Process Discovery of Operator Actions in Response to Alarms in Industrial Facilities

by

David Li

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

Master of Science

in

Control Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Alarm systems are considered to be a main line of defense against unforeseen situations and are of extreme importance in many industrial plants. These systems alert operators during the occurrence of an abnormality and sometimes could provide additional information on ways to resolve the issue. However, an alarm system will not always performing at an optimal level. In many cases, operators are overloaded with nuisance or chattering alarms and as a result are not able to focus on meaningful ones. In order to assist operators with effective decision making, this thesis focuses on ways to mine valuable operator actions from alarm and event logs.

Expert knowledge is an important factor to achieve operational effectiveness. Such expert knowledge does not stay within the company when employees retire. As a result of the work in this thesis, many alarm and action relationships are extracted and stored for future use. By using these relationships, new and inexperienced operators are able to learn from correct operating procedures, especially during critical situations. Two main problems have been examined in this thesis. First, the process discovery of operator actions in response to univariate alarms. To achieve this, case identification based on predetermined parameters along with several modeling algorithms will be utilized. Second, the process discovery of alarms and actions for multivariate alarms. To capture multivariate relationships, a frequency based algorithm is applied on the entire event log. Both process discovery algorithms are applied

to industrial pipeline data to test their effectiveness and the resulting process models are discussed. In this thesis, case identification methods, several process mining algorithms, and methods to interpret the results will also be discussed.

Dedicated to my parents Donglin Li and Qiming Zhang

Acknowledgements

I would like to express my sincerely gratitude to my supervisor, Dr. Tongwen Chen, who has guided me throughout my postgraduate studies journey. He has allowed me to explore many interesting areas within alarm management and has always kept me focused on my goals. This has truly been a life-changing experience and I will never forget it. I also wish to thank the NSERC CRD program for the financial support through out my thesis.

I would also like to show my appreciation for the Advances in Alarm Management and Design Team. Without these wonderful people my experience at university would have never been this fulfilling.

Contents

List of Tables

List of Figures

List of Acronyms

A&E	Alarm & Event
ALM	Alarm Occurrences
DCS	Distributed Control System
P&ID	Piping and Instrumentation Diagram
SCADA	Supervisory Control and Data Acquisition System
EEMUA	Engineering Equipment and Materials Users' Association
FAR	False Alarm Rate
MAR	Missed Alarm Rate
RTN	Return-To-Normal
L1D	Length One Dependency
L2D	Length Two Dependency
LDD	Long Distance Dependency
RTB	Relative To Best
RTN	Return To Normal
OA	Operator Action
XES	Extensible Event Stream
XML	Extensible Markup Language

Chapter 1

Preliminaries

1.1 Background

In the early days of alarm management, sensors were installed at set locations and physically wired back to the control panel. An alarm was raised through an analogue signal, such as a standard 4-20mA current loop. These alarms were then presented in forms of light and sound to the operator. This setup had quite a few flaws compared to the Distributed Control Systems in today's industry. First, the cost of installing sensors can be expensive due to the physical wiring. Second, the total number of variables that can be monitored is limited due to physical space on the control panel. Last, the optimization of variables is non-existent other than trial and error.

Due to the recent advancement of technologies, Process Control Systems (PCS) have become an important aspect in many industries. Before the introduction of Distributed Control Systems (DCS) and Supervisory Control and Data Acquisition (SCADA), a small number of variables were monitored. Nowadays all aspects of an industrial process are monitored due to the inexpensive cost associated with PCS. This, however, raises the problem of alarm overloading, where operators are faced with more alarm annunciations than they can effectively handle. According to both EEMUA [1] and ISA 18.2 standards [3], operators are only capable of dealing with up to 6 alarms every hour. Table 1.1 shows the observed alarms in different industries.

It is obvious from the table that the number of actual observed alarms in the industry is much higher than the standards set by EEMUA [1] and ISA 18.2 [3] standards. Therefore, alarm suppressing techniques such as filtering, dead-bands, and delay timers have been implemented. Research in the field of

Table 1.1: EEMUA standards and observed alarms in industry [1]

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

alarm systems has been fueled by the high demand for new alarm management techniques by the industry. In addition, companies are investing thousands of dollars to train new operators. According to a survey done by the Association for Talent Developments (ATD) State of the Industry report [4], companies on average invest $1252 and 33.5 hours in training on employees in 2015. If the work in this thesis can somehow train new operators to take the correct actions during alarm floods then this could potentially save money and time in the long term for many industries.

The propagation of alarms which results in multiple alarm announications can lead to what are known as alarm floods, where operators are unable to resolve all alarms at the same time. As mentioned in [5], it is highly unlikely for operators to read through an alarm response manual during an alarm flood that requires immediate attention. In general, incorrect operator actions are not due to a lack of experience, but rather caused by the overload of incoming alarms. During alarm floods, it is common for operators to overlook the sources of the problems that caused the propagation of alarms. If theses alarms are not addressed in a timely manner, this abnormal operation can lead to unexpected plant shut-down, economic loss and even loss of life as seen in past plant failure (Chernobyl disaster [6], BP oil spill [7], Texas City refinery explosion [8]). As a result, there is a high demand by the industry to develop tools to assist operators with decision making. A&E logs contain historical information about the process and can be used to capture the correct operator actions to clear each unique alarm. On the contrary, A&E logs can also be used to capture actions that should not be taken if an alarm is not cleared after several attempts. The concept of cry-wolf [9] can be applied to alarm systems. As the FAR increases, operators are less likely to pay attention to these alarms even if occasionally these alarms are true. On the other side of the spectrum, operators become inactive [10] when they rely heavily on decision

support systems. Thus, a right balance is needed between decision support systems and allowing operators to acknowledge and process new alarms.

1.2 Literature survey

Process mining takes existing data records, extracts all the variations of process and turns the results into understandable visualizations of the process.

Process mining consists of 3 main research areas [11]:

- Process Discovery: The construction of a model (petri-nets, fuzzy models [12], social networks [13], and instance graphs [14]) from event logs that represent how a process operates. There are many process discovery algorithms currently available since each algorithm has its advantages and disadvantages.

- Conformance Checking: Using the mined relationships to check with additional data or expert knowledge of the process. There are many quality dimensions associated with conformance checking as well.

- Process Enhancement: Any additional information discovered from conformance checking or interpreting the process model can be used to enhance the process. In terms of alarm management, process enhancement involves reducing the number of alarm annunciations and provideing decision support for operators.

One earlier work in the field of process mining [15] contains a great overview of the extensive ways to mine relationships using the process mining software, ProM. Although some earlier work was also conducted before the terminology process mining was introduced, some of which include workflow mining [16, 17, 18] and business process modeling [19]. As research in the field of process mining developed, new algorithms and softwares were developed. Some of the fundamental algorithms are, but not limited to, the alpha miner [18], heuristics miner [20], fuzzy miner [21], and genetic miner [22]. Over the last few years, many revised versions of these algorithms were also created [23, 24]. In particular several improvements to the α algorithm have been proposed in [25, 26, 27, 24, 28, 29]. An overview $\alpha+$ algorithm within the ProM software was described in [28] and the $\alpha++$ algorithm was discussed in [29] to cover non-free-choice constructs.

This thesis attempts to adopt the process mining algorithms discussed above into the field of alarm management. However, there is still plenty of room for the application of process mining techniques. One of the main foundations for this thesis is the "Process Discovery of Operator Actions in Response to Univariate Alarms" [30], which applied process discovery to text based messages to capture univariate relationship and then displayed the results using petri-nets. Dealing with industrial A&E logs may contain several problems. For example, missing records, duplicated events, and imprecise values all introduce difficulties. A more recent paper by Suriadi [31] covered some of the common imperfection patterns within event logs and the different approaches to clean them. Dasani [32] created work-flow models from event messages from a boiler operation. Conformance checking was applied to the resulting model to extract unique findings. Conformance checking is a method to test the quality of the process model, although the term "quality" is quite vague and can be interpreted in many different ways.

Two possible methods to evaluate the quality of mined process models were found in [33]. The authors used existing metrics (fitness, generalization and structure) and a k-fold cross validation method to evaluate the process models. A similar type of work was found in [34], where the authors stated four quality dimensions (replay fitness, precision, generalization and simplicity) along with their respective mathematical representations. The ETM algorithm, a genetic algorithm which optimizes the process discovery based on the weights of the four quality dimensions was created. Another paper based on assessing the quality dimensions was found in [35]. A complete overview of process discovery techniques along with accuracy dimensions was discussed. The authors categorized 12 dimensions into two categories, namely, recall and precision. A comprehensibility metric was created based on the three metrics to quantify complexity [36]. As the authors stated, "the future of process mining research should emphasize on developing insightful techniques for analyzing real-life event logs." This is a major motivation for this research. Process models based on real life data can sometimes be complex and difficult to interpret, therefore human interpretation of the results should also be examined.

There are several perspectives on the role of human factors in alarm systems design. Undoubtedly, the goal of any alarm system is to effectively display the state of the plant to the operator. The operator should be able

to clearly identify which alarm requires immediate attentions and how to fix the abnormality. In [37], the authors support the notion that operator performance is not directly affected by support tools, but rather the performance of an operator affects the way the support tool performs. As the authors stated in [38], the benefits of a decision support scheme was helpful in reaching a diagnosis more quickly, however, it did not improve the accuracy of correctly diagnosing the root cause. It is only true if the decision support is predicting the occurrence of alarms before they are triggered. This thesis attempts to improve on decision support systems by incorporating suggested actions for incoming alarms.

1.3 Thesis contributions

The major contributions in this thesis are summarized as followed:

1. Propose a data segmentation method to capture the univariate alarm relationships and the corresponding operator actions. The results are modeled using different process mining algorithms.

2. Propose a variant of the heuristics miner to capture the multivariate relationships between operator actions and alarms. The results are displayed in flow chart diagrams with frequencies of occurrence.

3. Develop a new graphical visualization framework to capture alarm data and operator actions.

1.4 Thesis organization

The remainder of the thesis is organized as follows.

Chapter 2 gives a brief introduction to common notations and terminology used in process mining, and proposes an algorithm to capture operator actions in response to univariate alarms. Finally, an industrial case study is used to test the effectiveness of the algorithm.

In Chapter 3, a variant of the heuristics miner is used to capture the multivariate relationships of alarms and operator actions. A process model based on flow chart design and a new graphical representation will be presented.

Process discovery results are presented and discussions on improvements are provided.

In Chapter 4, conclusion and future work in this research area are summarized.

Chapter 2

Process discovery for univariate alarms

2.1 Preliminaries

Several Assumptions are made about the input data before applying any process discovery algorithm:

- Each event in the log refers to a part in the mined model.
- Each event corresponds to a set of tasks known as a case or trace.
- All events are ordered in the same way as they originally occurred.

Without properly recorded raw data, process mining techniques are unable to recover useful models. The raw data can be saved in many forms, some common file formats are the Extensible Event Stream (XES), Comma-Separated Variables (CSV) and Extensible Markup Language (XML). According to IEEE Standard 1846 [39], the XES format is the standard format in event logs. Many process mining softwares, such as ProM, allows the interchangability of these file formats. Most A&E logs are saved in the CSV file format which are transformed into XES in ProM. The entries fround in A&E logs can be categorized into three main categories: alarms, actions/status, return to normal.

