DOKTORI (PhD) ÉRTEKEZÉS

Bántay László

Pannon Egyetem
2025

PANNON EGYETEM

Doktori (PhD) Értekezés

DOI:10.18136/PE.2025.952

Folyamatirányítási rendszerek gyakori mintázat keresés és folyamatbányászat alapú elemzése

Szerző:
Bántay László

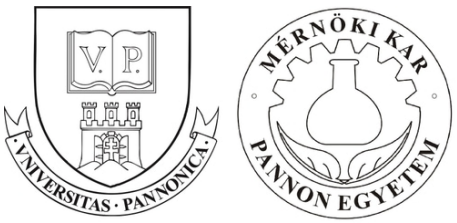
Konzulens:
Prof. Dr. habil. Abonyi János

Értekezés doktori (PhD) fokozat elnyerése érdekében

a **Pannon Egyetem**

Vegyészmérnöki- és Anyagtudományok Doktori Iskolájához tartozóan

Folyamatmérnöki Intézeti Tanszék



2025

UNIVERSITY OF PANNONIA

Doctoral (PhD) Thesis

Frequent pattern and process mining-based process operation analysis

Author:
László Bántay

Supervisor:
Prof. Dr. habil. János Abonyi

A thesis submitted in fulfillment of the requirements

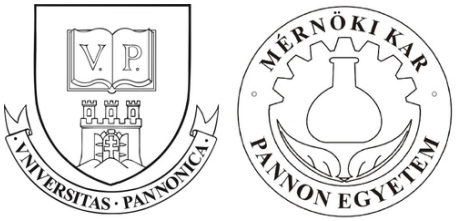
for the degree of Doctor of Philosophy

in the

Doctoral School in Chemical Engineering and Material Sciences

of **University of Pannonia**

Department of Process Engineering



2025

Folyamatirányítási rendszerek gyakori mintázat keresés és folyamatbányászat alapú elemzése

Az értekezés doktori (PhD) fokozat elnyerése érdekében készült a Pannon Egyetem Vegyészmérnöki- és Anyagtudományok Doktori Iskolája keretében

Bio-, környezet- és vegyészmérnöki tudományok tudományágban

Írta: Bántay László

Témavezető: Dr. Abonyi János

Elfogadásra javaslom (igen / nem)

…….

(témavezető)

Az értekezést bírálóként elfogadásra javaslom:

Bíráló neve: Dr. Fogarassyné dr. Vathy Ágnes igen /nem

…….

(bíráló)

Bíráló neve: Gyenesei Attila igen /nem

…….

(bíráló)

A jelölt az értekezés nyilvános vitáján ….....%-ot ért el.

Veszprém,

……...……

(a Bíráló Bizottság elnöke)

A doktori (PhD) oklevél minősítése….....

Veszprém,

……...……...….

(az EDHT elnöke)

Kivonat

Folyamatirányítási rendszerek gyakori mintázat keresés és folyamatbányászat alapú elemzése

Bántay László

Az Ipar 4.0 szemlélet megjelenésével a folyamatok mélyebb megértése iránti igény hatására megnőtt az eltárolt folyamat adatok jelentősége. Miután a folytonos adatok kezelésével kapcsolatban gazdag irodalom áll rendelkezésre, a diszkrét adatok elemzése is növekvő figyelmet kap. Munkám célja a meglévő technikák áttekintése mellet új, ipari napló fájlok feldolgozását támogató módszerek kifejlesztése.

Az ipari napló fájlok ritkán vannak azonnal feldolgozható formátumban; a tartalmukat és az adatok címkézését standardizált formába kell hozni. Probléma lehet még az adott folyamatokhoz tartozó események csoportosításának, a trace-eknek a hiánya is. Ez a probléma nem a rögzített üzleti folyamatok adatainál áll elő, hanem jellemzően komplex ipari létesítményeknél, ahol a diszkrét események csak egymás után kerülnek rögzítésre. A folyamatbányászat elveit és standardjait használva ezek a kihívások leküzdhetőek. Egy megfelelő formátumú napló fájl lehetővé teszi a tárolt folyamatok modellezését. Ahogy a trace-ek létrehozásra kerültek, lehetőség nyílik egy valószínűségi előrejelző modell megalkotására a leggyakoribb eseményláncok meghatározásával. Szintén megoldandó feladat az egyazon napló fájlban tárolt párhuzamos folyamatok kérdésköre. A gyakori szekvenciák feltárásával lehetőség nyílik az összetartozó események azonosítására, ami alapján partícionálhatjuk a napló fájlt. A megfelelő esemény mintázat kiválasztásához a gyakori szekvenciák utófeldolgozása szükséges. A szekvenciák tulajdonságainak felhasználásával (támogatottság, konfidencia, szekvencia hossz) létrehozható egy hálózat jellegű ábrázolás, ami az eseményláncok gyors megértését teszi lehetővé. Mindezen technikák alkalmazása biztosítja a folyamatok olyan mélységű megértését, ami támogathatja az ipari létesítmények ember-gép kezelőfelületeinek fejlesztését.

Az eredmények igazolták a feltevést, miszerint a kidolgozott eszköztár alkalmas ipari diszkrét adatok feldolgozására és elemzésére, valamint támogatja a folyamatirányítási feladatok optimalizálását.

Abstract

Frequent pattern and process mining-based process operation analysis

László Bántay

The appearance of the Industry 4.0 approach has raised the importance of stored process data with the increasing need for deep analysis of processes. With a reach literature in the field of continuous data handling, the topic of discrete process event analysis and modelling is gaining more and more attention. The goal of my work was to collect the available techniques and develop new supporting methods to process industrial log files.

Industrial log files rarely are in a processable form; their content and data labelling must be done according to a standard format. Another issue with this kind of log files that needs to be solved is the lack of traces, which are grouped events related to a certain process. This is not a problem with respect to business activities, but in complex production sites the discrete events are just logged after each other. Applying the principles and standards of process mining, these shortcomings can be dealt with. A proper log file format allows for the modelling of the stored processes. As traces are generated, the definition of the most frequent sequences of events may support a probabilistic discrete event scenario prediction system. Another challenge to be answered is that parallel running processes are stored in the same log file. Frequent sequential patterns can be used to identify related events and partition the log file according to them. To select the correct pattern, post-processing of the sequential patterns is required. Using the attributes of the sequences (support, confidence, sequence length) a network-like visualisation can be drawn that enables quick understanding of the different event propagation starting from the same trigger event. Using these techniques proposes a deep knowledge extraction of the processes that may support to optimise the human-machine interfaces of a production site.

The results proved that the developed toolkit is suitable for interpreting and analysing industrial discrete event data and supports the optimisation of process operations in the industry.

Abstracto

Análisis frecuente de operaciones de procesos basado en minería de patrones y procesos

László Bántay

La aparición del enfoque de la Industria 4.0 ha aumentado la importancia de los datos de procesos almacenados con la creciente necesidad de un análisis profundo de los procesos. Con una amplia literatura en el campo del manejo continuo de datos, el tema del análisis y modelado de eventos de procesos discretos está ganando cada vez más atención. El objetivo de mi trabajo fue recopilar las técnicas disponibles y desarrollar nuevos métodos de soporte para procesar archivos de registro industriales.

Los archivos de registro industriales rara vez están en un formato procesable; su contenido y etiquetado de datos debe realizarse de acuerdo con un formato estándar. Otro problema con este tipo de archivos de registro que necesita ser resuelto es la falta de rastros, que son eventos agrupados relacionados con un determinado proceso. Esto no es un problema con respecto a las actividades comerciales, pero en los sitios de producción complejos, los eventos discretos simplemente se registran uno tras otro. Aplicando los principios y estándares de la minería de procesos, estas deficiencias se pueden abordar. Un formato de archivo de registro adecuado permite el modelado de los procesos almacenados. A medida que se generan los rastros, la definición de las secuencias de eventos más frecuentes puede respaldar un sistema de predicción de escenarios de eventos discretos probabilísticos. Otro desafío que se debe resolver es que los procesos que se ejecutan en paralelo se almacenan en el mismo archivo de registro. Se pueden utilizar patrones secuenciales frecuentes para identificar eventos relacionados y dividir el archivo de registro en función de ellos. Para seleccionar el patrón correcto, se requiere un posprocesamiento de los patrones secuenciales. Utilizando los atributos de las secuencias (soporte, confianza, longitud de la secuencia) se puede dibujar una visualización similar a una red que permite una rápida comprensión de la propagación de diferentes eventos a partir del mismo evento desencadenante. El uso de estas técnicas propone una extracción de conocimiento profundo de los procesos que puede ayudar a optimizar las interfaces hombre-máquina de un sitio de producción.

Los resultados demostraron que el conjunto de herramientas desarrollado es adecuado para interpretar y analizar datos de eventos discretos industriales y respalda la optimización de las operaciones de proceso en la industria.

Köszönetnyilvánítás

Köszönöm a családomnak, a kollégáimnak és a barátoknak akik segítettek megvalósítani az elképzelhetetlent.

Juditnak, Panninak, Sárinak

Contents

Chapter 1

Introduction

The analysis and modelling of continuous process signals have a wide literature and a ready-to-use toolkit. However, industrial systems contain many discrete events that are critical to understanding the characteristics of processes. Some modelling and analysis solutions already exist, but they need to be extended or redesigned to get a deep and useful knowledge of industrial production systems. The motivation of the present work is to develop a methodology for the process mining-based analysis of industrial discrete event log databases to increase process safety and reduce operator workload. As a result, we will be able to understand the chain of events that trigger an operator action, as well to explore the effects of different operator action strategies; or from other point of view, to gain the models of processes leading potentially to malfunctions or safety incidents.

Process mining is a widely used technique to understand processes by exploring the regularity of events stored in log files [1]. In process mining, the grouped events in the cases are called traces. Traces can be considered as event sequences that enables the use of pattern mining techniques. Frequent sequence pattern mining is widely used to extract knowledge from event log files, e.g., it has been successfully applied for workload prediction [2], and prediction of transition probability [3]. The combined application of these techniques can help to solve the shortcomings of existing discrete industrial event analysis methods. The resultant frequent event patterns can be used for several purposes. With the help of sequence compression and alignment, an early warning system can be developed. The visualisation of sequences supports the targeted discovery of parallel subprocesses stored in the same log file. The combination of the discussed techniques can be the basis for a machine learning-supported method to optimise human-machine interfaces.

The roadmap of this work is the following. First, the problem description is presented to emphasise the objectives and goals of the research. Second, the theoretical background of the methods developed is discussed. In chapter 3 the proper log file design and trace generation method is defined. In chapter 4, a sequence compression and alignment event prediction method is presented. In chapter 5, a frequent pattern-based solution is provided to handle the parallel process storage issue in industrial log files. Chapter 6 contains the network-based visualisation methods of frequent sequences that supports the selection step of the log file partitioning technique discussed in chapter 5. Chapter 7 proposes a machine learning-based human-machine interface optimisation method. In the end, the results and findings are summarised.

1.1 The importance of data-driven alarm management

Alarm management is essential to any industrial process control system. Predictive and adaptive solutions can help to increase process safety. State-of-the-art industrial production systems contain complex process control solutions. The amount of recorded signals and process variables makes it difficult to have a clear view of the relationships between the different process elements, the control of the processes can be a demanding task for the workers. To lower the workload of the operators, a good understanding of the process element relationships is needed to predict the probable event scenarios, that can be the basis of a decision-supporting system.

The work of the operators can be reduced and supported in other ways as well. Generally speaking, predictable alarms do not contain useful information for the operators. Therefore, automation solutions should handle them before they occur, or completely useless alarms should be suppressed before an announcement is made [4]. Future alarm sequences can also be predicted using historical knowledge of the process [5]. The exploration of operational strategies holds promising opportunities for automation as well: the sequences of alarms and the corresponding operator actions as well as the best operational practices can be determined [6]. The analysis of the announced alarms can also facilitate root cause analysis [7]. A wide range of data-based solutions have been applied to reduce the operator workload, e.g. the conventional techniques of deadbands [8], delay timers [9] or filtering [10]. Advanced alarm management solutions aim to define more informative features for the operators by identifying redundant and co-occurring alarms. Two main approaches are common place. Firstly, correlation analysis-based techniques are widespread, where the aim is to find frequently co-occurring alarms over a short period of time, which can be considered redundant [11]. Secondly, the frequently occurring longer operational patterns can be considered to be the symptoms of the same malfunction and can be revealed by frequent pattern mining algorithms [12], as well as applied in terms of alarm prediction and suppression. It is also advantageous to apply highly efficient data-driven solutions, like deep learning [13] or decision tree-based classifiers [14].

To gain understandable models, that can be directly used in alarm management is challenging. To explore complex event chains or comparable models of operator action strategies, determining the correlation of event pairs, deep learning methods with hard to understand results are not satisfying. For compact and comprehensible models, we need to apply process mining. The log file design and trace generation rule suggestions will be discussed in Chapter 3, that is the basis of Thesis I.

1.2 Alarm scenario prediction

The effectiveness of industrial process monitoring depends heavily on alarm systems and the operators handling the alarms. As new sensor and network technologies appear, it has become much easier to configure alarms in industrial facilities. However, if alarm configurations are not rationally designed, the problem of excessive alarm messages would reduce the efficiency or even the safety of plant operations due to distracting information provided to operators. An alarm flood is an extreme case of this problem, during which the efficiency of the operator in handling important alarms is reduced significantly because of the overwhelming workload created by the numerous alarm messages. In chemical production plants, discrete events, e.g., warnings, alarms, operator actions, and system messages, are logged in alarm management databases. By analyzing the recorded discrete events, data model-based solutions can be constructed:

- to determine the root cause of the event series [13] ,
- to predict the future events that might occur [15] ,
- to determine the frequently occurring operation strategies for advanced automation solutions [16] ,
- and in general to help and reduce the workload of the operators.

There are two common strategies for the reduction of alarm floods. The first is to remove univariate alarms by applying techniques such as delay timers [17] , alarm limit dead-bands [18] and filtering [19] . The second strategy to reduce alarm floods is to study consequential alarms (alarms that correlate with each other) because they form another important part of an alarm flood. This type of consequential alarms are triggered by one or more certain faults and appears frequently in the alarm and event logs. Consequential alarms, as they are coupled, tend to form unique patterns in the sequence the alarms appear in, which opens up the possibility to apply pattern mining algorithms to identify their occurrences. Such sequences may be further investigated by the means of causality analysis, either based on process data [20] or alarm data (event logs) [21] to identify their root cause.

Identifying a set of the relevant sequences that describe best a sequence database is of huge value for the analysis and prediction of operational states of plants, eventually resulting in an increased economical and industrial safety performance. In the present study, the focus was on reducing, analyzing, and predicting the progression of alarm sequences in real time by utilizing the benefits of sequence compression and sequence alignment.

In Chapter 4, which is also part of Thesis I, the effectiveness of the proposed pattern-based event scenario prediction methodology is presented in terms of the analysis of the alarm and event-log database of an industrial delayed-coker plant.

1.3 Discovering parallel processes

One of the limitations of existing process mining techniques is that they cannot explore parallel and overlapping processes [22]. Moreover, it might be unclear which trace belongs to which process (Figure 1.1). Assuming that all events are related may be false in many cases, especially in alarm management. Once the log file is prepared, such connections and rules are added (in the form of traces), which may hinder the discovery of a satisfactory sub-process model. One solution to this issue is to identify and group closely related events.

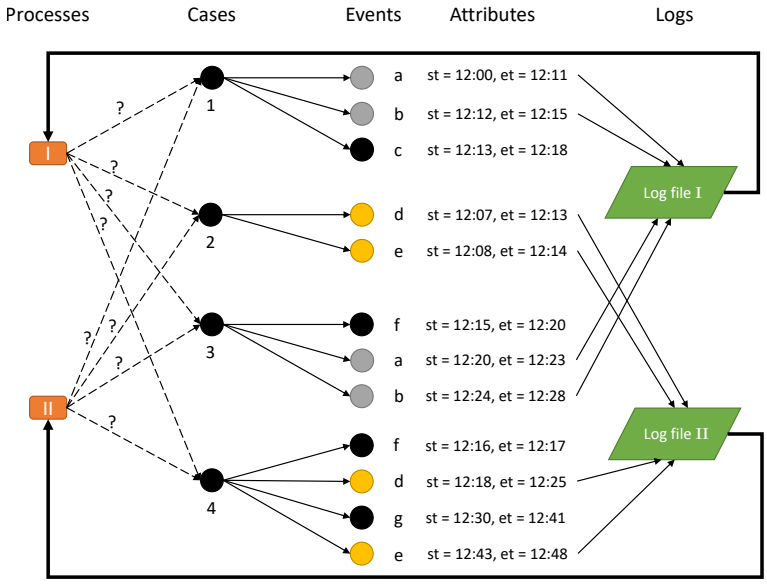


Figure 1.1: The problem of parallel as well as overlapping processes and the proposed method. *st* denotes the start time, while *et* stands for the end time of the event. A goal-oriented solution is to group the events according to their common occurrences.

Machine learning-based support of industrial operations is a developing area. Previous studies have already investigated the discovery of temporal patterns in process alarm sequences [23]. The extracted extracted knowledge can be used to assist operators [24], and for suppression of alarm floods [25]. This work aims to separate potentially important sub-processes automatically by segmenting the log file to obtain a goal-oriented process model. The segmentation is based on the co-occurrence frequency of relevant process events.

The frequent sequential pattern-based log file partitionong method is discussed in Chapter 5, that is the basis of Thesis II.

1.4 Visual analysation of event sequences

Frequent sequence pattern mining algorithms do not provide sufficient information on the relationship and context of the large number of sequences extracted, making the interpretation and utilisation of the results difficult. In the case of association rules, there are three main types of post-processing methods for the results to solve this kind of problem, pruning, grouping, and visualisation [26]. These post-processing methods can be applied to sequences as well.

Visualisation can be a powerful tool, especially when developing machine learning models [27], for example, pruning algorithms [28]. A network-like representation allows one to see and focus on relevant information without reading the data in detail [29]. A well-designed visualisation method can also allow the grouping of data by a relevant attribute. For example, sequential patterns contain the order of events, along with their frequency, which supports the development of event-propagation models. A wide range of sequential pattern visualisation techniques have been developed in the past and were summarised and compared in a study [30]. These techniques were individual representations [31], flow diagrams [32], aggregated pattern visualisations [33], placement strategies [34], and episode visualisations [35]. The comparison showed that a sequence pattern visualisation technique provides valuable information (support, confidence, sequence relations, etc.) or is easy to read and understand. With an increasing number of patterns, readability quickly worsens, affecting lucidity. If the goal is to analyse a system with many events containing many relevant sequences, this problem must be solved.

Based on the available metrics of the visualisation methods [36], we have chosen three key attributes that can serve as a metric to measure the conformity of the applicability. These are *Information content*, *Readability*, and *Flexibility*.

The motivation of the work is to suggest a solution that provides a good balance with respect to all attributes. It is also worth noting that most solutions are not open source and custom modifications are not easy. Although there are solutions that successfully overcome these shortcomings [37], we believe that our proposed method is a useful addition to these techniques. From a clarity point of view, finding the balance between information content and readability, and from an applicability point of view, allowing good flexibility in tailoring goal-oriented analysis tasks.

The use of network theory in the case of frequent sequence pattern postprocessing supports the identification of relevant event types, such as unifying and polarising events [38]. Existing visualisation techniques focus on the interpretation of well-defined processes, for example, in [39] and in [40]. The main difference between our approach and existing ones is that the proposed method does not aim to visualise one process model but rather to identify the different and common elements of parallel process models. In industrial log files, where technology is complex, stored events do not unequivocally describe existing subprocesses. One solution is to create sublog files based on frequently co-occurring events [41]. Our technique supports the selection of events that describe a particular process, enabling the definition of targeted process models.

