $\mathbf{W}_{A_1}$, $\mathbf{W}_{A_2}$, $\mathbf{W}_{B_1}$, and $\mathbf{W}_{B_2}$, given $\sigma = 1$. The fourth row and column of $\mathbf{W}_{B_2}$ are empty since the fourth element in B_2 is a gap.

$$\begin{aligned}\mathbf{W}_{A_1} &= [\mathbf{w}^T_{A_1,1}, \mathbf{w}^T_{A_1,2}, \mathbf{w}^T_{A_1,3}, \mathbf{w}^T_{A_1,4}, \mathbf{w}^T_{A_1,5}] \\ &= \begin{bmatrix} 1.00 & 0.61 & 0.32 & 0.01 & 0.01 \\ 0.61 & 1.00 & 0.88 & 0.14 & 0.09 \\ 0.32 & 0.88 & 1.00 & 0.32 & 0.24 \\ 0.01 & 0.14 & 0.32 & 1.00 & 0.98 \\ 0.01 & 0.09 & 0.24 & 0.98 & 1.00 \end{bmatrix}\end{aligned}$$

$$\begin{aligned}\mathbf{W}_{A_2} &= [\mathbf{w}^T_{A_2,1}, \mathbf{w}^T_{A_2,2}, \mathbf{w}^T_{A_2,3}, \mathbf{w}^T_{A_2,4}, \mathbf{w}^T_{A_2,5}] \\ &= \begin{bmatrix} 1.00 & 0.88 & 0.14 & 0.01 & 0 \\ 0.88 & 1.00 & 0.32 & 0.04 & 0 \\ 0.14 & 0.32 & 1.00 & 0.61 & 0.14 \\ 0.01 & 0.04 & 0.61 & 1.00 & 0.61 \\ 0 & 0 & 0.14 & 0.61 & 1.00 \end{bmatrix}\end{aligned}$$

$$\begin{aligned}\mathbf{W}_{B_1} &= [\mathbf{w}^T_{B_1,1}, \mathbf{w}^T_{B_1,2}, \mathbf{w}^T_{B_1,3}, \mathbf{w}^T_{B_1,4}, \mathbf{w}^T_{B_1,5}] \\ &= \begin{bmatrix} 1.00 & 0.61 & 0.14 & 0.04 & 0.01 \\ 0.61 & 1.00 & 0.61 & 0.32 & 0.14 \\ 0.14 & 0.61 & 1.00 & 0.88 & 0.61 \\ 0.04 & 0.32 & 0.88 & 1.00 & 0.88 \\ 0.01 & 0.14 & 0.61 & 0.88 & 1.00 \end{bmatrix}\end{aligned}$$

$$\mathbf{W}_{B_2} = [\mathbf{w}_{B_2,1}^T, \mathbf{w}_{B_2,2}^T, \mathbf{w}_{B_2,3}^T, \mathbf{w}_{B_2,4}^T, \mathbf{w}_{B_2,5}^T] = \begin{bmatrix} 1.00 & 0.88 & 0.14 & [] & 0.01 \\ 0.88 & 1.00 & 0.32 & [] & 0.04 \\ 0.14 & 0.32 & 1.00 & [] & 0.61 \\ [] & [] & [] & [] & [] \\ 0.01 & 0.04 & 0.61 & [] & 1.00 \end{bmatrix}$$

Let $\mu = -1$, the similarity score for the pair of tuples T_{A4} and T_{B4} is calculated as

$$\mathbb{S}(T_{A4}, T_{B4}) = -1 + \frac{1+1}{2 \times 2} \times \sum_{i=1}^{2} \sum_{j=1}^{2} \mathcal{S}((e_{Ai}^4, t_{Ai}^4), (e_{Bj}^4, t_{Bj}^4)) = -0.12,$$

where

$$\mathcal{S}((e_{A1}^4, t_{A1}^4), (e_{B1}^4, t_{B1}^4)) = \max\{w_{A_1,4}^3, w_{B_1,4}^3\} = 0.88,$$
$$\mathcal{S}((e_{A2}^4, t_{A2}^4), (e_{B1}^4, t_{B1}^4)) = \max\{w_{A_2,4}^3, w_{B_1,4}^3\} = 0.88,$$
$$\mathcal{S}((e_{A1}^4, t_{A1}^4), (e_{B2}^4, t_{B2}^4)) = \max\{0, 0\} = 0,$$

and

$$\mathcal{S}((e_{A2}^4, t_{A2}^4), (e_{B2}^4, t_{B2}^4)) = \max\{0, 0\} = 0.$$

Intuitively, this negative similarity score also makes sense since there is a gap in tuple T_{B4} and the type of the two alarm messages in tuple T_{A4} is "1" while the type of the alarm message in tuple T_{B4} is "3". The reason why this similarity score is -0.12 instead of the minimum value -1 is because there exists an alarm message $(1, 14)$ in tuple T_{B3} that was raised slightly ahead of $(3, 14.5)$ and its alarm type matches the alarm messages in tuple T_{A4}.

Generalized Dynamic Programming Procedure

Similarly, the dynamic programming procedure proposed in [18] for aligning two alarm sequences is generalized to aligning two alarm sequence alignments.

Figure 4.3 illustrates the way to achieve the generalized dynamic programming matrix from the tuples' point of view. To begin with, the first row and column of the matrix are filled in with zeros. Then, iteratively fill the rest of cells of the matrix with $H_{x,y}$ obtained from

$$\begin{aligned} H_{x+1,y+1} &= \max_{1 \le i \le x, 1 \le j \le y} (I(T_{A,i:x}, T_{B,j:y}), 0) \\ &= \max\{H_{x,y} + \mathbb{S}(T_{A,x+1}, T_{B,y+1}), \\ &\qquad H_{x,y+1} + \delta,\ H_{x+1,y} + \delta,\ 0\}, \end{aligned} \tag{4.8}$$

where the negative parameter δ is the gap penalty, and the similarity index $I(T_{A,i:x}, T_{B,j:y})$ is the alignment score of the segment pair $(T_{A,i:x}, T_{B,j:y})$.

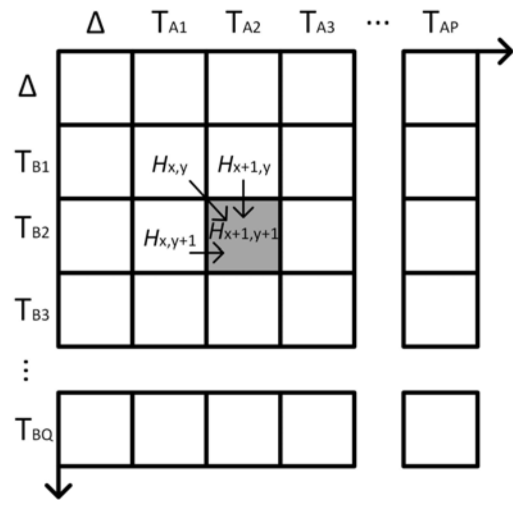


Figure 4.3: An illustrative example on how to obtain the generalized dynamic programming matrix

Once the whole dynamic programming matrix is achieved, the optimal alignment of the two alignments can be generated based on back-tracking. During back-tracking, the position of the maximum value in the dynamic programming matrix is located at first. Then, the algorithm tracks backwards to obtain the path along which the maximum score is generated. At last, one forward pass through the back-tracking path gives the optimal alignment.

The difference between the generalized and the standard dynamic programming procedures is that the former one treats the “sequence” from a “tuple” point of view, which allows the “sequence” to be either an alarm sequence or an alignment of multiple alarm sequences. Thus, the standard dynamic programming procedure can be seen as a special case of the generalized one.

4.2.3 Progressive Multiple Sequence Alignment Method

The computational cost of the traversal search approach in [62] grows exponentially with the number of sequences and their lengths, making the algorithm easily overwhelmed by real data. To improve the efficiency, we propose a progressive multiple sequence alignment method based on the generalized dynamic programming to find the optimal sequence alignment.

In the first step of the proposed method, the pairwise pattern matching algorithm in [18] is applied to calculate all the pairwise similarity scores between the alarm sequences. Then, a dendrogram is built based on these pairwise similarity scores using UPGMA (Unweighted Pair Group Method with Arithmetic Mean). Figure 4.4 shows an example of a dendrogram, in which there are 6 alarm sequences and the height of a link indicates the dissimilarity between two connected sequences.

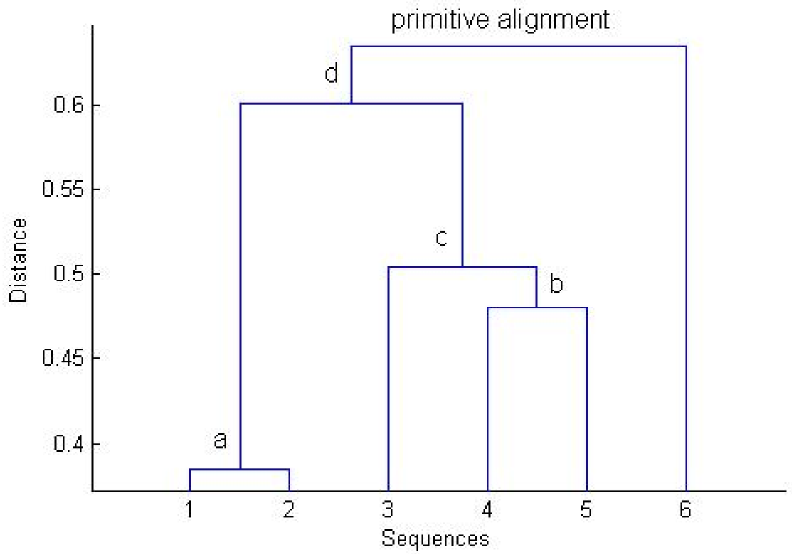


Figure 4.4: An example of dendrogram and the progressive multiple sequence alignment procedure based on it

Next, guided by the dendrogram, the primitive alignment of the sequences is obtained by progressively aligning all the sequences, from the most similar pair to the most dissimilar ones. In the example shown in Figure 4.4, the link connecting sequences 1 and 2 is the lowest one, indicating them to be the most similar sequence pair. Thus, these two sequences are aligned first and thus their alignment *a* is obtained. Similarly, sequences 4 and 5 are aligned and thus alignment *b* is obtained. Then, since there is a connection between sequence 3 and alignment *b*, the generalized pairwise sequence alignment method is conducted to get alignment *c* by aligning sequence 3 and alignment *b*. By iteratively aligning the sequences based on the dendrogram, the primitive alignment of all the 6 alarm sequences can be achieved.

Compared to the traversal search approach in [62], which conducts traversal search search to find the exact optimal alignment, the proposed method approximates the optimal alignment by progressively aligning all the sequences. This new approach greatly improves the efficiency of the algorithm; as the cost, however, the global optimum is no longer guaranteed.

4.2.4 Iterative Alignment Refinement Methods

Two iterative alignment refinement methods are developed to improve the accuracy of the primitive alignment generated by the progressive sequence alignment approach.

Leave-One-Out Alignment Refinement

Similar to the leave-one-out cross validation, which leaves one dataset out for testing and use the rest for training, during each iteration of the leave-one-out alignment refinement, one sequence in the alignment is replaced by its original sequence and then be re-aligned with the rest of sequences in the alignment (if any, common gaps in the rest of the alignment sequences should be removed) using the generalized pairwise sequence alignment method. After re-aligned, the new alignment score is compared with the old one before the current refinement iteration, and the alignment with the higher score will be kept for the next iteration. Iteratively repeat this procedure until the alignment score converges.

Figure 4.5 shows an illustration of one iteration of the leave-one-out refinement; respectively, the solid lines and "[ ]" represent alarm sequences and gaps in the alignments. In the example, alignment 1 is left out and replaced by its original alarm sequence, as indicated by the red bold line. Then the common gaps in the rest of the alignments $(2, 3, 4,$ and $5)$ are removed before they are re-aligned with the original alarm sequence 1.

Note the convergence is guaranteed by the leave-one-out refinement method. However, it is not guaranteed to converge to the global optimum. Details of convergence will be provided in the discussion section.

Random-division alignment refinement

The random-division alignment refinement can also improve the accuracy of an alignment. Compared to the leave-one-out approach, it has a relatively lower convergence speed, but in some cases, it could still improve the alignment

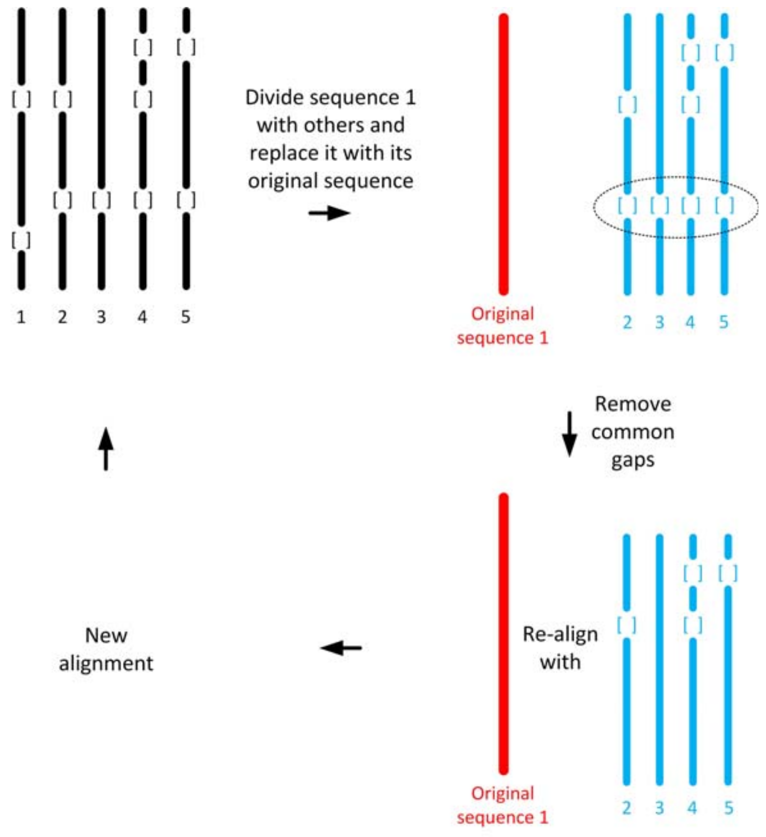


Figure 4.5: Illustration of the Leave-one-out refinement method

result even when the leave-one-out alignment converges. The detailed reason will be provided in the discussion section.

As illustrated in Figure 4.6, the primitive alignment is randomly divided into two groups (1, 2, 4, and 3, 5). Then, if any, the common gaps in each of the two groups of sequences should be removed, as shown in the example. Next, the two groups of sequences are re-aligned using the proposed generalized pairwise sequence alignment method. After re-aligned, the new alignment score is compared with the old one before the current refinement iteration, and the alignment with the higher score will be kept for the next iteration. Iteratively repeat this procedure until the upper limit number of iterations is reached.