As mentioned in [30], operator actions are classified into two categories:

1. Primary Action: this is an actions that is recorded in A&E logs. Such action can include changing the operating limits or opening and closing a valve.

2. Secondary Action: this is an actions that is performed before or after a primary action. These actions are not recorded in the A&E log and therefore are considered non-standard and non-documented actions. Example of secondary actions include verbal communication, and the use of communication equipment. The only way to recover these actions are from expert knowledge.

According to ISA standards 18.2 there are 3 transition of alarm states, these states are process state, acknowledgement state and alarm annunciation state. x_s^t will be used to denote the process state which checks if the process variable is within a normal operating range. The following notations are based on existing literature in the field of process discovery of A&E logs [30]:

$$x_s^t = \begin{cases} 1, & \text{if } \omega_t \notin \Omega, \\ 0, & \text{otherwise,} \end{cases} \tag{2.1}$$

where ω is the variable being monitored and Ω is the normal operating range. In an industrial setting when $x_s^t = 1$, a message is produced on the control panel to alert the operator. The process state will return to normal if the operator acknowledges the alarm and a suppression is applied or the proper operator actions are performed to resolve the issue (e.g., increasing the operating limit, opening a value, reducing the pump output pressure). The acknowledgment state is formulated as

$$x_t^c = \begin{cases} 1, & \text{if } b_t = 1 \ \& \ x_{t-1}^c = 0, \\ 0, & \text{if } x_{t-1}^c = 1 \ \& \ x_t^s = 1 \ \& \ x_{t-1}^s = 1, \\ x_{t-1}^c, & \text{otherwise,} \end{cases} \tag{2.2}$$

where b_t is the physical action performed by the operator to acknowledge the alarm. Finally, the alarm annunciation state is based on both the acknowledgement state and the process state:

$$x_a^c = \begin{cases} 1, & \text{if } x_t^s = 1, \\ 0, & \text{if } x_t^s = 0 \ \& \ x_t^c = 1, \\ x_{t-1}^a, & \text{otherwise,} \end{cases} \tag{2.3}$$

For any given unique alarm tag, the alarm occurrence and return to normal should always occur in pairs. An event is then defined as a 4-tuple system, E $= \{e_t, e_v, e_s, e_f\}$ where e_t is the time stamp, e_v is the tag ID, e_s is the state,

and e_f ([0,1]) indicates the event type. $e_f = 1$ if it is an alarm and $e_f = 0$ if it is an action performed. $e_s = [1, 2, 3]$ for ALM, RTN and OA. Depending on the DCS or SCADA system the way to record e_s can vary. Using the definition of an event, an event log is then defined as L = $\{E_1, E_2, , E_l\}$ where l is the total number of events.

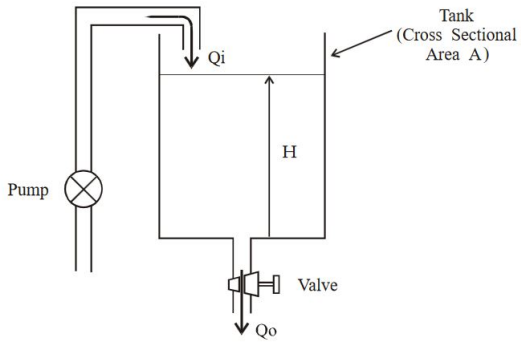

Figure 2.1: A single tank fluid level system [2]

Consider a single tank system, where the input flow rate is Q_i and the output flow rate is Q_o. The input flowrate quantity is controlled by the pump and the output rate is controlled by the valve. If the ratio between the pressure of the pump and the position of the value is not within the operating range then the level in the tank will give an alarm annunciation. Therefore in this system, three different variables are inter-related. Consider a scenario where the level in the tank is above the normal operating range set at 50-55%. Several operating procedures can be executed by the operator:

1. Increase the opening of the valve at the bottom of the tank.
2. Reduce the flow rate coming from the pump and to increase the opening of the valve.
3. Increase the normal operating threshold of the level.
4. Shut off the pump and let the level slowly return to normal.
5. Reduce the flow rate of the pump.

Any of the five sets of actions can clear the alarm, but it is also the goal of this research to determine the best operating procedure for operators. To achieve this several factors are considered:

- Time taken to reduce the alarm.
- Number of steps taken by the operator.
- Priority level of the alarms raised.
- The frequency of each unique set of actions

The events from different traces can occur simultaneously, therefore causing events to overlap in time. Several attributes are examined in order to properly segment the overlapping data. Occasionally A&E logs contain unlabled events or improperly labeled events, which increases the computational difficulty of data segmentation. Unlabeled events [40] are defined as event logs with case ID attributes missing. Due to unlabeled events, process discovery algorithms are unable to identify if any two events belong to the same process instance or trace. However, multivariate analysis deals with frequency of occurrence and do not rely trace labeling; therefore it can handle unlabeled events.

The segmentation of A&E logs consists of two parts:

1. First, decompose the event log into cases based on predefined domains. As mentioned in [40] expert knowledge is usually required to achieve such segmentation. Segmentation can also be achieved by studying Piping and Instrument Diagrams (P&ID).
2. Partitioning the cases into traces based on identifying the heads (ALM) and tails (RTN). This is also known as trace labeling.

2.2 Data pre-processing

Pre-processing the data is one of the most important steps in order to achieve a sound process model. Feeding the mining algorithm with poorly recorded event logs will only result in undecipherable and useless relationships. Therefore several categories within the A&E log will be examined:

1. Time: A uniform format is the objective in pre-processing. One common format is Date-Month-Year-Hour-Minutes-Seconds for a time stamp. Some commonly found errors are around noon or midnight of each day due to the time changing to 00:00:00 or 12:00:00. Other errors include

missing digits, repeating symbols, and incorrect symbols. Although "inadvertent time travel" [31] is not covered in this thesis due to lack of expert knowledge, this problem should be addressed if any additional knowledge is known about the process. The inadvertent time travel refers to the occurrence of an event which should never occur before another event. A simple example would be the clearing of an alarm should never occur before the same alarm is raised. More complicated relationships will have to be gathered from P&IDs, operators or other process knowledge sources. The table below shows a very common occurrence found in A&E logs.

Table 2.1: Example of events with the same time stamp

Time stamp	Type	Area	Tag ID	Parameter	State	Priority
01-Jan-17 09:00:01	Response	Pump_1	P_001 Pressure	CMD	Limit Change	0
01-Jan-17 09:00:01	Alarm	Pump_1	P_001 Pressure	RTN	LO	0
01-Jan-17 09:00:01	Alarm	Pump_1	P_001 Pressure	RTN	LOLO	0
01-Jan-17 09:00:01	Alarm	Pump_1	P_001 Pressure	RTN	LOLOLO	0

By observing table 2.1, the operator action along with the clearing of three alarms occurred at the same stamp. This would never occur in reality since the clearing of an alarm will always occur after an action but never at the same time. Imprecise data as mentioned in Table 2.2 is one way to categorize this type of behavior.

2. Type: This field in the A&E log contains one of two types of information:

   (a) An indication of an ALM or RTN in the form of 'Alarm'.

   (b) An operator response has been performed indicated by 'Response'.

   This field along with several other fields is used to find all the tail/end event of each univariate alarm.

3. Area & Tag ID: The tag ID identifies each unique component that is monitored and the area is usually the location of that unique component. A univariate alarm is then defined as a set of instances with the same area and tag ID. Case identification requires sorting the heads/starting points and tails/ending points for each unique area and tag ID.

4. Parameter: Indicates one of four possible outcomes:

   (a) ALM: an indication that an alarm has been raised.

   (b) CMD: an operation action was performed.

   (c) RTN: an indication that an alarm has returned to normal.

   (d) Status: a change in the status of a tag. Some commonly found states associated with status include 'Error', 'Open', 'Close', and 'Reset'. Some of these events with the status parameter can also be treated as an operator action since a command is required to change the status.

5. State: Indicates the type of alarm raised. For most tags, three different levels below the nominal operating point and three different levels above the nominal operating point are set in place: (LO, LOLO, LOLOLO, HI, HIHIHI, HIHIHI). Both the annunciation and return to normal for any tag will display the same state within the A&E log.

6. Priority: Usually ranges from 0-3 to separate the normal alarms with the alarm that requires immediate attention. The alarm priority distribution standards can be found under ISA 18.2, a standard used by most industries for the management of alarm systems.

Other issues related to A&E logs includes missing, incorrect and unclear data. Techniques such as inserting extra zeroes, spaces or symbols can correct missing and improper data. However, unclear data or over generalized data cannot be corrected through pre-processing. Most A&E logs are set up in such a manner, although they may slightly differ in the number of attributes. To summarize, we can refer to the imperfection patterns discussed in [31].

Table 2.2: Imperfection pattern found in A&E logs

Imperfection pattern	**Data quality dimensions**	**Effect**
Same time stamp	Imprecise data	Reducing the temporal ordering of events
Unanchored events	Imprecise data, Incorrect data	Algorithms cannot interpret Events
Elusive case	Missing data	Events do not contain case ID
Polluted label	Data is too precise	Too many unique events

The first step to pre-processing is removing the following from each field:

1. Leading spaces, tailing spaces and duplicate spaces.

2. Symbols: commas, periods, brackets, dashes, and so on.

2.3 Data segmentation

The head of a trace indicates the alarm annunciation of any tag and the tail refers to the return to normal of the same tag. All information between the head and the tail corresponding to that tag is assigned the same trace ID. The goal of Algorithm 1 is to search the whole A&E log and assign trace numbers to all the events; this process is refered to as data segmentation. Depending on how the A&E log was recorded, the algorithm to find the head and tail will vary. In the industrial pipeline data used, two examples of commonly found alarms are shown below.

Table 2.3: ALM and RTN of a univariate alarm example 1

Time Stamp	Type	Area	Tag ID	Parameter	State	Priority
01-Jan-17 10:01:13	Alarm	Pump_1	P_001 Flow	ALM	LO	0
01-Jan-17 10:01:34	Alarm	Pump_1	P_001 Flow	ALM	LOLO	0
01-Jan-17 10:01:58	Alarm	Pump_1	P_001 Flow	ALM	LOLOLO	0
01-Jan-17 10:06:01	Response	Pump_1	P_001 Flow	CMD	Limit Change	0
01-Jan-17 10:06:01	Alarm	Pump_1	P_001 Flow	RTN	LO	0
01-Jan-17 10:06:01	Alarm	Pump_1	P_001 Flow	RTN	LOLO	0
01-Jan-17 10:06:01	Alarm	Pump_1	P_001 Flow	RTN	LOLOLO	0

Table 2.4: ALM and RTN of an univariate alarm example 2

Time Stamp	Type	Area	Tag ID	Parameter	State	Priority
01-Jan-17 11:01:01	Alarm	Valve_1	V_001 Pressure	ALM	Hi	3
01-Jan-17 11:01:51	Alarm	Valve_1	V_001 Pressure	RTN	Hi	3

For the first type of univariate alarms, 'RTN' in the parameter column of the A&E indicated the tail of a trace. The corresponding ALM will have the same Area, Tag ID and State (columns 3, 4 and 6) as the tail. In this particular case it can be noticed that the operator took approximately four minutes to address the alarms. Again, the issue of events with the same time stamp is present. The second type of alarm contains single head and tail without any operator action in between which provides no additional knowledge about the

process. This type of univariate alarms will be removed before applying any process discovery algorithms.