The network-like visualisation method of frequent sequential patterns is discussed in Chapter 6, that is the basis of Thesis III.

1.5 Machine learning-supported Human Machine Interface optimisation

User experience (UX) is identified as an essential topic for design and functionality in the development of user interfaces (UI) [42]. Today, machine learning-based solutions are widely used in all types of production development work [43], including the topic of human-machine interactions. The potential of machine learning techniques was used in the areas of air traffic management [44], collaborative robot control [45], and industrial PLC programming [46]. Industrial Human Machine Interfaces (HMIs) are designed primarily based on Piping & Instrumentation Diagram (P&ID) or the physical layout of the plant, rather than from a process perspective. As an effect of this design principle, operator actions must be performed through several displays, causing a time loss during the intervention. Time loss can cause decreased production volume, poor product quality, or even hazardous personal and environmental situations. Therefore, it is necessary to reduce the workload of plant personnel to ensure more efficient and safer process management, which is consistent with the Industry 5.0 paradigm [47], focussing on human-centric solutions [48]. The industry is aware of the shortcomings of P&ID-based HMI design, and the solution is to develop high performance HMIs [49]. However, "high performance" was limited on the information visualisation part of the problem, not touching on the content of the display of hierarchy levels 2 and 3. Drawing level 1 displays based on P&IDs is logical, but levels 2 and 3 need a more process control-related perspective.

Our suggestion is to create the HMI hierarchy and displays based on typical process scenarios and operator interactions, which helps to apply the "Navigation and Layout" perspective of the ISA 101 standard [50]. This kind of information can be extracted from the system log files, but not directly. As an operator can perform several corrective actions belonging to different subprocesses in the same time frame, a segmentation of the log file is needed to accurately identify the subprocesses [41]. The operator action log must be merged with the plant log, pairing the alarm management data with all the actions taken on the HMI by the workers. This way, the number of steps taken to react to an alarm event can be analysed, indicating the needed amendments on the HMI (shortcuts, pop-up notifications, structure hierarchy). Alarm floods are common in DCS systems. In most cases, few alarms indicate the source of the problem, but they can initiate a massive amount of 'consequence' alarms. If too many alarms are generated (which can be expected), the source alarms may be thrown out of the display, and the operator may not be able to identify them. A more transparent alarm overview display can be developed by forming alarm groups and labelling source and consequence alarms. Based on frequent alarm co-occurrences, alarms can be labelled by their priority.

The machine learning-based HMI optimisation method is proposed in Chapter 7, that is the basis of Thesis IV.

Chapter 2

Theoretical Background

This chapter discusses all relevant theoretical considerations. First, the fundamentals of frequent pattern mining are presented. Second, process mining is introduced along with the topic of trace generation. Finally, the connection between frequent pattern and process mining is discussed.

2.1 Frequent sequential pattern mining

This section discusses the necessary concepts and considerations about frequent patterns and their mining. To help the reader read more easily, the used notations are collected in Table 2.1.

Frequent Pattern Mining	
$\mathcal{I}$	The set of items
$\mathcal{I}_i$	The i-th item
$\mathcal{T}$	The set of transaction
$\mathcal{T}_i$	The i-th transaction
$\mathcal{D}$	$\mathcal{D}$ Database
a, b	Items
P	Probability
$s_{\mathcal{T}}$	Support of an association rule
$c_{\mathcal{T}}$	Confidence of an association rule
Φ	Sequence
$sup(\Phi)$	Support of a sequence
$conf(\Phi)$	Confidence of a sequence
Process Mining	
E	The set of events
$S(\phi)$	The set of transactions containing sequence ϕ
$e_{i,j}$	The j-th event of the i-th trace
T	The set of traces
T_i	The i-th trace
L	Log
a, b	events

Table 2.1: List of used notations

A sequence is an ordered list of itemsets [51], in this work, the items are e events. Let us denote the set of all events with I, $I = \{e_1, ..., e_k, ..., e_p\}$. The set of $T = \{T_1, ..., T_i, ..., T_n\}$ transactions contain events, T_i is the i-th transaction, $T_i = \{e_1, ..., e_p\}$ and $T_i \subseteq I$. A sequence $\phi_i = \langle e_1, ..., e_k \rangle$ is a sequence in T_i, $\phi_i \subseteq T_i$. The set of all sequences is $\phi = \langle \phi_1, ..., \phi_i, ..., \phi_l \rangle$. There are two important attributes of sequences, support and confidence. The support of a sequence is the number of times it occurs or the number of transactions containing it. Let S^{ϕ_i} be the set of transactions that support ϕ_i, $\phi_i \subseteq S^{\phi_i}$. The cardinality of S^{ϕ_i} is the support of ϕ_i:

$$\mid S^{\phi_i} \mid = Sup(\phi_i). \tag{2.1}$$

One sequence can occur in many transactions, but one transaction can occur only once. To consider a sequence as frequent, we must define a threshold $minSup$:

$$\{\phi_i \in \phi_{freq} : Sup(\phi_i) > minSup\}, \tag{2.2}$$

where ϕ_{freq} is the set of all frequent sequences and $\phi_{freq} \subseteq \phi$.

The confidence ($Conf$) is the conditional probability of the occurrence of a sequence of k length, assuming that its predecessor of $k-1$ length (or parent sequence) has already occurred. If $\phi_i = \{e_b, e_k, e_m\}$ is the one event extension of sequence $\phi_h = \{e_b, e_k\}$, $\phi_i = (\phi_h \rightarrow e_m)$, than the probability of the occurrence of ϕ_i under the condition, that ϕ_h already occurred is the confidence value of ϕ_i and can be calculated as:

$$Conf(\phi_i|\phi_h) = \begin{cases} \frac{Sup(\phi_i)}{Sup(\phi_h)} \times Conf(\phi_h) & i > 0, \\ 1 & i = 0. \end{cases} \tag{2.3}$$

The higher the confidence is, the bigger the possibility is that the given sequence will be followed by the item of interest. The confidence value is an essential part of the suggested event prediction method.

The $D^T = \{T_1, \ldots, T_i, \ldots, T_n\}$ log file is the set of T traces. A T_i trace contains $e \in E$ events with $t(e)$ timestamps, and $t(e_b) < t(e_d) < ... < t(e_m)$:

$$T_i = \{(e_b, t(e_b)), (e_d, t(e_d)), (e_f, t(e_f)), ..., (e_k, t(e_k)), (e_m, t(e_m))\}. \tag{2.4}$$

This model considers events as point-like, where the timestamps mark their occurrence time. In the case of temporal events, there can be overlaps that require special solutions [52]. Assuming this temporal follow-up relationship structure combined with even the finest temporal resolution, there is still a possibility that events occur exactly at the same time. In this case, the order of the events will be decided based on a predefined rule so that the structure of the database is kept consequent and well-defined. If no rule can be defined, the order of these kinds of events will be set alphabetically.

There are also two essential definitions in the case of itemsets and sequences, which enable extra filtering layers to be applied in the database.

- **Closed and maximal frequent itemsets** A closed frequent itemset has no superset with the same support, and a maximal frequent itemset has no superset that is frequent [53]. By identifying these types of frequent itemsets, an excellent base is gained from which to choose the attributes of a log file for filtering or partitioning.

- **Closed and maximal frequent sequential patterns** Although their definitions are the same as in the case of *frequent itemsets*, they have an extra dimension of information, namely the sequence rule of the events, which allows the filtering to be more strict. Keeping items or sequences only above a specific confidence value is up to consideration.

The selection of the pattern-mining tool can be affected by the adjustable parameters offered by the algorithm, besides the minimum support value that must be set. It is suggested to use three additional parameters, namely the gap (sequences), minimum sequence length (sequences), and maximum sequence or pattern length (itemsets). For the initial setup, the recommended values are 1 for the gap (default), 2 for the minimum sequence length, and 5 for the maximum sequence/pattern length. The fine-tuning of these parameters depends on the resultant set of frequent sequential patterns. The mentioned parameters and their effect on the result of the mining are as follows:

- **Support** *Support* yields the occurrence probability of the pattern. By adjusting the minimum support value, we can define the frequency level from where a chain of events can be considered relevant.

- **Confidence** *Confidence* is the occurrence probability of an itemset or sequence if another itemset or sequence has already occurred before. This parameter provides a deeper insight into the process as it is the conditional probability of an event when another event or sequence has already occurred. It helps explore processes where an event has a clear effect, e.g., an alarm signal results in a well-defined series of operator actions.

- **Sequence length** The minimum and maximum sequence lengths define the minimum and the maximum number of items in the sequences. This parameter can adjust the depth of the sought process model.

- **Gap** The maximum gap is the maximum allowed number of events occurring between two consecutive events in the pattern. More precisely, if the value of the maximum gap is N, then N-1 events are allowed. This parameter describes the long-term relationship between events, i.e., events that are connected but don't follow each other directly. Its value affects the width of the process model, with a bigger gap, new branches are added.

With these parameters, targeted pattern mining can be performed, but it is hard to apply them in automated systems. In the case of continuous data streaming, the input parameters should be changed continuously, this kind of application calls for an aimed tool, regarding parameter adjustment, data size, and processing speed.

2.2 Process mining

Process mining is a collection of techniques which support the understanding of processes based on event logs [54]. There are two basic requirements of process mining: a properly designed *log file* that contains events grouped in well-defined *traces*. Industrial discrete events, e.g. alarms and warnings, are recorded in the process control unit of almost every production site when a variable exceeds its associated threshold. An industrial alarm & event-log database is usually composed of these alarms and warnings, operator actions, system messages as well as any further timestamped information logged by the control system. For the purpose of further mathematical formulation, every event can be regarded as a state of the technology (denoted by s) represented by $< pv,\ a >$ data pairs. pv indicates the name of the process variable, while a represents the attribute that indicates the state occurring on the given variable, e.g. high or low alarm in the case of an alarm announcement or increase or decrease (open or close) in the case of an operator action. An *event* is the occurrence of a given state, which can be an event of temporal duration such as alarm messages or point-like in time such as operator actions. For example, the description of an alarm and an operator action can be mathematically represented as $<$ *Column Inlet Temperature, High Alarm* $>$ and $<$ *Cooling Water Inlet Valve, Open* $>$, respectively. The events are represented by $< s,$ *st, et* $>$ triplets, where s denotes the state that occurs between *st* starting and *et* end times. For an event with a point-like temporal characteristic, even though the *st* and *et* times are regarded as equal to maintain a uniform mathematical formulation, it must be decided whether to keep both or utilise only the *st* starting time for further processing.

Log files

The files of industrial alarm & event-logs are usually composed of timestamped events of alarms, operator actions, system messages and any further temporal information. Additional information can help us to determine which sensor generates the alarm in the process and the priority of that alarm, e.g. the location of the event, that is, in which part of the plant/organization it occurred. Each event that occurs is labelled with a tag name. Every alarm and operator action can be considered to be the states of the process. The different alarms in the event log have starting and end times. The starting time is when the alarm in the process was raised, and the end time is when the process returned to its proper operational zone. The operator actions are usually considered to be point-like events, i.e. their starting and end times coincide. A series of events can be defined as a trace. Let L denote a set of events; $\sigma \in L$ stands for an event trace, that is, a sequence of events; and $T \subseteq L$ represents an event log, i.e. a set of event traces. Besides information on the occurrence of an event, the log files usually contain other information as previously discussed, e.g. the location of the sensor that raised the alarm in the process, which is possibly categorized into units or production units, etc. This work presents the importance of task-specific trace definitions. The methodology of trace segmentation will be discussed later.

Trace definition

The input of process mining tools is a log file in a standard format, in this work we have chosen the XES standard. XES is an XML-based standard for event logs. Its purpose is to provide a generally acknowledged format for exchanging event log data between tools, applications and domains. The main reason for choosing the XES model is the support it provides for traces. As mentioned above, it is very important that traces are well defined. Usually, in industrial log files the events are sequenced independently of one another, unlike in the XES Standard. The majority of process mining algorithms take into consideration traces in addition to events.

For the effective mining of the alarm & event-log files, a definition of the time window applied for the segmentation of the alarm and event log files into traces that is dependent on the purpose is required.

As previously mentioned, since traces play a significant role in process mining methods, the task-dependent determination of trace-defining time windows is required. As the main event of the alarm log file is the appearance of the alarm event itself (that is, its *starting time*), the basis for the trace generation are the following:

Let α and β denote two consecutive events in log L. Let $T(\alpha)$ and $T(\beta)$ stand for the times of occurrence of events α and β, respectively, and σ represents the spillover constant. If $T(\beta) - T(\alpha) < \sigma$, then α and β are located in the same trace. However if $T(\beta) - T(\alpha) \geq \sigma$, then β is placed in the following trace (Figure 2.1. The spillover constant can be tuned based on the knowledge about the dynamic of the system. Obviously, it can be calculated with the help of the experience of the operators and identification of data driven dynamical models.

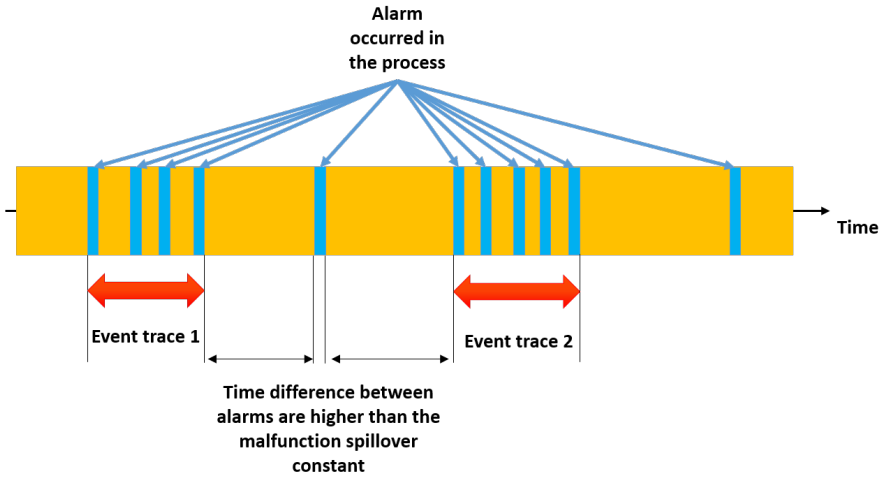


Figure 2.1: The segmentation of an event log database into event traces of potential propagation of error.

2.3 The similarity and connection between frequent pattern mining and process mining

The basis of Process Mining are *Event*s, grouped into *Trace*s, that form a *Log*. If these definitions are compared to the ones related to Frequent Itemset Mining, they are quite similar to each other. *Event*s correspond to *Item*s, *Trace*s to *Transaction*s and a *Log* to a *Database*. This similarity enables the synergy of Process Mining and Frequent Itemset/Sequence Mining (Figure 2.2).

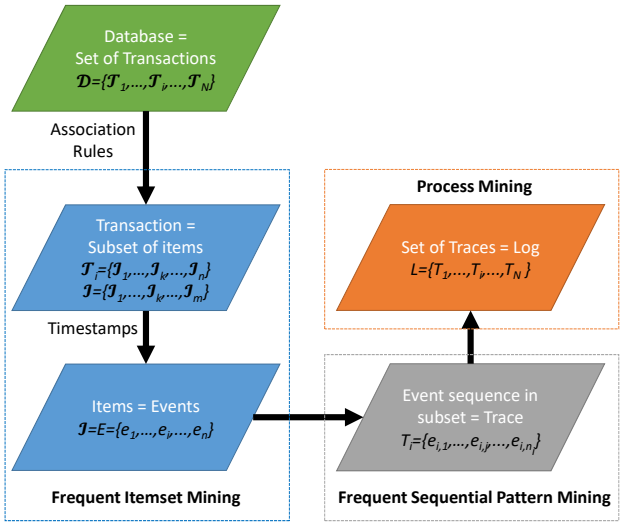


Figure 2.2: The connection between Frequent Itemset Mining, Frequent Sequential Pattern Mining and Process Mining

Let $\mathcal{D}$ denote a *Database* that contains N *Transaction*s, a *Transaction* $\mathcal{T}$ refers to a set of n *Item*s, $\mathcal{I}_k$ represents an *Item* and $\mathcal{I}$ stands for the set of all *Item*s.

$$\begin{aligned} \mathcal{D} &= \{\mathcal{T}_1, ..., \mathcal{T}_i, ..., \mathcal{T}_N\} \\ \mathcal{T}_i &= \{\mathcal{I}_1, , ..., \mathcal{I}_k, ..., \mathcal{I}_n\} \\ \mathcal{I} &= \{\mathcal{I}_1, ..., \mathcal{I}_k, ..., \mathcal{I}_m\} \end{aligned} \tag{2.5}$$

The connection between the *Item*s can be described by *Association Rules*. An *Association Rule* $a \Rightarrow b$ is where $a \in \mathcal{I}$ denotes the antecedent, $b \in I$ represents the consequent and $a \neq b$. Again, let $\mid \mathcal{T} \mid$ be the number of all *Transaction*s and $\mid a \cup b \mid$ the number of *Transaction*s that contains a and b. The *Support* of rule $a \Rightarrow b$ ($s_\mathcal{T}$) can also be considered, as the probability that a *Transaction* contains a and b.

$$s_\mathcal{T}(a \Rightarrow b) = \frac{\mid a \cup b \mid}{\mid \mathcal{T} \mid} = P(a \cup b) \tag{2.6}$$

The *Confidence* of an *Association Rule* is the probability of finding b under the condition, that the *Transaction* contains a.

$$c_{\mathcal{T}}(a \Rightarrow b) = P(b \mid a) = \frac{P(a \cup b)}{P(a)} = \frac{s_{\mathcal{T}}(a \Rightarrow b)}{s_{\mathcal{T}}(a)} \tag{2.7}$$

The next step is to add the order of *Item*s as an attribute. Having *Process mining* in mind, the obvious way is to give a timestamp to *Item*s, transforming them into $E = \{e_1, ..., e_i, ..., e_n\}$ *Event*s. By using the timestamps to determine the order of *Event*s, *Sequence*s are obtained.

As *Transaction*s contain *Sequence*s, they can be reffered to as *Trace*s, which is where the field of *Process Mining* (PM) is entered. The set of *Trace*s (T) form *Log* (L), where $e_{i,j}$ denote the j-th *Event* of the i-th *Trace* containing n_i *Event*s:

$$\begin{aligned} L &= \{T_1, ..., T_i, ..., T_N\} \\ T_i &= \{e_{i,1}, ..., e_{i,j}, ..., e_{i,n_i}\}, e_i \in E \end{aligned} \tag{2.8}$$

Similarly to Frequent Itemset Mining (FIM), the relationships between *Event*s have to be identified. These relationships are referred to as the *Dependency Relations*. Let L be a *Log* and $a, b \in E$ denote *Event*s. The possible *Dependency Relations* are the following:

1. a **is directly succeeded by** b ($a >_L b$): if a trace $T_i = \{e_{i,1}, ..., e_{i,j}, ..., e_{i,n_i}\}$ is present where $i \in \{1, ..., N\}$ and $j \in \{1, ..., n_i\}$, such that $T_i \in L$, $e_{i,j-1} = a$ and $e_{i,j} = b$,

2. a **is directly succeeded by** b**, but** b **is never directly succeeded by** a ($a \rightarrow_L b$): if $a >_L b$ and $b \not>_L a$,

3. a **and** b **never succeed each other directly** ($a \#_L b$)**:** if $a \not>_L b$ and $b \not>_L a$,

4. a **and** b **succeed each other** ($a \parallel_L b$): if $a >_L b$ and $b >_L a$,

5. a **is succeeded directly by** b**, but before another occurrence of** a ($a >>_L b$): if a trace $T_i = \{e_{i,1}, ..., e_{i,j}, ..., e_{i,n_i}\}$ is present where $i \in \{1, ..., N\}$ and $j \in \{1, ..., n_i\}$, such that $T_i \in L$, $e_{i,j-1} = a$, $e_{i,j} = b$ and $e_{i,j+1} = a$,

6. a **is succeeded indirectly by** b**, but before another occurrence of** a ($a >>>_L b$): if a trace $T_i = \{e_{i,1}, ..., e_{i,n_i}\}$ is present where $j < k$ and $j, k \in \{1, ..., n_i\}$, such that $T_i \in L$, $e_{i,j} = a$ and $e_{i,k} = b$.