Similar to the leave-one-out refinement, the convergence is guaranteed for the random-division approach when no upper limit iteration number is set. However, the convergence is hard to be identified without knowing the optimal alignment score beforehand, also the convergence value is not guaranteed

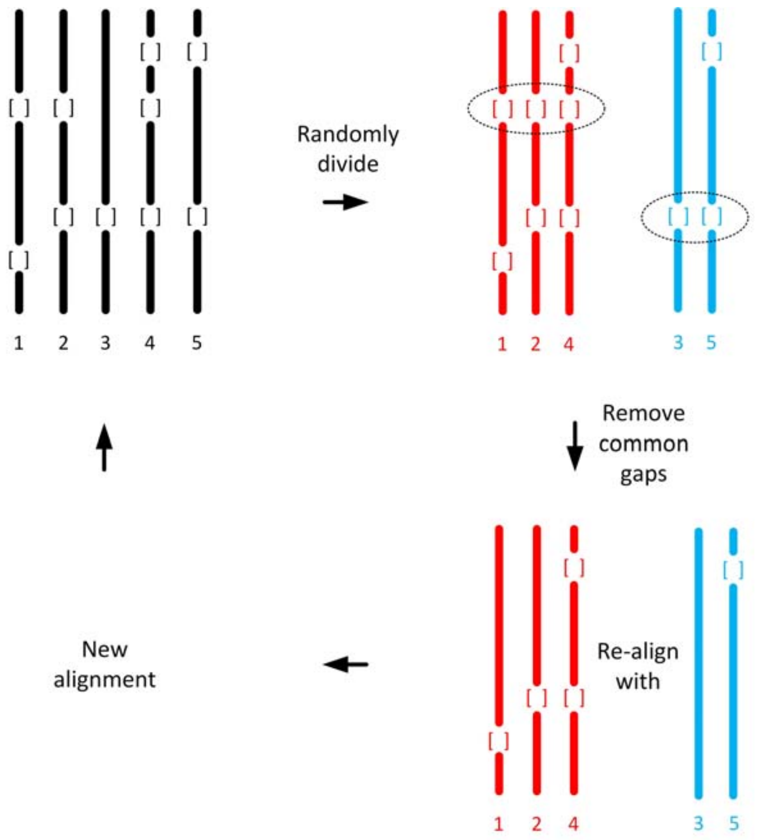


Figure 4.6: Illustration of the Random-division refinement method

to be global optimum. Moreover, since no sequence in the alignment will be replaced by its original sequence during the refinement, the primitive alignment becomes much more critical; and the deleted alarm messages in the sequence of an alignment will never be brought back.

4.3 Industrial Case Study

The proposed algorithm has been tested on an industrial dataset from Suncor Energy Inc. The dataset comes from a plant that is already in production but need rationalization to deal with chattering alarms and alarm floods. Off-delay timers of 300 seconds have been applied as a preprocessing step at the beginning of the analysis to remove chattering alarms. Alarm floods were extracted based on the ISA standard: 10 alarms per 10 minutes per operator. General descriptions of the dataset are shown in Table 4.1.

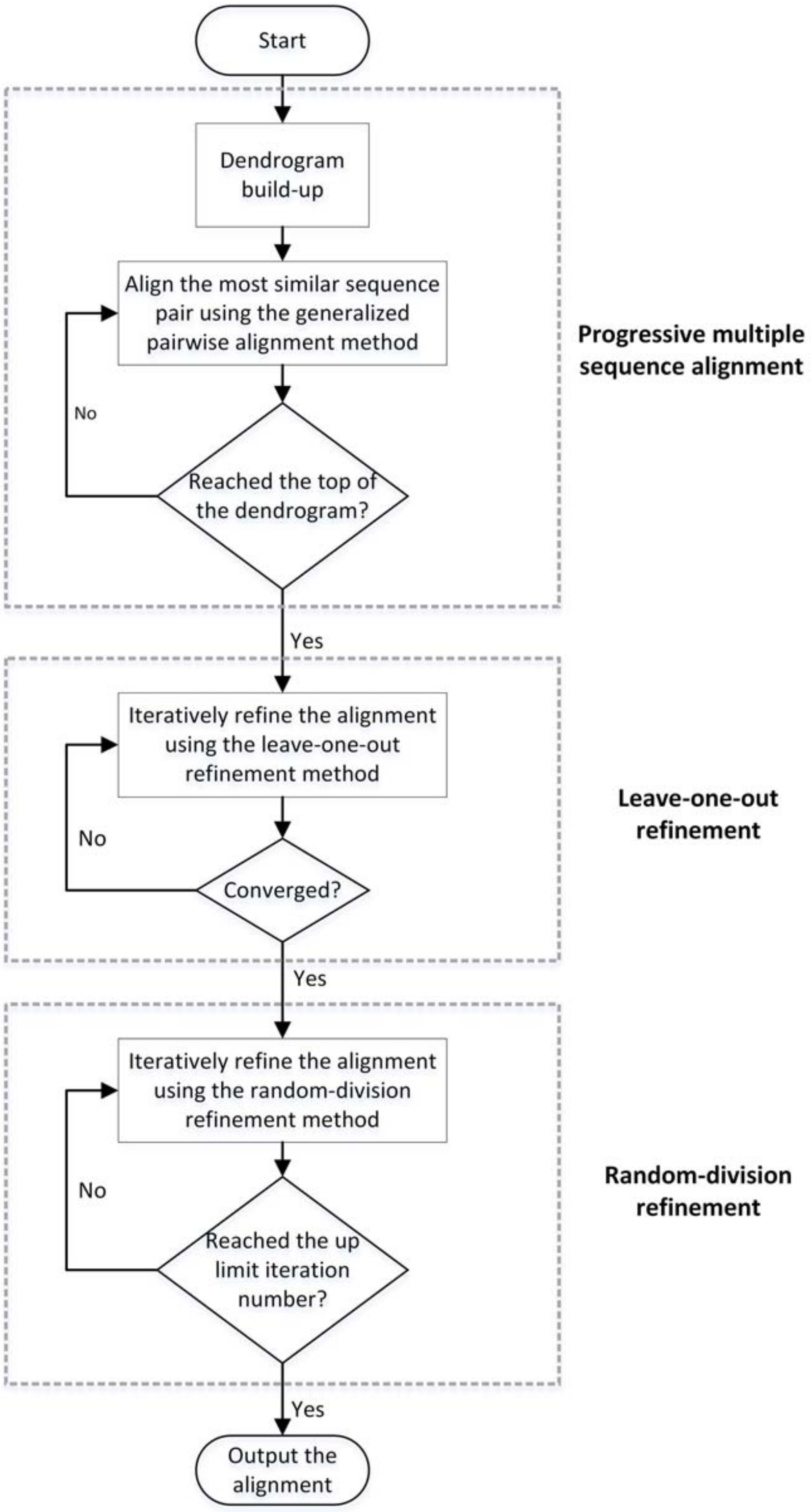

Start
Dendrogram build-up
Align the most similar sequence pair using the generalized pairwise alignment method
No
Reached the top of the dendrogram?
Progressive multiple sequence alignment
Yes
Iteratively refine the alignment using the leave-one-out refinement method
No
Converged?
Leave-one-out refinement
Yes
Iteratively refine the alignment using the random-division refinement method
No
Reached the up limit iteration number?
Random-division refinement
Yes
Output the alignment

Figure 4.7: Flowchart of the proposed algorithm

The flowchart of the algorithm is shown in Figure 4.7. The parameters were chosen as: $\sigma = 0.2$, $\mu = -1$, $\delta = -1$, and the upper limit iteration number for random-division refinement is set as 50. All the tests were carried out on a 64-bit Windows PC with Intel i7-4770 3.40GHz CPU and 24.0 GB memory.

Table 4.1: Statistics of the dataset

Description	Number
Total time period	336 days
Total number of tags	1502
Total number of alarms	109393
Average alarm rate	14/h
Highest peak alarm rate	334/10 min
Number of alarm floods	359
Average length of alarm floods	39

Pairwise similarity scores were calculated using the method in [18] and the UPGMA clustering was conducted based on the obtained similarity scores thereafter. Similar to the proposed approach, the method in [18] finds the similarity score between a pair of sequences by searching for their best alignment. It is able to find the exact optimal alignment; however, it can only be applied on a pair of sequences rather than multiple ones. The clustering result is shown in Figure 4.8, where each pixel represents the similarity score between two corresponding alarm flood sequences; darker color means a higher similarity. Four groups of alarm floods were selected to be the test dataset, denoted as clusters A, B, C, and D. The reason for choosing them as the test datasets was because they each contains sufficient number of alarm floods for testing, and a pattern of alarm sequence exists in each of the clusters. Detailed information about the four selected clusters can be found in Table 4.2.

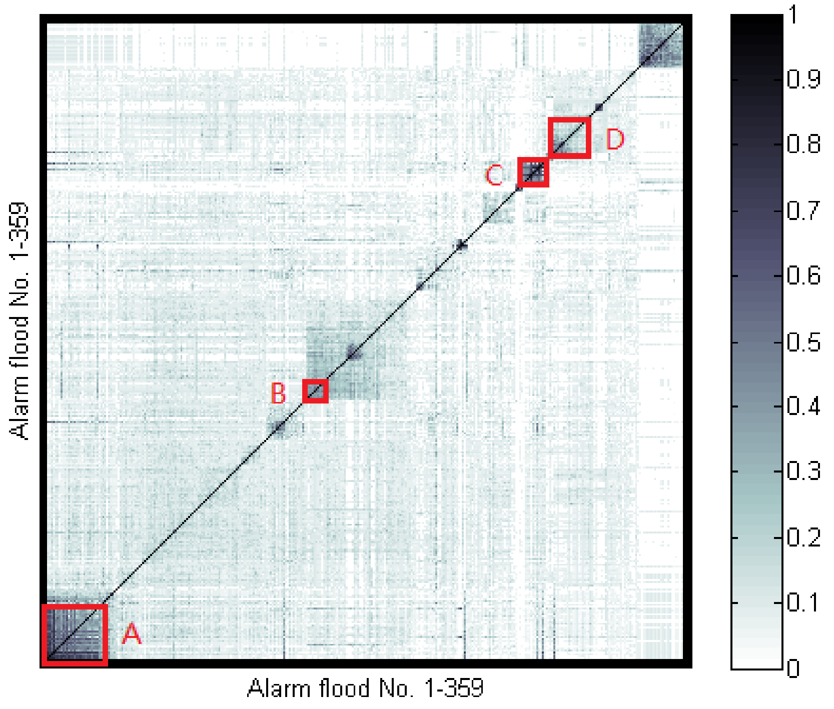


Figure 4.8: Clustering result of the extracted alarm floods and the four selected clusters

Table 4.2: Information of the alarm sequences in each selected cluster

Cluster	A	B	C	D
Number of alarm floods	19	12	9	17
Average sequence length	26.6	14.6	31.6	20.1
Longest sequence length	46	23	64	36
Shortest sequence length	15	10	14	10
Pattern length	14	4	11	2

Both accuracy and efficiency tests have been conducted to compare the proposed algorithm with the traversal search approach in [62]. Same parameters were set for both algorithms.

4.3.1 Accuracy Tests

From each of the four clusters, 3 sequences were randomly selected, and the two algorithms were applied respectively to find the best alignment. Since the traversal search approach guarantees the global optimum, the accuracy of the proposed algorithm can be evaluated by the alignment score difference between the two algorithms. Repeat the whole procedure for 50 times and obtain the average alignment score difference for each cluster; the result is shown by the green line in Figure 4.9. From the left to the right, the four points on the green line indicates the average alignment score difference between the two algorithms in clusters B, D, A, and C respectively (in the ascendant order of the average sequence length in each cluster). Similarly, the average alignment score difference between the two algorithms was obtained when 4 and 5 sequences were randomly selected from each of the four clusters, presented by the red and blue lines respectively.

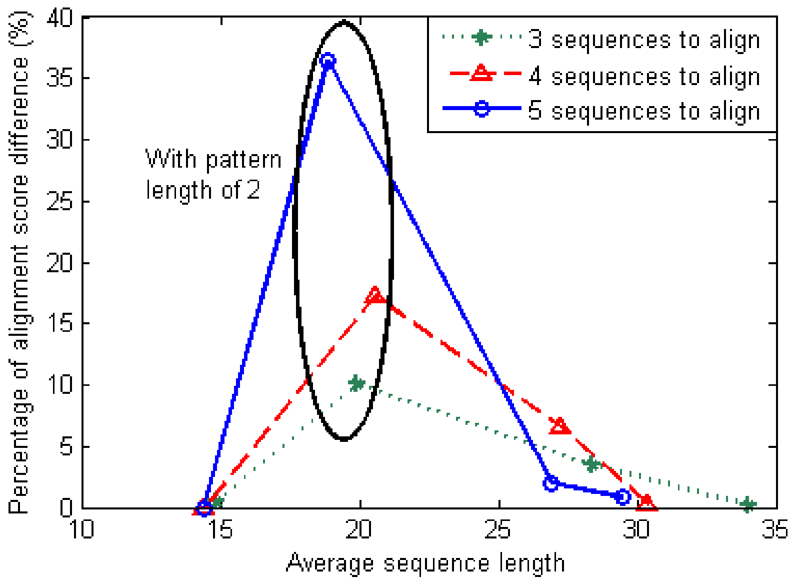


Figure 4.9: Percentage of alignment score difference between the proposed algorithm and the traversal search approach

A very noticeable spike can be found in all the three lines in Figure 4.9, showing that the average alignment score difference between the two algorithms was large for the tests in cluster D. The reason for this is because the pattern length in cluster D was too short (only 2). Thus, the alignment can be easily influenced by the similar terms among the sequences that are not related to the pattern. Also, since the pattern length is too short, the percentage of alignment score difference caused by a small defect in the alignment would be amplified. In addition to the big spike caused by cluster D, the accuracy of the proposed algorithm is high, with no more than 10% difference compared to the traversal search approach. Figure 4.9 also shows that there is no evident relation between the accuracy of the proposed algorithm and number of sequences to be aligned or the average sequence length.

4.3.2 Efficiency Tests

Execution times of both algorithms during the accuracy tests were recorded to compare their efficiency. Figure 4.10 shows the efficiency of the two algorithms as the number of sequences to be aligned grows. The execution times of both algorithms increases when the number of sequences grows; however, the growth rate of the proposed algorithm is much smaller than that of the traversal search approach. When the number of sequences reached 5, the proposed algorithm was almost 10,000 times faster than the traversal search approach. Figure 4.11 indicates there is a positive relation between the computational cost of the traversal search approach and the average sequence length, but this trend is not evident for the proposed algorithm. The reason is because the pattern length, as numbered on the figure, influences the execution time of the proposed algorithm as well. Detailed explanation will be provided in the discussion section.