Table 2.5: Operator action example

Time Stamp	Type	Area	Tag ID	Parameter	State	Priority
13-Jan-17 11:00:01	Response	Valve_1	V_001	Status	CLOSED	3
13-Jan-17 11:01:51	Response	Valve_1	V_002	Status	RESET	2
13-Jan-17 11:01:51	Response	Valve_1	V_003	Status	RESET	2

The figure below displays a typical continuous alarm sequence. It also is very common for alarms to repeat, therefore only the location between the last repeating alarm and return is recorded.

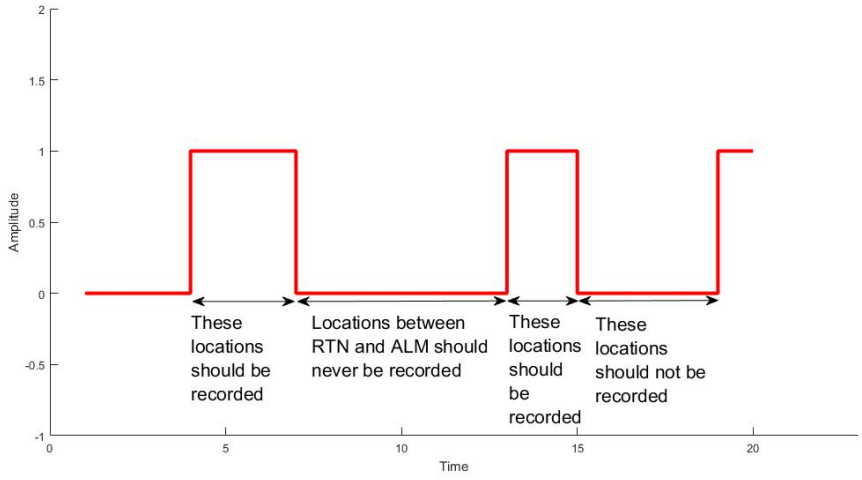

Figure 2.2: Example of an alarm sequence: amplitude of 1 indicates an alarm and amplitude of 0 indicates a return to normal

After the data segmentation, the process mining algorithm can then be applied to the A&E log. In Algorithm 1, UniqueTags contains all the unique tag IDs and Location stores the locations between each individual head and tail. For this particular input data, 13917 events are recorded over an one year period. 38 unique tag IDs and 2883 traces are mined from the A&E log.

Algorithm 1 Algorithm to mine univariate relationships

```
1: procedure
2:     for i = 1 : length(EventLog) do
3:         for j = 1 : length(UniqueTags) do
4:             Find all alarms within UniqueTags(j) in EventLog
5:             Store the location in variable head
6:             Find all RTN with tag UniqueTags(j) in EventLog
7:             Store the location in variable tail
8:         end for
9:     end for
10:    f, k, h = 1
11:    while K ≤ length(head) ∧ h ≤ length(tail) do
12:        if K < length(head) ∧ Head(k + 1) < Tail(h) then
13:            Skip or remove head(k)
14:            k = k+1
15:        else
16:            if Head(k) > Tail(h) then
17:                Skip or remove Tail(h)
18:                h = h+1
19:            else
20:                if Head(k) < Tail(h) then
21:                    Store all locations between head(k) ∧ tail(h) in Location{f}
22:                    f=f+1
23:                end if
24:            end if
25:        end if
26:        k=k+1
27:        h=h+1
28:    end while
29:    for i = 1 : length(Location) do
30:        w=1
31:        while W ≤ length(Location{i}) do
32:            Compare the tag ID of the head with the rest of the tag IDs in Location{i}
33:            if The tag ID matches with the head then
34:                W=W+1
35:            else
36:                if It is not a match then
37:                    Remove from Location{i}
38:                end if
39:            end if
40:        end while
41:    end for
42: end procedure
```

In order to segment the data the start of an alarm always ends with the return to normal, we then consider this as a trace. In between the head and tail contains operator actions and status updates. Occasionally there are traces where no useful information can be mined, such a trace contains only a single head and tail. However in multivariate analysis, some of theses traces containing two events can be related to other events. This scenario will be covered in Chapter 3 of this thesis. Algorithm 1 can be divided into 3 main section:

1. The loop from lines 1 to 9 in Algorithm 1 searches through the entire event log for the locations of all unique tags. Depending on how the events are recorded, the way to identify the annunciation and clearing of an alarm will differ. The locations for all the tags will be unique and will range from 1 to the length of the A&E log.

2. The loop from lines 11 to 28 in Algorithm 1 attempts to sort through the locations of the heads and tails. The conditions are set up to capture the locations as described in Figure 2.2. To simply state it, the head of a trace should always occur before the tails and no other heads or tails should occur in-between. This information is then stored in variable Locations{N}, where N is the total number of unique tags.

3. The loop from lines 29 to 42 extracts the useful information from variable Locations. Since this is an algorithm to capture univariate relationships, only events with the same tagID as the head and tail will be retained. Therefore each individual traces will contain the same tagID. The issue of completeness is evident after implementing Algorithm 1. With an industrial A&E log of 13917 events, only 6805 events were considered to create the process model in Figure 2.3 and Figure 2.5. No conclusions can be made whether the removed events contain any additional information but improvement should be made to consider all the given information.

Table 2.6 shows an example of the A&E log after applying Algorithm 1.

Table 2.6: Example of A&E log after data segmentation

Time Stamp	Tag ID	Case ID
15-01-01 06:55:49	ST1.HI.ALM	1
15-01-01 06:56:55	ST1.HIHI.ALM	1
15-01-01 06:58:30	ST1.HIHIHI.ALM	1
15-01-01 07:34:21	ST1.Limit Change.CMD	1
15-01-01 07:34:21	ST1.HIHIHI.RTN	1
15-01-01 07:34:21	ST1.HIHI.RTN	1
15-01-01 07:34:21	ST1.HI.RTN	1
15-01-01 08:02:08	ST1.HI.ALM	2
15-01-01 08:13:34	ST1.Limit Change.CMD	2
15-01-01 08:13:34	ST1.HI.RTN	2
15-01-03 11:01:51	ST3.LO.ALM	3
15-01-03 11:02:34	ST3.MANUAL INPUT.CMD	3
15-01-03 11:02:34	ST3.PUT IN SCAN.CMD	3
15-01-03 11:02:34	ST3.LO.RTN	3

2.4 Graphical representation

Firstly some research containing the graphical representation of alarm data will be covered. The high density alarm plot (HDAP) and alarm similarity color map (ASCM) have been utilized for the purpose of alarm rationalization [41]. HDAP examines the alarm count for all unique tags over a time window. The results are then examined to find potential chattering alarms, redundant alarms and plant instability. ASCM examines the alarm sequence of all tags and applies the Jaccard similarity measure [42] to find potentially redundant or related alarms. Workflow models [26] are another visualization tool based on textual message to monitor procedual compliance and process safety [32]. It is important to distinguish between workflow nets and petri-net models due to their similarity. Workflow nets are petri nets with a single source and a single sink: this means all process start with the same event and end with the same event. Representing univariate alarms will always be in the form of workflow nets since all events begin with an alarm and end with a return to

normal. Petri nets, widely used models for the execution of events, were also utilized in A&E logs [30].

Some new notations are created to construct the process model. In the new graphical representation two main parts are defined:

1. Nodes: indicate the type of events that have occurred. These events can be separated into alarms, actions, and return to normal. Three levels of alarm annunciation are observed in Figure 2.4. A yellow downwards triangle indicates the annunciation of an alarm in the High state, while its counter part, the return to normal of an alarm in the high state is indicated by the upwards yellow triangle. The same logic applies to HIHI alarms and returns indicated by orange triangles and HIHIHI alarms and returns by red triangles. Three levels of alarms are also defined for low alarms. Low alarms are indicated be downwards light blue triangle. LOLO and LOLOLO alarms are indicated by purple and dark blue downwards arrows respectively. If more than three alarm levels are utilized, then more colors would have to be defined for each level. Operator actions and status updates are indicated by oval nodes. No clear distinctions are made between action and status due to two main reasons. First, it can be difficult to distinguish between action and status due to the lack of expert knowledge. Second, operator actions are usually followed by a status update even if the operator action was not recorded. A node with solid color indicates that a self loop is present, this suggests that repeating of the same event was found in the event log.

2. Edges: Three types of edges are defined in this graphical representation. A green arrow indicates a forward occurrence while a red arrow indicates a reverse or feedback event. In most cases, a well structured or ideal process model should not contain self loops and reverse events. A dashed line indicates a partial process model and another dashed line of the same color connects both models. Dashed lines are useful for large and complex process models.

The list below summarizes the notations for the representation of A&E logs:

- $\triangleleft$: The return to normal of an tag. Light blue indicates low state,

purple indicates LOLO states, and dark blue indicates LOLOLO states. Yellow indicates hi states, orange indicates HIHI states, and red indicates HIHIHI states.

- ▷: The alarm annunciation for an individual tag. The same color scheme as the return to normal will be utilized.
- ◀: The return to normal of an tag with a feedback. The same color scheme will be used.
- ▶: The annunciation of an individual tag with a feedback. The same color scheme will be utilized.
- $\longrightarrow$: The connection of two events in the forward direction.
- $\longleftarrow$: The connection of two events in the reverse direction.
- $\cdots$: The connection of two partial process models.
- - - -: The connection of two incomplete process models. This suggests that one or more events are missing between the symbols.
- ⬭: The indication that an operator action or status update has occurred. A solid symbol indicates that a feedback is present.

2.5 Modeling univariate relationships

Several process mining software are available to model traces. ProM, Disco, Celonis, myInvenio and Minit are just a few of the many process mining software tools. In this thesis ProM, a commonly used process mining software in academic research will be used to model the univariate behavior.