The numerical value of the dependency relation certainty of two events can be calculated, the *Dependency Measure* is denoted as $a \Rightarrow_L b$.

$$a \Rightarrow_L b = \left(\frac{|a >_L b| - |b >_L a|}{|a >_L b| + |b >_L a| + 1} \right) \tag{2.9}$$

These corresponding definitions and rules are summarized in Table 2.2. This comparison proves the theory that *Frequent Pattern Mining* is a great tool to support special *Process Mining* cases, where the pattern can be an *itemset* or a *sequence*.

	Frequent pattern mining	**Process mining**
Item base	$\mathcal{I} = \{\mathcal{I}_1,...,\mathcal{I}_k,...,\mathcal{I}_m\}$ $\mathcal{I}$, for example, represents a product	$E = \{e_1,...,e_i,...,e_n\}$ e_i stands for an event, for example, an alarm
Transaction	$\mathcal{T}_i = \{\mathcal{I}_1,...,\mathcal{I}_k,...,\mathcal{I}_n\}$ denotes a set of items	$T_i = \{e_{i,1},...,e_{i,j},...,e_{i,n_i}\}$ denotes a trace, which is a set of events
Database	$D = \{\mathcal{T}_1,...,\mathcal{T}_i,..., \mathcal{T}_N\}$ the set of transactions	$L = \{T_1,...,T_i,...,T_N\}$ represents a log, which is a set of traces
Association rule	$a \Rightarrow b$, where $a, b \in \mathcal{I}$ and $a \neq b$ a: antecedent, b: consequent	Dependency relations, where $a, b \in E$ $a >_L b$, $a \rightarrow_L b$, $a\#_L b$, $a \parallel_L b$, $a >>_L b$, $a >>>_L b$
Important measures	Support: $s_\mathcal{T}(a) = P(a)$; $s_\mathcal{T}(a \Rightarrow b) = P(a \cup b)$ Confidence: $c_\mathcal{T}(a \Rightarrow b) = P(b \mid a) = \frac{P(a \cup b)}{P(a)} = \frac{s_\mathcal{T}(a \Rightarrow b)}{s_\mathcal{T}(a)}$	Dependency measure: $a \Rightarrow_L b = \left(\frac{\lvert a >_L b\rvert - \lvert b >_L a\rvert}{\lvert a >_L b\rvert + \lvert b >_L a\rvert + 1} \right)$

Table 2.2: Corresponding nomenclature of frequent pattern mining and process mining.

One crucial issue is the handling of the activities in terms of time. They can have a dimension in time by having a *start* timestamp and an *end* timestamp, or they can be point-like, that is, only having one timestamp. In this work, they are taken as point-like, as the aim is to identify the typical chains of events to create the sub-logs. Taking the time dimension into consideration is the next step of process discovery. The parallelism of the events facilitates a more precise model of the processes. Although this aspect is beyond the scope of my work, time is considered in sequence-based log file partitioning, as the order of events is bounded by time.

Chapter 3

Process mining of industrial log files

3.1 Introduction

Process mining algorithms are applied in various fields, without the aim of completeness: in healthcare to improve the operational efficiency of processes [55], in business management to analyse the processes and reveal their bottlenecks [56], for the automatized analysis of financial statements during audit processes [57], or the support of e-learning are among the applications as well [58]. Moreover, a recent and highly promising application field is the identification of repetitive processes using process mining to discover potential processes for robotic process automation [59]. A collection of research fields and application areas in process mining can be found in [60].

Process mining algorithms are tailored for the purpose of discovering process flows, where the events of the different processes are organized into traces. A trace is a collection of events which are considered to be related in some way. Definition of the traces is essential for the purpose of process discovery, especially in the case of alarm management, where the connection between alarms and operator actions requires accurate trace identification or generation. The difference between them is, that although there are no traces in the log file, identification supposes some level of system knowledge that can be the base of the trace definition. In the case of generation, the user defines the trace rules, as is the case in the case study section. As generation methods can be considered a subset of identification methods, the term identification will be used. Therefore, the structures of alarm & event-logs is unsatisfactory in terms of process mining as they lack this very important component, that is, trace indicators. A process mining-based approach was applied to evaluate the behaviour of plants and support the rationalisation of alarms. Its basic concepts are presented in [61], while the applicability of process mining algorithms for the determination of alarm performance metrics and the exploration of process behaviour is presented in [62], which can be considered to be the motivation of the present chapter.

Although these studies usually focus on different aspects of alarm management, e.g. operator actions or alarm rationalization, they lack the general formalization of the problem and the goal-oriented definition of the input database for different purposes of analysis. Given the lack of a comprehensive study on the applicability

of the techniques of process mining with regard to large-scale industrial alarm & event log databases, the contribution of the present work is the following:

- The goal-oriented tasks and the related definition of the events are discussed, as well as further characteristics of the events and resources. Traces must be defined to provide a suitable structure of alarms and the related operator actions, resulting in the most appropriate input dataset for the applied mining algorithms.
- The effectiveness and applicability of the proposed methodology are presented with regard to the analysis of the large-scale alarm & event-log database of an industrial plant. Understandable and comprehensive information have been gained, that is useful not only in alarm management and operator training systems, but in the process mining of other industrial sectors.

According to the contributions, the core applicability of the proposed methodology is not narrowed down to the industrial alarm systems but can be transferred to other fields of applications as well, where temporal events are present and their follow-up or cause and effect type of relationship is to be analysed (similarly to the back and forth relationship of alarms and operator actions). In the case of alarm systems, the type of process control system, let it be a distributed control system (DCS), supervisory control and data acquisition system (SCADA), or any other type of system, is irrelevant, as long as the provided alarm (or event) data is timely and accurate. Using the proposed methodology, the alarm evolution paths can be identified, the triggering alarms of operator actions can be revealed and recommendations for the reduction of the probability of hazardous situations can be provided.

The structure of the chapter is the following. In Section *Materials and methods*, a brief overview of industrial log files (structure and content) is provided; the goal-oriented definition of the traces is introduced; the necessary rules to generate traces are identified; the process mining tools necessary to achieve the goals are discussed. In Section *Results and discussion*, the previously defined process mining tasks, the preparation of the log file, the time distribution analysis of the events (alarms and operator actions), the alarm spillover analysis between the production units and the discovery of the connection between alarms and operator actions are examined. In Section *Conclusion*, some concluding remarks are provided, experiences discussed and possible future research directions identified.

3.2 Materials and methods

This section introduces the basis of process discovery (log files), the event-clustering method (trace identification rules) and the goal-oriented selection of process exploration tools. As in the case of any project-like activity, an execution plan to achieve our predefined goals must be drawn up. In terms of the present case study, a schematic summary is provided in Fig 3.1.

Our goal is to identify basic patterns in the chain of alarms to focus on frequent sequences that can help us compile a prediction model of the alarms. In Table 3.1, an example of an industrial alarm and event-log can be seen. The column labeled "Tag" has been added to support the analysis and is a summarized representation

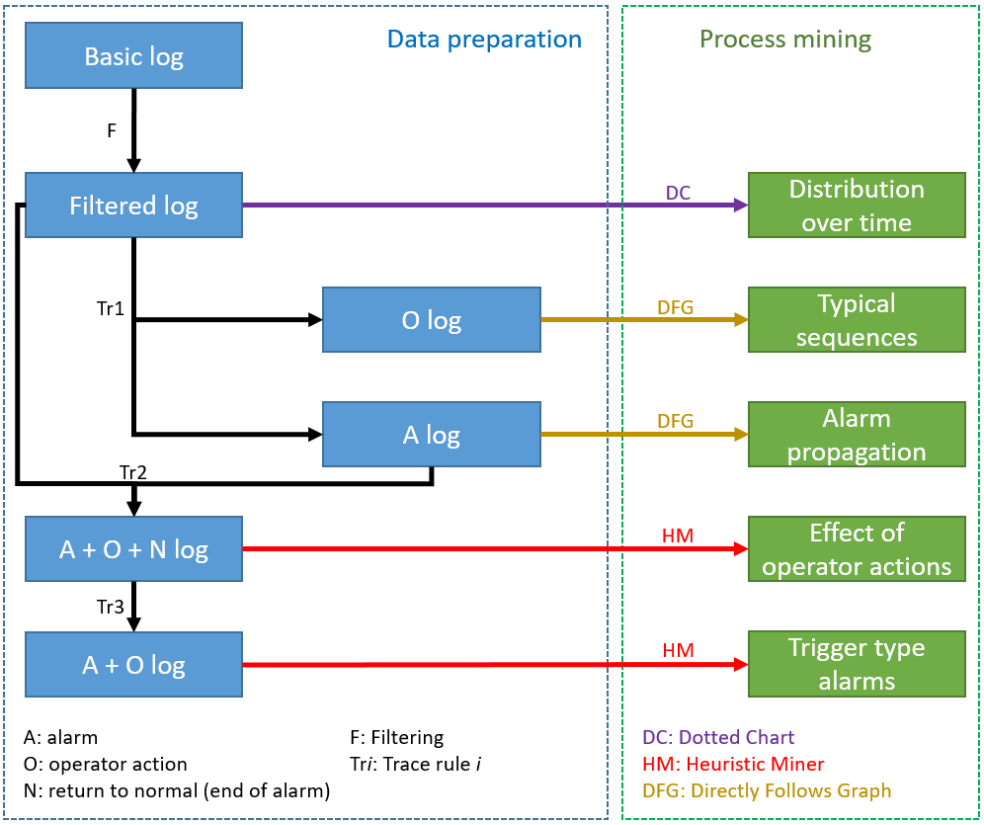


Figure 3.1: The concept of process mining-based alarm management. After preparing the data, we can perform our process discovery tasks on our sub-logs, that were generated by applying the suggested trace rules.

of the sensor's name. The first part is the name of the Tag, the second and third are those of the Unit and Production unit, respectively, while the last one is the type of event (A: *Alarm*, O: *Operator Action*, N: *Return to Normal*). Different "sub-logs" are necessary to examine various mining tasks. The required types of events are summarized in Table 3.2. The object of the analysis indicates what information is of interest with regard to the analysis provided in the task column. The event types indicate the types of events available for the analysis. Finally, the suggested tools to process are mentioned. The three types of objects are the following:

- **Basic log**: all three types of events are needed, after a filtering/cleaning step, the log file has to be put into a standardized format. This format is XES (Section 3.2.1).
- **Alarms**: Dotted chart, Directly-Follows Graph and Heuristic Miner are proper tools to analyse the different aspects of alarm propagation. Adding Operator Actions to the traces, their triggering alarms can be identified.
- **Operator Actions**: the tools and tasks are more or less the same as in the case of Alarms. The most complex task is to explore the effect of the Operator Actions, this needs all three types of events, as Alarms are the trigger events and Return To Normal events are the consequences of Operator Actions.

There are two kinds of tools to process the data, preparation type and the ones responsible for the Process Mining part itself. The filtering/cleaning step is a stan-

dard data processing task, it can be tailored and automatized by using *python* programming language, as well as its transformation to XES standard. The three Process Mining tools are also available in *python*, this way it is quite easy to develop an integrated solution tailored for the actual purpose.

- **Dotted chart analysis**

  The most transparent method by which to visualize the event log is the dotted chart analysis. In these charts, a dot represents a single event in the log with two orthogonal dimensions, namely time and component types. Component types like instance, originator, task, event type or data elements are shown on the vertical axis. Time is measured on the horizontal axis of the chart. Many measures related to events can be determined such as the average number of occurred events over a certain time period, the maximum number of events in that time period, the maximum and minimum time interval between events, etc. Time can be presented factually or relatively. The relative time can be used to abstract the log file. For every component type, the first event is positioned at time 0 and subsequent events are placed relative to the time of occurrence of the first event. Moreover, the shape and colour of the dots can be changed depending on the examined event attributes, adding dimensions to our chart.

  From a chart like this, a lot of useful information can be obtained, e.g. where the alarms occur more frequently or which production units are affected more by alarm events.

- **Directly Follows Graph**

  On a Directly-Follows Graph (DFG) an edge is represented between two nodes when at least one trace where the target event follows the source event is present. In a nutshell, this method by witch a DFG is obtained [63]:

  - defines 3 frequency parameters, namely τ_{var}, τ_{act}, τ_{df};
  - removes cases with a frequency lower than τ_{var} from the log;
  - remove events with a frequency lower than τ_{act} from the filtered log, and adds a node for each of the remaining activities;
  - connects the nodes where 2 activities follow on from each other at least τ_{df} times.

  On these graphs two metrics can be represented, namely frequency (the number of times the target event follows on from the source event) and performance (the average time elapsed between the source and target events), the decision depends on the type of required information.

- **Heuristic Miner Algorithm**

  The Heuristic Miner Algorithm explores the control-flow perspective of the process model. The log is analyzed for the presence of causal dependencies. If an event is always followed by another event, a dependency relationship probably exist between these events. The log should be analyzed for these causal dependencies. The advantage compared to α-miner is that the Heuristic Miner Algorithm considers frequencies and can handle skipping activities [54]. Several parameters can be adjusted, e.g. *minimum activity count*. Events that occur under this threshold are not shown on the nets, which can be a *Heuristic net* or a *Petri net*. Another parameter is the *minimum DFG occurrences*, which is the minimum number of occurrences of an edge to be considered. This attribute shows that the Heuristic Miner is based on *DFG*. Heuristic mining requires a clear starting and end event, assuming that every activity is located on a path from the starting activity to the end activity. As is the case in *DFG*, frequency and performance parameters can be entered on the net.

Table 3.1: Example of an industrial alarm and event log file

ID	Tag name	Event	Start time	End Time	Unit	Prod. unit	Tag
1	321	A	2018.05.01 00:01:01	2018.05.01 00:03:08	3	1	321_3_1_A
2	632	A	2018.05.01 00:01:03	2018.05.01 00:10:06	4	5	632_4_5_A
3	421	A	2018.05.01 00:01:10	2018.05.01 00:05:10	2	5	421_2_5_A
4	312	A	2018.05.01 00:01:30	2018.05.01 00:03:08	3	1	312_3_1_A
5	321	O	2018.05.01 00:02:10	2018.05.01 00:02:10	3	1	321_3_1_O
7	321	N	2018.05.01 00:03:08	2018.05.01 00:03:08	3	1	321_3_1_N
8	421	O	2018.05.01 00:04:01	2018.05.01 00:04:01	2	5	421_2_5_O
10	632	N	2018.05.01 00:10:06	2018.05.01 00:10:06	4	5	632_4_5_N

Table 3.2: Task oriented event types in sub-logs and suggested tools to process (A: *Alarm*, O: *Operator Action*, N: *Return To Normal*, PM: *Process Mining*)

Task	Object	Event types	Tool
Data preparation for PM	Basic log	A,N,O	Filtering and XES generator
Distribution over time	Alarms	A	Dotted chart
Typical processes	Alarms	A	Directly-Follows Graph
Spillover among units	Alarms	A	Directly-Follows Graph/Heuristic Miner
Trigger type events	Alarms	A,O	Heuristic Miner
Distribution over time	Operator actions	O	Dotted chart
Typical processes	Operator actions	O	Directly-Follows Graph
Effect of operator actions	Operator actions	A,N,O	Heuristic Miner

3.2.1 Goal-oriented definition of the traces

There are three areas to discuss the rules concerning trace identification, the analysis of alarms, operator actions and their relations.

- **Analysis of alarms**

  The first concept to explore in an alarm management system is the spillover effect of the alarms. These sequences of alarms are caused primarily by the decline in product streams, as well as the spread of pressure anomalies or attributes (temperature, concentration) connected to technology streams. According to this, probable propagation times are related to the sojourn times of equipment, the length of pipelines and the logic of the control system. Hazard and Operability Analysis (HAZOP) provides options to explore this malfunction propagation, in addition, more and more attention is being paid to Dynamic HAZOP. As its automated use is challenging [64] and these methods are very resource-demanding (require expert engineers), it would be beneficial to explore these potential relationships automatically from the log files. In this case, it is practical to use a rule of thumb to define the possible propagation times. This rule can be determined by analysing the time of occurrence of consecutive events originating from different production units.

- **Analysis of operator actions**

  A similar analysis can also be performed on operator actions. Despite being a complex troubleshooting process that can last for hours, our primary goal is to identify the sequences of correlated operator actions (similar to parent-children type alarms). One way achieving this is to define a time window lasting between 10 and 60 seconds (based on the cognitive model of operators and the attributes of the existing system). A new series of actions is identified if the time gap between two consecutive actions exceeds this time window. Another way is to regard alarm acknowledgements as the end of intervention activities (if this type of event is found in our log file). This way, they generate the groups of action series.

- **Connection between alarms and operator actions**

  The most complex task is to analyse operator actions with regard to the alarms that trigger them and qualify the efficacy of the interventions.

  To determine which operator action trigger alarm events, operator actions should be placed into our existing alarm traces (which commence after the starting time of the trace and finish before the end time of the trace or the starting time of the following trace).

  If the aim is to explore the effect of the operator actions, the end times of the alarms should be put into the aforementioned traces, as the *Return to Normal* pair of alarm events.

Rules of generating traces

In the previous list, trace generating methods were identified, now the formalization of these is provided. At the start, we consider one trace, which contains all events, and with the following methods we will split it step by step.

$A_{j,n}$, $O_{k,n}$ and $N_{p,n}$ are the jth alarm, kth operator action and pth return to normal event of the nth trace (σ_n), where n $\in \{1, ..., |\sigma_i|\}$, j $\in \{1, ..., |A_i|\}$, k $\in \{1, ..., |O_i|\}$ and p $\in \{1, ..., |N_i|\}$. tw_1 is the time window constant for alarms, tw_2 is the time window constant for operator actions, $t(A_{j,n})$, $t(O_{k,n})$ and $t(N_{p,n})$ are the timestamps of the related events. The explanation of the rules and their mathematical description is as follows.

Trace rule 1: $t(A_{j+1,n}) - t(A_{j,n}) > tw_1, A_{j+1,n} \Rightarrow A_{1,n+1}, A_{j,n} \Rightarrow A_{|A_i|,n}$: if the difference between the timestamps of two consecutive alarms is greater, than tw_1, then the alarm with the higher index will be the first event of the next trace, the one with the lower index will be the last event of the actual trace. This rule can be applied to Operator Actions as well. These traces provide the input to gain the distribution of events over time, the identification of typical event sequences and the spillover of the alarms among production units.

Trace rule 2: From traces made by Trace rule 1, we generate the return to normal events from the end timestamps of the alarms. This means, that the number of return to normal events will be equal to the number of alarm events ($|A_i| = |N_i|$) and traces will lap over each other, as an alarm of a trace can end later, than the start time of the next trace's first alarm. We put an operator action into the trace, if it's timestamp is later than the first alarm of the trace and sooner, than the last return to normal event of the trace ($t(O_{1,n}) > t(A_{1,n})$, $t(O_{|O_i|,n}) < t(N_{|N_i|,n})$). A visualized example is shown on Figure 3.2. From these traces the effect of Operator Actions can be gained. To identify trigger type alarms, Return To Normal events have to be removed from the traces made by Trace rule 2. Obviously, they shouldn't be excluded, but process mining a log without unnecessary events results in a more clear process model.