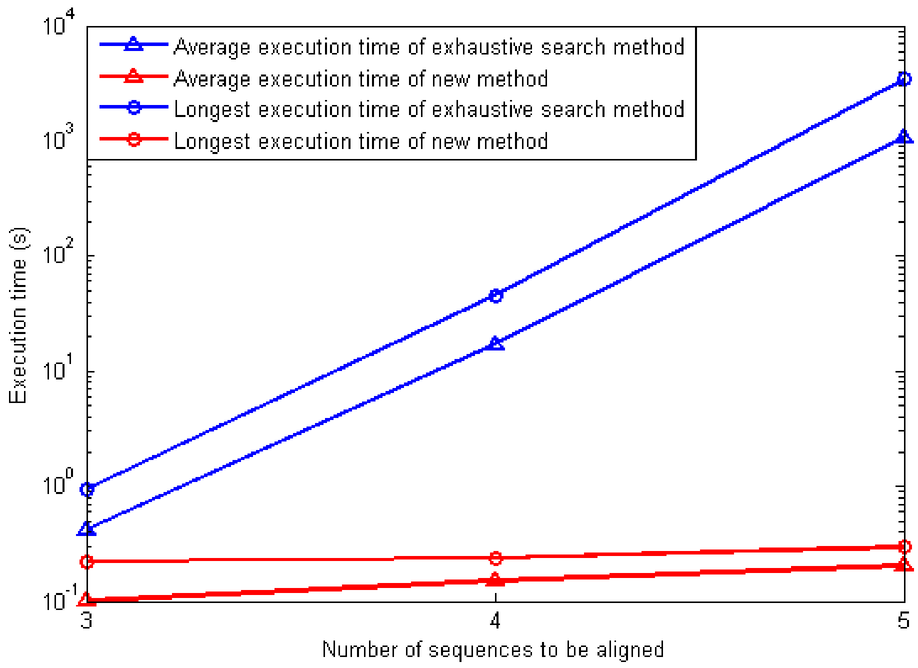


Figure 4.10: Efficiency comparison of the two algorithms when the number of sequences increases

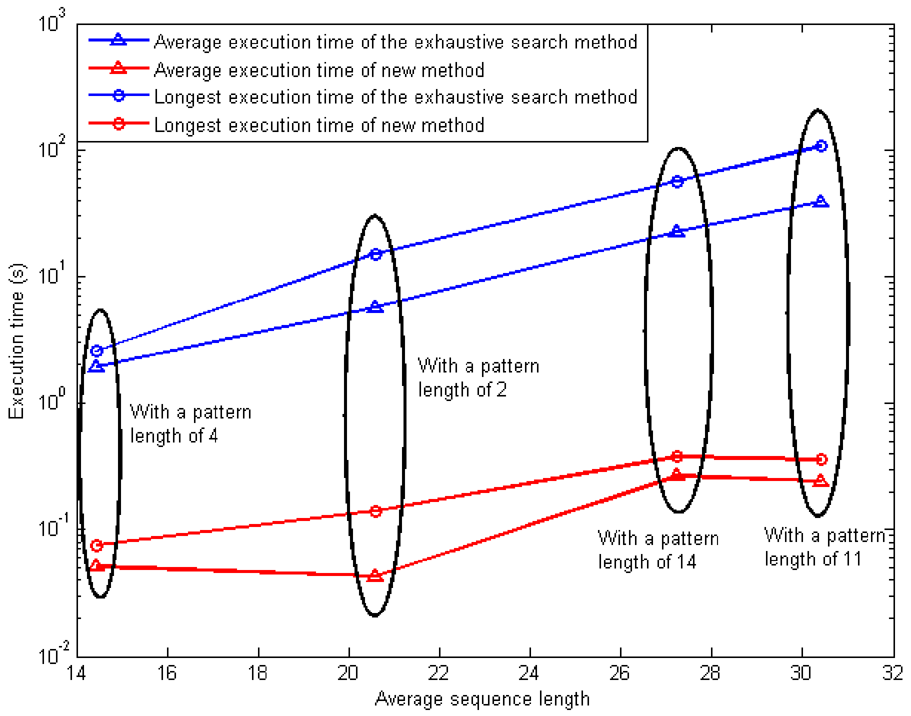


Figure 4.11: Efficiency comparison of the two algorithms when the average sequence length increases

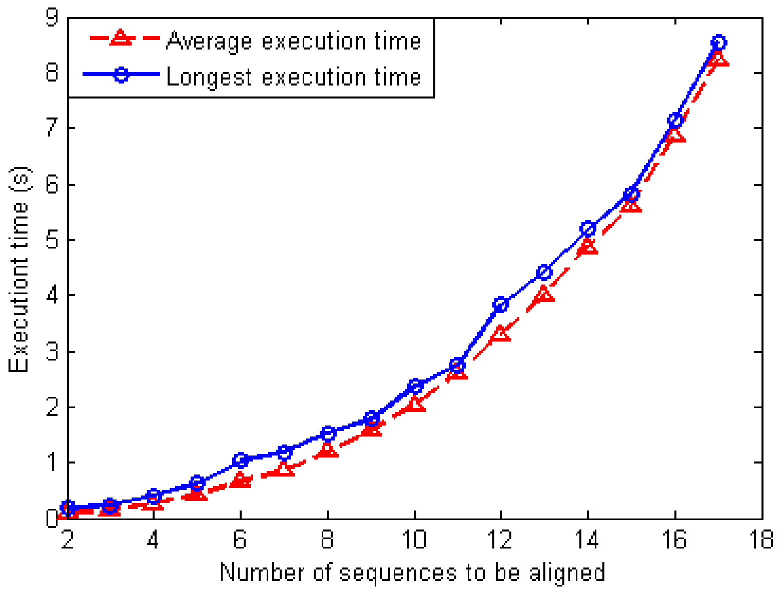


Figure 4.12: Execution time of the proposed algorithm when the number of sequences grows

More efficiency tests were conducted to cover the testing region that had not been touched by the accuracy tests. Since the traversal search approach could hardly finish in most of the tests, only the result of the proposed algorithm is shown.

Figure 4.12 shows the execution time of the proposed algorithm when the number of sequences to be aligned increases. The tests were conducted by randomly selecting sequences from cluster A (average sequence length 26.6); 50 repeated tests were carried out for each number of sequences. The result reveals a quadratic trend in the computational cost as the number of sequences to be aligned grows. Even when the sequence number exceeds 250, as shown in Figure 4.13, the execution time of the proposed algorithm was still within 1 minute.

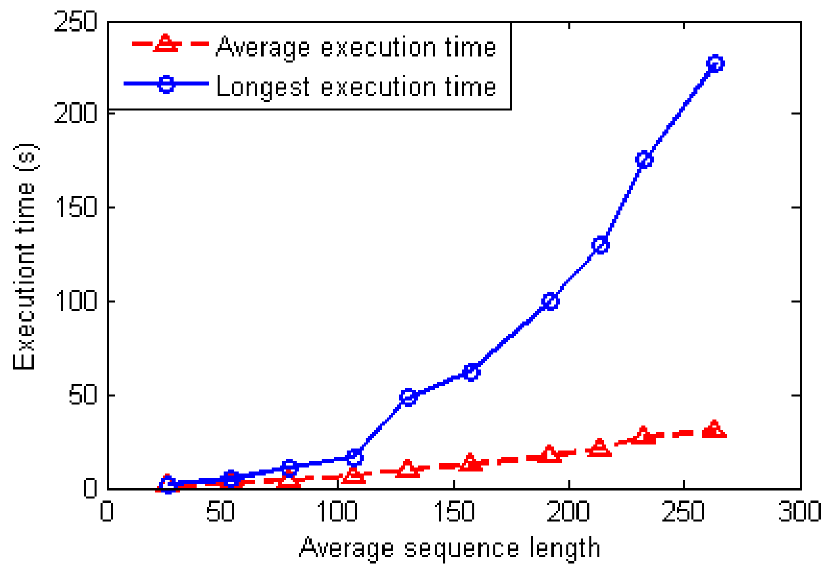


Figure 4.13: Execution time of the proposed algorithm to align 10 sequences as the average sequence length grows

Figure 4.13 reveals the relationship between execution time of the algorithm and average sequence length. The tests were carried out by applying the proposed algorithm to align 10 alarm sequences randomly selected from cluster A. In order to increase the sequence length in each test, the sequences were extended by duplicating and connecting themselves with their duplicates. Each test was repeated 50 times and both the average and the longest execution time were recorded. The result shows the longest execution time grows quite fast with the growth of average sequence length; however, the average execution time grows much slower and is always less than 50 seconds. The reason is because the sequence length in cluster A varies from 15 to 46 and thus the sequence growth rates were quite different when the sequences were extended by duplicating themselves.

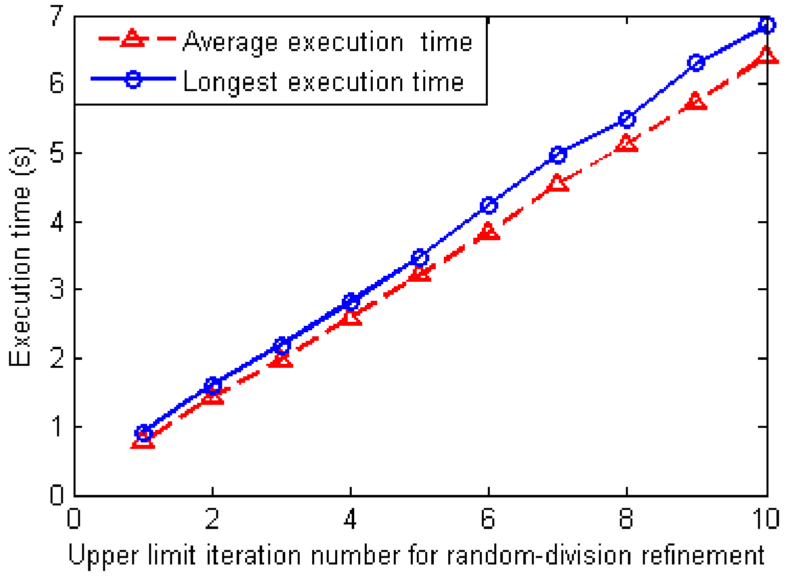


Figure 4.14: Execution time of the algorithm with respect to the upper limit iteration number for the random-division refinement

Figure 4.14 shows the influence of the upper limit of iteration number for the random-division refinement on the execution time of the algorithm. The results were obtained by aligning 10 sequences randomly selected from cluster A, with different upper limits of iteration numbers for the random-division refinement in the algorithm. Each test was repeated 50 times and both the average and the longest execution times were recorded. The result shows a linear increasing trend of the execution time of the algorithm when the upper limit increases, as is expected. The detailed reason will be explained in the discussion section.

4.3.3 Convergence Tests

Tests for the convergence of the two refinement methods were conducted and the results are shown in Figures 4.15 - 4.17.

Since the convergence speed of the random-division refinement is much slower than the leave-one-out refinement and can hardly be identified, only the convergence speed of the leave-one-out refinement was tested. Figure

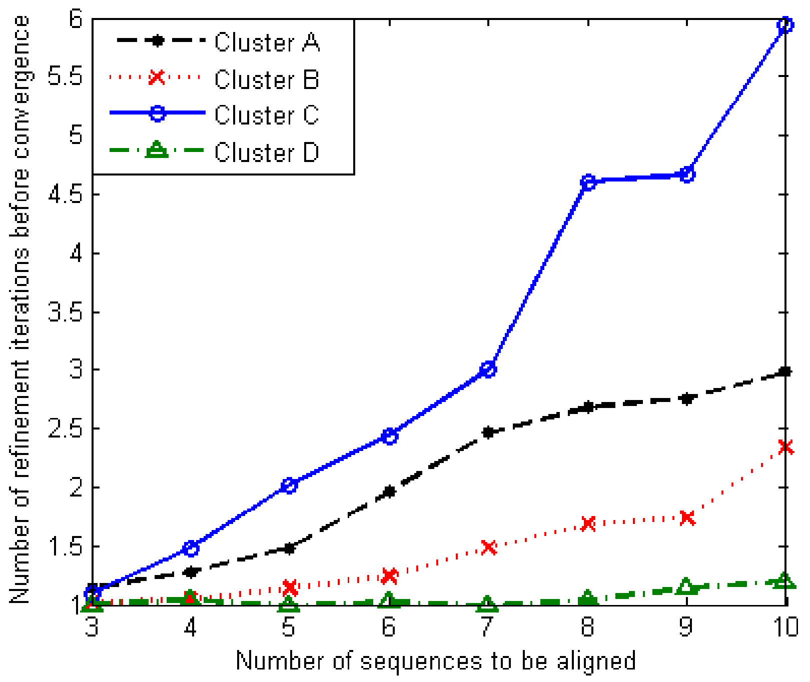


Figure 4.15: The number of refinement iterations needed before reaching the convergence when the number of sequences to be aligned increases.

4.15 shows the refinement iteration number needed before the convergence was reached by leave-one-out refinement when aligning different numbers of sequences in each cluster. As the result shows, more iterations were needed when the number of sequences to be aligned was large. However, there is no clear relationship between the refinement iteration number needed for convergence and the average sequence length, as the descending order of average sequence length in each cluster is $C > A > D > B$.

The two refinement methods were also compared by evaluating the amount of improvement they can make on a given primitive alignment. Since for a certain primitive alignment, the leave-one-out refinement always has the same convergence trail, while the random-division method doesn't, the final alignment score of the leave-one-out refinement was compared with the results from 30 runs of the random-division refinement. Two cases where the two methods achieved a higher alignment score compared with each other are shown in Figure 4.16. The red bold lines represent the converging trails of the leave-one-out method, while the blue lines show the trails of the random-

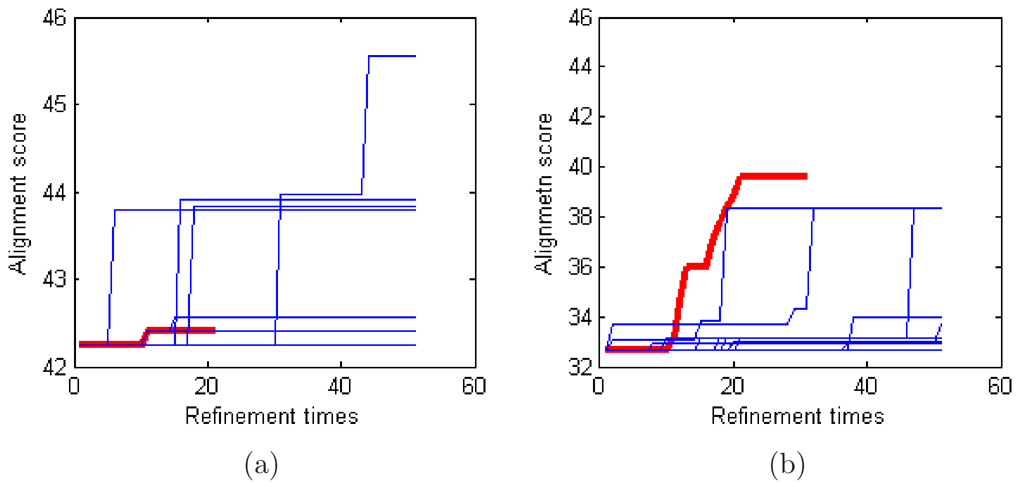


Figure 4.16: Examples of the optimums achieved by the two types of refinement methods: (a) when the random-division refinement method was better; (b) when the leave-one-out refinement method was better.

division method. In case (a), the leave-one-out refinement was stuck in a local optimum and outperformed by most of the runs of the random-division refinement. However, in case (b), the leave-one-out refinement was better than all runs using the random-division refinement. The reason for this was because the leave-one-out refinement method could recover from the incorrectly deleted part for the primitive alignment, but the random-division refinement method could not. Thus, during the procedure of forming the primitive alignment, if a part of the sequence that should appear in the optimal alignment was deleted incorrectly because of the approximation, then we should expect it to be recovered by the leave-one-out refinement method rather than the random-division refinement.

When applying the leave-one-out refinement method, for each iteration we always select the sequence by their orders in the primitive alignment. However, will the order we select the sequences to be left out make a difference in the final alignment given by the leave-one-out refinement method? The answer is yes, as the result in Figure 4.17 shows, where 10 different orders were tried to improve a certain primitive alignment using the leave-one-out refinement method. The result shows the order we use to select the sequences will influence not only the convergence value but also the speed for the leave-one-out refinement method.