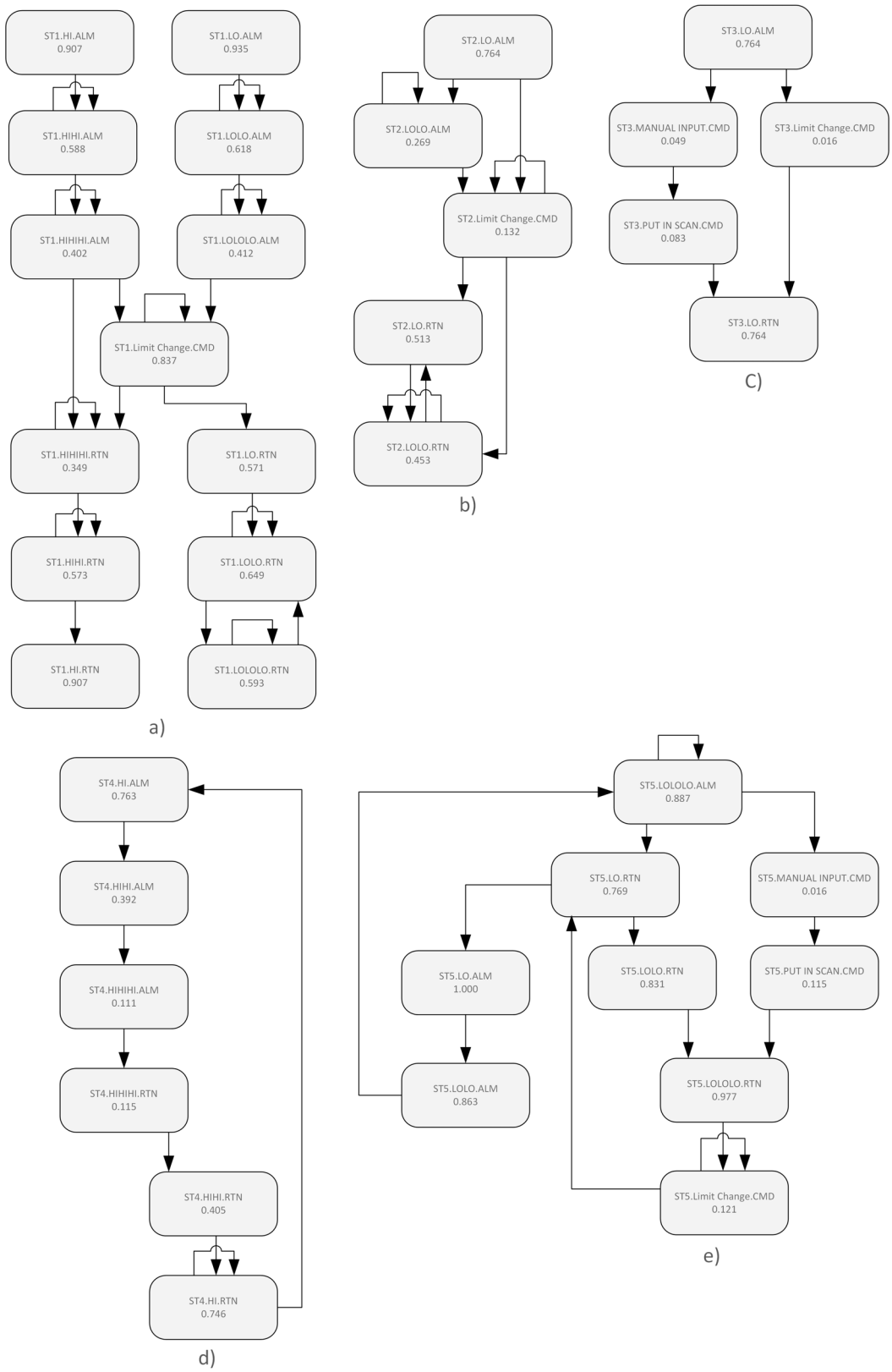

Figure 2.3: Fuzzy model to capture univariate relationships

By inpecting Figure 2.3, the model is quite structured and all events begin with ALM and end with RTN which reconfirms Algorithm 1. Some key results can be concluded from the fuzzy model.

- The sequences Tag_HI → Tag_HIHI → Tag_HIHIHI → Operator Action → Tag_ HIHIHI_CL → Tag_ HIHI_ CL → Tag_ HI_ CL is very common. Intuitively this is reasonable, if an alarm variable reaches high alarm threshold it is very likely that it will continue to escalate to HIHI and HIHIHI. Also, the clearing of an alarm will always occur in the same order because the last alarm will always be cleared before any previous alarms.

- There are cases where the same set of operator actions are able to clear multiple different alarm variables. in 2.3 ST1.Limit.Change.CMD is able to clear both alarms in the high and low states of tag ST1.

- Two sets of operator actions are able to clear the same alarm variable.

- Some events tend to repeat by observing the self feedback in the fuzzy model.

- Operator action Manual_Input.CMD is always followed my Put_In_Scan.CMD

By examining the log summary, some unique characteristics about the A&E log are listed below:

- The total number of process instances/traces is 1536.

- The total number of events is 6805, which indicates that more than half of the events are removed through pre-processing.

- On average, only 4 events are within each trace with a maximum of 15 event.

Out of 2883 traces, only 4-5 useful sets of relationships are mined out of a total of 7 process models. This is due to two main reasons, first, the mining algorithm is restricting the trace search to only events with the same tag ID. Second, the fuzzy mining algorithm tends to over generalize the input data thereby removes less frequently occurring traces. This problem will be examined in multivariate design of event logs in the next chapter.

Using the new notation discussed in section 2.4, Figure 2.4 was constructed.

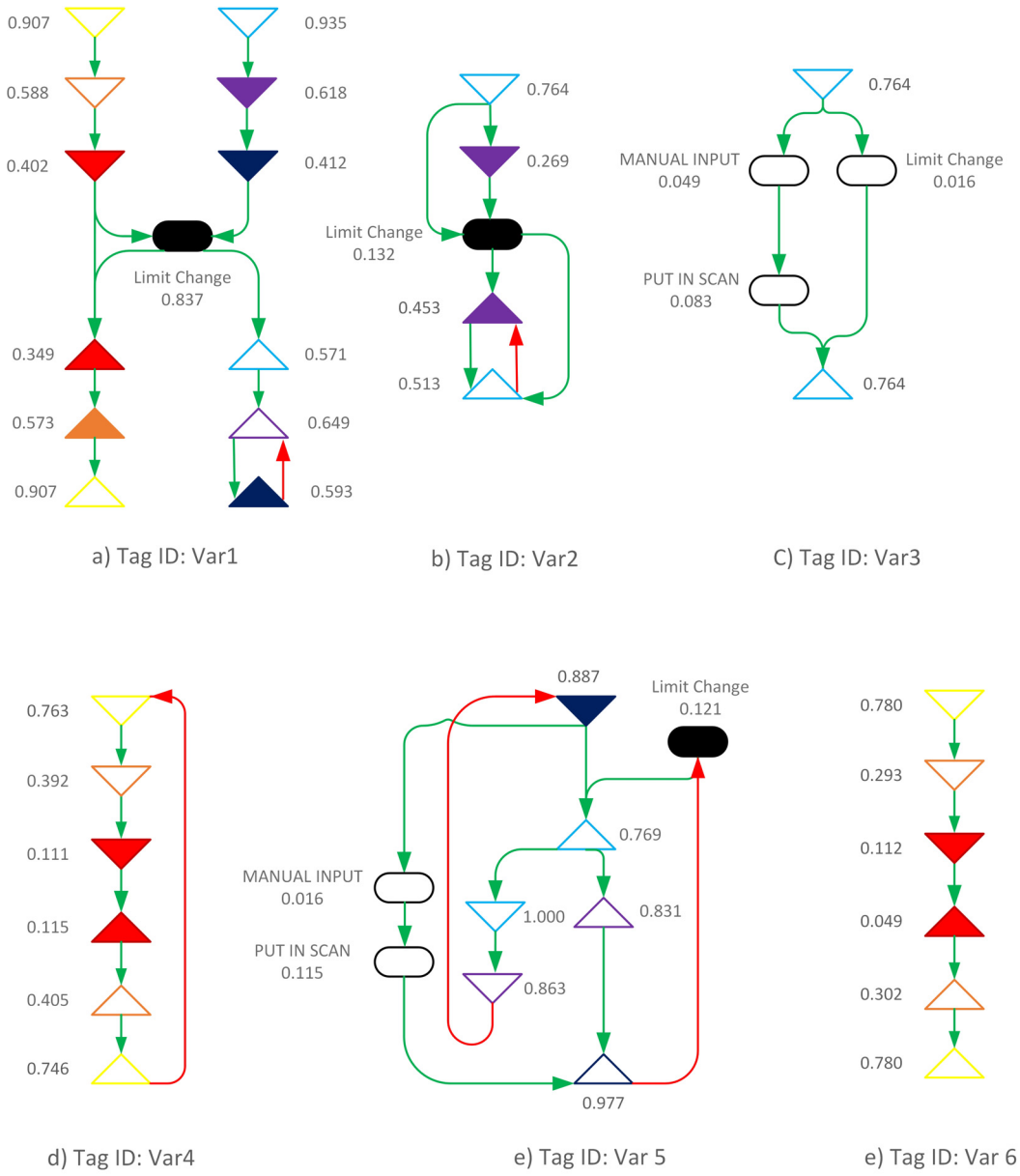

Figure 2.4: New graphical representation to capture univariate relationships

To reconfirm the results collected from the fuzzy model, a heuristics miner is applied to the data. Using the default parameters set by ProM, a heuristics net is produced in Figure 2.5.

Table 2.7: Default parameters of the heuristics miner

Parameters	**Percentage**
Relative to Best	5
Direct Dependency	90
Length-Two Dependency	90
Long Distance Dependency	90

The general structure of the model remained consistent and feedbacks observed in the fuzzy model were removed. More connections were made between operator actions and alarms/returns. Overall the heuristics net provided a more detailed relationship while keeping the model manageable. For these reasons the foundation of multivariate analysis will be based on the heuristics miner algorithm.