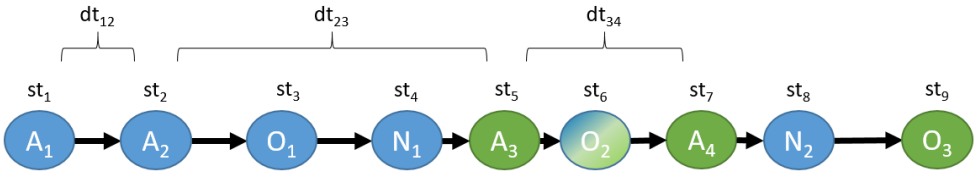


Figure 3.2: st_i denotes the occurrence time of the event, dt_{ij} the time difference between alarm occurrence times, dt_w the time window and T_i the trace. If $dt_{12}, dt_{34} < dt_{\mathrm{w}}$ and $dt_{23} > dt_{\mathrm{w}}$ and $\forall st : st_{\mathrm{i}} < st_{\mathrm{i+1}}$, then $A_1, A_2, O_1, N_1, O_2, N_2 \in T_1$ and $A_3, O_2, A_4, O_3 \in T_2$. It is worth to note, that operator action O_2 belongs to two traces.

3.2.2 Process mining-based alarm management solutions

In this section, the theoretical background of the applied analysis methods is presented.

Different algorithms of process mining can help us to identify patterns within the swarm of data placed in the log files. Given the need to find a solution to this common problem, different process mining techniques and several software products to evaluate the data mining tasks can be used. In this work, instead of using the well-known and usual process mining tools (like ProM [65] or EMiT [66]), I have used an open-source Python programming language-based solution, *PM4Py*. This library provides a wide range of process mining tools and since it is based on Python, which is a great tool for manipulating huge data sources (like industrial log files), *PM4Py* is an excellent option to develop semi- or fully automated process discovery methods from scratch on one platform.

3.3 Results and discussion

In order to show the industrial applicability of the formerly introduced process mining method, the alarm management system of an industrial hydrofluoric acid alkylation plant will be analyzed. The process flow diagram of the plant can be seen in Figure 3.3. The plant consists of four production units and more than 400 tags, the distributed control system is a Honeywell product. The logic of the tag names is X...X_YY_Z, where X...X is the identifier of the tag, YY is the identifier of the production unit (where the tag is located) and Z is the type of the event. The identifiers of the production units are the following: *CH*: isostripper, propane-depleting and propane handling unit; *U1*: utility streams; *AC*: reactor and acid generating unit; *FD*: raw material and drying unit; *02*: "virtual" unit, collection of sensors, that cannot be assigned to a specific unit. There are three types of events, namely Alarm (denoted with A), Operator Action (denoted with O) and Return To Normal (denoted with N).

Even though an alarm rationalization was performed on the plant, so the events in the log files are all considered to be relevant by the operating personnel, the log files need to be processed carefully, as remaining problems can be present, which can cause issues while undertaking the process discovery tasks.

3.3.1 Preparation of the log file of the alarm system

The log file of the aforementioned plant contains a lot of data, in excess of 200,000 events over a time period of four months. As previously discussed, the log file must be filtered to ensure only relevant and valuable data for the purpose of process mining is retained. The minimum number of attributes for process mining is three: an identifier of the event, at least one timestamp of the event (start or complete) and an identifier of the trace. Although not mandatory, it is useful to have additional attributes, for example, the name of the resource that triggered the event, the name of the organizational group in which the resource is located, in case of temporal-type events the counterpart of the timestamp (start or complete) and the type of the event.

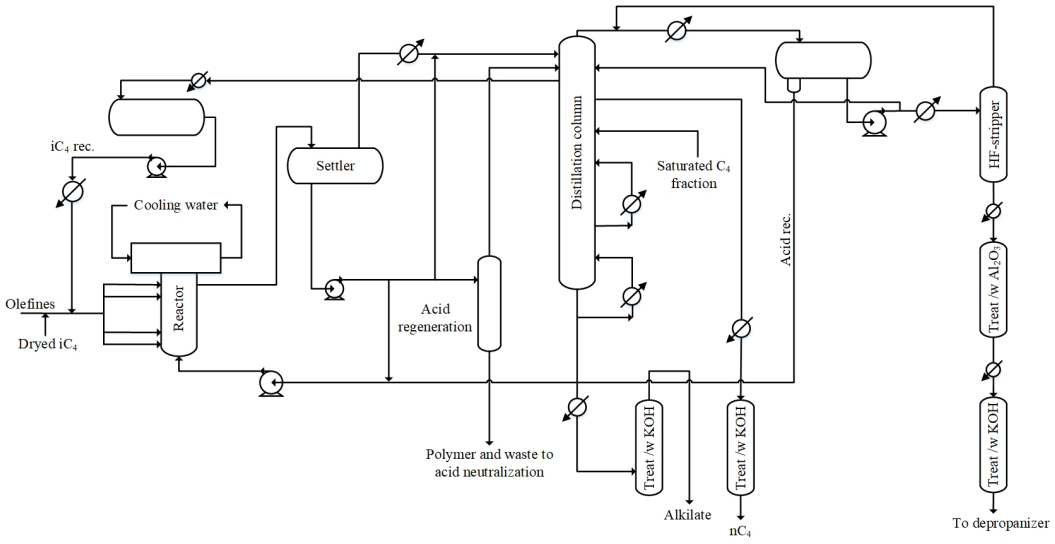


Figure 3.3: Schematic diagram of the plant in the case study

Another aspect that must be taken under consideration is what type of events to keep. Ten event types are present in the analysed log file, namely *Alarm, Return To Normal, Acknowledge, Operator Action, System, Operator Message, Suppress, Shelved, Unshelved, Process Event*. According to our goals, first it was decided to retain three types of events, that is, *Alarm, Return To Normal and Operator Action*. *Alarm* and *Operator Action* will be used to explore their number of occurrences and the sequences in which they occur, moreover, all three will be used to determine the causal relations between the alarms and operator actions. As a result of filtering types of events, approximately 80,000 events remained.

As soon as the sub-logs with the needed types of events are obtained, traces must be generated. Trace window constants have to be defined for every sub-log, the object and the contained event types of these sub logs are collected in Table 3.2. The value of these constants depends on the actual system, literature data and industrial experiences.

3.3.2 Distribution over time and typical event chains

To visualize the distribution of the events over time, a dotted chart is an excellent tool to use. Formerly, time distribution analysis has been identified as a task for both alarms and operator actions. As it can be interesting to compare the time of occurrence of alarms with the time of occurrence of operator actions, both have been visualized on one dotted chart. Figure 3.4 shows a one-day-long time window of production units CH and U1, the colours represent the types of events. It can be seen that although tags are present where both *Alarm* and *OperatorAction* events occur, many can be identified where this is not the case. It is not inevitable, that interventions are made at the same place where the alarm occurs. For example, only alarms are located in production unit U1, in production unit CH more operator actions are present than alarms. It can be supposed that the alarms in U1 trigger alarms in CH which in turn trigger the operator actions.

By briefly examining the time distribution statistics, it can be seen that either extremely long-lasting and near-to-zero seconds long alarms are present. These

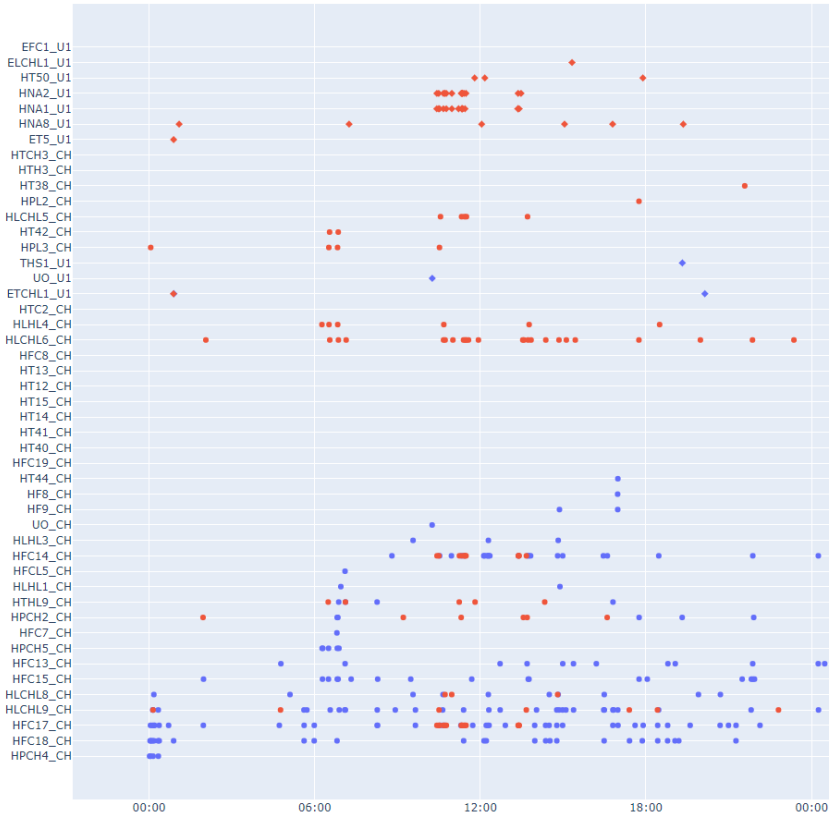


Figure 3.4: Distribution of alarms and operator actions over time (red: *Alarm*, blue: *Operator Action*). The x axis shows the time, the y axis shows the name of the tags.

extreme values can have a biased effect on our process mining results. These events contain little or no timely information for the operators were filtered out from the log file. Only alarms, which lasted between 5 and 28,800 seconds (8 hours) were retained. Of course, the upper and lower limits should be considered based on the given system and task.

Now that our log file has been "normalized", the next step is to generate the traces. If the needed trace windows are to be determined, the statistics regarding the time difference between the starting times must be examined. According to the statistics concerning the alarm starting times, for exploring the typical alarm chains, the trace window (the minimum elapsed time between two events to start a new trace) was chosen to be 220 seconds (the median value). To ensure at least two events are found in a trace (which is the minimum to be considered in a chain), the traces consisting of one event must be removed.

The first tool that can be used to explore the typical alarm chains is the *Directly-Follows Graph (DFG)*. The frequency of both the events and nodes can be put on the graph. Figure 3.5 is a portion of the *DFG* of the alarms, as the original graph is too large to be presented in full here.

Suppose the typical alarm propagation time between the units is sought, a *DFG* can be used once again, where the average elapsed time between two alarms occurring in different units is added, as is presented in Figure 3.6:

Although the average times are more or less identical, the frequency at which the alarms occur in unit *U1* is much higher than in the other units. Zero seconds

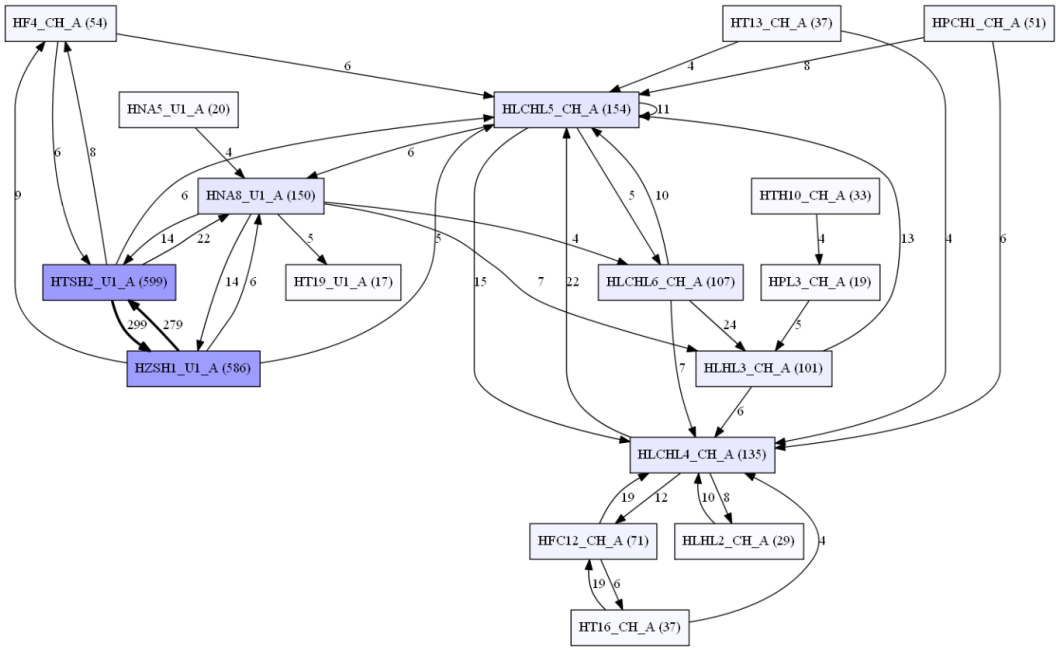


Figure 3.5: Alarm series (part of the *DFG*). CH: isostripper, propane-depleting and propane handling unit; U1: utility streams. The numbers on arrows and in boxes represent the number of transitions and occurrences, respectively.

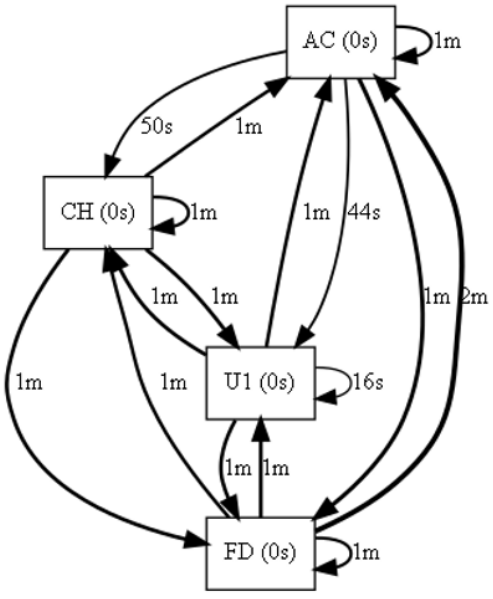
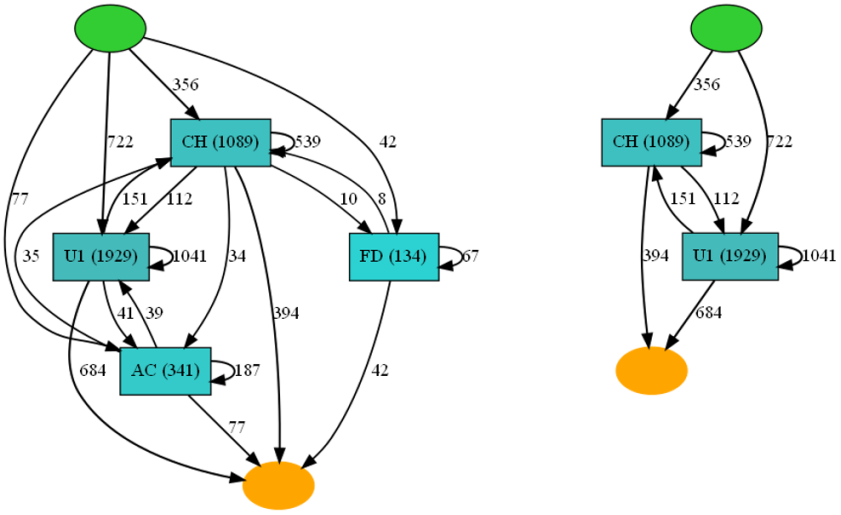


Figure 3.6: Average alarm propagation times between units. CH: isostripper, propane-depleting and propane handling unit; U1: utility streams; AC: reactor and acid generating unit; FD: raw material and drying unit.

can be seen in the boxes of the units because the events were regarded as point-like so they are dimensionless in terms of time.

Even though a *DFG* can provide a good overview of typical sequences, providing a quick overview of the event environment, a different tool must be used to explore

processes. One option is the Heuristic Miner Algorithm to obtain a so-called Heuristic Net, which is one form of visualizing the typical processes. As frequent event chains are to be explored, the parameter *minimumactivitycount* was used and set at 100 and 500, as presented in Figure 3.7. On a Heuristic net, the green ellipsis represents the start, the orange one the end of the process.



a) Minimum activity count = 100 b) Minimum activity count = 500

Figure 3.7: Heuristic nets of alarm propagation between production units. CH: isostripper, propane-depleting and propane handling unit; U1: utility streams; AC: reactor and acid generating unit; FD: raw material and drying unit

Obviously, the biggest proportion of the alarm events occurs in unit *U1*, which has a high rate of interaction with units *CH* and *AC*. Although our assumption that alarms in *U1* trigger alarms in *CH* is proven, the opposite can also occur. This shows the advantage of a Heuristic Net over a dotted chart or a *DFG*.

To explore the typical series of operator actions, the formerly presented *DFG* is used. Figure 3.8 shows the frequency of two consecutive operator actions. It can be seen, that the most operator actions are located in unit *CH*, as was presumed from Figure 3.4. Nodes denoted in a darker colour with thicker edges represent more frequent events and transitions. From this graph, the group of tags where the most of the operator actions occur can be identified, along with information about frequent series of operator actions.

3.3.3 Correlation between alarms and operator actions

To understand the connection between operator actions and alarms, two different questions may have to be answered:

1. Which alarm trigger an operator action?
2. What is the effect of the operator actions?

To answer these two questions, different log file and trace identification rules are required. The first requires the event types *Alarm* and *Operator Action*, while the second requires *Return To Normal* events, which can be a result of an *Operator*

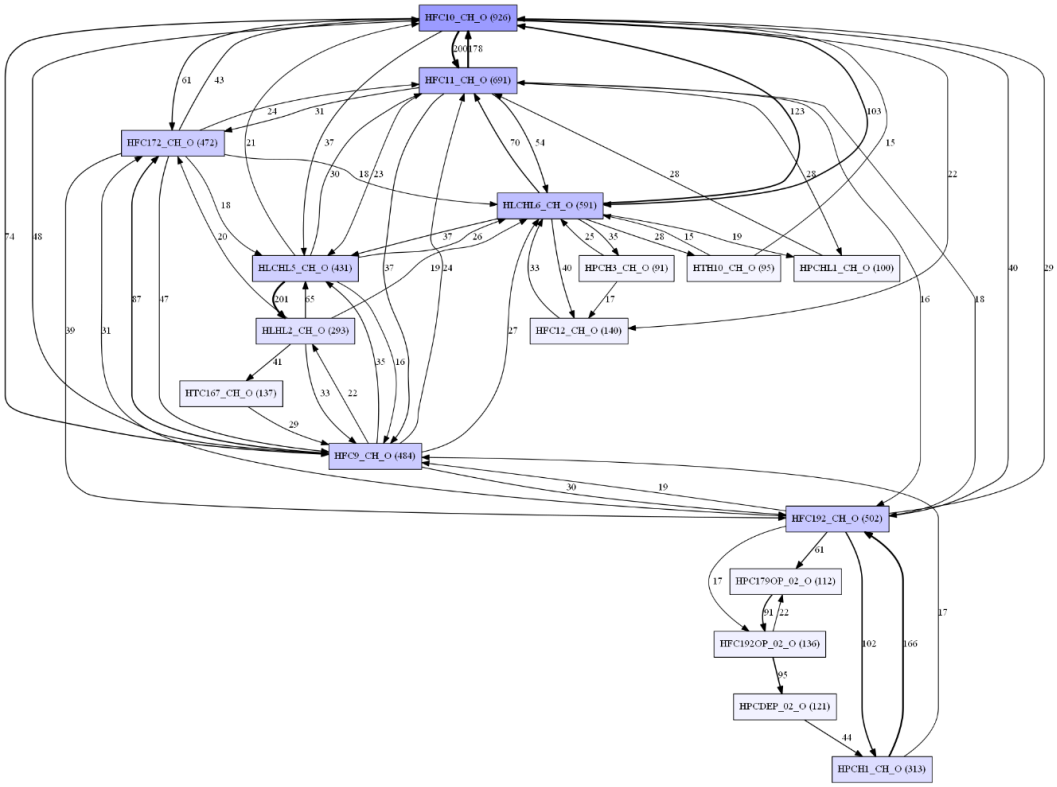


Figure 3.8: Series of operator actions. CH: isostripper, propane-depleting and propane handling unit; 02: virtual unit.