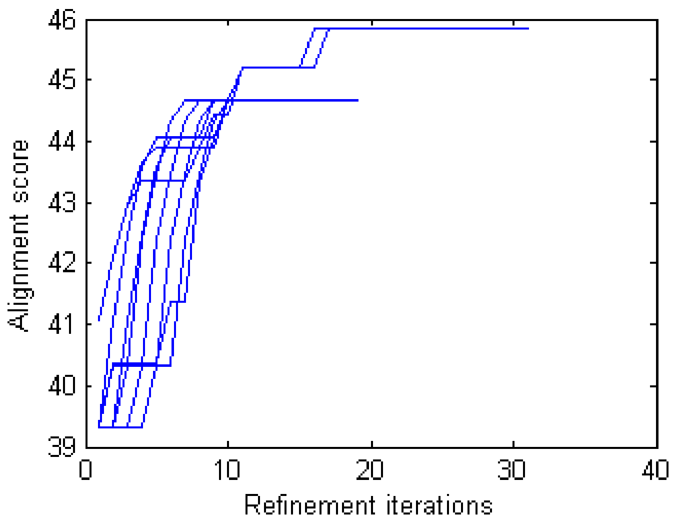


Figure 4.17: An example of convergence curves when the order of sequences to be left out for the leave-one-out refinement method varies

4.4 Discussions

Compared to the traversal search approach in [62], the proposed algorithm does not guarantee the global optimum. However, as shown in the industrial case study section, the accuracy of the proposed algorithm is acceptable (within 10% range) when the pattern length is not too small (larger than 3). For the cases where the pattern sequence is too short, the algorithm may not perform well since the alignment can be easily influenced by the similar terms among the sequences that are not related to the pattern, and the improvement on the primitive alignment that could be made by the two refinement methods will be very limited as well.

The computational complexity of the proposed algorithm is composed of five parts: (1) calculation of time weight matrices $\mathcal{O}(\sum_{i=1}^{N} L_i^2)$, where N is the sequence number and L_i is the sequence length; (2) formation of the dendrogram $\mathcal{O}(N^3)$; (3) generation of the primitive alignment using progressive multiple sequence alignment method $\mathcal{O}(\sum_{j=1}^{N-1} A_{j1}A_{j2})$, where A_{j1} and A_{j2} are the lengths of the two sequences (can be either a sequence or an alignment) for the generalized pairwise sequence alignment; (4) leave-one-out refinement $\mathcal{O}(\sum_{k=1}^{I_1} A_{k1}A_{k2})$, where I_1 is the number of iterations before convergence is reached, and A_{k1} and A_{k2} are the lengths of the two sequences (can be either a sequence or an alignment) for the generalized pairwise sequence alignment;

and (5) random-division refinement $\mathcal{O}(\sum_{t=1}^{I_2} A_{t1}A_{t2})$, where I_2 is the upper limit iteration number, and A_{t1} and A_{t2} are the lengths of the two sequences (can be either a sequence or an alignment) for the generalized pairwise sequence alignment. Thus, the whole computational complexity is

$$\mathcal{O}(\sum_{i=1}^{N} L_i^2 + N^3 + \sum_{j=1}^{N-1} A_{j1}A_{j2} + \sum_{k=1}^{I_1} A_{k1}A_{k2} + \sum_{t=1}^{I_2} A_{t1}A_{t2}).$$

Based on the computational complexity analysis, the efficiency of the proposed algorithm is determined by many factors, among which the number of sequences, sequence length, and alignment length are the three main elements. The highest order of sequence or pattern length in the computational complexity is only 2 for the proposed algorithm. Notice the computational complexity of the traversal search approach in [62] is

$$\mathcal{O}(2 \cdot (\sum_{k=1}^{N} L_k) \times (\prod_{k=1}^{N} L_k)),$$

which increases exponentially with the sequence number and length. Thus, the proposed algorithm achieves a significant improvement in efficiency, which has also been confirmed in the industrial case study section.

The convergence can be guaranteed by both refinement methods. The proof is simple and does not require any mathematical derivations. Since in every refinement iteration, the new alignment score will be compared with the old one, and the alignment with a higher score will be fed to the next iteration. Thus, the alignment score is monotonically increasing, while there must be an upper bound for the alignment score of certain sequences. Therefore, the convergence is guaranteed for both refinement methods. The leave-one-out refinement method converges when no more improvement can be made to the alignment for a full round of iterations (every sequence has been left out once). However, the convergence of the random-division refinement can hardly be identified, as there does not exist a full round of iterations. Thus, an upper limit number of iterations should be configured for the random-division refinement method.

Convergence speeds of the two refinement methods are worthy studying too. The leave-one-out refinement method usually, but not always, converges faster than the random-division method. The reason is because for both of the refinement methods, there are two main steps: separate the alignment into two bunches of sequences, and re-align them. The leave-one-out refinement method has a certain order to separate the alignment, and only one sequence is separated from the rest during one iteration. While the random-division refinement method does not; thus, there is a much bigger pool of ways of separating the sequences in the alignment that the random-division refinement method can select from.

Regarding the convergence value, both refinement methods cannot guarantee the global optimum, and neither of them can always achieve a better alignment score than the other. However, by combining them together we could have a better chance to reach the global optimum, because each method improves the alignment from a different perspective. The leave-one-out refinement method can recover the part of the sequence that has been incorrectly deleted during the generation of the primitive alignment because in each iteration, one sequence in the alignment is replaced by its original sequence and re-aligned with the rest of the sequences in the alignment. However, the random-division refinement method cannot do so because no separated sequences will be replaced by their original ones. Even though it cannot recover the incorrectly deleted sequence part, it still has an advantage over the leave-one-out refinement method — a much bigger pool of ways of separating the sequences in the alignment to choose from, which means more chances to improve the alignment by re-arranging the gaps and matchings. Thus, by applying the leave-one-out refinement followed by the random-division refinement method, we will be able to first recover the incorrectly deleted sequence parts, and then modify the ways of placing gaps and matchings in the existing alignment to achieve a better alignment score. Therefore, in the algorithm flowchart shown in Figure 4.7, the leave-one-out refinement is applied before the random-division refinement.

Parameter tuning can be an obstacle for applying the proposed algorithm. As mentioned in [18], the variance of the Gaussian function σ, the gap and mismatch penalties δ and μ need to be tuned to meet users' requirements. The variance of the Gaussian function σ controls the size of the time span within which the algorithm blurs the occurrence orders between the alarm messages. When the value of σ goes to infinity, the algorithm totally ignores the orders of alarms in the alignment and simply counts the alarm occurrences. On the contrary, if $\sigma = 0$, the orders of alarms must be exactly the same in the two sequences in order to get a match. The gap and mismatch penalties, σ and μ, determine the algorithm's tolerance to including gaps and mismatch terms in alignments. When these two parameters get larger, the algorithm places more tolerance on the irrelevant alarms raised within a pattern sequence. Besides the ones in [18], a new parameter is introduced in the proposed algorithm: the upper limit number of iterations I_2 for the random-division refinement. With a higher I_2, there will be a greater chance that the global optimum is reached, but at the cost of higher computational cost.

During the generation of the primitive alignment, the sequences are progressively aligned by the descending order of their pairwise similarity. The reason for choosing this order is because once an alignment is obtained, the gaps and the matchings between the two sequences are fixed and will be propagated into the primitive alignment. Thus, by aligning the most similar sequence pair first and leaving the most divergent pair to the last would improve the correctness of placing gaps and matchings in the alignment.

4.5 Summary

In this chapter, an accelerated multiple sequence alignment algorithm for pattern mining in multiple alarm sequences has been proposed. A progressive multiple sequence alignment mechanism has been introduced to accelerate the generation of the primitive alignment. Two types of refinement methods have been developed to improve the primitive alignment. A dataset from a real chemical plant has been used to test the effectiveness of the proposed algorithm and compare it with the traversal search approach in [62]. The results has shown that the proposed algorithm has a significant improvement in efficiency with a small accuracy cost.

Chapter 5

Application of Pattern and Causality Analysis on Alarm Floods

5.1 Overview

As discussed in the previous chapters, the parameter selection and tuning for the proposed pattern mining and pattern matching algorithms can be tough. In this chapter, parameter robustness tests are conducted using industrial data to reveal the insights on how those parameters affect the results of the algorithms. Specifically, robustness tests for the influence of the parameters σ, μ, and δ on the accelerated pattern mining algorithm proposed in Chapter 4 are carried out. Moreover, an extended accuracy test with both α and σ as the manipulated variables is conducted for the online pattern matching algorithm proposed in Chapter 3. Discussions on parameter selection and tuning are also provided.

In addition to the robustness tests, an industrial case study on the application of Granger causality towards the root cause analysis in alarm floods is carried out. Datasets from two consoles in an operating utility plant are used in the study. The causal maps generated from the frequency domain Granger causality are confirmed by process diagrams from the plant manual.

5.2 Robustness Test of Parameters in Pattern Mining and Pattern Matching Algorithms

The computational complexity analysis in the previous chapters showed that the efficiency of the three proposed algorithms did not depend on the value of their parameters. Thus, the parameter robustness tests carried out in this section are mainly focused on the influence of parameters on the alignment result and accuracy. Datasets from a real chemical plant are used for parameter robustness test of the pattern mining and pattern matching algorithms proposed in Chapters 3 and 4. At the end of the section, discussions on how to select and tune the algorithm parameters are provided.

5.2.1 The Influence of Parameters σ, μ, and δ on the Alignment Result

Parameter robustness tests were conducted on the same dataset used in the industrial case study in Chapter 4 to show the influence of parameters σ, μ, and δ on the alignment output of the accelerated pattern mining algorithm proposed in Chapter 4. Since the three algorithms proposed in Chapters 2-4 are all based on the similar alignment technique, the results from the carried-out robustness tests can represent the influence of parameters on all three algorithms. Three alarm floods from cluster A in Figure 4.8 were used as the input of the accelerated pattern mining algorithm.

Table 5.1: Alignment result with $\sigma = 2$, $\mu = -0.6$, and $\delta = -0.4$.

AF 1	Time	AF 2	Time	AF 3	Time
Tag355	00:03:04	Tag355	22:21:45	Tag355	19:16:11
Tag352	00:03:59	Tag356	22:21:49	Tag354	19:16:17
Tag360	00:04:00	[]	[]	[]	[]
Tag350	00:04:00	[]	[]	Tag357	19:16:17
Tag351	00:04:01	Tag357	22:21:56	Tag352	19:16:17
Tag357	00:04:02	[]	[]	Tag351	19:16:21
Tag356	00:04:04	Tag358	22:22:00	Tag358	19:16:24
[]	[]	[]	[]	Tag353	19:16:26
Tag359	00:04:05	[]	[]	Tag356	19:16:28
Tag358	00:04:05	Tag359	22:22:05	Tag359	19:16:31

For the first set of three tests, μ and δ were fixed as -0.6 and -0.4, and the manipulated variable σ was set as 0.1, 2, and 10 in each test. Alignment results are shown in Tables 5.1, 5.2, and 5.3. By comparing the alignments in the three tables we can notice that as the value of σ grows from 0.1 to

Table 5.2: Alignment result with $\sigma = 10$, $\mu = -0.6$, and $\delta = -0.4$.

AF 1	Time	AF 2	Time	AF 3	Time
Tag354	00:03:03	Tag350	22:21:20	Tag355	19:16:11
Tag355	00:03:04	Tag351	22:21:25	Tag354	19:16:17
Tag352	00:03:59	Tag352	22:21:28	Tag357	19:16:17
Tag360	00:04:00	Tag353	22:21:34	Tag352	19:16:17
Tag350	00:04:00	Tag354	22:21:38	[]	[]
Tag351	00:04:01	Tag355	22:21:45	[]	[]
Tag357	00:04:02	Tag356	22:21:49	Tag351	19:16:21
Tag356	00:04:04	Tag357	22:21:56	Tag358	19:16:24
Tag359	00:04:05	Tag358	22:22:00	Tag353	19:16:26
Tag358	00:04:05	Tag359	22:22:05	Tag356	19:16:28
Tag353	00:04:07	Tag360	22:22:09	Tag359	19:16:31

Table 5.3: Alignment result with $\sigma = 0.1$, $\mu = -0.6$, and $\delta = -0.4$.

AF 1	Time	AF 2	Time	AF 3	Time
Tag355	00:03:04	Tag355	22:21:45	Tag355	19:16:11
Tag352	00:03:59	Tag356	22:21:49	Tag354	19:16:17
Tag360	00:04:00	[]	[]	[]	[]
Tag350	00:04:00	[]	[]	[]	[]
Tag351	00:04:01	[]	[]	Tag357	19:16:17
Tag357	00:04:02	Tag357	22:21:56	Tag352	19:16:17
Tag356	00:04:04	[]	[]	[]	[]
[]	[]	[]	[]	Tag351	19:16:21
Tag359	00:04:05	Tag358	22:22:00	Tag358	19:16:24
[]	[]	[]	[]	Tag353	19:16:26
[]	[]	[]	[]	Tag356	19:16:28
Tag358	00:04:05	Tag359	22:22:05	Tag359	19:16:31

10, the number of tags included in the alignment increases and the amount of gaps decreases. The reason of this result is very clear: the variance of the Gaussian function σ controls the size of the time span within which the algorithm blurs the occurrence orders between the alarm messages. When $\sigma = 0.1$, the algorithm becomes very strict with the order of alarms in the alignment, many alarm tags were eliminated from the alignment because their appearance orders did not match. This is also the reason why there were more gaps in the alignment results when $\sigma = 0.1$. On the contrary, when $\sigma = 10$, the order of alarms becomes unimportant; this explains why more alarms were included and the amount of gaps was reduced.

Table 5.4: Alignment result with $\sigma = 2$, $\mu = -1$, and $\delta = -0.4$.

AF 1	Time	AF 2	Time	AF 3	Time
Tag357	00:04:02	Tag357	22:21:56	Tag352	19:16:17
Tag356	00:04:04	Tag358	22:22:00	Tag351	19:16:21
[]	[]	[]	[]	Tag358	19:16:24
[]	[]	[]	[]	Tag353	19:16:26
Tag359	00:04:05	[]	[]	Tag356	19:16:28
Tag358	00:04:05	Tag359	22:22:05	Tag359	19:16:31

Parameter μ was chosen as the manipulated variable in the second set of

Table 5.5: Alignment result with $\sigma = 2$, $\mu = -0.1$, and $\delta = -0.4$.

AF 1	Time	AF 2	Time	AF 3	Time
Tag352	00:03:59	Tag350	22:21:20	Tag350	19:16:10
Tag360	00:04:00	Tag351	22:21:25	Tag355	19:16:11
Tag350	00:04:00	Tag352	22:21:28	Tag354	19:16:17
Tag351	00:04:01	Tag353	22:21:34	Tag357	19:16:17
Tag357	00:04:02	Tag354	22:21:38	Tag352	19:16:17
[]	[]	Tag355	22:21:45	Tag351	19:16:21
[]	[]	Tag356	22:21:49	Tag358	19:16:24
Tag356	00:04:04	Tag357	22:21:56	Tag353	19:16:26
Tag359	00:04:05	Tag358	22:22:00	Tag356	19:16:28
Tag358	00:04:05	Tag359	22:22:05	Tag359	19:16:31
Tag353	00:04:07	Tag360	22:22:09	Tag360	19:16:33

tests, where the value of σ and δ were fixed to be 2 and -0.4, and μ was set as -0.1, -0.6, and -1 in each of the three tests. Results are shown in Tables 5.1, 5.4, and 5.5. Comparing the three tables, we can see fewer alarm tags and more gaps were included in the alignments when the value of mismatch penalty μ went from -0.1 to -1. This is because a bigger value on μ puts less penalties on mismatches in the alignments, allowing more alarm tags to be included even if they do not match the alarms in the corresponding positions of other aligned sequences.