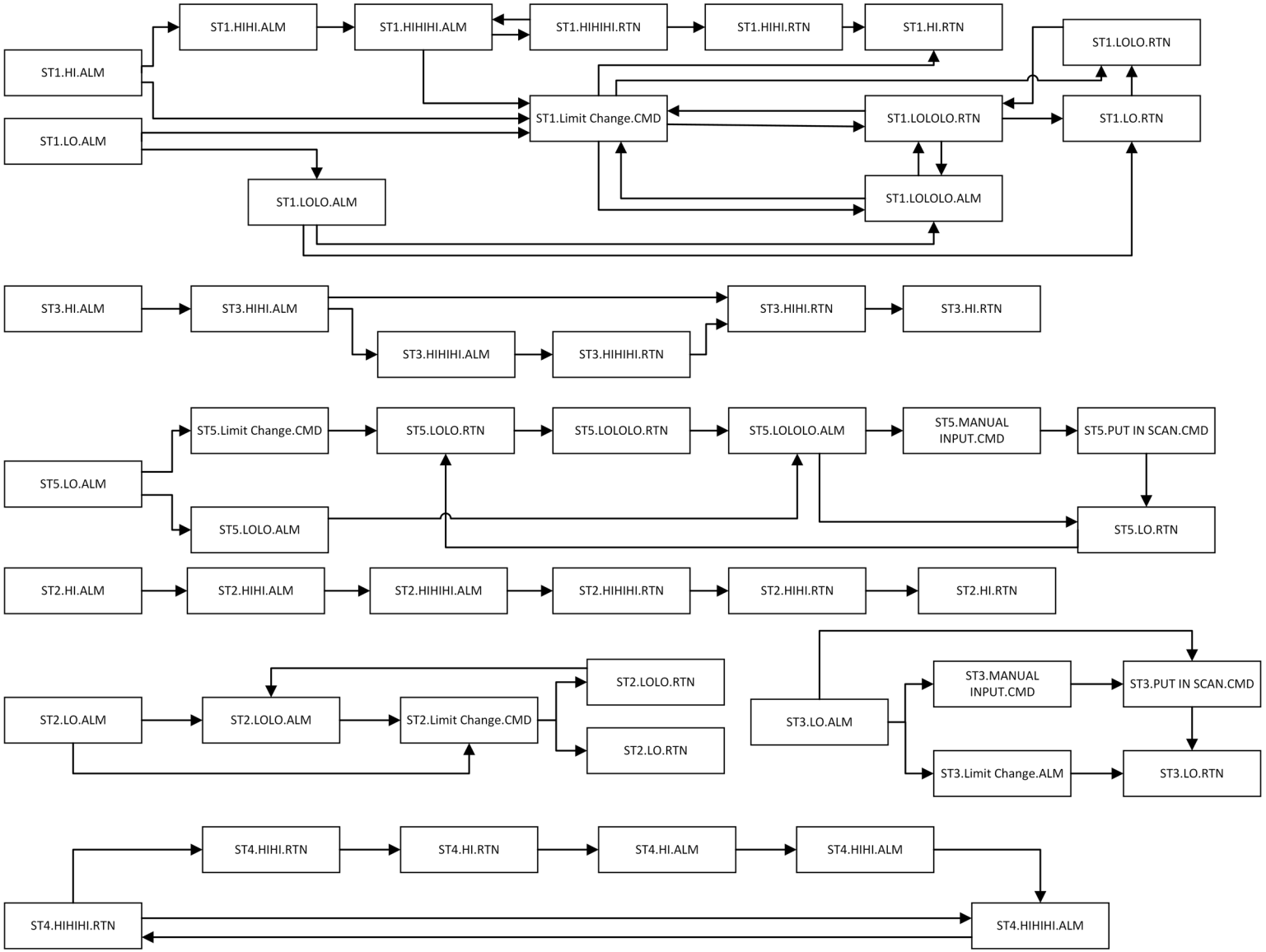

Figure 2.5: Heuristics net of univariate relationships

This chapters examined the common attributes of an A&E logs and provided some pre-processing techniques for imperfection patterns. The algorithm for case identification based on predefined parameters was presented in Algorithm 1. Finally, two process mining algorithms were used on the data to compile the process models. In the next chapter, methods to extracts multi-variate alarm relationships will be examined.

Chapter 3

Process discovery for multivariate alarms

3.1 Introduction

Process mining of multivariate alarms involves examining the relationships between alarms of different tag IDs. In most industrial processes, many process variables being monitored are physically interconnected. It was mentioned in [30] that for industrial A&E logs alarms and actions are often related to many processes. Due to such inter-dependent relationships in industrial plants, the dependencies among all variables should be examined. To the authors knowledge, this work on multivariate alarms is the first of its kind. Unlike univariate analysis, there is no "head" or "tail" in multivariate process mining. The whole event log is treated as a sequence and bivariate relationships are extracted based on frequencies of occurrence. Simple preprocessing techniques similar to univariate analysis are applies for consistency in the data.

Several notations will be introduced [20, 23]:

Let W be an A&E log of length T.

- A $>$ B iff within the event log $W = t_1\ t_2\ t_3\ \ldots\ t_T$ there exsists i $\in \{1, 2, \ldots, T-1\}$ such that $t_i = A$ and $t_{i+1} = B$.
- A $\gg$ B iff within the event log $W = t_1\ t_2\ t_3\ \ldots\ t_T$ there exsists i $\in \{1, 2, \ldots, T-2\}$ such that $t_i = A$, $t_{i+1} = B$ and $t_{i+2} = A$.
- A $\ggg$ B iff within the event log $W = t_1\ t_2\ t_3\ \ldots\ t_T$ there exsists i<j such that $t_i = A$ and $t_j = B$.

A simple example demonstrates how the notations are applied to an event log W. Given event log W=ACBACACCBAAA where A B and C are unique events, the table below shows the frequencies of occurrence between any two events.

Table 3.1: Frequencies of the occurrence table

>	A	B	C
A	2	1	2
B	3	0	0
C	0	2	1

Dependency is then defined as the presence of certain variables tends to imply the presence of other variables. To be more specific, within the heuristics miner algorithm, 5 types of dependencies are defined [20, 23].

1. Direct Dependency: The occurrence of one event directly causes another event to occur. A value close to 1 will indicate a strong bi-variate relationship between A and B:

$$A \rightarrow B = \frac{|A > B| - |B > A|}{|A > B| + |B > A| + 1} \tag{3.1}$$

2. Self Dependency: The occurrence of the same event consecutively within an event log. This type of dependency usually occurs when the same alarm is raised multiple times in succession.

$$A \rightarrow A = \frac{|A > A|}{|A > A| + 1} \tag{3.2}$$

3. Length-Two Dependency: This type of dependency can be regarded as a feedback system. The occurrence of A followed by B is followed again by A (ABA):

$$A \rightarrow_2 B = \frac{|A \gg B| - |B \gg A|}{|A \gg B| + |B \gg A| + 1} \tag{3.3}$$

4. Long Distance Dependency: similar to direct dependency except A is followed by B within a set window size.

$$A \rightarrow_l B = \frac{|A \ggg B| - |B \ggg A|}{|A \ggg B| + |B \ggg A| + 1} \tag{3.4}$$

5. AND/XOR Dependency: If an event has multiple direct dependencies then those events must be either be AND or XOR dependencies. For example, A is followed by B and A is followed by C, an AND relationship suggests that both B and C will occur after A. XOR dependency suggests that A followed by B and A followed by C are two separate events with no relation:

$$A \to B \wedge C = \frac{||B > C| - |C > B||}{|A > B| + |A > C|} \tag{3.5}$$

Table 3.2: Default parameters for the heuristics miner

Parameter	Value
Length One Dependency (L1D)	0.9
Relative to Best (RTB)	0.05
Length-Two Dependency (L2D)	0.9
Long Distance Dependency (LDD)	0.9

Several key parameters can be adjusted to fit the user's requirement. High restrictions on all three dependencies will result in over generalized or partial models, while low restrictions will result in unwanted relationships. The user's expert knowledge about the process and interpretation skills can greatly improve the results. Given the following direct dependency and length-two dependency matrices, we can construct a process model.

The direct dependency matrix is calculated from equations (3.1) and (3.2) and are based on the frequencies of occurrence table. Self dependency is only displayed on the diagonals of the direct dependency matrix. Any dependency that is negative will not be considered in the heuristics miner algorithm and are be treated as zero. If these negative dependencies are not treated as zeroes then the dependency matrix is skew-symmetric. This statement can be verified by examining equation (3.1). Dependencies highlighted in green are relationships that are the highest in their respective row or column; however they do not meet the direct dependency threshold of 0.9. Dependencies highlighted in red and orange are relationships that are accepted or 'mined' from the direct dependency matrix. Dependencies that are in orange are not the highest values in their respective row or column, but they are accepted due to the relative to best threshold of 0.05. This means that any dependency within a range of 0.05 to the highest dependency of that row or column will

Table 3.3: Direct dependency matrix

→	A	B	C	D	E	F	G	H	I	J	K	L
A	0	.991	.982	0	0	0	0	0	.988	0	0	.175
B	0	.902	.114	.911	.985	0	0	0	.323	.384	0	0
C	0	0	0	.328	.272	.981	.542	0	.912	0	0	0
D	0	0	0	0	0	.650	0	0	0	0	.833	.130
E	0	0	0	0	0.75	0	.620	0	0	0	0	.345
F	0	0	0	0	.125	0	0	.940	.084	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	.485
H	0	.012	0	0	0	0	0	.905	.205	0	0	0
I	0	0	0	.073	0	0	.845	0	0	0	0	0
J	0	0	0	.435	0	0	.062	.144	0	0	0	0
K	0	0	0	.134	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	.21	0	0	0	0

be collected and used for process modeling. If the highest dependency does not meet the minimum requirement of 0.9 then the relative to best threshold will not be considered for that row or column. Length-two dependency will also be examined to capture any feedback relationships. By applying equation (3.3) with the frequency of occurrence table the following matrix is calculated.

Table 3.5: XOR/AND dependencies for multiple outgoing edges

Notation	**XOR/AND Dependency**
$A \rightarrow B \wedge C$	0.058
$A \rightarrow B \wedge I$	0.163
$A \rightarrow C \wedge I$	0.482
$B \rightarrow E \wedge D$	0
$F \rightarrow D \wedge H$	0
$C \rightarrow F \wedge I$	0.044

For the purpose of demonstration and simplicity, long distance dependency will not be considered for the final process model. As we will see in industrial case studies, long distance dependencies tend to over complicated the process model. By examining the process model several conclusions can be made:

- All traces begins with event A.
- Three main sets of traces are observed in the model.

Table 3.4: Length-two dependency matrix

→	A	B	C	D	E	F	G	H	I	J	K	L
A	0	0	.042	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	.412	0
E	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	.080
H	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	.120	0	0	.842	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	.341	0	0	0	0

- There are multiple inter-dependencies among all three sets of traces.
- Some events (B and H) tends to repeat in a traces which can potentially indicated chattering.

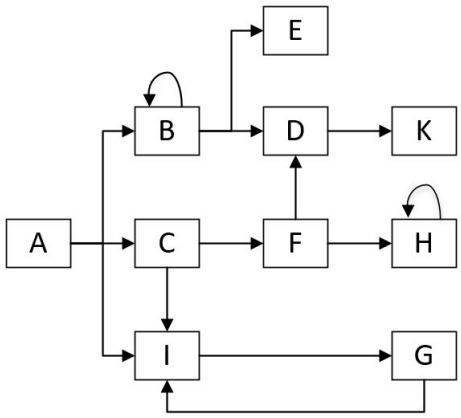

Figure 3.1: Process model based on direct and length-two dependency matrix

3.2 Heuristics miner algorithm

The key parameters in the heuristics miner algorithm are applied to the event log to extract a frequency based matrix. Several predefined variables are use in this algorithm which were from Algorithm 1.

- TagState: The string combination of Area, Tag ID and state. For example, 'ST5.LOLO' is a TagState indicating that this alarm contains a Area named ST5 and the current state is LOLO(low low) compared to the normal operating limit.
- Locations: Contain n number of individual cells, where n is the number of unique TagStates. Each cell will contain the locations of all the corresponding tagstate. The size of each cell will vary depending on the frequency of occurrence for each tagstate.

The following table shows the threshold parameters used in the heuristics miner algorithm.

Table 3.6: Parameters for the heuristics miner

Parameter	Value
Length One dependency (L1D)	0.7
Relative to best (RTB)	0.1
Length-two dependency (L2D)	0.7
Long distance dependency (LDD)	0.7
Positive observation (PO)	10
Long distance length	10

The dependency graphs are of size $n \times n$ where n is the number of unique events. In this particular case n is equal to the length of Unique_Tagstate. Three dependency graphs are created using the hybrid heuristics miner algorithm: Direct Dependency, Length-Two Dependency and Long Distance Dependency.