Action. The trace identification rules were determined in Section 3.2.1. Firstly, our formerly generated *Alarm* traces are taken and *Return To Normal* events are generated from the end timestamps of the alarms. Subsequently, operator actions are placed into our traces with timestamps between the first and last events in the trace (practically speaking, the first *Alarm* and the last *Return To Normal* events). By considering this method, trigger-type alarms and the effect of operator actions can be handled in one task as the log for trigger-type alarms would also be the one for exploring the effect of operator actions in the absence of the return to normal events.

Figure 3.9 shows the explored processes (*minimum dfg* was set at 55). By closely examining the net, two main types of processes (in addition to a third one) can be identified. For the purpose of better readability, starting and end points of the alarm sequences have been highlighted with coloured boxes (same colour for each pair). The boxes with dashed lines denote those processes where no *Operator Action* occurred between the starting and the end of an alarm sequence (Process type 1). The ones denoted with solid lines mark processes containing *Operator Action* (Process type 2). The third type, denoted by dotted lines belongs to both, as this alarm (HNA8_U1_A) can end either with or without an *Operator Action* (Process type 3). Furthermore, an area which is worth examining closely is the green ellipse, as this part definitely appears to be a typical process (this "group" can also be seen in Figure 3.8). From this net, which tag is relevant in which kind of process discovery task can be determined, moreover, our log file can be filtered further and the mining conducted again.

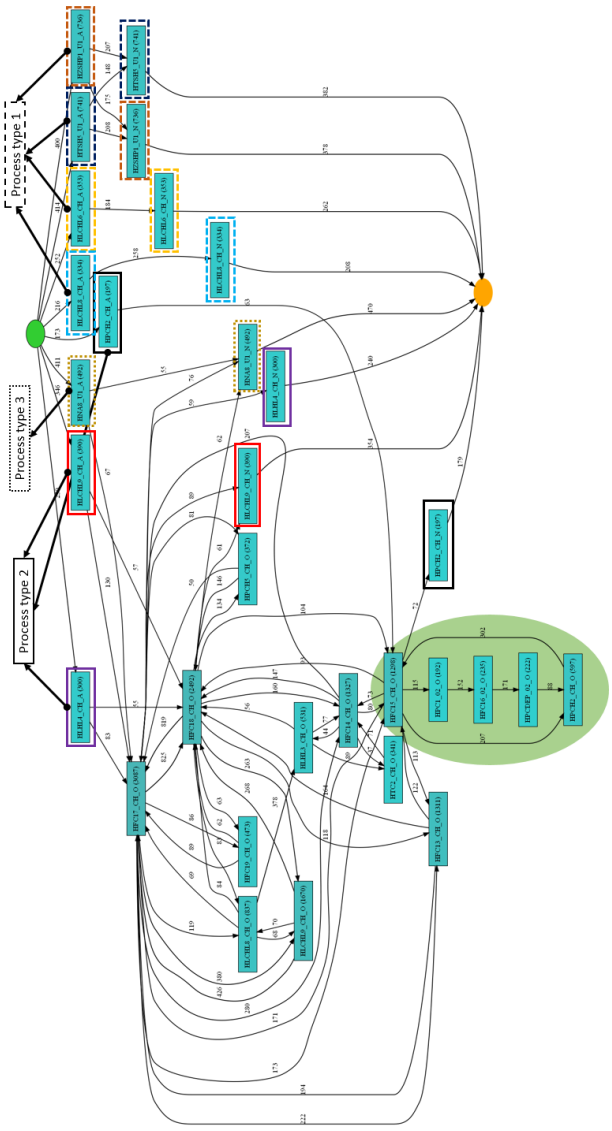


Figure 3.9: *Alarm(A)-Operator Action(O)-Return To Normal(N)* sequences. CH: isostripper, propane-depleting and propane handling unit; U1: utility streams; 02: virtual unit.

The HLCHL9_CH_A alarm indicates problems with the liquid level at the HF stripper bottom. It is well visible that the operators frequently apply the HFC17_CH_O action in this situation, which modifies the bottom inlet of the HF stripper. Similarly, they apply the HFC18_CH_O action in this situation, which controls the steam inlet of the reboiler of the stripper. In the case of the events marked by black brackets, the HPCH2_CH_A alarm indicates problems with the depropanizer pressure, while the action applied in this situation, HFC15_CH_O, controls the blow off of the technology. The HTSH5_U1_A and HZSHP1_U1_A alarms (dark blue and brown dashed brackets) almost always co-occur. As these alarms both related to the problem of the same pump, they tend to be redundant and their definition should be revised by the process experts.

3.4 Conclusion

As industrial technologies are becoming more and more complex, the work of operators required to ensure the safe and optimal operation is getting increasingly challenging. With the introduction of the Industry 5.0 approach in progress, there is an emerging need for solutions to balance the productivity and efficiency with the life quality (work and home) of the workers, as well with the affection of the industry on society. One way to achieve this is to design alarm management systems based on tools that were developed to answer the challenges of the 4th Industrial Revolution (Industry 4.0). A properly built alarm management system can lower the workload of the operators significantly, and reduces the probability of hazardous situations affecting the environment (including civilians). The primary goal of this work was to explore useful information from the historical data of industrial alarm management systems that can support the reduction of the operator workload and learn optimal operating strategies.

This work proposes a process mining-based method to discover the fundamental relations between alarms and the related operator actions. Standard process mining techniques are not suitable for the analysis of historical process data of similar type. The benefits of the goal-oriented design of the log files that allows the extraction of information available to more effective alarm management and operator training have been demonstrated.

The method was applied in the alarm management rationalisation project of an industrial hydrofluoric acid alkylation plant. The project demonstrated that with the help of process mining, alarm signals could be rationalised; therefore, the work of the operators will become safer, as well as more effective and, last but not least, the workload has been decreased.

The discovered process models are easy to understand and provide some kind of improved digital visualization of alarms and operator actions. Lee *et.al.* summerized the applicability of digital twins in Industry 4.0-driven process safety management [67]. Our process mining-based method is also suitable to support some of the improvement actions collected in that study as a complementary tool. These improvement actions, related to alarms are: generate alarm signatures that can be useful in abnormal situation management, identify critical operator interventions, improve procedural risk assessments, and reduce the time and risk of errors during traditional risk assessment processes. They can support operator action-related tasks as well, namely processing of procedures and operator actions: enhancing work design and operator performance, and representation and assessment of people and procedure related performance deviations and failures.

The gained information can be used to improve control systems, get a better insight into plant failure and behaviour (process hazard analysis), review process safety incidents (incident investigation), conducting what-if scenarios to understand how the plant may continue to run during unplanned maintenance or examining specific abnormal operation scenarios in more depth. It also gives the ability to assess initiating causes systematically and in an automated way based on historical data, due to the many failure modes of equipment that exist in a process plant. This is considered a key attribute to reduce resource intensive analysis.

Traditional hazard analysis processes have some shortcomings. A data science-based approach can address some of them, for example when there is a lack of

- depth of analysis,
- follow through to final consequences,
- completeness in identifying initiating causes and scenarios.

The outcome of this study proved, that the process mining-based analysis of events, along with the goal-oriented design of log files, should be added to the digital toolkit of process safety management.

Chapter 4

Sequence compression and alignment-based process alarm prediction

4.1 Introduction

Different approaches were proposed, developed, and used over time to reduce the number of alarms and present only the relevant ones to the operators. These methods were focusing on specific aspects of industrial alarm processing, however, no method - being that simple or composite - was offered to combine state-of-the-art approaches to tackle multiple challenges offering a more generic solution to multiple problems of the domain at once. To overcome these limitations (computational demand, generalization) I turned to a method based on the Needleman-Wunsch [68] algorithm that is often used for sequence alignment problems as it guarantees to find the best alignment regardless of the length or complexity of the sequences. In the present work, the pairwise2 method has been used, an implementation included in the Biopython package. A big advantage of this method is that it accepts integers as sequence items, making it applicable for any number of alarms. It handles gaps, penalization of gaps can be set as a parameter that reflects in the scoring value and implements a dynamic programming algorithm that suggests good performance which will be demonstrated during the study.

Frequent sequence mining algorithms suffer from the well-known problem of pattern explosion: a huge number of highly redundant frequent sequences are retrieved if the given minimum support threshold is too low. Modern approaches have surfaced to tackle the problem, which use the minimum description length (MDL) principle [69] to find the set of sequences that best summarize the data. The most remarkable algorithms employing MDL are GoKrimp [70] and SQS [71]. In the present study, the GoKrimp algorithm is suggested to build a concise and yet descriptive set of sequences from historical data, which can serve as the pattern database for a lightning-fast pairwise comparison of the real-time alarms coming from online devices. GoKrimp is based on the Krimp [72] algorithm, but mines compressing patterns directly. The algorithm applies a greedy procedure to gain patterns that have a positive compressing benefit. This method unlocks the possibility not only for real-time filtering of the alarm data but also sets the ground for its interpretation and contextualization of it.

With the help of the proposed method, the process control system can be transformed into a so-called state-based alarm system [73]. The main contribution of a state-based alarm system is that long-lasting [74] and nuisance [75] alarms can be removed, this way the workload of the operators can be lowered [76] to reduce their mental fatigue [77]. These kinds of systems are supported by machine learning algorithms to recognize alarm situations and to predict possible future scenarios by using probability values [78]. Usual state-based alarm systems need expert knowledge, and first principles based on this knowledge. The biggest advantage of the proposed method is its data-driven character, it does not need the aforementioned expert knowledge. On the other hand, it consumes a lot of time and works with data to have a reliable alarm prediction system. The decision on which type of method to choose depends on the circumstances of the task.

To be able to compare the incoming alarm data with the concise format of the database, a sequence alignment method was employed. The usage of sequence alignment methods in alarm processing methods has been suggested by several studies before, mainly when solving the problem of alarm floods. Similar patterns were mined from historical data so that methods could be constructed to avoid and handle alarm floods generated in an industrial environment. Common subsequences in alarm floods were explored by using the dynamic time warping (DTW) algorithm [79] . An improved Smith-Waterman algorithm for pattern matching of alarm floods was suggested considering the time stamps for similarity calculation of alarm floods [80] . These methods are computationally demanding; therefore, their biggest limitation is to present a processable output within a tolerable period. One solution is to introduce an alarm priority to be used together with the Basic Logical Alignment Search Tool (BLAST) method making the method more sensitive to alarms of higher priorities [81].

My work offers the following novel contributions:

- An integrated method is proposed, that utilizes sequential pattern mining and sequence alignment algorithms to identify and forecast in real-time operational states and their progression based on alarms.
- I offer a generic, parameter-free method that can be used with many industrial assets and many alarm management systems.
- The performance of the method is adjusted to the real-life requirements of the industry by giving high-confidence predictions in the sub-second region and updating the model with new information in a few minutes.

The roadmap of the chapter is as follows. A logical backbone of the proposed composite method and a detailed description of the applied algorithms are provided in the next section. This is followed by the analysis and discussion of the proposed method on real-life data. Finally, the work is wrapped up with the concluding remarks and outlooks provided.

4.2 Integrated sequence compression and alignment method to predict alarm scenarios

The main goal of the present work is to offer a self-consistent and straightforward method to recognize the operational states of complex systems in real time based on previously recognized historical patterns.

The proposed solution is divided into the following steps:

1. pre-processing of historical data, data cleaning, and sequence generation;
2. mining for frequent sequential patterns;
3. real-time comparison of the alarms to the frequent pattern database;
4. visualisation and evaluation of the results.

It is important to emphasize that this cleaning process aims to restrain the applied techniques to the alarm messages being potentially important for the process operators. Therefore, the applied steps, as in the case of any system-specific analysis, should aim to remove inconsistent or uninformative events from the data sets and be tailored for the specific system. This work aims to produce compressed frequent sequential patterns that are comparable with live event data streams using the sequence alignment technique. For that reason, the first step is not discussed in detail, as the used techniques and principles are well-known in data- and process mining, however, some information is provided in the case study section.

The schematic workflow of the suggested method is visualized in Figure 4.1. The workflow of the proposed composite method starts with the collection of historical alarm data (H) from the Distributed Control System (DCS) or Supervisory Control And Data Acquisition (SCADA) system of the analyzed industrial plant. After the collected data is pre-processed and cleaned (E), event sequences are formed from the series of alarms. These sequences are time window-driven and are called traces (T) that form the log file of the alarm management system (D^T). A compression method is applied to the sequences produced in the previous step resulting in the compressed sequence database, $C = \{C_1, ..., C_k, ..., C_n\}$. During prediction, after similar preparation steps, the actual online sequence $T_* = \{e_1, ..., e_i\}$ is compared to C using a sequence alignment algorithm returning not only the matches but their confidences too. Finally, the generated predictions are evaluated based on real events.

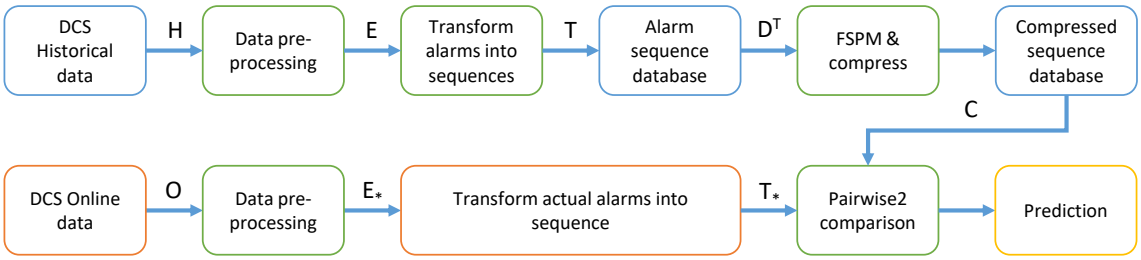


Figure 4.1: The workflow of the suggested event sequence similarity-based method.

4.2.1 Finding frequent patterns in the alarm database

An alarm management database contains $E = \{e_1, ..., e_i, ..., e_w\}$ alarms stored with some attributes, like a timestamp, the ID of the sensor, the ID of the equipment, etc., forming a D^T log file. A well-designed log file is essential for in-depth process analysis tasks [82]. An alarm represents a state of the system where a process variable exceeds the warning or alarm limits [25]. To predict potential alarm escalation scenarios, typical alarm sequence patterns have to be explored. To minimize the computation time to compare the actual alarm data and the stored sequences, the sequence database has to be compressed.

Sequential pattern mining algorithms provide a lot of overlapping patterns, that are variations of the same sequence. As the goal is to identify event chains quickly, these similar patterns have to be grouped. Several algorithms [83] have been proposed to find sequential patterns effectively, but very few researchers addressed the problem of redundancy reduction. Using the minimum description length (MDL) principle, the Krimp algorithm mines patterns that summarize the data in a well-compress representation [84]. This approach reduces the redundancy and generates patterns that are useful for classification, component identification [85], and change detection [86].

The Krimp algorithm, inspired by the gap handling of the SQS algorithm that is developed further into the GoKrimp algorithm to both penalize gaps and handle interleaving patterns. It is of crucial importance that **GoKrimp has no input parameters**, which makes it extremely attractive for the application in alarm sequence compression, as there is no need to fine-tune the method depending on the endless variations of different alarm handling systems. With this profile, GoKrimp algorithm is an ideal candidate to transform the sequence database into a compressed sequence database that contains characteristic and frequent sequential patterns from the original set. From D^T the compressed sequence database, $C = \{C_1; ...; C_k; ...; C_n\}$ can be gained with the method described in Algorithm 1.

Algorithm 1 GoKrimp(D^T) algorithm

Input: $D^T = \{T_1, T_2, \ldots, T_N\}$
$C \leftarrow \{\emptyset\}$
while $Benefit(\phi_*) > 0$ **do**
  $\phi_* \leftarrow$ **GetNextPattern**(D^T) ▷ picks a pattern from D^T
  if $Benefit(\phi_*) \leq 0$ **then**
    break
  end if
  $C \leftarrow C \cup \{\phi_*\}$
  use **Compress**$(D^T|\phi_*)$ ▷
end while
Output: C ▷ the compressed sequence database

In a nutshell, the algorithm is searching for those patterns (ϕ_*), that are adding compressing benefit to D^T. ϕ_* is initialized by the **GetNextPattern** algorithm, it is a sequence of frequent events where the most compressing benefit is provided. Then ϕ_* is added to C and replaced with a pointer in D^T. The detailed description of **GoKrimp**, including algorithms **GetNextPattern** and **Compress**, can be found in [70].

4.2.2 Comparing actual event chains with the compressed sequence database

To predict the most probable event chains after a certain event (that can be a sequence of events as well), the actual online sequence T_* has to be compared to the compressed sequence database C by using a sequence alignment method (Figure 4.2). Sequence alignment methods compare two (pairwise alignment) or more sequences (multiple alignment) by searching for a series of individual characters or character patterns that are in the same order in both sequences.

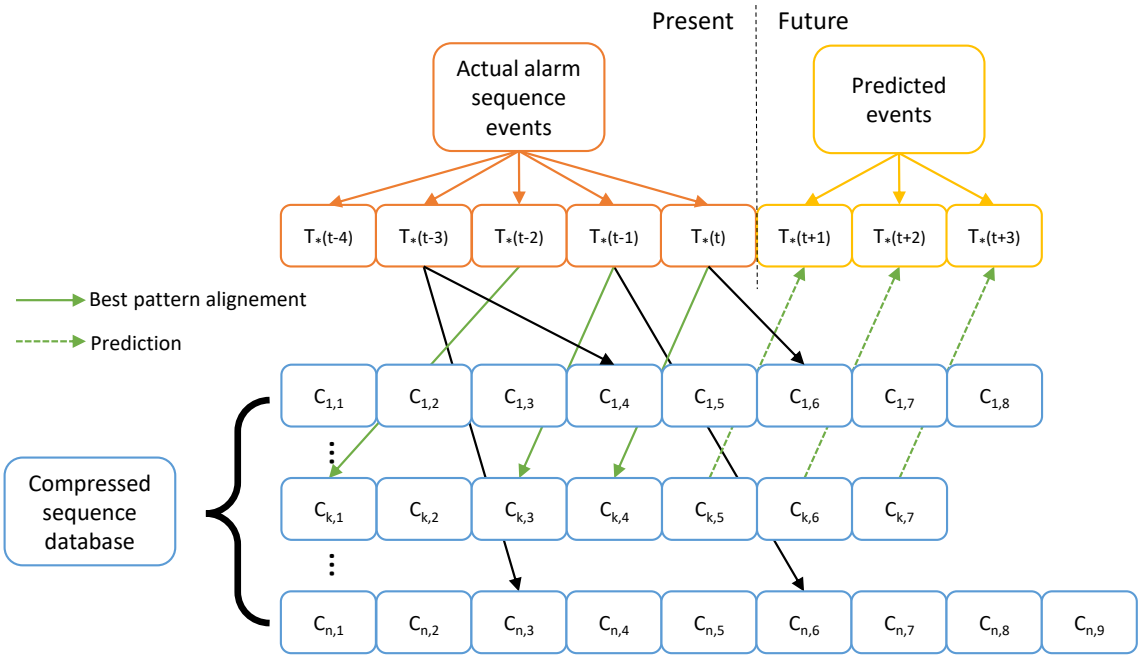


Figure 4.2: The event sequence prediction method. By comparing (arrows) the already occurred events (orange boxes) until t time ($T_*(t-4)...T_*(t)$) with the frequent event sequences of the compressed sequence database C (blue boxes), the most probable event path (golden boxes) can be predicted (green dashed arrows) based on the best alignment (green arrows). The allowed *gap* is 2 in this example.