Table 5.6: Alignment result with $\sigma = 2$, $\mu = -0.6$, and $\delta = -0.8$.

AF 1	Time	AF 2	Time	AF 3	Time
Tag359	00:04:05	Tag359	22:22:05	Tag359	19:16:31

Table 5.7: Alignment result with $\sigma = 2$, $\mu = -0.6$, and $\delta = -0.1$.

AF 1	Time	AF 2	Time	AF 3	Time
Tag360	00:04:00	Tag350	22:21:20	Tag350	19:16:10
Tag350	00:04:00	Tag351	22:21:25	Tag355	19:16:11
Tag351	00:04:01	Tag352	22:21:28	Tag354	19:16:17
[]	[]	Tag353	22:21:34	[]	[]
[]	[]	Tag354	22:21:38	Tag357	19:16:17
[]	[]	Tag355	22:21:45	[]	[]
[]	[]	Tag356	22:21:49	[]	[]
Tag357	00:04:02	Tag357	22:21:56	Tag352	19:16:17
Tag356	00:04:04	Tag358	22:22:00	Tag351	19:16:21
[]	[]	[]	[]	Tag358	19:16:24
[]	[]	[]	[]	Tag353	19:16:26
Tag359	00:04:05	[]	[]	Tag356	19:16:28
Tag358	00:04:05	Tag359	22:22:05	Tag359	19:16:31

Based on a similar procedure, the third set of tests was carried out with σ and μ fixed at 2 and -0.6, and the manipulated variable δ set as -0.1, -0.4, and -0.8, respectively. The alignment results are shown in Tables 5.1, 5.6, and 5.7. We can notice the manipulation of parameter δ has brought more drastic changes to the alignments, compared to the first two sets of tests, where parameters σ and μ were changed. The explanation for this phenomenon is that decreasing δ put bigger penalties on the gaps, which were critical to extending the length of the alignment. If there are gaps in the middle of an alignment, by increasing the gap penalty (decreasing the value of δ) the alignment may be broken in half or into several segments. The effect of changing gap penalty, however, can be minimum if an alignment does not require any gaps.

5.2.2 An Extended Accuracy Test for the Online Pattern Matching Algorithm with Respect to the Variance of the Gaussian Function σ

The same dataset from the industrial case study in Chapter 3 was used in the extended accuracy test. Most of the test procedures for false and missed detection rates stayed the same, except more prediction threshold values α were considered and the variance of the Gaussian function σ was set as a manipulated variable. By changing the value of σ during the tests, insights were revealed on the effect of parameters.

Table 5.8: Results of the extended accuracy test.

α	$\sigma = 2$ (default)		$\sigma = 0.5$		$\sigma = 8$	
	FDR (%)	MDR (%)	FDR (%)	MDR (%)	FDR	MDR
0.5	17.2	0	15.7	0	18.8	0
1	10.7	0	8.8	0	11.9	0
2	8.5	2.5	7.2	2.5	10.3	0
3	6.6	2.5	5.0	5.0	7.8	2.5
4	3.8	5.0	2.8	5.0	4.7	2.5
5	2.5	32.5	1.9	37.5	3.1	30.0
6	1.6	37.5	1.6	45.0	2.2	32.5
7	1.6	42.5	1.3	52.5	1.6	35.0
8	0	45.0	0	60.0	0	35.0

Table 5.8 shows the extended accuracy test results for the online pattern matching algorithm proposed in Chapter 3. Both the false detection rate (FDR) and the missed detection rate (MDR) are listed for different values of the prediction threshold α and the variance of Gaussian function σ. The table reveals clearly a trade-off between the missed and false detection rates

while changing the prediction threshold. What the results also revealed is the influence of σ on the detection rates: FDR increases and MDR decreases when σ gets larger and vice versa. The data in Table 5.8 is visualized using ROC curves, shown in Figure 5.1.

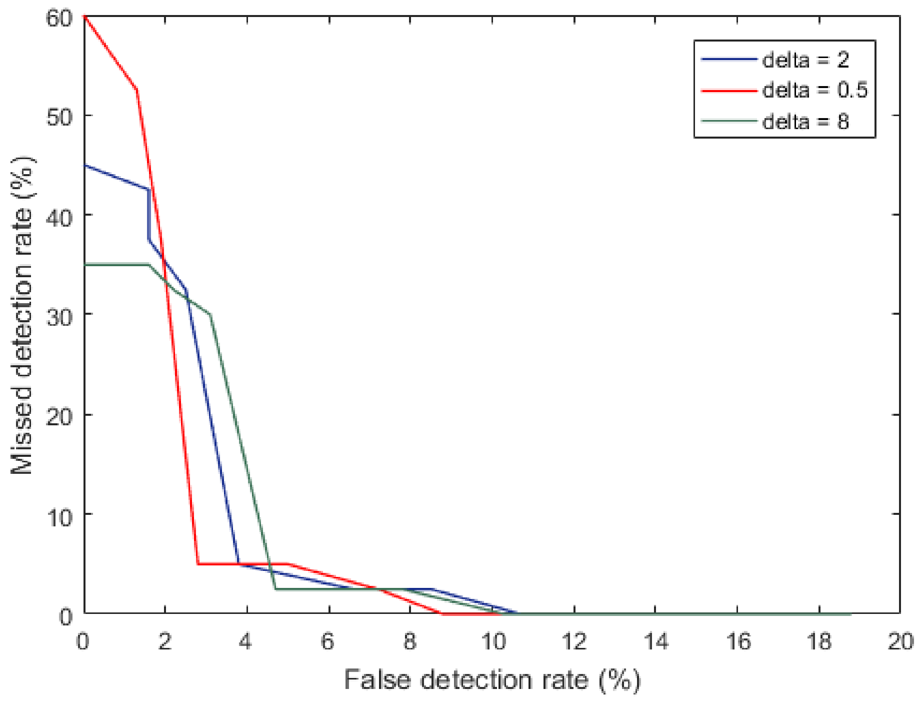


Figure 5.1: ROC curves of the data in Table 5.8.

5.2.3 Discussions on Parameter Selection and Tuning

Parameter selection and tuning can be a tough task for the proposed pattern mining and matching algorithms. As discussed in the previous chapters, the variance of the Gaussian function σ controls the size of the time span within which the algorithm blurs the occurrence orders between the alarm messages. When the value of σ increases, the algorithm tends to ignore the order of alarms in the alignment and simply takes the alarm occurrences into consideration. On the contrary, if σ is small, the algorithm becomes strict at the appearance order of the alarms. The gap and mismatch penalties, δ and μ, determine the algorithm's tolerance to including gaps and mismatch terms in alignments. When these two parameters become larger, the algorithm is more tolerant to irrelevant alarms raised within a pattern sequence.

In practice, the two parameters for the penalties, δ and μ, are normally selected first based on user's preferences on the amount of gaps they could allow in the alignment. Based on our experience from analyzing alarm floods in chemical plants, $\mu = -0.6$ and $\delta = -0.4$ is a good starting point. By allowing a medium amount of gaps in the alignment, this set of values could capture the existence of patterns. When the pattern is confirmed to exist, the user can gradually increase the penalties (decrease the parameter values) to reduce the amount of gaps in the alignment in order to get a better view of the pattern.

Regarding the parameter tuning of σ, its value should be determined by both the user preference and type of processes. If occurrence orders are not important to the user or the process operates slowly, the value of σ can be increased. Based on our experience, $\sigma = 2$ is a good starting point for oil production related processes.

5.3 Causality Analysis for Root-Cause Detection in Alarm Floods

In this section, basics of Granger causality, conditional Granger causality, and frequency domain conditional Granger causality are reviewed. Datasets from two consoles in an operating utility plant are used in industrial case studies, where frequency domain conditional Granger causality is applied to investigate the root causes.

5.3.1 Basics of Granger Causality

Granger causality was first proposed in [35] by C.W.J Granger for investigating econometric models. It measures the causality by calculating the residue variances from the autoregressive models with and without the signals. If a causal relation exists, the autoregressive model with the corresponding signals should provide better predictions, namely, smaller residue variances.

Pairwise Granger Causality

Consider a pair of process signals X_t and Y_t, with autoregressive representations

$$\begin{aligned} X_t &= \textstyle\sum_{j=1}^{\infty} a_{1j} X_{t-j} + \epsilon_{1t}, \quad \mathrm{var}(\epsilon_{1t}) = \Sigma_1 \\ Y_t &= \textstyle\sum_{j=1}^{\infty} d_{1j} Y_{t-j} + \eta_{1t}, \quad \mathrm{var}(\eta_{1t}) = \Gamma_1, \end{aligned} \tag{5.1}$$

where ϵ_{1t} and η_{1t} are noise terms. Jointly, the two signals can be described as

$$\begin{aligned} X_t &= \textstyle\sum_{j=1}^{\infty} a_{2j} X_{t-j} + \sum_{j=1}^{\infty} b_{2j} Y_{t-j} + \epsilon_{2t} \\ Y_t &= \textstyle\sum_{j=1}^{\infty} c_{2j} X_{t-j} + \sum_{j=1}^{\infty} d_{2j} Y_{t-j} + \eta_{2t}, \end{aligned} \tag{5.2}$$

where ϵ_{2t} and η_{2t}, representing the noise terms, have a covariance matrix

$$\Sigma = \begin{pmatrix} \Sigma_2 & \Upsilon_2 \\ \Upsilon_2 & \Gamma_2 \end{pmatrix}. \tag{5.3}$$

The entries in the covariance matrix Σ are defined as $\Gamma_2 = \mathrm{var}(\eta_{2t})$, $\Sigma_2 = \mathrm{var}(\epsilon_{2t})$, and $\Upsilon_2 = \mathrm{cov}(\epsilon_{2t}, \eta_{2t})$. Based on the variances in Equations (5.1) and (5.3), three metrics were proposed in [35] as

$$F_{X,Y} = \ln \frac{\Sigma_1 \Gamma_1}{|\Sigma|} \tag{5.4}$$

$$F_{Y \to X} = \ln \frac{\Sigma_1}{\Sigma_2} \tag{5.5}$$

$$F_{X \to Y} = \ln \frac{\Gamma_1}{\Gamma_2}. \tag{5.6}$$

$F_{X,Y}$ represents the total interdependency between X_t and Y_t. If $F_{X,Y} = 0$, X_t and Y_t are independent; otherwise $F_{X,Y} > 0$. $F_{X \to Y}$ and $F_{Y \to X}$ captures the causality from X_t to Y_t and from Y_t to X_t, respectively. A causality influence exists when $F_{X \to Y} > 0$ or $F_{Y \to X} > 0$. If $F_{X \to Y} = 0$, there is no causality from X_t to Y_t; similarly, if $F_{Y \to X} = 0$, there is no causality from Y_t to X_t.

Conditional Granger Causality

The pairwise Granger causality described in the previous subsection can only detect the causality and its direction. However, it cannot identify if the causality is direct or indirect. For example, in Figure 5.2, the causality from signal X_t to Y_t can be induced directly from X_t to Y_t, it can also be induced via an intermediate signal Z_t.

$$\begin{tikzcd} X \arrow[r] & Y \end{tikzcd}$$

$$\begin{tikzcd} X \arrow[r] & Z \arrow[r] & Y \end{tikzcd}$$

Figure 5.2: An example of direct and indirect causality.

A conditional Granger causality was introduced in [35] to distinguish whether the causality between two time series is direct or intermediated by another signal. Here we describe the method using a case of tri-variate time series, it can easily be extended to cases with more variables. Consider three process signals X_t, Y_t, and Z_t. The joint autoregressive model of X_t and Z_t can be presented as

$$\begin{aligned} X_t &= \textstyle\sum_{j=1}^{\infty} a_{3j} X_{t-j} + \sum_{j=1}^{\infty} b_{3j} Z_{t-j} + \epsilon_{3t} \\ Z_t &= \textstyle\sum_{j=1}^{\infty} c_{3j} X_{t-j} + \sum_{j=1}^{\infty} d_{3j} Z_{t-j} + \gamma_{3t}, \end{aligned} \tag{5.7}$$

where the noise terms ϵ_{3t} and γ_{3t} have a covariance matrix

$$\Sigma_3 = \begin{pmatrix} \Sigma_3 & \Upsilon_3 \\ \Upsilon_3 & \Gamma_3 \end{pmatrix}. \tag{5.8}$$

The entries in the covariance matrix Σ_3 are defined as $\Gamma_3 = \mathrm{var}(\gamma_{3t})$, $\Sigma_3 = \mathrm{var}(\epsilon_{3t})$, and $\Upsilon_3 = \mathrm{cov}(\epsilon_{3t}, \gamma_{3t})$. The autoregressive model for the three signals X_t, Y_t, and Z_t is defined in

$$\begin{aligned} X_t &= \textstyle\sum_{j=1}^{\infty} a_{4j} X_{t-j} + \sum_{j=1}^{\infty} b_{4j} Y_{t-j} + \sum_{j=1}^{\infty} c_{4j} Z_{t-j} + \epsilon_{4t} \\ Y_t &= \textstyle\sum_{j=1}^{\infty} d_{4j} X_{t-j} + \sum_{j=1}^{\infty} e_{4j} Y_{t-j} + \sum_{j=1}^{\infty} g_{4j} Z_{t-j} + \eta_{4t} \\ Z_t &= \textstyle\sum_{j=1}^{\infty} u_{4j} X_{t-j} + \sum_{j=1}^{\infty} v_{4j} Y_{t-j} + \sum_{j=1}^{\infty} w_{4j} Z_{t-j} + \gamma_{4t}, \end{aligned} \tag{5.9}$$

of which the covariance matrix of ϵ_{4t}, η_{4t}, and γ_{4t} is

$$\Sigma_4 = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{xz} \\ \Sigma_{yx} & \Sigma_{yy} & \Sigma_{yz} \\ \Sigma_{zx} & \Sigma_{zy} & \Sigma_{zz} \end{pmatrix}. \tag{5.10}$$

Based on the covariance matrices in Equations (5.8) and (5.10), the Granger causality from Y_t to X_t conditioned on Z_t can be measured by the metric

$$F_{Y \to X|Z} = \ln \frac{\Sigma_3}{\Sigma_{xx}}. \tag{5.11}$$

If $F_{Y \to X|Z} > 0$, it means there is direct causality from signal Y_t to X_t; on the contrary, if $F_{Y \to X|Z} = 0$, there is no direct causality from Y_t to X_t. The interpretation of the metric is very clear. When Y_t does not cause X_t directly, $\mathbf{b}_{4j}$ are all zeros in Equation (5.9), which leads to $\Sigma_3 = \Sigma_{xx}$ and $F_{Y \to X|Z} = 0$. On the other hand, if there is direct causality from Y_t to X_t, $\mathbf{b}_{4j}$ are not all zeros and the autoregressive model for X_t in Equation (5.9) is a better fit of signals X_t, Y_t, and Z_t compared to the one in Equation (5.7); thus, $\Sigma_3 > \Sigma_{xx}$ and $F_{Y \to X|Z} > 0$.

Frequency Domain Conditional Granger Causality

The derivation of frequency domain conditional Granger causality involves a transformation procedure introduced by [33] for the purpose of frequency decomposition and is very tedious. The full derivation can be found in [23]; we will not repeat it in this thesis. What is worth pointing out, equality exists relating the spectral and time domain measurements as

$$F_{Y \to X|Z} = \frac{1}{2\pi} \int_{-\pi}^{\pi} f_{Y \to X|Z}(w) dw \tag{5.12}$$

under general conditions. As pointed out in [23], "this certainly holds true on purely theoretical grounds, and it may very well be true for simple mathematical systems. For actual physical data, however, this condition may be very hard to satisfy due to practical estimation errors."