Algorithm 2: Lines 1 to 5 checks each unique tagstate for meeting the minimum positive observation count. If this condition is not met then these tagstates will not be considered for the remaining algorithm. After the positive observation threshold is checked, lines 6 to 19 creates three dependency graphs, namely, direct dependency, length-two dependency and long distance dependency. This part of the algorithm can be divided into three main parts. First, lines 10 to 14 calculates the direct dependency values along the diagonal; these values differ from equation (3.1) and are referred to as self-dependency. Second, lines 15 to 19 uses equation (3.1), (3.2) and (3.4) to calculate the remaining dependency values. Finally, lines 20 to 15 checks for any negative dependencies and removes them from the dependency matrix.

Algorithm 2 Hybrid heuristics miner algorithm to create depedency matrix

1: **for** $i = 1 : length(Locations)$ **do**
2: **if** $length(locations\{i\}) \leq PO\_threshold$ **then**
3: TagState will not be considered in rest of the algorithm
4: **end if**
5: **end for**
6: Create a three dependency graph of size length(Unique_Tagstate) x length(Unique_Tagstate)
7: **for** $x = 1 : length(Unique\_Tagstate)$ **do**
8: y=1
9: **while** $y \leq length(Unique\_Tagstate)$ **do**
10: **if** $x = y$ **then**
11: Using algorithm 3.2 to calculate the self dependency values and store in Direct_Dependency(x,x)
12: A value of 0 is assigned to Length_Two_Depdency(x,x)
13: A value of 0 is assigned to Long_Distance_Dependency(x,x)
14: **else**
15: A=location{x}
16: B=location{y}
17: Using eqn 3.1 to calculate the direct dependency values and store in Direct_Dependency(x,y)
18: Using eqn 3.3 to calculated the length-two dependency value and store in Length_Two_Depdency(x,y)
19: Using eqn 3.4 to calculate the long distance dependency values and store in LDD_Dependency(x,y)
20: **if** $Direct\_Dependency\_value < 0$ **then**
21: Set the direct dependency value to 0
22: **end if**
23: **if** $Length\_Two\_Dependency\_value < 0$ **then**
24: Set the length-two dependency value to 0
25: **end if**
26: **end if**
27: y=y+1
28: **end while**
29: **end for**

Algorithm 3 Mining multivariate relationships from dependency graphs

1: **for** $x = 1 : length(Unique\_Tagstate)$ **do**
2: Find the highest value within column x and row x of the direct dependency graph
3: Find the highest value within column x and row x of the length-two dependency graph
4: Find the highest value within column x and row x of the long distance dependency graph
5: **if** $The\_highest\_column\_value > L1Dthreshold$ **then**
6: Take all values greater than highest_column_value-RTB Threshold and storing it into DDresult_Follower{x}
7: **else**
8: no value will be assigned to DDresult_Follower{x}
9: **end if**
10: **if** $The\_highest\_row\_value > L1Dthreshold$ **then**
11: Take all values greater than highest_row_value-RTB_Threshold and storing it into DDresult_Cause{x}
12: **else**
13: no value will be assigned to DDresult_Cause{x}
14: **end if**
15: **if** $The\_highest\_column\_value > L2Dthreshold$ **then**
16: Take all values greater than highest_column_value-RTB Threshold and storing it into L2Dresult_Follower{x}
17: **else**
18: no value will be assigned to L2Dresult_Follower{x}
19: **end if**
20: **if** $The\_highest\_row\_value > L2Dthreshold$ **then**
21: Take all values greater than highest_row_value-RTB_Threshold and storing it into L2Dresult_Cause{x}
22: **else**
23: no value will be assigned to L2Dresult_Cause{x}
24: **end if**
25: **if** $The\_highest\_column\_value > LDDthreshold$ **then**
26: Take all values greater than highest_column_value-RTB Threshold and storing it into LDDresult_Follower{x}
27: **else**
28: no value will be assigned to LDDresult_Follower{x}
29: **end if**
30: **if** $The\_highest\_row\_value > L2Dthreshold$ **then**
31: Take all values greater than highest_row_value-RTB_Threshold and storing it into LDDresult_Cause{x}
32: **else**
33: no value will be assigned to LDDresult_Cause{x}
34: **end if**
35: **end for**

36: x=1
37: **for** $x = 1 : length(DDresult_Follower$ **do**
38: **if** $length(DDresult_Follower1, x) > 1$ **then**
39: Find all possible pairs of combinations in DDresult_Follower{1,x} and store in variable Choose
40: **for** $i = 1 : length(choose(:, 1)$ **do**
41: Apply algorithm 3.5 to find the XOR/AND dependency values and store in XOR_AND_Result
42: **end for**
43: **if** $XOR\_AND\_Resultxisnon - empty$ **then**
44: Find all values within XOR_AND_Result{x}>0.5 and store the results in AND_Result{z}
45: z=z+1
46: **end if**
47: **end if**
48: **end for**

Algorithm 3: The for loop from lines 1 to 4 selects the highest row and column values from all three dependency matrices. Since the dependency matrices are always square, each individual dependency can be called by their respective x and y values. The remaining algorithm checks whether each of the highest row or column values meet the threshold requirement. If this condition is met then the algorithm selects all dependencies in that respective row or column with dependencies greater than the highest row/column minus the relative to best threshold.

3.3 Industrial case study

The A&E log data discussed in this section are obtained from industrial facilities. This event log consists of 13917 instances over a period of one year. Priority levels ranges from 0 to 4; however priority 4 was removed from the data since it was only encountered 6 times and no actions were recorded.

Table 3.7: Priority distribution for the industrial A&E log

Priority level	Alarm Count	Proportion
priority 0	9495	68.2%
priority 1	107	0.768%
priority 2	1769	12.71%
priority 3	2540	18.25%

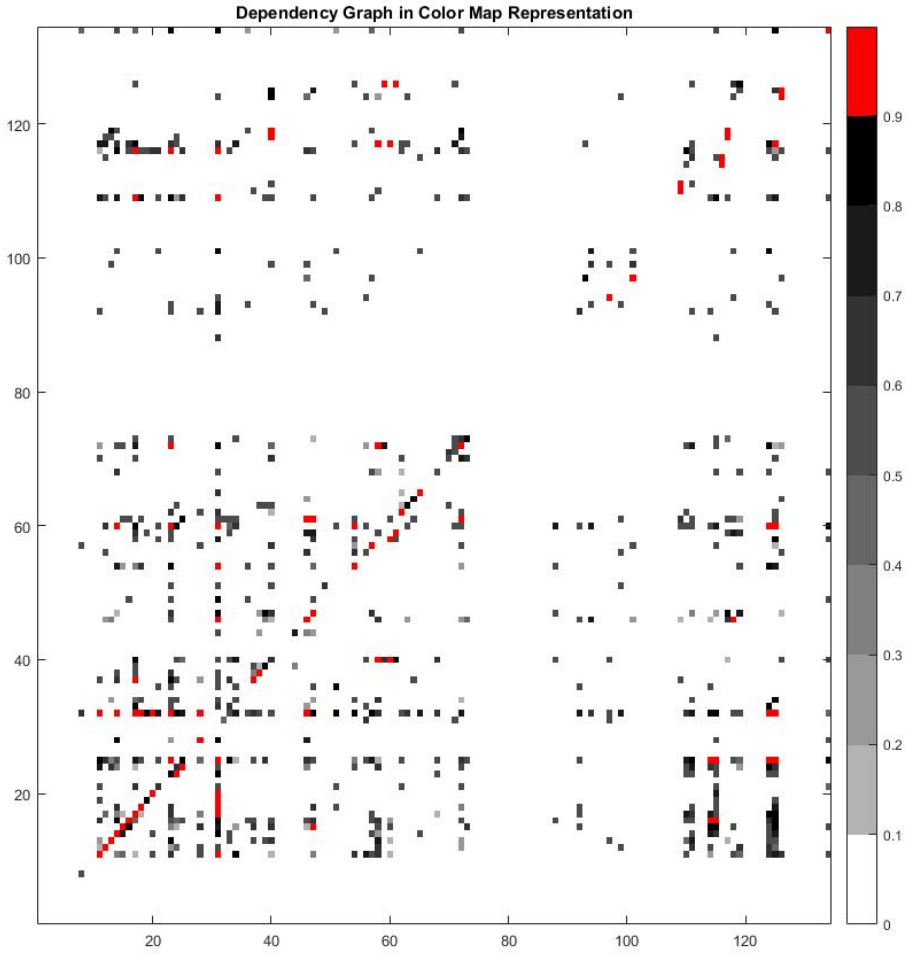

Figure 3.2: Color map representation of direct dependency matrix

Since there are 134 unique events in this A&E log, the size of all the dependency matrices are therefore 134 $\times$ 134. All dependencies greater than

0.9 are highlighted in red and are used to create the process model. Two interesting patterns are seen in the direct dependency matrix:

1. There is a high number of self-dependencies along the diagonals of the matrix that are greater than the threshold of 0.9. This behavior can be explained by examining equation 3.2. Since self dependency is directly related to the occurrence of event repeating, if any event repeats more than 8 times then self dependency will be greater than 0.9.

2. There are no dependencies between events 74 to 87. By going back into the algorithm and examining variable "location" it was concluded that all the variables from 74 to 87 occurred less than the positive observation threshold of 10. Therefore on both the horizontal and diagonal axes events 74 to 87 contains no dependency relationships.

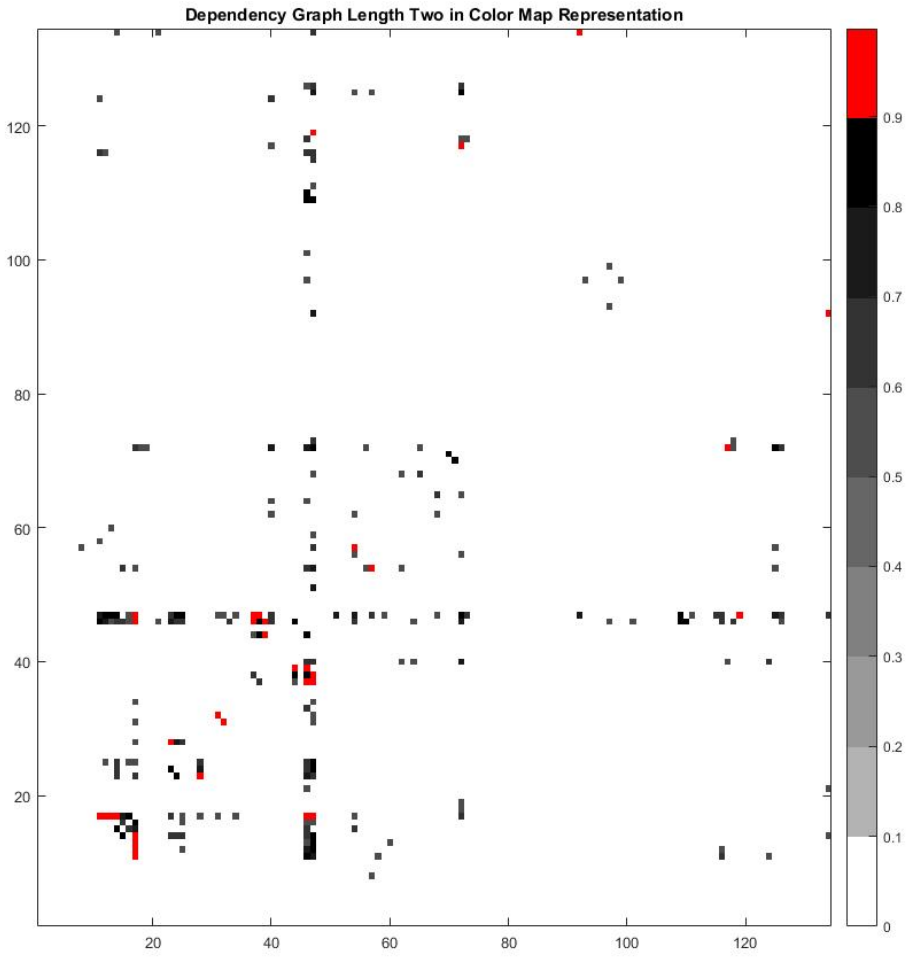

Figure 3.3: Color map representation of length-two dependency matrix

The length-two dependency matrix contained fewer relationships compared to the direct dependency matrix, however, a clear pattern is found between events 46 to 47. This suggested that both events were related to most of the other events in the log. These events corresponded to 'PCOUT_FROM_C1' and 'PCOUT_FROM_C2' in Figure 3.5 e). One explanation for this behavior could be due to the over generalization of commands, where multiple different operator actions are recorded in the same format.