The alignment the methods are looking for may be global-, overlap-, and local alignment. A global alignment finds the best concordance between all characters in two sequences. A local alignment finds just the sub-sequences that align the best. It is also pivotal to handle the problem of the similarity of sequences as some may be more similar to each other than others, several scoring functions have been developed that penalize substitutions (mismatches), insertions, and deletions (gaps) in the sequences.

The most important step in sequence alignment tasks is to create the score matrix (**S**). This matrix contains the alignment scores of the elements of the two sequences to compare. The score of the actual sequence element pair can be calculated based on the scores of antecedent pairs, and with two parameters. The substitution score (s) is the cost of substituting an event with another [87], it can be positive or negative either. A *match* has a positive score, obviously a *mismatch* has a negative. *Match* and *mismatch scores* can even vary for every event pair, in this case the values have to be identified in a so-called *substitution matrix*. The gap penalty (g) represents

the cost of an insertion or deletion of a residue [87]. The value of the gap penalty depends on the actual system, it can be zero, or it can be several times greater than the mismatch score (for example when comparing DNA sequences). The direction of the calculation is from left to right, and from upside to down. Considering T_* and $C_k \in C$ sequences to be compared, with i and j-length, the initial values and the calculation of the scores are the following:

$$\begin{gathered} \mathbf{S}_{0,0} = 0,\ \mathbf{S}_{i,0} = i \times g,\ \mathbf{S}_{0,j} = j \times g; \\ \mathbf{S}_{i,j} = \max \begin{cases} \mathbf{S}_{i-1,j-1} + s(T_{*,i}, C_{k,j}) \\ \mathbf{S}_{i-1,j} + g \\ \mathbf{S}_{i,j-1} + g \end{cases} \end{gathered} \tag{4.1}$$

For a better understanding, Figure 4.3 gives a visual explanation of the score calculation.

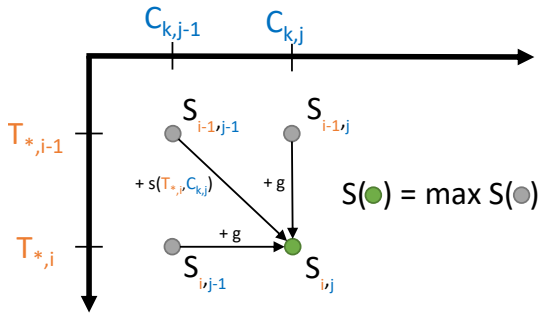


Figure 4.3: The score calculation method described in Equation 4.1, where the cost functions are g (gap penalty) and s (substitution score)

.

Having all scores calculated enables the creation of the traceback matrix ($\mathbf{B}$), which will define the optimal alignment. The values of $\mathbf{B}$ can be determined as follows:

$$\begin{gathered} \mathbf{B}_{0,0} = done,\ \mathbf{B}_{i,0} = up,\ \mathbf{B}_{0,j} = left; \\ \mathbf{S}_{i,j-1} = \max(\mathbf{S}_{i-1,j-1}, \mathbf{S}_{i,j-1}, \mathbf{S}_{i-1,j}) \rightarrow \mathbf{B}_{i,j} = left; \\ \mathbf{S}_{i-1,j} = \max(\mathbf{S}_{i-1,j-1}, \mathbf{S}_{i,j-1}, \mathbf{S}_{i-1,j}) \rightarrow \mathbf{B}_{i,j} = up; \\ \mathbf{S}_{i-1,j-1} = \max(\mathbf{S}_{i-1,j-1}, \mathbf{S}_{i,j-1}, \mathbf{S}_{i-1,j}) \rightarrow \mathbf{B}_{i,j} = diag. \end{gathered} \tag{4.2}$$

Starting from the right bottom cell, the optimal alignment can be constructed by following the path described, e.g., moving to the direction of the neighbor cell with the highest alignment score. The alignment method is summarized in Algorithm 2.

Let's take an example for a set of alarm events (E), where

$$\begin{gathered} E = \{e_1; e_2; e_3; e_4; e_5; e_6; e_7; e_8; e_9\}, \\ T_* = \{e_2; e_3; e_5\}, \\ C_1 = \{e_1; e_3; e_5; e_9; e_7\}, \\ C_2 = \{e_4; e_2; e_6; e_3; e_9; e_7\}, \\ C_3 = \{e_1; e_4; e_8; e_6; e_7\}, \end{gathered}$$

match score=1, *mismatch score*=-2 (s), and *gap penalty*=-10. This is a simplified example to present the working of the method. The values of *match score*, *mismatch score* and *gap penalty* depend on the events to compare. The optimal alignment and

Algorithm 2 The sequence alignment algorithm

Input: T_*, C_k ▷ $|T_*| = i$, $|C_k| = j$
Initialize $\mathbf{S}_{i\times j}$ and $\mathbf{B}_{i\times j}$
For $\forall i, j$ **do**
Calculate $\mathbf{S}_{i,j}$ and $\mathbf{B}_{i,j}$
End for
Find the best alignment using $\mathbf{B}$
Calculate Score of the optimal alignment
Output: $Score(T_*, C_k)$ ▷ the score of the optimal alignment

its *Score* have to be calculated for every (T_*, C_k) pair. For example, if T_* has i events and C_1 has j events, the scores for the pair $T_{*,1}(e_2), C_{1,1}(e_1)$ according to Equation 4.1 are

$$S_{0,0} + s(T_{*,1}, C_{1,1}) = 0 + (-2) = (-2),$$
$$S_{0,1} = (-10),$$
$$S_{1,0} = (-10),$$

so $S_{1,1} = (-2)$ and $B_{1,1} = diag$. The score and traceback matrices for (T_*, C_1) can be seen in Table 4.1:

	C_1	e_1	e_3	e_5	e_9	e_7
T_*	0	-10	-20	-30	-40	-50
e_2	-10	-2	-12	-22	-32	-42
e_3	-20	-12	-1	-11	-21	-31
e_5	-30	-22	11	0	-10	-20

Score Matrix

	C_1	e_1	e_3	e_5	e_9	e_7
T_*	done	left	left	left	left	left
e_2	up	diag	left	left	left	left
e_3	up	up	diag	left	left	left
e_5	up	up	up	diag	left	left

Traceback Matrix

Table 4.1: Example score and traceback matrices

Starting from the right-bottom cell of the Traceback matrix, following the directions resulted from the values of the Score matrix, the optimal alignment in this case is:

$$e_1, e_3, e_5, e_9, e_7$$
$$e_2, e_3, e_5, -, -$$

$Score(T_*, C_1)$=$e_1e_2 : mismatch + e_3e_3 : match + e_5e_5 : match + e_9- : gap + e_7- : gap = (-2)+1+1+(-10)+(-10)$=-20. Using the same technique, $Score(T_*, C_2)$=-30 and $Score(T_*, C_3)$=-26. With the highest $Score$ value, C_1 will be the basis of the prediction, so the most probable upcoming events after e_5 are e_9 and e_7.

In the present study the *pairwise2* method of *Biopython* package is used which employs a dynamic programming implementation of the Needleman-Wunsch algorithm. The algorithm reduces the number of possible global alignments by breaking down the problem into smaller sub-problems (sub-sequences), finds an optimal alignment for smaller sub-sequences, based on which the optimal solution of the original problem is constructed [88].

4.3 Application of the method on real life industrial data

In this section, the analysis of historical alarm data of a delayed-coker plant at the Danube Refinery of the MOL Group is presented alongside the algorithm used to predict the next element of an online sequence opening up the possibility to the creation of numerous data-driven Decision Support Systems (DSSs). A detailed description of the plant is provided in [82] , the process flow diagram of the analyzed technology can be seen in Figure 4.4. The analyzed alarm log contains more than 2,000 process tag names on the level of sensors and actuators recorded in almost 400 units (equipment, vessel), located in 19 production units (sub-parts of the plant) so our example of application can be considered a realistic and challenging case study. The section will follow and describe the stages of the data analysis. All the tag names and identifiers mentioned in this study have been altered from their original value due to confidentiality considerations. Data preparation, sequence generation, compression, and alignment processes were performed using Python programming language, that enables automated execution of the tasks.

Pre-processing of the historical data and building the compressed sequence database

The historical data of the alarm management system has been presented in the form of CSV database exports containing records for multiple events logged like alarms, warnings, operator actions, system messages, etc. for more than four months of standard operation time. Every alarm has a start and an end timestamp. The resolution of the timestamp is in the order of a millisecond. Attached to each alarm a TagName is specified that uniquely identifies the plant, the unit, the asset, and the sensor that produced the measured data triggering the alarm. Furthermore, an EventTypeID is given that unambiguously identifies the type of the record (alarms, warnings, operator actions, system messages, etc.). The dataset contains other

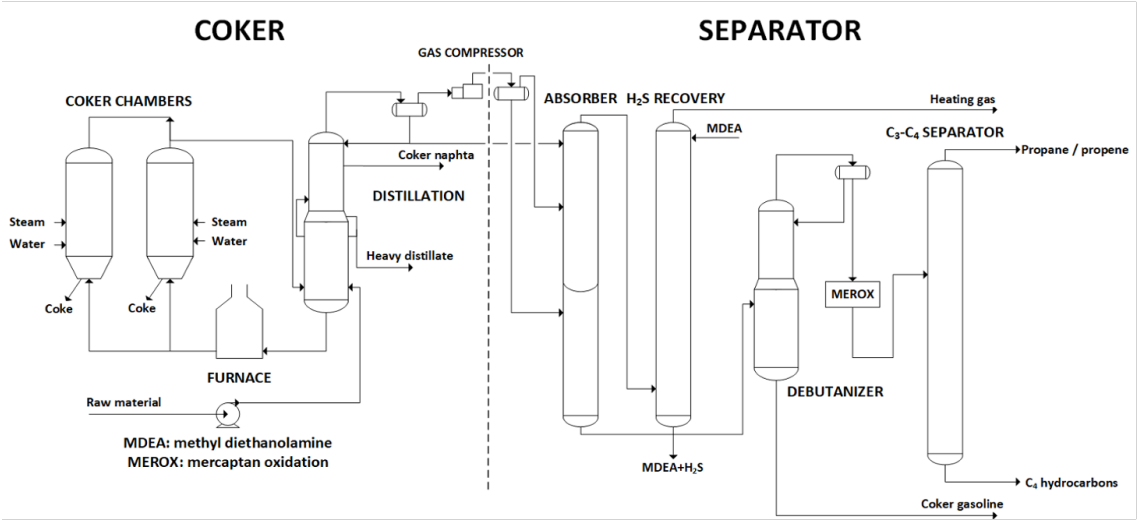


Figure 4.4: The process flow diagram of the analyzed industrial delay coker plant.

meta-data fields too, like alarm description, priority display name, etc. that did not play any role in the present analysis.

The historical dataset has been filtered for EventTypeID to select the alarms only which resulted in a dataset of more than 500,000 records. This filter has been run on KNIME Analytics Platform v.4.3.0 using freely available nodes. The records were further filtered in Jupyterlab v.2.2.9 with python v.3.8.8 environment using python code applying the following actions sequentially:

- deduplication executed on the level of alarms (no duplicate has been found),
- exclude unfinished alarms,
- exclude alarms that do not show up at least ten times over four months,
- exclude alarms with chatter index [89] above 0.05,
- exclude alarms that are shorter than 20 s and longer than eight hours [90] .

The effect of each filter on the number of records is illustrated in Figure 4.5.

Following data cleansing, the next step to be executed is to generate sequences from the stored events and format the output in such a way that the SPMF-GoKrimp [91] java package will accept it as valid input. The events were loaded with the help of python libraries and restructured following the logic:

- Assign unique numeric IDs for each TagName. This step is needed to generate an acceptable input for the sequential pattern mining algorithm.
- Identify the end of the sequences. One sequence ends if no alarms were coming up for more than 120 seconds. It is assumed, that having a 2-minute silence between the alarms means that the system is back to a normal operational state and any upcoming alarm describes an irrelated, separate behavior/state of the system.
- Generate the SPMF-GoKrimp input file.

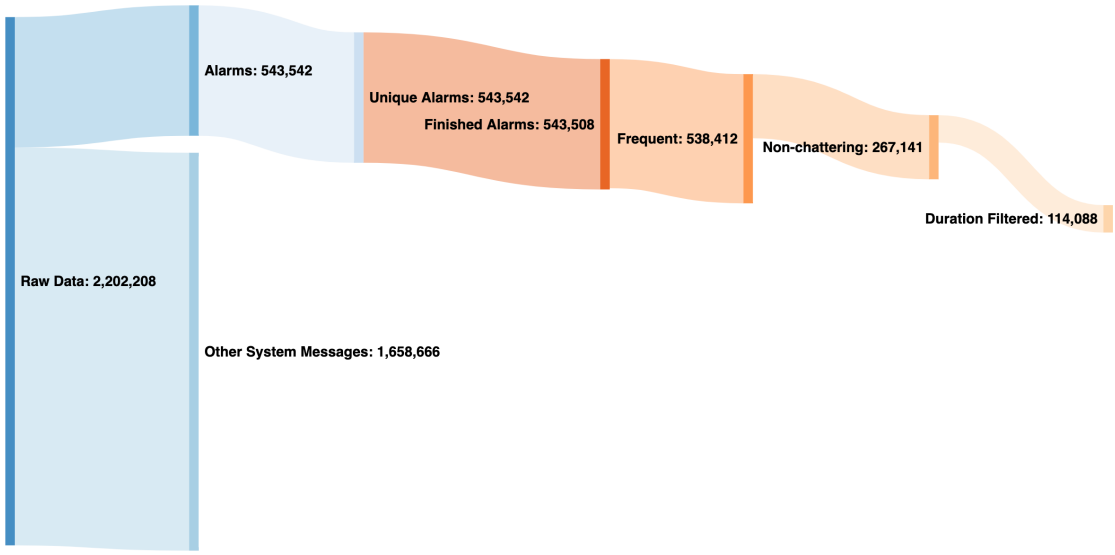


Figure 4.5: The effect of each filtering condition on the dataset size.

The GoKrimp algorithm was executed using the SPMF package invoking the GoKrimp algorithm via command line command having an input of 30,846 sequences, from which the GoKrimp algorithm identified 118 frequent sequence patterns. The run length of the compression step was 844 seconds on a commodity MacBook Pro 2018 laptop (with 2,3 GHz Quad-Core Intel Core i5 CPU). The first few rows of the output are presented in Table 4.2.

Scanning through the frequent sequences we might find interesting ones like: “10006 10006” where the same alarm repeats. It is tempting to label it as a sequence that the chatter index filter did not eliminate, but if we check the occurrences of the sequence more carefully, we will see that its support is 194 and the chatter index for this given alarm is well below the threshold suggesting that the reason of having recurring alarms showing up might be something different.

4.3.1 Evaluation of the proposed predictive method

This section provides the performance evaluation of the proposed alarm progression prediction method. The results of predicting the next possible alarm with the calculated probabilities to the [’10022’, ’10023’, ’10005’, ’10024’, ’10025’, ’10026’, ’10027’, ’10028’, ’10029’, ’10030’, ’10029’, ’10031’, ’10032’, ’10033’] sequence is presented in Table 4.3. This step may be repeated arbitrarily times minding that the further we predict from measured data, the less certain we can assume the prediction.

Sequence	Compression Contribution	$supp(C_k)$
10024 10025	15579.092	0.938
10019 10020	8329.556	0.423
10685 10686 10698 10635 10702 10699 10687 10700 10688 10689 10610 10611 10612 10613 10614 10615 10616 10617 10618 10619	6516.723	0.550
10072 10073	3844.644	1.000
10105 10106 10107	3521.141	0.745
10193 10195 10198 10199	2407.887	0.990
10108 10110 10111	2294.050	0.360
10665 10720	2234.178	0.620
10121 10146	1939.403	0.435
10089 10090 10098 10100	1874.949	0.924
10040 10071	1839.222	0.421
10074 10075	1673.417	0.816
10610 10611 10612 10613 10614 10615 10616 10617 10618 10619	1617.523	0.926
10174 10176 10174	1586.388	0.846
10068 10186 10187	2235.531	0.229
10051 10053 10054 10051	1471.014	0.743
10052 10050	1434.265	0.876
10106 10105	1419.308	0.564
10040 10035	1336.292	0.391
10172 10173	1253.315	0.914
10095 10097 10101 10102	1196.325	0.606
10006 10006	1076.289	1.000

Table 4.2: The first seqences returned by GoKrimp algorithm, their contribution to compression and the calculated support

Similarity Score	End index	Sequence	Next Sequence	Confidence
2.0	4	10024	10025	0.938398
2.0	5	10025	10024	0.042532

Table 4.3: Prediction results calculated for sequence: ['10022', '10023', '10005', '10024', '10025', '10026', '10027', '10028', '10029', '10030', '10029', '10031', '10032', '10033']

The prediction in Table 4.3 contains the following data:

1. Similarity Score: is the similarity score returned by the pairwise2 algorithm with the following parameters: $2, -1, -0.5, -0.1$ which stand for match/mismatch scores of $2/-1$ and gap penalties (open/extend) of $-0.5/-0.1$. The lower the SimilarityScore is the less similar the sequences are.

2. End index is the index of the last found sequence in the real-time sequence. This position describes how early the known pattern has been recognized in

the real-time alarm sequence. The smaller this number is, the earlier the algorithm has identified the possible operational states.

3. Sequence shows what sequence fragment has been recognized.

4. Next Sequence shows the predicted alarm.

5. Confidence shows the probability of Sequence being followed by Next Sequence. It is calculated according to equation 2.3.

The content of Table 4.3. reads as follows: a known pattern – '10024' – has been found for sequence ['10022', '10023', '10005', '10024', '10025', '10026', '10027', '10028', '10029', '10030', '10029', '10031', '10032', '10033'] – in the compressed sequence database – with a similarity score of two (that is considered to be low). The pattern ends at position four meaning that it is located at the beginning of the alarm sequence, hence the sequence has been recognized early. There is a 0.938398 probability that alarm '10024' is going to be followed by alarm '10025' – as is the case in our sequence. Reading the second row we can determine that the prediction is that alarm '10025' may be followed by '10024' with a probability of 0.042532. Checking the original sequence, we can say that alarm '10025' was not followed by '10024' even if we consider gaps. However, this prediction seems to be a false one, the probability assigned to it is also very low which is in line with the capabilities of the model.

To explain the gap handling of the algorithm, the prediction for sequence ['10002', '10003', '10002', '10004', '10008', '10053'] is presented in Table 4.4.

Similarity Score	End index	Sequence	Next Sequence	Confidence
2.0	2	10003	10053	0.260000
2.0	4	10004	10008	0.207792

Table 4.4: Prediction results calculated for sequence: ['10002', '10003', '10002', '10004']

While the second row of Table 4.4. can be easily interpreted based on the description applied for Table 4.3., the first row needs more explanation. In this case, it is predicted that alarm '10003' will be followed by '10053' in the sequence with a probability of 0.26. This is a correct prediction as the algorithm does allow gaps in the sequences, it does not restrict the prediction to be valid for the next alarm only. In the case of the first row, alarms '10003' and '10053' are following each other in the correct order having '10002', '10004', and '10008' intercalated in between '10003' and '10053'. This is an intended behavior of the system as in reality the timing of sensor reads is superposed on the change of the system, and it can happen that other unrelated alarms – even from other assets – do intercalate in between the alarms of interest.