One advantage of spectral Granger causality over time domain is that the result of frequency domain Granger causality shows the amount of causality from one signal to the other across the whole frequency space, which can be useful for determining the frequency band of signals that are interdependent.

5.3.2 Industrial Case Study

The frequency domain Granger causality analysis has been applied on datasets from a real utility plant. Two alarm floods from the two consoles in the utility plant have been selected for application.

Application on the first console

Figure 5.3 shows the two alarm burst plots from the first console. The blue line shows the original alarm rate of the panel, the red line is the alarm rate with 60 sec off-delay timers uniformly applied on all the tags, and the horizontal black line is the alarm flood threshold suggested by the ISA standard (10 alarms per 10 min per operator). By comparing the blue and red lines in the figure we can notice chattering alarms contributed a large amount to the alarm floods in the first console. Moreover, even with off-delay timers, there is still a big spike on the alarm burst plot around 8:00PM. The corresponding alarm flood occurred during this period was extracted and studied in this example.

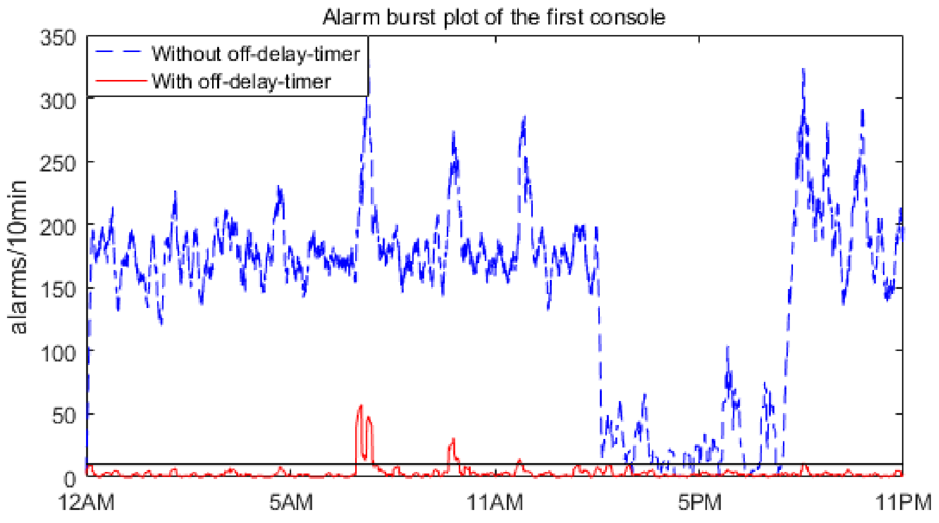


Figure 5.3: Comparison of alarm burst plots with and without off-delay timers for the first console.

Alarm tags that were of high and emergency priorities raised during the alarm flood are listed in Table 5.9. The alarm tags that were associated with process variables have been highlighted in bold fonts. One-day data of the corresponding four process variables were searched and obtained by typing the alarm tag names into the DCS server. For convenience, we name the four process variables after their corresponding alarm tag names. Table 5.10 shows an example of a part of the extracted process data. It is easy to notice the sampling times of the process data are not synchronized, as Tag9 started at 00:01:04 but Tag18 started at 00:01:01. The second problem is the changing sampling time. The highlighted time stamps for the last two samples of Tag9 have 10 sec intervals in between, different from the 5 sec sampling time. These two data problems are very common when the data is extracted from a DCS server. Due to the two problems, a data padding procedure was applied to up-sample the data to a sampling time of 1 sec based on a zero-order hold

(ZOH) operation. Then, frequency domain Granger causality analysis was applied on the four processed variables and the results are shown in Figure 5.4.

Table 5.9: High and emergency priority alarms raised during the alarm flood from the first console.

Alarm tag	Priority
Tag1.OFFNRM	EMERGENCY
Tag2.OFFNRM	EMERGENCY
Tag3.OFFNRM	EMERGENCY
Tag4.UNCMD	EMERGENCY
Tag5.OFFNRM	HIGH
Tag6.OFFNRM	HIGH
Tag7.ALARM	HIGH
Tag8.OFFNRM	HIGH
Tag9.PVHIGH	HIGH
Tag10.PVLOLO	HIGH
Tag11.OFFNRM	HIGH
Tag12.OFFNRM	HIGH
Tag13.OFFNRM	HIGH
Tag14.OFFNRM	HIGH
Tag15.PVLOW	HIGH
Tag17.OFFNRM	HIGH
Tag18.PVHIGH	HIGH
Tag19.OFFNRM	HIGH
Tag20.OFFNRM	HIGH
Tag21.OFFNRM	HIGH
Tag22.OFFNRM	HIGH
Tag23.OFFNRM	HIGH
Tag24.OFFNRM	HIGH
Tag25.OFFNRM	HIGH

Table 5.10: An example of a part of the extract process data.

Time	Tag9.PVHIGH	Time	Tag18.PVHIGH
00:01:04	-26.03682327	00:01:01	615.0713501
00:01:09	-26.03117371	00:01:06	614.9962158
00:01:14	-26.04812431	00:01:11	614.8083496
00:01:19	-26.03682327	00:01:16	614.977417
00:01:24	-26.03682327	00:01:21	614.8646851
00:01:29	-26.02552223	00:01:26	614.9586182
00:01:34	-26.02552223	00:01:31	615.0337524
00:01:39	-26.01987267	00:01:36	614.9962158
00:01:49	-26.01987267	00:01:41	614.8459473

The horizontal axis in Figure 5.4 shows the cause tags, on the vertical axis are the effect tags. If there is a big spike on the corresponding sub-figure, say the sub-figure from Tag15 to Tag10, there is a causality from Tag15 to Tag10. Notice the values in the vertical axis of the sub-figures are percentages, i.e., the real values can be obtained by multiplying 0.01. By setting a uniform threshold of 1% on the frequency domain causality measurements, Figure 5.4 can be converted to a causal map in Figure 5.5.

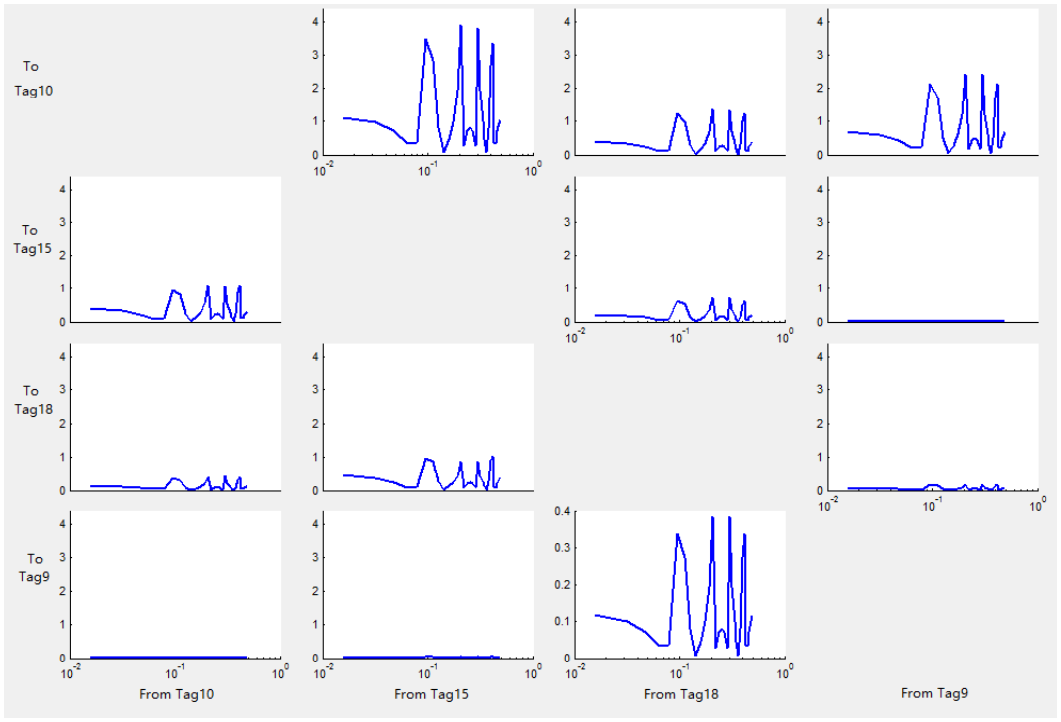


Figure 5.4: Result of frequency domain Granger causality for the first console.

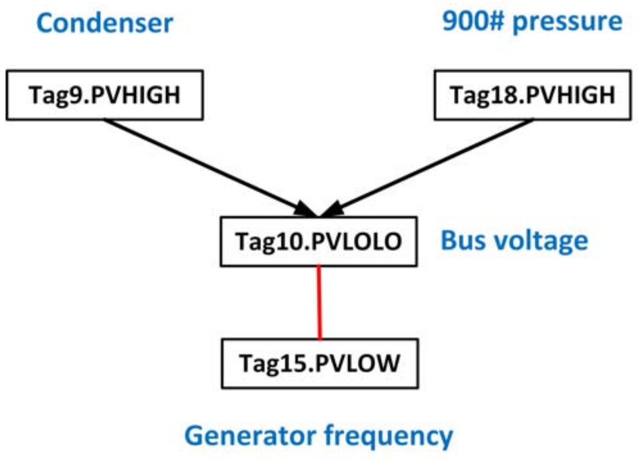


Figure 5.5: Causal map of the four extracted process variables in the first console.

In Figure 5.5, the black arrows represent cause and effect directions and the red line represents a bi-directional causality. Descriptions are provided alongside every tag. As shown in the figure, Tag9 and Tag18, which are the condenser value and the 900# pressure value, are both the causes of Tag10, the bus voltage value. Since there is a red line between Tag10 and Tag15, they are both the cause and effect tags of each other. Based on the causal map, the relations between the tags are very clear, and this information can be very useful in root cause analysis.

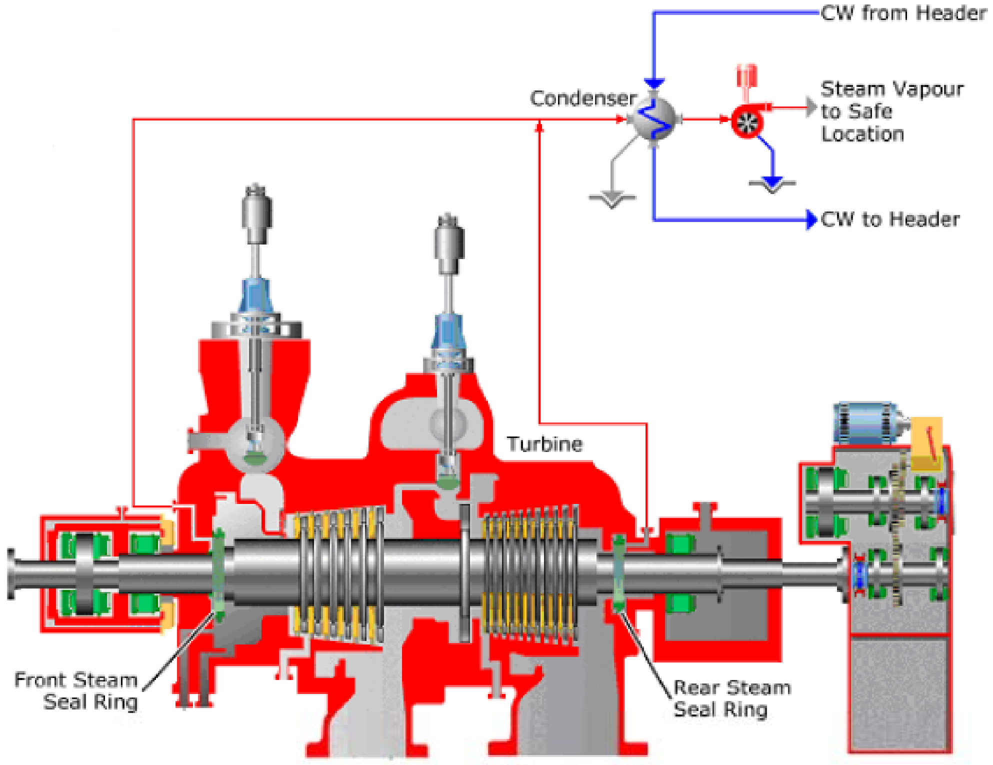


Figure 5.6: Diagram of the steam turbine generator process.

To confirm the causal map, two process diagrams from the plant manual are used, shown in Figure 5.6 and Figure 5.7. Figure 5.6 describes a steam turbine generator (STG) process, where the high-pressure steam goes in from the left side of the turbine. When the steam passes through the turbine it drives the rotors, which are connected to the generator; the spinning of the rotors in the generator generates power, consisting of voltage and frequency. Thus, the causality direction is steam pressure $\rightarrow$ spinning of the turbine $\rightarrow$ voltage and frequency. In Figure 5.6, there is also a condenser taking in the extra steam and cool it down. Since the flow in the condenser is related to the high-pressure steam, it explains why Tag9 is the drive of Tag10 and Tag15. The left-hand side of the causal map is confirmed.

The right-hand side of the causal map can be confirmed in Figure 5.7, which shows the pressure line connections around the steam turbine generator. Notice the 900# pressure line is the input of the STG process. Thus, 900# pressure value (Tag18) can be confirmed to be the drive of the Tag10.

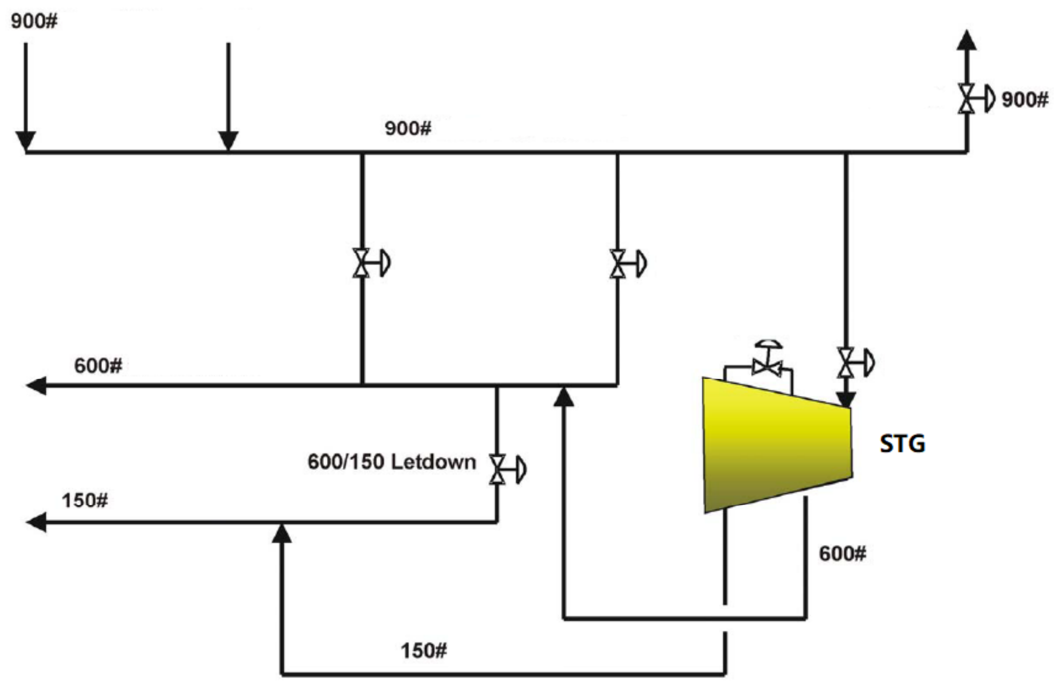


Figure 5.7: Diagram of pressure lines in the first console.