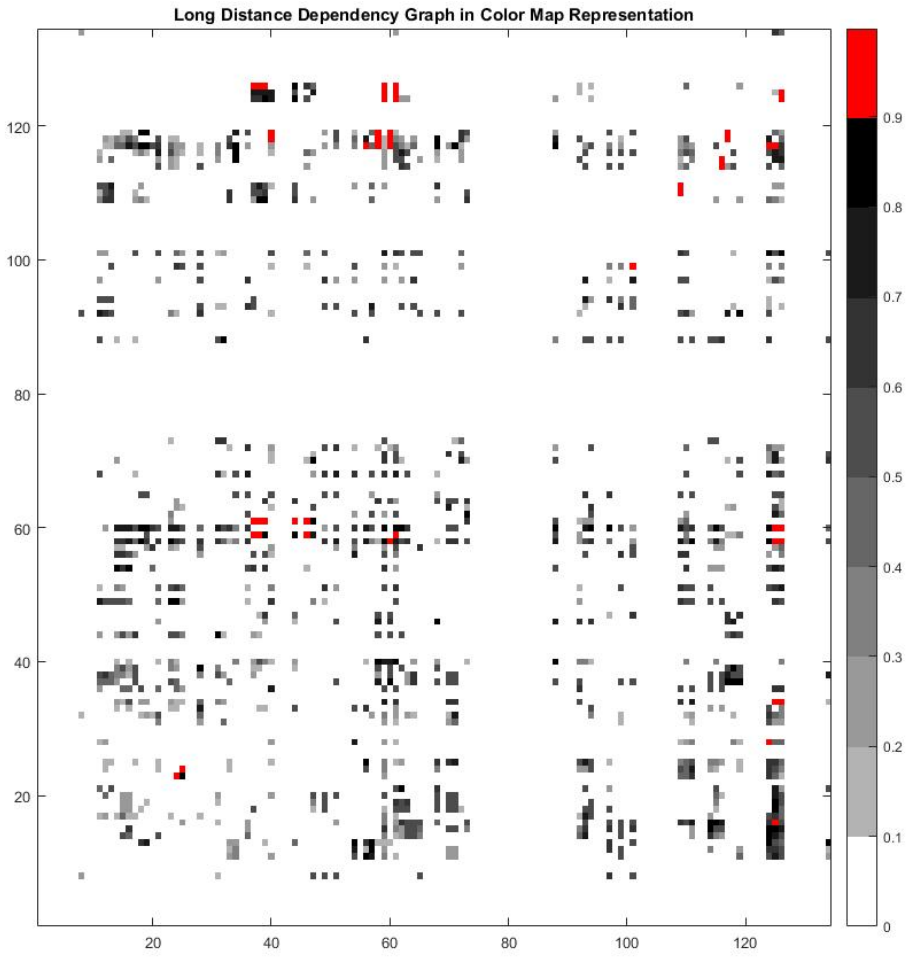

Figure 3.4: Color map representation of long distance dependency matrix

Although long distance dependency is not incorporated into the process model, this matrix could be used to mine additional relationships. As expected, long distance dependency contains most relationships among all the dependency graphs. If relationships are 'mined' from the direct dependency matrix, these relationships should also be found within the long distance matrix.

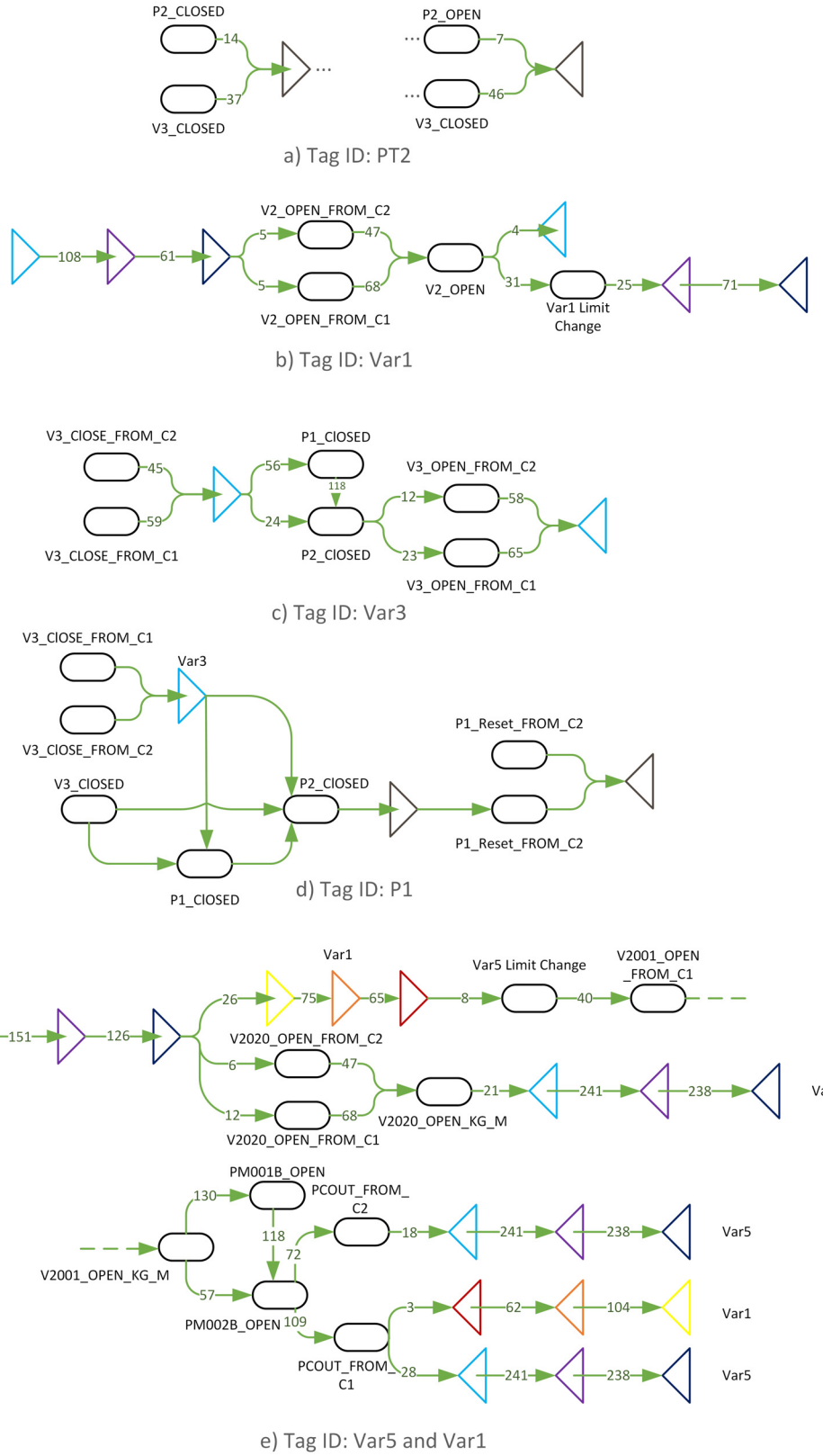

Figure 3.5: New graphical representation of multivariate relationships

Although the XOR/AND dependency for $A \rightarrow C \wedge N$ is close to 0.5, the threshold for AND dependency is >0.5. Therefore all outgoing arrows are treated as XOR. As we will see in the later industrial examples, XOR/AND dependencies are captured in the direct dependency matrix.

By observing Figure 3.5 a) we can see that there are two separate process models. A close inspection tells us that both process models belong to the same tag PT2. Sometimes a complete process model gets separated into many parts due to the weak dependencies between events. The only way to fix partial process models is to lower the dependency thresholds; but doing so can also result in many unwanted relationships which can over complicated the process. It can be concluded that two possible causes for this tag are P2_CLOSED and V3_CLOSED. Operator actions P2_Open is presumably the best action to clear the alarm and V3_CLOSED is unlikely to clear the alarm since this is a direct effect of PT2_alarm. This is a practical solutions since opening the same pump (P2) that causes the alarm will return it back to normal. Figure 3.5 c) contained a slightly more complicated but complete process model. We can see that the alarm Var3_LO is caused by the closing of valve V3 from two separate locations C1 and C2. Two very similar set of actions are used to clear tag Var3:

1. P1_ClOSED → P2_CLOSE → V3_OPEN
2. P2_CLOSE → V3_OPEN

In Figure 3.5 b), a similar pattern (LO → LOLO → LOLOLO) found in univariate analysis is also found here for tag Var1. Although this process model is also slightly incomplete, operator action V2_OPEN followed by Var1_LIMIT_CHANGE clears Var1_LO, Var1_LOLO and Var1_LOLOLO.

In Figure 3.5 d), the alarm for tag P1 is caused by the closing of pump P2. However we can see that there are several events that causes pump P2 to close. This is similar to root cause analysis, where we try to extend the process model as far back possible from the instance the alarm is raised to figure out the source that caused this ripple effect. Therefore closing of valve V3 is ultimately what caused tag P1 to go into the alarm state. Operator actions to clear this alarms are relative simple: by resetting the same tag from either location C1 or C2 will clear tag P1. Again, here we see an incomplete process model since the operator action to reset tag P1 from C2 contains no incoming arcs.

There are two paths that occur after annunciation of Var5_LO → Var5_LOLO → Var5_LOLOLO. The first path is a simple operator action to open V2020 that clears all three alarms associated with tag Var5. The second path shows a multivariate behavior, making connection between tag Var5 and Var1. This type of behavior would have never been captured in univariate analysis. It is also important to note that only tag Var5 in the low alarm state is connected to the high alarms of tag Var1. The same cannot be said for any other combinations. The following are the correct operating procedures with the highest confidence level to clear both sets of alarms: Var5 Limit Change → Open Valve V2001 → Open Pump PM001B → arrow Open Pump PM002B → PCOUT_FROM_C1 or C2.