The examples presented above were used to explain the basic behavior of the algorithm, but both of them were making a prediction based on a match for a single alarm – the sequence section did contain only one alarm – while in reality patterns might be formed from multiple alarms. Therefore, a score threshold setting has been introduced into the algorithm, with which it is possible to rule out matches exhibiting small similarity scores for predictions like the ones shown in the example.

In the case of every method where prediction has involved the question of the applicability and precision of the method arises. To address these questions, cross-validation of the method on the data used for compression is presented. The measurement has been done using python code. All sequences (30846 pieces) of the Cleared Sequence DB were fed into the model as if they were real-time sequences that need predictions.

In the Cleared Sequence DB, 11555 sequences out of 30846 were composed of a single alarm, therefore predictions on these sequences were skipped. From the 19,291 remaining sequences in the case of 5434, no prediction was made by the method. In 13857 cases the algorithm did recognize at least one sequence pattern and made a prediction(s). As discussed earlier the predictions are made based on confidence values and one sequence may have multiple possible predictions, therefore the number of false predictions plus the number of true predictions will be bigger or equal to 13857. This test resulted in 7354 (43,03%) true predictions and 9737 (56,97%) false predictions.

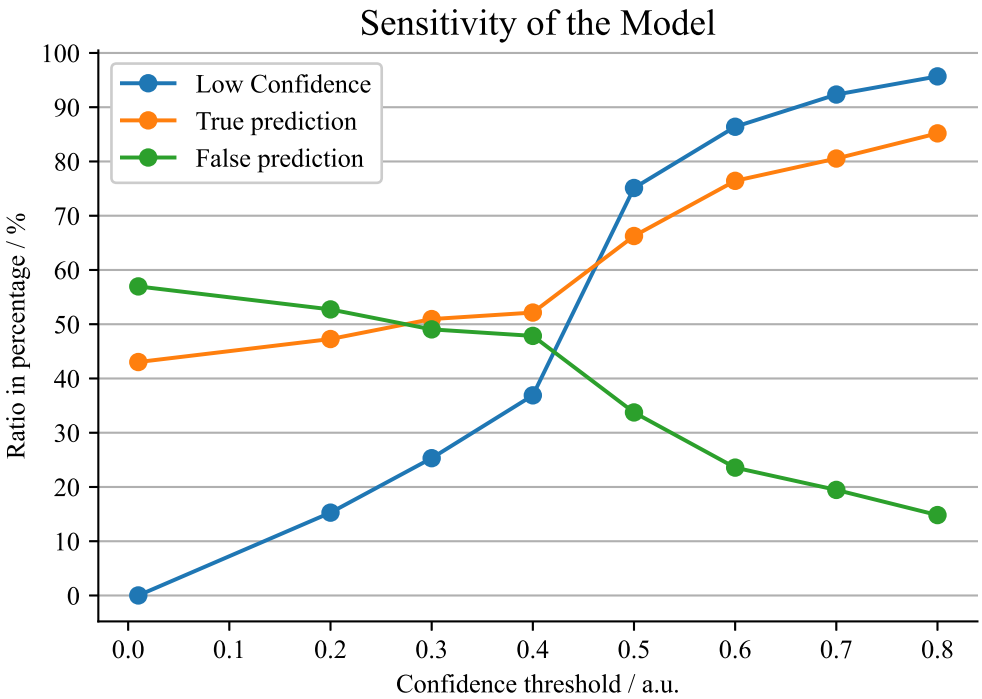


Figure 4.6: The sensitivity of the model. The ratio of filtered-out predictions (because of low confidence), true predictions, and false predictions at different confidence thresholds.

It is worth mentioning the number (5434) of real-time sequences to which the algorithm could not return any prediction as it can be interpreted also as a measure of the applicability of the suggested method. The reasons for this number might be manifold, on one hand, side it shows us the usability and limitations of sequence compression methods in alarm and alarm flood handling algorithms, on the other hand, it also describes the alarm management system, as the fewer patterns can be recognized, the more random the alarm sequences are, the less deterministic and accurate the system is. Some predictions may have low confidence, therefore, a filter

was introduced that will allow predictions to be considered if they are above a predefined threshold. The validation measurement was repeated for different confidence threshold values. The sensitivity of the method can be seen in Figure 4.6. We can see that there is a steep change in the ratio between 0.4 and 0.5 confidence threshold. Similarly, we can also see the ratio of the true and false predictions amongst the predictions that pass the confidence threshold. It is visible from the graph, that the ratio of true and false predictions reaches 50% - 50% around a confidence threshold of 0.28.

4.4 Conclusion

Proper handling of industrial alarms, especially reducing the galore of alarms to the relevant ones is of huge interest in the past decades as the number of sensors installed on assets, and therefore the number of alarms grows magnitudes faster than what the human operators can handle. Multiple methods were presented to tackle a specific aspect of these alarms like reducing alarm floods, analyzing logs for relevant sequences, and many others. While these methods enhance the information content of the alarm, they still miss to convey the most important information to the operators: the operational state of the industrial asset. A state-based alarm management system can solve this issue, however, it requires deep process-relevant knowledge and first principles. In the case of rare events, like not typical malfunctions, or if the mentioned knowledge is not fully available, a different approach is needed. Our work aims to answer this challenge in the form of a data-driven state-based alarm management system. A novel method blending a sequence compression algorithm and a sequence alignment algorithm has been proposed to recognize in real-time the operational states of industrial assets based on raw alarms and predict the progression of the system based on historical data. The method has been examined and characterized using real-life data originating from a delayed coker, and its usability and limitations have been determined. The results show that the method has a very effective pattern mining capability that extended with the sequence alignment method can recognize an operational state just after a few typical alarms and match it with historical patterns in less than a second. High-confidence predictions could be obtained easily. It has to be noted, as it is a data-driven type approach, it may consume more preparation time from a data processing point of view compared to first principle-based solutions.

The method presented in the current study heavily relies on the patterns identified in the historical data of the given plant. Fortunately, these patterns may change over time as the operational conditions of the plant might change including, but not limited to seasonal changes, quality of raw material, changes in technology, or even enhancements and development of parts of the plant. All these changes call for an easy method to measure the efficacy of the method and update it if it falls out of the optimal range. During an operational application of the method described in the present study, a vast amount of real-life data may be available and analyzed. Therefore, Key Performance Indicators (KPIs) have been suggested to design an automatic evaluation method triggering automatic updates to the Compressed Sequence DB incorporating new information like newer alarms, and operator feedback. An automatic evaluation may be integrated into the method presented that triggers an adjustment as soon considerate deviations are observed from the optimal KPIs.

These updates and corrections may be done very easily with high frequency as only the update of the Compressed Sequence DB is needed.

Integrating the algorithm proposed into industrial software packages opens up the possibility to adjust the alarms transmitted to the operators according to the operational state of the assets reducing the negative effect of irrelevant alarms. Integrating, operating, and maintaining a new module in an industrial IT environment is not an easy task, and has to be thought through. Therefore, the environment used for execution has been containerized and may be available as a docker image. The algorithm is parameter-free making it widely usable - with no further customization needed - for a multitude of industrial assets, alarm management systems, and Decision Support Systems (DSSs) present in the industry.

Chapter 5

Frequent pattern mining-based log file partitioning

5.1 Introduction

As discussed in the Introduction, industrial log files may contain parallel processes, which have to be separated. This phase of process exploration assumes that the traces have already been identified. In the proposed method, two steps are followed:

1. the search for traces where a specific group or sequence of events occurs, and only analyze these,
2. before filtering out infrequent events from the remaining traces.

Although many articles discuss pre-processing, only a handful deal with pattern-based or embedded techniques [92]. This work highlights the benefits and possible applications of a novel, iterative log-file pre-processing method that adds measures for evaluation. The applicability of the method is also emphasized in alarm management systems, which may be a novel research direction in industrial safety management. The main contributions of this chapter are the following:

- To identify those events that coincide (assuming they belong to the same sub-process) and perform frequent itemset and sequential pattern mining on the original *log*, the basis of process mining.
- To pick and filter out the relevant itemsets and sequences. The co-occurrence of the events is represented on a heat map.
- A three-step method is suggested. Firstly, only those traces are kept where the identified set of events is represented. Secondly, traces are chosen where the targeted sequence patterns could be found. Finally, infrequent events are removed from the sequence-filtered traces. The sub-logs gained in this step can be process mined directly. How many steps are needed depends mainly on the number of sub-processes. The more sub-processes present, the stricter filtering must be.
- The method is implemented in the alarm management system of an industrial plant. Compact, goal-oriented sub-process-models are identified from all three

kinds of filtered logs (sub-logs) using the *Heuristic Miner* algorithm. The results show a reasonable degree of process selectivity. The method allows more targeted process-model filtering than the available tools offered by *Heuristic Miner* (minimum activity count or minimum DFG) alone. Moreover, it provides the most benefits in cases where the hierarchical structure of the log file is inappropriate.

- Performance metrics are defined and evaluated. The metrics-based comparison of the gained models proves the applicability and effectiveness of the method.
- The conditions and relevant parameters of the sequential pattern mining algorithms are identified. A summary of the advantages and limitations of the suggested method is also provided.

The remainder of the chapter is organized as follows. In Section 5.2, the proposed frequent-pattern-mining-based method is discussed in details. Section 5.2.2 gives an overview of related works regarding the combined application of process mining and frequent pattern mining, highlighting the novelty of the proposed method. In Section 5.3, the proposed method is used on the alarm management system of an industrial Hydrofluoric Acid Alkylation (HFA) plant. The frequent pattern mining-based process models are compared with the one gained from the original log file, and performance analysis of the mined process models is also provided. Section 5.4 consists of the discussion and a proposal for future research directions.

5.2 The concept of frequent pattern-based log-file partitioning

This work aims to support the analysis of historical event data when the *log* file consists of events collected from more than one *process*. The *case-process* relationship might be unclear, or, in the worst-case scenario, even the *event-case* pairing can be controversial. One solution is to identify frequent patterns among the events to create *sub-log*s, from which the targeted *sub-process-model*s can be generated (Figure 5.1).

One of the most critical questions is identifying the relevant events that describe the processes. The tool used depends on whether key events or event chains are needed. *Frequent Itemset Mining* and *Frequent Sequential Pattern Mining* are powerful tools to support the discovery of independent and compact process models by identifying dominant event groups. Initially, the selection from frequent patterns is based on their support, that is, the number of their occurrence or from any formerly defined key attributes. With the discovery of each new sub-process, more information for selecting the next itemset will become available, and better models will be obtained. This loop must be repeated until the models that satisfy our previously determined demands are formulated (Figure 5.2).

The need to identify and separate sub-processes is not novel. One way to achieve this is implementing the OLAP (Online Analytical Processing) method on event data, e.g., log files, which is referred to as the process cube approach [93]. Process cube operations like *slice* (Figure 5.3a), *dice* (Figure 5.3b), *drill-up* and *roll-down* (Figure 5.3c) change the granularity or size of a dimension of our log file, thereby

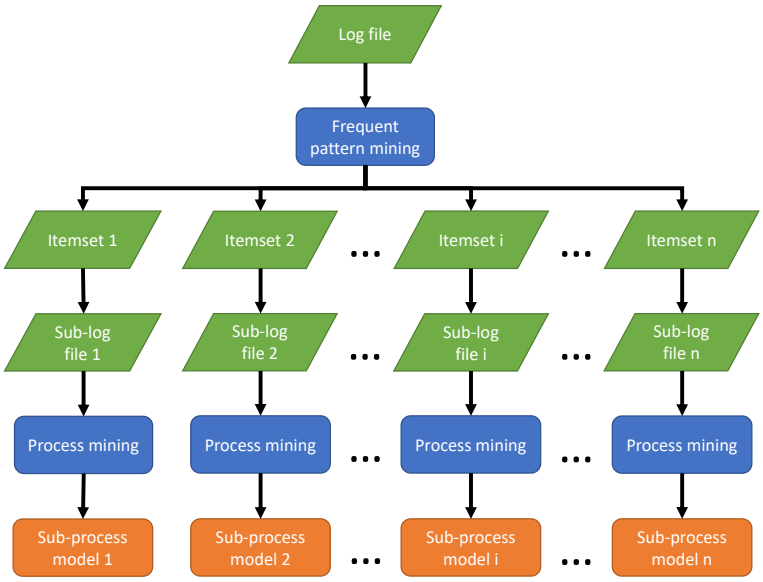


Figure 5.1: The model of the proposed frequent pattern mining-based log preparation method that enables the targeted process mining of complex processes.

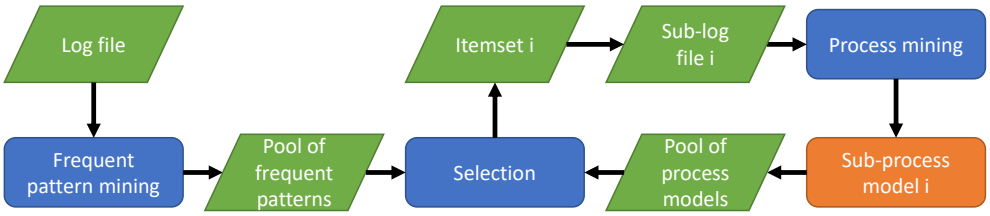
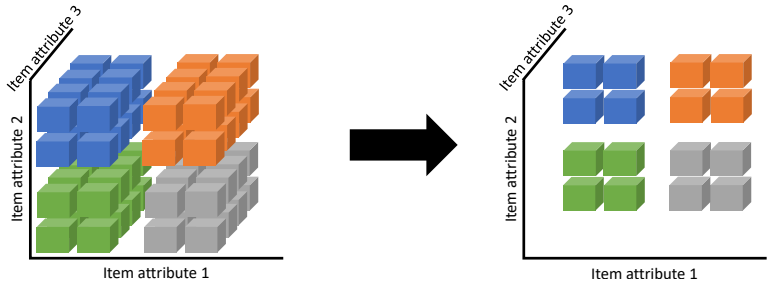


Figure 5.2: The identification method of the most compact process-models.

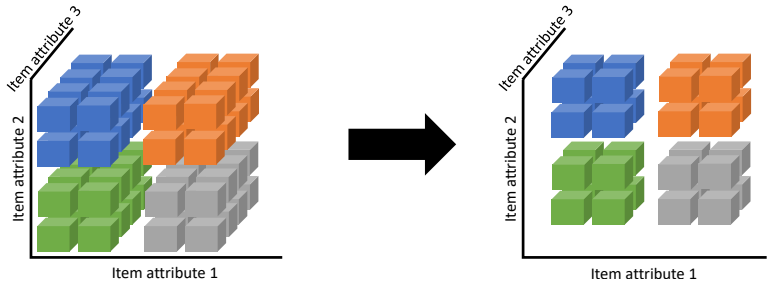
generating sub-logs, but only existing event connections are taken into consideration, and all events are kept within the actual view of the process cube.

In cases where log files are not well structured, parallel processes are hard to find. A Hybrid Feature Set Clustering method was applied to generate trace clusters, which is an excellent way to identify the log data related to sub-processes [94]. A shortcoming of these methods is that they suppose that all events in the log file are part of a process, which is not valid in special cases. Log files containing events from processes where the order or number of actions is not fixed exist. For example, in alarm management systems, since operator actions triggered by an alarm event are not handled equally by the different operators, unnecessary events can be present in the set of actions. Those extra actions behave as noise in our system, and their removal can result in a more accurate process model.

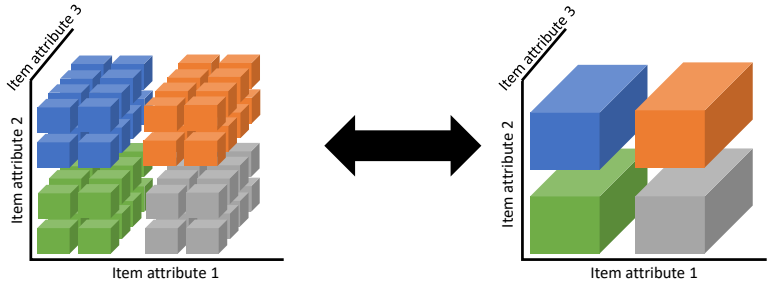
The essence of the proposed method is that the log file is not split according to the predefined attributes of events, e.g., time, shift, plant, equipment, etc., but in line with their co-occurrence, i.e., frequent process patterns. Since the added connection layer of the items (events) is the 4th dimension of the process cube, infrequent process elements are also eliminated, which is an additional function compared to its traditional operations (Figure 5.4).



(a) Slice operation



(b) Dice operation



(c) Drill-up and Roll-down operations

Figure 5.3: Process-Cube operations

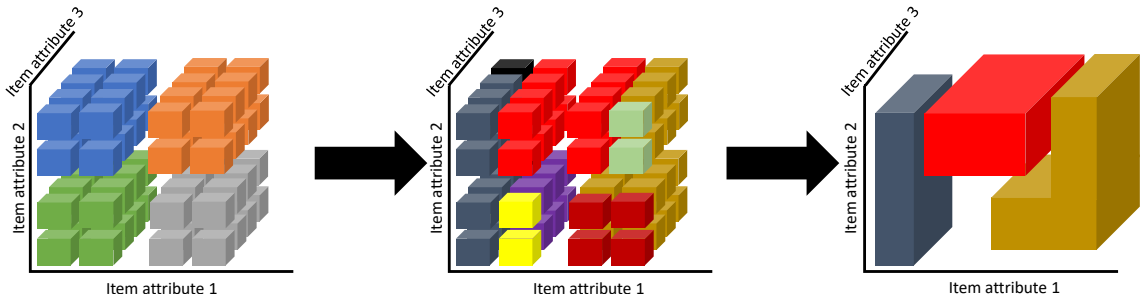


Figure 5.4: The proposed log file partitioning method

5.2.1 Description of the sequence pattern-based log file partitioning method

In this section, the concept of the proposed method is presented and compared to traditional process-cube operations. Secondly, the connection between Frequent Itemset Mining (FIM) and Process Mining (PM) is discussed based on their definitions, corroborating the applicability of the method concerning the process mining of poorly structured log files. As frequent pattern mining and process mining techniques are widely known, this section focuses on their connection only. A deeper discussion of these techniques can be found in the placed citations.

The process mining tool used was the Heuristics Miner Algorithm, which explores the control-flow perspective of the process-model [95] and only considers the order of the events within a case, hence why it is a suitable tool for this itemset-based method. The Heuristics Miner Algorithm has an advantage over the α-miner, namely that it considers frequencies and can handle skipped activities [96]. The log analysis is based on causal dependencies, that is, the so-called *Dependency Graph*, e.g. since other events always follow some events, it can be assumed that a dependency relationship exists between them. The calculated *Dependency Measure* values (Equation 2.9) are used by the miner algorithm to construct the process-model of the log. It is possible to limit the minimum dependency threshold to retain only edges in the dependency graph with specific dependency relations. However, this step may not be necessary if the log has been filtered using the proposed method, as it already only contains events with strong dependency relationships. Note that frequent itemset/sequence-based log filtering is a more precise and targeted approach.

The algorithmic description of the method is presented in Algorithm 3.