The root cause of this alarm flood was suspected to be a turbine lock-out trip. A series of high pressure alarms were raised because the turbine lock-out permissive blocked the path of the high pressure steam. Also, due to the turbine lock-out, the rotors in the generator stopped and alarms were raised for low bus voltage and frequencies.

Application on the second console

Similar to the causality analysis application procedure for the first console, two alarm burst plots for the alarm rate of the second console with and without 60 sec off-delay timers were generated, shown in Figure 5.8. The horizontal black line is the alarm flood threshold suggested by the ISA standard, 10 alarms per 10 min per operator. We can notice chattering alarms contributed a lot to the alarm floods in the second console by comparing the blue and red lines in the figure. The alarm flood occurred during the time period of the spike on the red line was studied.

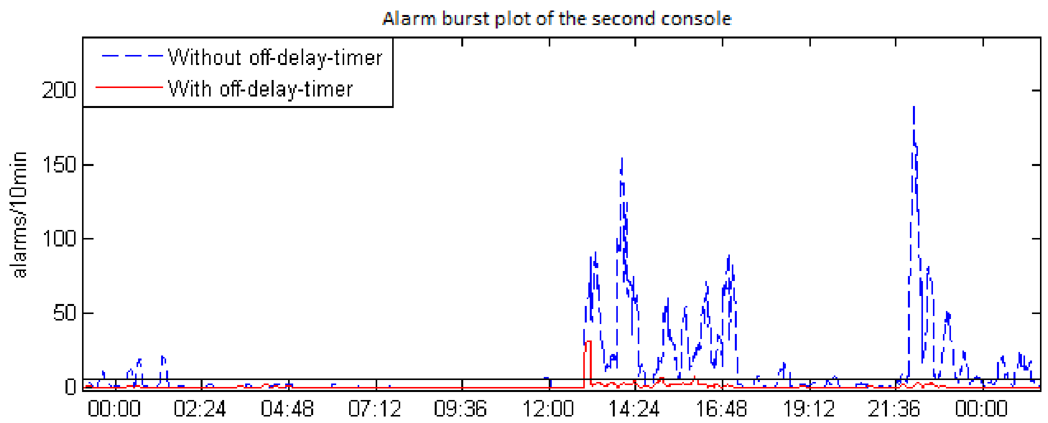


Figure 5.8: Comparison of alarm burst plots with and without off-delay timers for the second console.

Table 5.11: High and emergency priority alarms raised during the alarm flood from the second console.

Alarm tag	Priority
TagA.PVHIGH	EMERGENCY
TagB.OFFNRM	EMERGENCY
TagC.OFFNRM	EMERGENCY
TagD.PVLOW	HIGH
TagE.OFFNRM	HIGH
TagF.PVLOW	HIGH
TagG.OFFNRM	HIGH
TagH.OFFNRM	HIGH
TagI.PVLOW	HIGH
TagJ.UNCMD	HIGH
TagK.UNCMD	HIGH
TagL.ALARM	HIGH
TagM.UNCMD	HIGH
TagN.PVLO	HIGH
TagO.OFFNRM	HIGH
TagP.ALARM	HIGH
TagQ.OFFNRM	HIGH
TagR.OFFNRM	HIGH
TagS.OFFNRM	HIGH

Alarm tags that were of high and emergency priorities raised during the alarm flood are listed in Table 5.11. The five alarm tags that were associated with process variables have been highlighted in bold fonts. One day data of the corresponding five process variables were extracted from the DCS server and named after their corresponding alarm tags. The extracted process data were up-sampled to 1 sec using a ZOH filter because of the two data problems mentioned in the previous subsection. Then, frequency domain Granger causality was applied on the five processed variables and the results are shown in Figure 5.9.

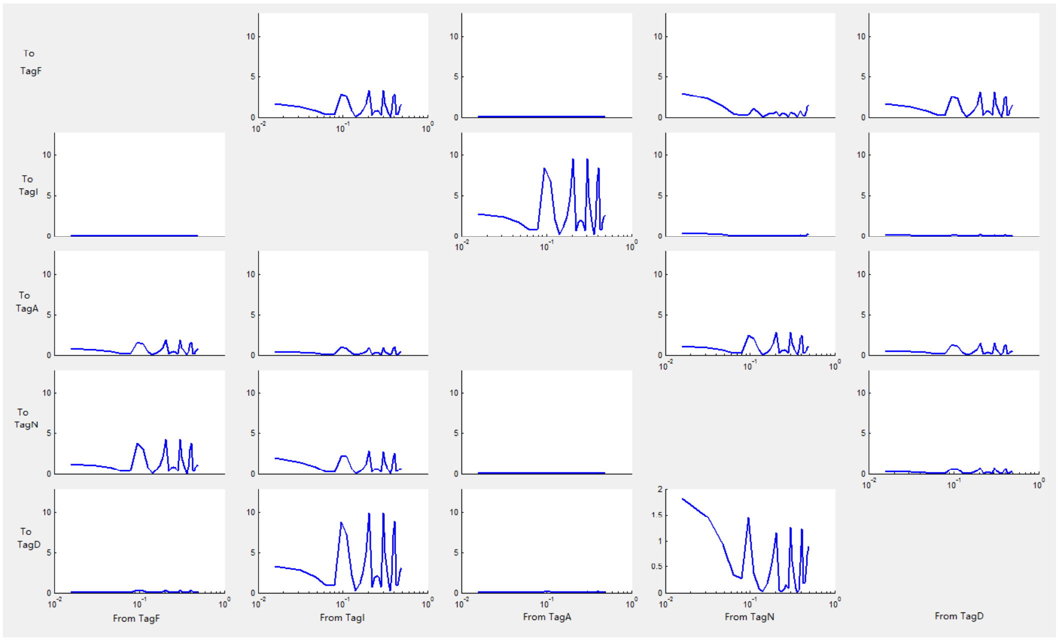


Figure 5.9: Result of frequency domain Granger causality for the second console.

The horizontal and vertical axis in Figure 5.9 shows the cause and effect tags, respectively. The values in the vertical axis of the sub-figures are percentages, i.e., the real values can be obtained by multiplying 0.01. By setting a uniform threshold of 3% on the frequency domain causality measurements, Figure 5.9 can be converted to a causal map in Figure 5.10.

Figure 5.10 shows a serious of causal relations: TagA $\rightarrow$ TagI $\rightarrow$ TagD $\rightarrow$ TagF $\rightarrow$ TagN and one parallel relation TagI $\rightarrow$ TagF. These relations between the tags in the causal map can be used in root cause analysis.

The alarm flood occurred in a CO boiler process. Figure 5.11 shows a typical CO boiler process diagram, where the burner takes in the gas and oxygen to heat up the fluid. This explains the causal relations between TagI, TagD, TagF, and TagN. The circulate flow controller (TagI) determines the amount of gas and oxygen fed into the burner (TagD), and the fan controller (TagF) takes the signals from both circulate flow and flue gas oxygen controllers. The burner pressure (TagN), as the result of the reaction, is at the bottom.

Figure 5.12 was found in the process manual for the downstream process. As the input of the process, the boiler feed water passes through a series of

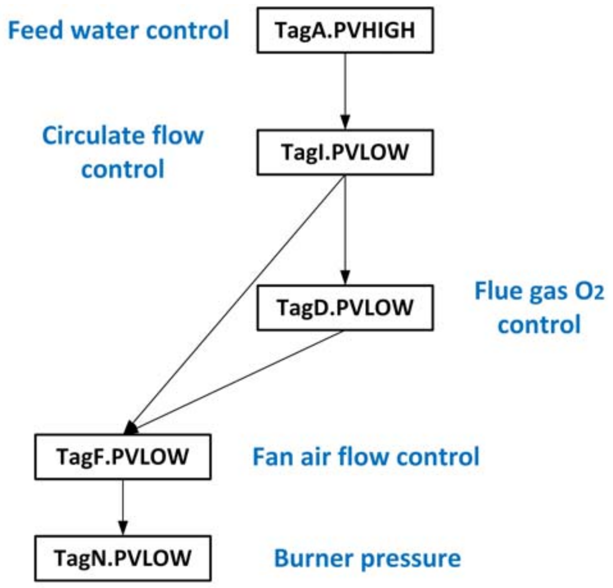


Figure 5.10: Causal map of the five extracted process variables in the second console.

desuperheaters and is eventually supplied to skid heat exchangers in other plants. This confirms why the feed water controller (TagA) is on top of the causal map in Figure 5.10; it controls the amount of water to be boiled in the boiler process by controlling the circulate flow in the boiler.

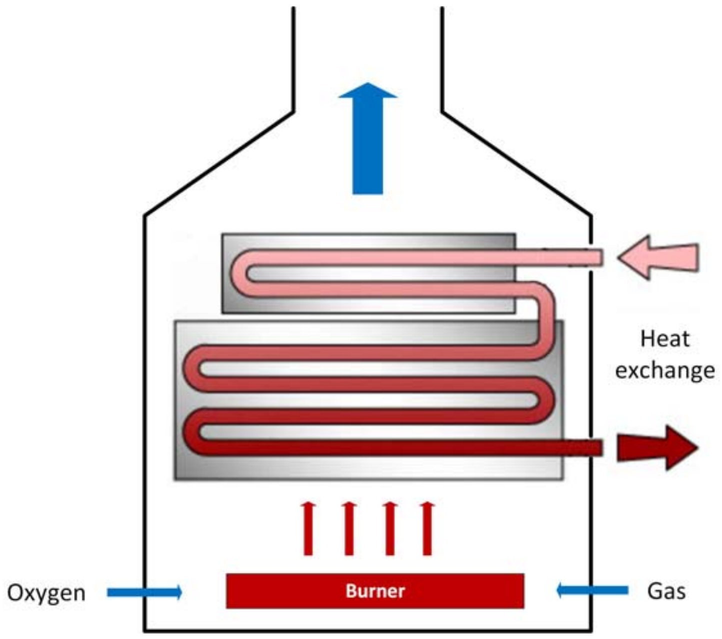


Figure 5.11: Diagram of a typical CO boiler process.

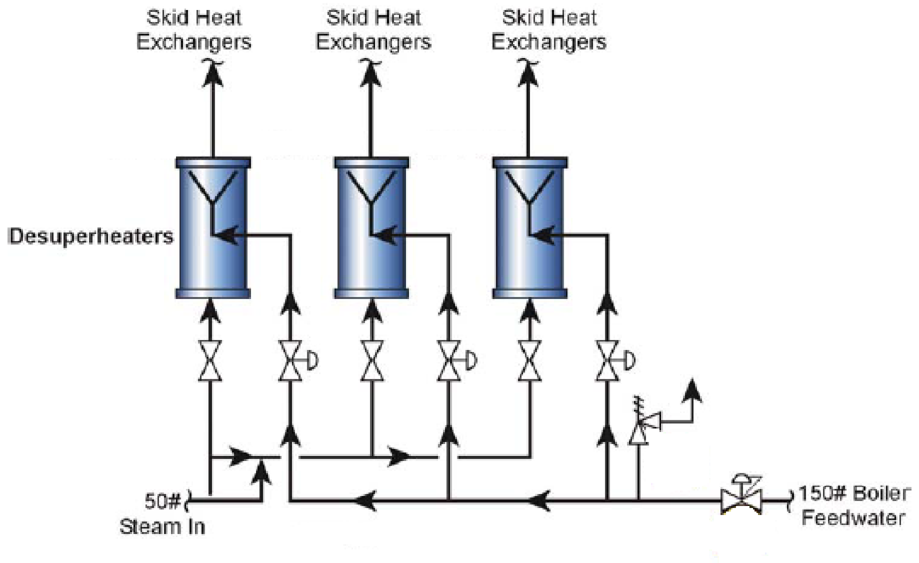


Figure 5.12: Diagram of desuperheater process.

The root cause of this alarm flood was suspected to be a circulate flow pump trip. Because of the pump trip, a series of alarms were raised for the fans and air flows in the boiler. Eventually, the insufficient amount of oxygen and gas lead to burner trips.

5.4 Summary

In this chapter, robustness and accuracy tests were carried out for the pattern mining and pattern matching algorithms proposed in previous chapters. The influence of the parameters were discussed and a guideline for selecting the parameters was given. In addition, applications of causality analysis for the root cause detection of alarm floods have been conducted on a dataset from a real utility plant. Results show the causal map generated from the causality analysis can provide valuable information for root cause analysis of alarm floods.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The purpose of the work reported in this thesis is to develop data-driven techniques for industrial alarm flood analysis. The outcomes of the studies in this thesis are summarized as follows:

1. An algorithm to find the optimal alignment of multiple alarm flood sequences and obtained a common pattern of them thereafter is proposed. The algorithm is an extension of the algorithm in [18] to the case of multiple sequences. It makes up one of the missing steps in alarm flood analysis. A lot of analysis and management such as root cause analysis, dynamic alarm suppression, and analysis on potential problems in alarm systems can be carried out based on the results of this proposed algorithm.

2. An algorithm for online pattern matching and prediction of incoming alarm floods is proposed. A chattering window filter was introduced to eliminate chattering alarms online; a sequence window filter, a modified calculation of the time weight matrix, and an incremental dynamic programming strategy have been introduced to improve the efficiency of the algorithm. The computational complexity analysis has shown that the proposed algorithm is applicable to large scale plants since the computational cost increases linearly with the number of patterns and the lengths of patterns and the online sequence. Accuracy tests revealed a trade-off between the missed and false detection rates.

3. An accelerated multiple sequence alignment algorithm for pattern mining in multiple alarm sequences has been proposed. A progressive multiple sequence alignment mechanism has been introduced to accelerate the generation of the primitive alignment. Two types of refinement methods have been developed to improve the primitive alignment. A dataset from a real chemical plant has been used to test the effectiveness of the proposed algorithm and compare it with the exhaustive search approach in [62]. The results show the proposed algorithm has a significant improvement in efficiency with a small accuracy cost.

4. Parameter tuning and robustness tests of the parameters in pattern mining and pattern matching algorithms are provided. The influence of the parameters on the algorithms are discussed and a guideline for selecting the parameters is given. In addition, applications of causality analysis for the root cause detection of alarm floods are conducted on a dataset from a real utility plant. Results show the causal map generated from the causality analysis can provide valuable information for root cause analysis of alarm floods.

6.2 Future Work

Alarm management is a relatively new research area in academia. Alarm flood analysis, as one of the tools in alarm management and rationalization framework, imposes a big challenge because of the severe outcome if not managed properly. This thesis provides a guideline for alarm flood analysis, which include: univariate alarm analysis, pattern mining of alarm floods, online pattern matching of alarm floods, and causality analysis for root cause detection in alarm floods. However, there still remain opportunities for further exploration and improvements. The following promising directions deserve efforts for future work.