This chapter adopted some of the dependency metrics from the heuristics miner to capture how operators react to multi-variate alarms. Color map representation of the dependency matrix as well as a new graphical visualization was presented. Algorithms 2 and 3 presented in this chapter can be applied to any event logs under the condition that the heads and tails for all traces can be identified.

Chapter 4

Conclusions and future work

4.1 Conclusions

This thesis established a framework for the process discovery of operator actions in response to both univariate and multivariate alarms. Two algorithms are studied in this thesis and their effectiveness is tested against industrial alarm and event logs. Suggestions to improve the existing process are also mentioned in this thesis. Common imperfection patterns and method to clean the messages into a standardized format are discussed in Chapters 2. Temporal ordering rules and ways to identify the head and tail of a trace are established in both Chapters 2 and 3. This is followed by Trace labeling/Case Identification which is the main goal of Algorithm 1. The results from Algorithm 1 are then the inputs into the process mining software ProM. Two mining algorithms are used to compare the mined process model. The limitations of Chapter 2 are outlined which motivated the work in the following chapter. The concept of trace labeling is not incorporated into Chapter 3, which is a major difference compared to existing research in this field. The dependency metrics are adopted from [20] and [23] to form Algorithms 2 and 3. The dependency matrices are then represented in color maps which are commonly used in correlation analysis [43] and root cause analysis [44]. Threshold parameters are applied to the matrices to build the knowledge base for the process models. Mined results and ways to enhance the process are outlined.

The major outcomes of this thesis are summarized below:

1. Developed methods to pre-process commonly found imperfection patterns in alarm and event logs.

2. Developed an algorithm to assign caseID based to temporal ordering rules and tagID.

3. Discussed visualization of the mined traces using process mining software ProM as well as a new visualization tool specifically designed for A&E logs.

4. Developed an algorithm to mine multivariate relationships between alarm tags as well as operator actions in response to both univariate and multivariate alarms.

5. The color map representation is used to visualize the dependency matrices.

4.2 Future work

The possible future work in the field of process mining of A&E logs can be summarized below:

1. Since locating the heads and tails for the univariate alarms are case specific, finding a general algorithm still remains a challenge. Finding a general solution seems to be a common problem among most process mining research areas.

2. Numerous authors have suggested quality dimension metrics to measure resulting process models [36, 34, 33, 35], however there is still no widely accepted metric. It is also important to extend process discover of operator actions to cover conformance checking in real time and enhancement of the process.

3. Ultimately the goal of this thesis is to used the mined results to assist operators with decision making in real time. Extension of this thesis can include average time duration for operators to clear alarms. Using the average time duration as another threshold, users can specify the maximum time an operator is allowed to clear an alarm.

4. Currently, it is possible to calculate the time duration for a single trace. Finding the average time duration for all traces seems difficult with the current algorithms. Additionally, verifying the mined results with

expert knowledge will be beneficial. Occasionally, actions performed by the operator will not be recorded in the A&E log, therefore having the expert knowledge will greatly improve the completeness of the overall process model.

5. If there exists only one definitive set of actions to clear an alarm then these actions should be automated. This will save time and allow operators to focus on more complex alarm annunciations.

Bibliography

[1] *Engineering Equipment and Materials Users' Association (EEMUA), Alarm Systems: A Guide to Design, Management and Procurement.* EEMUA Publication 191, 2007.

[2] E. Laubwald, "Coupled tanks systems 1."

[3] *International Society of Automation, Management of Alarm Systems for the Process Industries.* ANSI/ISA 18.2, 2009.

[4] "Atd Releases 2016 State of the Industry Report." https://www.td.org/Publications/Blogs/ATD-Blog/2016/12/ATD-Releases-2016-State-of-the-Industry-Report. Accessed: 2017-07-19.

[5] C. Mattiasson, "The alarm system from the operator's perspective," in *Human Interfaces in Control Rooms, Cockpits and Command Centres, 1999. International Conference on*, pp. 217–221, IET, 1999.

[6] "Chernobyl disaster." https://en.wikipedia.org/wiki/Chernobyl_disaster. Accessed: 2017-07-04.

[7] "Deepwater Horizon oil spill." https://en.wikipedia.org/wiki/Deepwater_Horizon_oil_spill. Accessed: 2017-07-04.

[8] "Texas City Refinery explosion." https://en.wikipedia.org/wiki/Texas_City_Refinery_explosion Accessed: 2017-07-04.

[9] S. Breznitz, *Cry wolf: The psychology of false alarms.* Psychology Press, 2013.

[10] R. Parasuraman and D. H. Manzey, "Complacency and bias in human use of automation: An attentional integration," *Human Factors: The*

Journal of the Human Factors and Ergonomics Society, vol. 52, no. 3, pp. 381–410, 2010.

[11] W. M. P. van der Aalst, *Getting the Data*, pp. 95–123. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[12] T.-P. Hong, K.-Y. Lin, and S.-L. Wang, "Fuzzy data mining for interesting generalized association rules," *Fuzzy sets and systems*, vol. 138, no. 2, pp. 255–269, 2003.

[13] W. M. Van Der Aalst, H. A. Reijers, and M. Song, "Discovering social networks from event logs," *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 6, pp. 549–593, 2005.

[14] B. F. Van Dongen and W. M. Van der Aalst, "Multi-phase process mining: Building instance graphs," in *International Conference on Conceptual Modeling*, pp. 362–376, Springer, 2004.

[15] B. F. Van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, and W. M. Van Der Aalst, "The prom framework: A new era in process mining tool support," in *International Conference on Application and Theory of Petri Nets*, pp. 444–454, Springer, 2005.

[16] W. M. Van der Aalst and B. F. van Dongen, "Discovering workflow performance models from timed logs," in *Engineering and Deployment of Cooperative Information Systems*, pp. 45–63, Springer, 2002.

[17] W. M. Van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. Weijters, "Workflow mining: A survey of issues and approaches," *Data & knowledge engineering*, vol. 47, no. 2, pp. 237–267, 2003.

[18] W. Van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[19] J. Becker, M. Rosemann, and C. Von Uthmann, "Guidelines of business process modeling," in *Business Process Management*, pp. 30–49, Springer, 2000.

[20] A. Weijters, W. M. van Der Aalst, and A. A. De Medeiros, "Process mining with the heuristics miner-algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP*, vol. 166, pp. 1–34, 2006.

[21] C. W. Günther and W. M. Van Der Aalst, "Fuzzy mining-adaptive process simplification based on multi-perspective metrics," in *International Conference on Business Process Management*, pp. 328–343, Springer, 2007.

[22] A. K. de MEDEIROS, A. J. Weijters, and W. M. van der Aalst, "Genetic process mining: an experimental evaluation," *Data Mining and Knowledge Discovery*, vol. 14, no. 2, pp. 245–304, 2007.

[23] A. Weijters and J. Ribeiro, "Flexible heuristics miner (fhm)," in *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pp. 310–317, IEEE, 2011.

[24] L. Wen, J. Wang, and J. Sun, "Detecting implicit dependencies between tasks from event logs," in *Asia-Pacific Web Conference*, pp. 591–603, Springer, 2006.

[25] A. Alves de Medeiros, B. Van Dongen, W. Van Der Aalst, and A. Weijters, "Process mining: Extending the α-algorithm to mine short loops," tech. rep., BETA Working Paper Series (WP 113), Eindhoven University of Technology, Eindhoven, 2004.

[26] A. K. A. de Medeiros, W. M. van der Aalst, and A. Weijters, "Workflow mining: Current status and future directions," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 389–406, Springer, 2003.

[27] L. Wen, J. Wang, W. M. van der Aalst, B. Huang, and J. Sun, "A novel approach for process mining based on event types," *Journal of Intelligent Information Systems*, vol. 32, no. 2, pp. 163–190, 2009.

[28] A. K. A. de Medeiros, B. F. van Dongen, W. M. van der Aalst, and A. Weijters, "Process mining for ubiquitous mobile systems: an overview and a concrete algorithm," in *International Workshop on Ubiquitous Mobile Information and Collaboration Systems*, pp. 151–165, Springer, 2004.

[29] L. Wen, W. M. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 145–180, 2007.

[30] W. Hu, A. W. Al-Dabbagh, T. Chen, and S. L. Shah, "Process Discovery of Operator Actions in Response to Univariate Alarms," *IFAC-PapersOnLine*, vol. 49, no. 7, pp. 1026–1031, 2016.

[31] S. Suriadi, R. Andrews, A. H. ter Hofstede, and M. T. Wynn, "Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs," *Information Systems*, vol. 64, pp. 132–150, 2017.

[32] S. Dasani, S. L. Shah, T. Chen, J. Funnell, and R. W. Pollard, "Monitoring safety of process operations using industrial workflows," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 451–456, 2015.

[33] A. Rozinat, A. A. De Medeiros, C. W. Günther, A. Weijters, and W. M. Van der Aalst, "Towards an evaluation framework for process mining algorithms," *BPM Center Report BPM-07-06, BPMcenter. org*, vol. 123, p. 142, 2007.

[34] J. C. Buijs, B. F. Van Dongen, and W. M. van Der Aalst, "On the role of fitness, precision, generalization and simplicity in process discovery," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 305–322, Springer, 2012.

[35] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Information Systems*, vol. 37, no. 7, pp. 654–676, 2012.

[36] K. B. Lassen and W. M. van der Aalst, "Complexity metrics for Workflow nets," *Information and Software Technology*, vol. 51, no. 3, pp. 610–626, 2009.

[37] J. Meyer and Y. Bitan, "Why better operators receive worse warnings," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 44, no. 3, pp. 343–353, 2002.

[38] A. Adhitya, S. F. Cheng, Z. Lee, and R. Srinivasan, "Quantifying the effectiveness of an alarm management system through human factors studies," *Computers & Chemical Engineering*, vol. 67, pp. 1–12, 2014.

[39] "IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability inEvent Logs and Event Streams," *IEEE Std 1849-2016*, pp. 1–50, Nov 2016.

[40] M. Walicki and D. R. Ferreira, "Sequence partitioning for process mining with unlabeled event logs," *Data & Knowledge Engineering*, vol. 70, no. 10, pp. 821–841, 2011.

[41] S. R. Kondaveeti, I. Izadi, S. L. Shah, and T. Black, "Graphical representation of industrial alarm data," *IFAC Proceedings Volumes*, vol. 43, no. 13, pp. 181–186, 2010.

[42] M.-J. Lesot, M. Rifqi, and H. Benhadda, "Similarity measures for binary and numerical data: a survey," *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 1, no. 1, pp. 63–84, 2008.

[43] F. Yang, S. L. Shah, D. Xiao, and T. Chen, "Improved correlation analysis and visualization of industrial alarm data," *ISA transactions*, vol. 51, no. 4, pp. 499–506, 2012.

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