Offline Analysis of Alarm Floods

There are four tasks in offline analysis of alarm floods: targeting important alarm floods (classification and pattern mining), problem localization (root cause analysis), preventing alarm floods from happening again (alarm system rationalization), and finding proper actions against alarm floods (operator action and workflow mining). The pattern mining methods proposed in this thesis focus on the first task, targeting important alarm floods. The proposed methods are very effective in finding the alarm floods occurred repeatedly in the process. The methods, however, are unable to locate an important alarm flood if it only happened once in the given dataset. To overcome this issue, methods that take operator actions and alarm priorities into account are worth study. In terms of root cause detection, causality analysis can be an effective tool, as shown in the thesis. But in practice, a process may switch between multiple modes and each mode may come with its unique causal map. Thus, it is important to include operating mode information into causality analysis. Alarm system rationalization for alarm floods can be divided into: univariate alarm analysis and consequence (multivariate) alarm analysis. The output (alarm flood patterns) of the proposed pattern mining methods can be used as a valuable information for consequence alarm analysis. Causality analysis and correlation analysis can be studied to rationalize consequence alarms more efficiently. For the last task, to come up with proper actions against an alarm flood, since alarm data and operator data are greatly related, methods that consider both alarm events and operator actions are worth developing.

Online Management of Alarm Floods

Three types of methods can be valuable for online management of alarm floods: the methods that can predict the whole or part of alarm floods, the methods that can accurately suppress alarms of low criticality, and the methods that can provide tips to the operators in addressing the alarm floods. The online alarm flood pattern matching algorithm proposed in this thesis falls into the first type of methods. The causal map generated in causality analysis has potentials to be used in the alarm flood prediction as well. Operator action analysis and alarm criticality ranking algorithms are worth study to provide accurate alarm shelving (suppression) during alarm floods. For the last type of methods, in order to provide tips to the operators during alarm floods, process and workflow mining can be helpful to study the actions performed by super operators and convert them into tips.

Implementation of Alarm Management Techniques

Computational speed is important for improving user experience. However, for techniques such as causality analysis, it is not very realistic to develop new methods that can improve the computational efficiency by a great deal. Cluster computing techniques and frameworks targeted at dealing with large scale datasets can be incorporated into the existing methods to improve their computation speed drastically. Another popular technique that evolves fast nowadays is cloud computing. Rather than installing the toolbox or software on a personal computer, cloud computing uses remote servers hosted on the Internet to store and process data. The advantage of cloud computing is the ability to reduce the requirements of installing and using the toolbox, such as operating system and software requirements, providing better user experience. In addition, due to the variety of DCS models, a lot of formats exist for alarm, process, and operator action data. Moreover, sometimes a DCS server may record errors (e.g., wrong characters) into the database. Algorithms need to be developed to load data robustly from DCS database. Another important issue is how to integrate the online methods into operating industrial processes. In this case, DCS programming and friendly HMI design may be necessary.

Bibliography

[1] N. A. Adnan, Y. Cheng, I. Izadi, and T. Chen. A generalized delay-timer for alarm triggering. In *American Control Conference (ACC), 2012*, pages 6679–6684. IEEE, 2012.

[2] 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.

[3] N. A. Adnan, I. Izadi, and T. Chen. On expected detection delays for alarm systems with deadbands and delay-timers. *Journal of Process Control*, 21(9):1318–1331, 2011.

[4] M. S. Afzal and T. Chen. Analysis and design of multimode delay-timers. *Chemical Engineering Research and Design*, 120:179–193, 2017.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14. IEEE, 1995.

[6] K. Ahmed, I. Izadi, T. Chen, D. Joe, and T. Burton. Similarity analysis of industrial alarm flood data. *IEEE Transactions on Automation Science and Engineering*, 10(2):452–457, 2013.

[7] J. Ahnlund, T. Bergquist, and L. Spaanenburg. Rule-based reduction of alarm signals in industrial control. *Journal of Intelligent & Fuzzy Systems*, 14(2):73–84, 2003.

[8] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[9] J. Arnhold, P. Grassberger, K. Lehnertz, and C. E. Elger. A robust method for detecting interdependences: application to intracranially recorded eeg. *Physica D: Nonlinear Phenomena*, 134(4):419–430, 1999.

[10] L. A. Baccal and K. Sameshima. Partial directed coherence: a new concept in neural structure determination. *Biological Cybernetics*, 84(6):463474, 2001.

[11] M. Bauer, J. W. Cox, M. H. Caveness, J. J. Downs, and N. F. Thornhill. Finding the direction of disturbance propagation in a chemical process using transfer entropy. *IEEE transactions on control systems technology*, 15(1):12–21, 2007.

[12] M. Bauer, J. W. Cox, M. H. Caveness, J. J. Downs, and N. F. Thornhill. Nearest neighbors methods for root cause analysis of plantwide disturbances. *Industrial & Engineering Chemistry Research*, 46(18):5977–5984, 2007.

[13] M. Bauer and N. F. Thornhill. A practical method for identifying the propagation path of plant-wide disturbances. *Journal of Process Control*, 18(7):707–719, 2008.

[14] M. Bransby and J. Jenkinson. *The management of alarm systems*. HSE Books Suffolk, UK, 1998.

[15] E. Burnell and C. Dicken. Handling of repeating alarms. In *Stemming the Alarm Flood (Digest No: 1997/136), IEE Colloquium on*, pages 12–1. IET, 1997.

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

[17] 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.

[18] 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.

[19] P. Cisar, E. Hostalkova, and P. Stluka. Alarm rationalization support via correlation analysis of alarm history. In *19th International Congress of Chemical and Process Engineering, Prague, Czech Republic*, 2010.

[20] I. I. E. Commision). *Management of Alarm Systems for the Process Industries*. 2014.

[21] C. R. Dal Vernon, J. L. Downs, and D. Bayn. Human performance models for response to alarm notifications in the process industries: An industrial case study. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48, pages 1189–1193. SAGE Publications, 2004.

[22] S. Dash and V. Venkatasubramanian. Challenges in the industrial applications of fault diagnostic systems. *Computers & chemical engineering*, 24(2):785–791, 2000.

[23] M. Ding, Y. Chen, and S. Bressler. Granger causality: basic theory and application to neuroscience. 2006. *arXiv preprint q-bio/0608035*.

[24] P. Duan, T. Chen, S. L. Shah, and F. Yang. Methods for root cause diagnosis of plant-wide oscillations. *AIChE Journal*, 60(6):2019–2034, 2014.

[25] P. Duan, F. Yang, T. Chen, and S. L. Shah. Detection of direct causality based on process data. In *American Control Conference (ACC), 2012*, pages 3522–3527. IEEE, 2012.

[26] P. Duan, F. Yang, S. L. Shah, and T. Chen. Transfer zero-entropy and its application for capturing cause and effect relationship between variables. 2014.

[27] S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[28] S. R. Eddy et al. Multiple alignment using hidden markov models. In *Ismb*, volume 3, pages 114–120, 1995.

[29] E. E. Equipment and M. U. Association). Alarm systems - a guide to design, management and procurement. 2013.

[30] D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, 1987.

[31] J. Folmer and B. Vogel-Heuser. Computing dependent industrial alarms for alarm flood reduction. In *9th International Multi-Conference on Systems, Signals and Devices (SSD)*, pages 1–6. IEEE, 2012.

[32] W. A. Freiwald, P. Valdes, J. Bosch, R. Biscay, J. C. Jimenez, L. M. Rodriguez, V. Rodriguez, A. K. Kreiter, and W. Singer. Testing non-linearity and directedness of interactions between neural groups in the macaque inferotemporal cortex. *Journal of neuroscience methods*, 94(1):105–119, 1999.

[33] J. F. Geweke. Measures of conditional linear dependence and feedback between time series. *Journal of the American Statistical Association*, 79(388):907–915, 1984.

[34] R. Govindan, J. Raethjen, F. Kopper, J. Claussen, and G. Deuschl. Estimation of time delay by coherence analysis. *Physica A: Statistical Mechanics and its Applications*, 350(2):277–295, 2005.

[35] C. W. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438, 1969.

[36] C. W. Granger. Time series analysis, cointegration, and applications. *American Economic Review*, pages 421–425, 2004.

[37] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 355–359. ACM, 2000.

[38] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[39] A. Henningsen and J. P. Kemmerer. Intelligent alarm handling in cement plants. *IEEE Industry applications magazine*, 1(5):9–15, 1995.

[40] B. Hollifield, E. Habibi, and J. Pinto. Alarm management: A comprehensive guide. *International Society of Automation*, 2011.

[41] 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.

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

[43] A. J. Hugo. Estimation of alarm deadbands. *IFAC Proceedings Volumes*, 42(8):663–667, 2009.

[44] S.-L. Hwang, J.-T. Lin, G.-F. Liang, Y.-J. Yau, T.-C. Yenn, and C.-C. Hsu. Application control chart concepts of designing a pre-alarm system in the nuclear power plant control room. *Nuclear Engineering and Design*, 238(12):3522–3527, 2008.

[45] H. Hyyrö. An efficient linear space algorithm for consecutive suffix alignment under edit distance (short preliminary paper). In *String Processing and Information Retrieval*, pages 155–163. Springer, 2009.

[46] ISA. *Management of Alarm Systems for the Process Industries*. 2009.

[47] I. Izadi, S. L. Shah, and T. Chen. Effective resource utilization for alarm management. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 6803–6808. IEEE, 2010.

[48] I. Izadi, S. L. Shah, D. S. Shook, and T. Chen. An introduction to alarm analysis and design. *IFAC Proceedings Volumes*, 42(8):645–650, 2009.

[49] I. Izadi, S. L. Shah, D. S. Shook, S. R. Kondaveeti, and T. Chen. A framework for optimal design of alarm systems. In *Proceedings of 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 651–656, 2009.

[50] H. Jiang, R. Patwardhan, and S. L. Shah. Root cause diagnosis of plant-wide oscillations using the concept of adjacency matrix. *Journal of Process Control*, 19(8):1347–1354, 2009.

[51] M. S. Johnson and R. F. Doolittle. A method for the simultaneous alignment of three or more amino acid sequences. *Journal of Molecular Evolution*, 23(3):267–278, 1986.

[52] H. S. Kim and K. C. Lee. Fuzzy implications of fuzzy cognitive map with emphasis on fuzzy causal relationship and fuzzy partially causal relationship. *Fuzzy Sets and Systems*, 97(3):303–313, 1998.

[53] S.-R. Kim and K. Park. A dynamic edit distance table. In *Combinatorial Pattern Matching*, pages 60–68. Springer, 2000.

[54] L. Kocarev and U. Parlitz. Generalized synchronization, predictability, and equivalence of unidirectionally coupled dynamical systems. *Physical Review Letters*, 76(11):1816, 1996.

[55] S. R. Kondaveeti, I. Izadi, S. L. Shah, and T. Black. Graphical representation of industrial alarm data. In *Analysis, Design, and Evaluation of Human-Machine Systems*, volume 11, pages 181–186, 2010.

[56] S. R. Kondaveeti, I. Izadi, S. L. Shah, T. Black, and T. Chen. Graphical tools for routine assessment of industrial alarm systems. *Computers & Chemical Engineering*, 46:39–47, 2012.

[57] S. R. Kondaveeti, I. Izadi, S. L. Shah, and T. Chen. On the use of delay timers and latches for efficient alarm design. In *Proceedings of 19th Mediterranean Conference on Control & Automation (MED), 2011*, pages 970–975. IEEE, 2011.

[58] 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.

[59] S. Kordic, P. Lam, J. Xiao, and H. Li. *Analysis of Alarm Sequences in a Chemical Plant*. Springer, 2008.

[60] J. M. Koscielny, M. ł Bartys, and M. ł Syfert. Method of multiple fault isolation in large scale systems. *IEEE Transactions on Control Systems Technology*, 20(5):1302–1310, 2012.

[61] T. Kourti. Process analysis and abnormal situation detection: from theory to practice. *IEEE control systems*, 22(5):10–25, 2002.

[62] S. Lai and T. Chen. A method for pattern mining in multiple alarm flood sequences. *in press in Chemical Engineering Research and Design (in press)*, 2015.

[63] G. M. Landau, E. Myers, and M. Ziv-Ukelson. Two algorithms for lcs consecutive suffix alignment. In *Combinatorial Pattern Matching*, pages 173–193. Springer, 2004.

[64] G. M. Landau, E. W. Myers, and J. P. Schmidt. Incremental string comparison. *SIAM Journal on Computing*, 27(2):557–582, 1998.

[65] J. Liu, K. W. Lim, W. K. Ho, K. C. Tan, R. Srinivasan, and A. Tay. The intelligent alarm management system. *IEEE software*, 20(2):66–71, 2003.

[66] V. M. Marques, C. J. Munaro, and S. L. Shah. Data-based causality detection from a system identification perspective. In *Control Conference (ECC), 2013 European*, pages 2453–2458. IEEE, 2013.

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

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

[69] 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.

[70] J. Nishiguchi and T. Takai. Ipl2 and 3 performance improvement method for process safety using event correlation analysis. *Computers & Chemical Engineering*, 34(12):2007–2013, 2010.

[71] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448, 1988.

[72] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 0215–0215. IEEE Computer Society, 2001.

[73] J. Petersen. Causal reasoning based on mfm. In *Proceedings International Symposium on Cognitive Systems Engineering in Process Control (CSEPC), Taejon Korea*, 2000.

[74] S. H. Rich and V. Venkatasubramanian. Model-based reasoning in diagnostic expert systems for chemical process plants. *Computers & Chemical Engineering*, 11(2):111–122, 1987.

[75] J. T. Rickard, J. Aisbett, and R. R. Yager. A new fuzzy cognitive map structure based on the weighted power mean. *IEEE Transactions on Fuzzy Systems*, 23(6):2188–2201, 2015.

[76] D. H. Rothenberg. *Alarm management for process control: a best-practice guide for design, implementation, and use of industrial alarm systems.* Momentum Press, 2009.

[77] E. N. Satuf, E. Kaszkurewicz, R. Schirru, and M. C. M. M. de Campos. Situation awareness measurement of an ecological interface designed to operator support during alarm floods. *International Journal of Industrial Ergonomics*, 53:179–192, 2016.

[78] T. Schreiber. Measuring information transfer. *Physical review letters*, 85(2):461, 2000.

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

[80] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[81] 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.

[82] 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.

[83] 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.

[84] G. Weidl, A. L. Madsen, and S. Israelson. Applications of object-oriented bayesian networks for condition monitoring, root cause analysis and decision support on operation of complex continuous processes. *Computers & chemical engineering*, 29(9):1996–2009, 2005.

[85] 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, 2012.

[86] F. Yang, S. Shah, and D. Xiao. Signed directed graph based modeling and its validation from process knowledge and process data. *International Journal of Applied Mathematics and Computer Science*, 22(1):41–53, 2012.

[87] F. Yang, S. L. Shah, D. Xiao, and T. Chen. Improved correlation analysis and visualization of industrial alarm data. *ISA transactions*, 51(4):499–506, 2012.

[88] F. Yang and D. Xiao. Progress in root cause and fault propagation analysis of large-scale industrial processes. *Journal of Control Science and Engineering*, 2012, 2012.

[89] Z. Yang, J. Wang, and T. Chen. Detection of correlated alarms based on similarity coefficients of binary data. *Automation Science and Engineering, IEEE Transactions on*, 10(4):1014–1025, 2013.

[90] Y. Yuki. Alarm system optimization for increasing operations productivity. *ISA transactions*, 41(3):383–387, 2002.

[91] M. J. Zaki. Efficient enumeration of frequent sequences. In *Proceedings of the seventh International Conference on Information and Knowledge Management*, pages 68–75. ACM, 1998.