Algorithm 3

Require: L (log file),E (set of events), Method (itemset-based or sequential pattern-based),
Non-frequent filtering (True or False)
PATTERN MINING
M:{*minsup,max gap,min pattern length,max pattern length*}
if Method is *itemset-based* **then**
$\{I_{freq}\} \leftarrow$ FIM(L,M) ▷ I_{freq} set of frequent itemsets
heat map $\leftarrow \{I_{freq}\}$
else if Method is *sequential pattern-based* **then**
$\{\phi_{freq}\} \leftarrow$ FSPM(L,M) ▷ ϕ_{freq} set of frequent sequential patterns
heat map $\leftarrow \{\phi_{freq}\}$
end if
SELECTION
$e^{filt} \leftarrow$ heat map ▷ e^{filt} set of filtering events
PROCESS MINING
$S^{filt} \leftarrow e^{filt}$ ▷ S^{filt} set of traces containing the filtering events
if *Non-frequent filtering* is `False` **then**
Sub-log file $\leftarrow S^{filt}$
else if *Non-frequent filtering* is `True` **then**
$e^{infreq} =$ E - e^{freq} ▷ e^{infreq} set of infrequent events
$e^{S^{filt}} = e^{S^{filt}}$ - e^{infreq} ▷ $e^{S^{filt}}$ set of events in the selected traces
Sub-log file $\leftarrow S^{filt}$
end if
Heuristic Miner $\leftarrow$ Sub-log file
Sub-process-model $\leftarrow$ Heuristic Miner
CHECKING
if Sub-process-model conformance is `False` **then**
`return to` **SELECTION** or **PATTERN MINING**
end if
Ensure: Sub-process-model ▷

The complexity of the method depends mainly on the number of different events ($|E|$) stored in the log (L). Process mining the original log file has a complexity of $O_P = O(|L|, |E|^2)$ [95]. After the partitioning, the number of sub-logs is R, the number of traces in the i-th sub-log is $|R_i|$, and the number of different events in the i-th sublog is $|E^{R_i}|$. The complexity of the proposed method is the combination of the sequence mining complexity (O_S) and the process mining complexity of the reduced log (O_{P^R}).

$$O_S = O(|L| \times z) + O(|L| \times |E|) + O(2^{|E|-1}) \times O(|L|),$$
$$O_{P^R} = \sum_{i=1}^{R} O(|R_i|, |E^{R_i}|^2), \tag{5.1}$$

where $z = \frac{\sum_{i=1}^{N} |T_i|}{|L|}$ is the average length of a trace, $|T_i|$ is the number of events in the i-th trace and N is the number of traces stored in L, so

$$O_S = O(\sum_{i=1}^{N} |T_i|) + O(|L| \times |E|) + O(2^{|E|-1}) \times O(|L|). \tag{5.2}$$

The additional complexity of the proposed method compared to the direct process mining of the log file can be expressed as: $O_P - (O_S + O_{P^R})$. It is clear, that the complexity of the method increases exponentially with $|E|$ due to the task of sequence mining pointing to the importance of filtering out the irrelevant events already at the beginning of the analysis.

The selected pattern mining method depends on the data content of the log file. If only several sub-processes are stored, or the number of stored events is relatively low, frequent itemset mining should be chosen. If there are a lot of sub-processes with significant overlaps, and the amount of stored data is big, the use of frequent sequential pattern mining is recommended. The minimum support value is a key point, it has a high impact on the Pattern Mining step, this way on the results of the method. There is no rule of thumb for setting this value, as every log file is different regarding its size and stored processes. The goal is to choose a value, where the amount of found patterns is enough to take it as representative for the log file. Too many patterns may increase the processing time without adding a significant amount of information. There is no strict rule of minimum support value determination, but the occurrence frequency of the events of interest can be helpful. Take Frequent Sequence Pattern Mining into consideration. First, mining is done with a very low minimum support value to have a lot of sequence patterns. After that, the support value of the sequence of interest ($sup(\Phi)$) is calculated. The chosen minimum support value of the second round of mining should be one order lower than the calculated support value (for example 10% of it). Based on the results, the value can be fine-tuned.

The resulting pattern matrix can be visualized with a heat map to select easily the events of interest that drive the filtering step. If the size of the remaining log file demands it, additional filtering can be performed by deleting the non-frequent events from the sub-log file. Based on the evaluation of the gained sub-process model, the Selection or even the Pattern Mining step should be repeated.

5.2.2 Related works

Using the techniques mentioned above, an interactive process exploration tool similar to the one introduced by Vogelsang, Rinderle-Ma, and Appelrath [97] can be developed. Although their work provides a good framework concerning targeted process mining tasks, creating sub-log files may require a more detailed and goal-oriented approach in some cases. The tools used to partition the log file were frequent itemset and sequential pattern mining. With the help of the identified frequent itemsets and sequences, traces can be grouped, and infrequent (irrelevant) events and sequences can be removed. Sani, Zelst, and van der Aalst [22] proposed a sequence mining-based filtering solution to detect and remove outliers, while Zhu et al. [98] used an improved PrefixSpan Algorithm to discover alarm signal patterns. Frequent itemset mining was combined with k-means algorithm-based clustering of objects to improve business intelligence [99]. The need for log-file partitioning is not novel; a trace clustering method to explore simple process models from complex logs has been suggested [100], while other researchers have performed slice operations on process cubes [101]. Sequence-supported process discovery was used before, but mainly in student behavior analysis [102] or in healthcare-related solutions [103].

The methods above are all useful from specific perspectives of process-model discovery tasks. However, real-life log files can be far from ideal regarding process selectivity, complicating process mining. Although existing methods filter the log file by starting events or other specified attributes, they assume a thorough knowledge of the processes. The suggested method tries to rise to this challenge when a sufficient understanding of the system is unavailable.

This work aims to prove that goal-oriented log-file transformation-based process mining is more effective than a complex process model segmentation. Pre-processing is essential in process mining tasks, especially concerning process-model discovery. In a recent review [92], two main techniques were identified:

- *Transformation techniques* are mainly filtering or time-based. Filtering-based solutions focus on the removal of abnormal events [104] [105], while time-based ones attempt to deal with quality issues related to the timestamps in the log file [106] [107].

- *Detection and visualization techniques* are meant to handle outlier events by identifying and grouping them. Two main methods are known, namely clustering and pattern-based. Clustering aims to identify event-level deviations [108] [109], while pattern-based tools work on the event level of groups, e.g., sequences [110] [111].

The techniques mentioned above can be mixed to resolve special issues concerning process discovery, providing embedded solutions. The suggested method can be considered one of them but follows a different approach. Instead of focusing on outlier detection, it searches and identifies frequent patterns to cluster the traces of the various sub-processes and simultaneously remove infrequent behavior. The common goal of the discussed techniques - including ours - is to design more explicit and more understandable models (Figure 5.5).

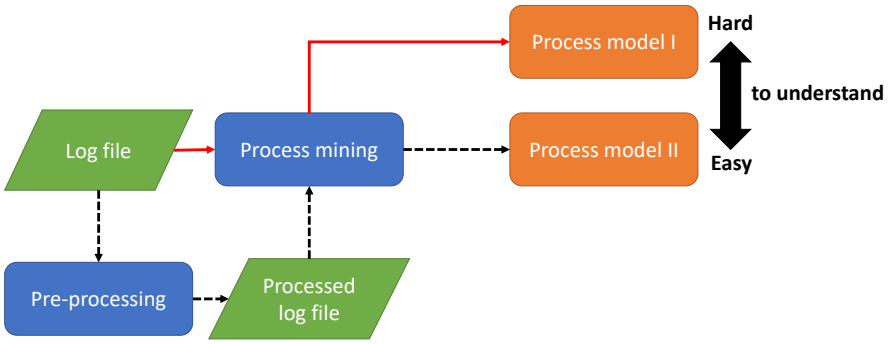


Figure 5.5: The importance of pre-processing the log file.

5.3 Application of the method to the alarm log of an industrial site

After a brief overview of alarm management systems, this Section discusses the method's applicability. The input data was the log file of the alarm management system of a Hydrofluoric Acid Alkylation (HFA) plant. The results proved that the proposed method effectively gained targeted sub-process models.

5.3.1 Alarm management systems

Alarm management refers to the effective design, implementation, operation, and maintenance of industrial manufacturing/process plant alarms. Alarm management is necessary for a process-plant environment controlled by an operator using a control system such as a Distributed Control System (DCS) or a Programmable Logic Controller (PLC) [112]. The files of industrial alarms & event logs are usually composed of timestamped events of alarms, operator actions, system messages, and any further temporal information. Additional information can help to determine which sensor generates the alarm in the process and the priority of that alarm, e.g., the location of the event, that is, in which part of the plant/organization it occurred. Each event that occurs is labeled with a tag name. Every alarm and operator action can be considered to be the state of the process. Alarm management systems generate huge log files that consist of many variables and collect information from several parts of the plant. Dörgő et al. [113] suggested performance measurement approaches for alarm systems, highlighting the importance of data-driven alarm management. The plant consists of four production units and more than 400 tag names. The log file contains over 200,000 events that occur over four months. The existence of data from parallel processes is apparent, and the need for the suggested log file partitioning-based solution is confirmed.

Generally, alarm management aims to understand the effect of a particular event, an alarm signal, or an operator action. Frequent itemset and sequential pattern mining techniques are suitable solutions if we want to explore certain relationships between alarms and operator actions.

5.3.2 Application of the method

As an antecedent of this work, a process mining technique was performed on this log file, which identified several processes recorded in our single log file (Figure 5.6). The identifiers in the boxes consist of three parts with the general form X_Y_Z. X denotes the identifier of the tag, Y represents the part of the plant, and Z represents the type of event (A: alarm, O: Operator action, N: Return to normal).

It can be seen that several transparent processes exist. Moreover, some overlaps can also be identified. The events denoted by red boxes will be of importance later. As process mining tasks are cyclical, amendments must be made to the input. This amendment will be the frequent itemset- and frequent sequential pattern-based filtering of the original log file, creating goal-oriented sub-log files.

To be more specific, the algorithm AprioriClose from the SPMF toolkit was applied to search for closed itemsets. From the gained itemsets, key events were identified. Sub-log files were created using these events, and only traces containing them were collected. It should be noted that during the itemset mining, return-to-normal events were removed from the transactional database. The reason for this is that in every trace, every alarm event has its return-to-normal pair. Therefore, return-to-normal events are redundant pieces of information in terms of itemsets. Of course, they remained in the filtered traces prepared for process mining. The log files were converted into the XES format and the process mining tool used was the Heuristics Miner algorithm. All tasks were executed in *Python*.

As the mining algorithm requires numbers, event names were coded with numbers. The logic was as follows: numbers from 100 to 499 represent alarms, from 500 to 999 return-to-normal events, and from 1000 upwards operator actions. This way, it is easier to read the results, and events can be more targeted. There are options to have the names in the database, but with numbers, the selection can be made without preconceptions, thereby obtaining more objective results. Only itemsets with more than one element were kept, and their connection strength was visualized on a heat map (Figure 5.7). The darker a tile is, the greater the support an itemset has. Since a three-element itemset was present, two operator actions (1084+1085) were merged to include it in this figure.

Four sections can be seen on the heat map, separated by red lines. The top-left one represents alarm-alarm, the top-right one denotes alarm-operator action, the bottom-left one stands for operator action-alarm, and the bottom-right one refers to operator action-operator action connections. Based on this heat map, it is easy to pick the itemset [235,1084,1085] with original names [HLCHL9_CH_A, HFC17_CH_O, HFC18_CH_O]. This alarm indicates problems with the liquid level of the stripper. The operator actions HFC17_CH_O and HFC18_CH_O control the bottom inlet and the steam inlet of the stripper, respectively. The first filtering on traces containing these events resulted in the first sub-process-model (Figure 5.8). A minimum activity-count filtering of 100 was applied to the Heuristics Miner algorithm.

By analyzing the itemsets further, it can be seen that many contain a lot of two-item sets. The additional selection of itemsets depends on the set goals, that is, what information has to be extracted from the log data. Suppose that the aim of interest is operator actions related to alarms. In this case, at least one alarm event is needed so itemsets containing only operator actions (numbers above 1000) can be dismissed. Besides the itemsets holding events 235 and 1084/1085, the set

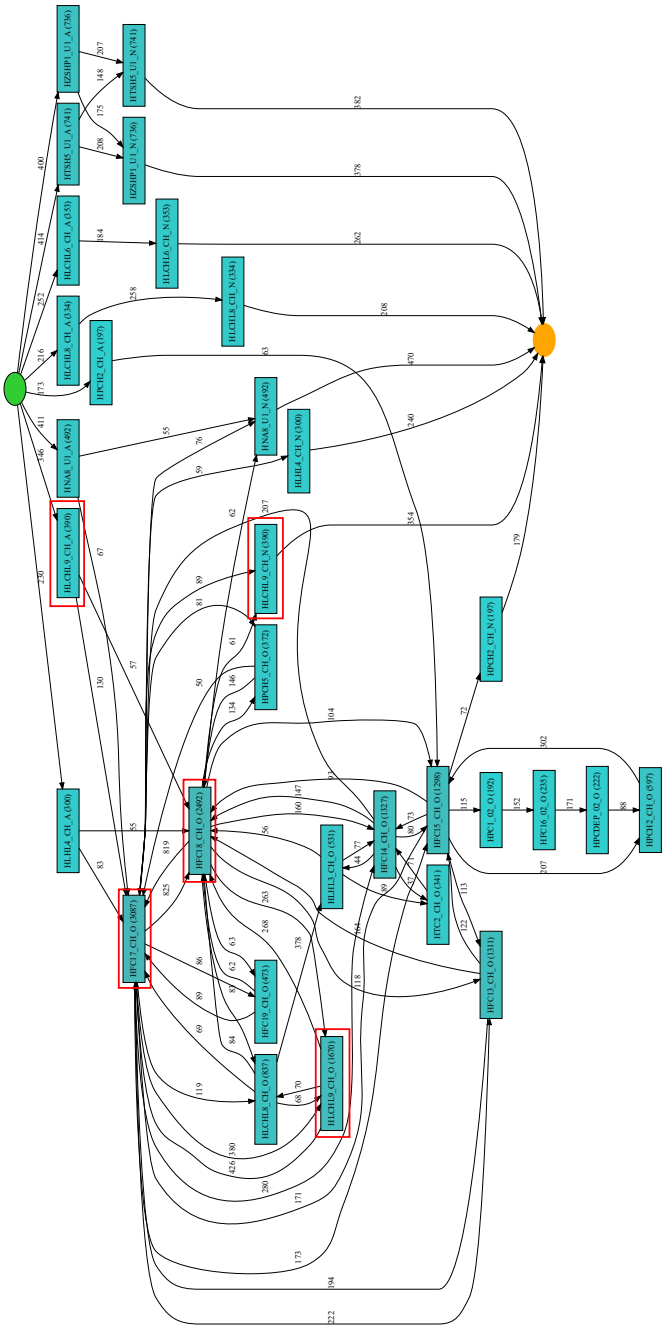


Figure 5.6: process-model from the original log file

[309,330] has a high support rate, and their original names are HTSH5_U1_A and HZSHP1_U1_A. These alarms are related to the problem with the same pump, so they must be checked as maybe they are redundant signals.

The discovered process model can be seen in Figure 5.9a (minimum activity count was set to 50). The model shows that these two alarms rarely require operator actions. Filtering on the frequency of the edges (minimum dfg parameter in the Heuristics Miner algorithm), the process model becomes very simple (Figure 5.9b). Therefore, these alarms are more or less independent of operator actions, so they must be treated differently.

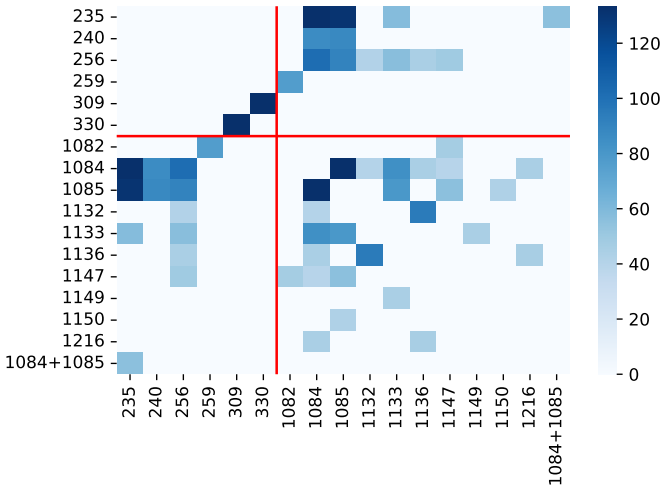


Figure 5.7: Heat map of frequent itemsets

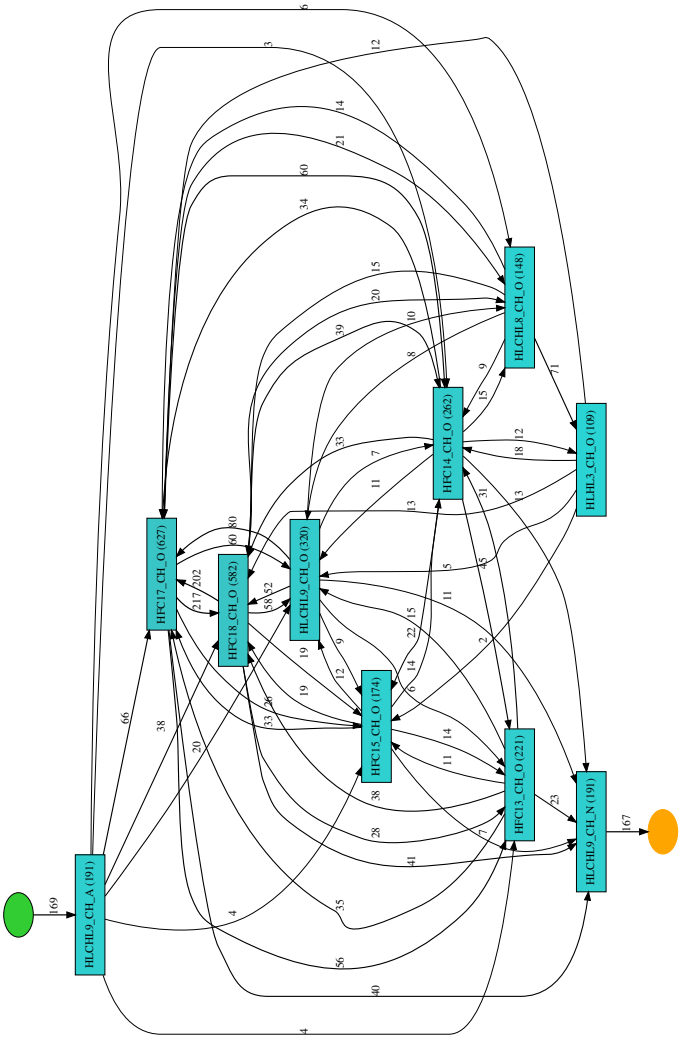
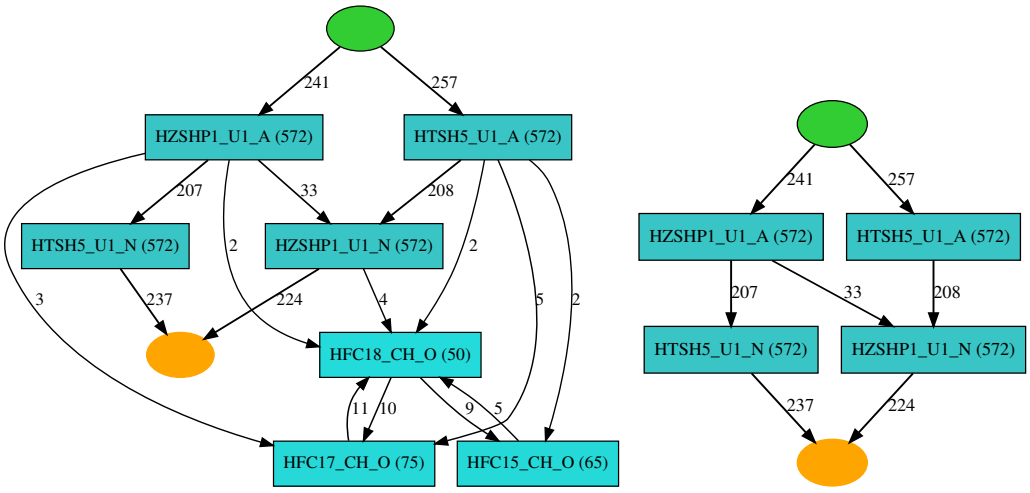


Figure 5.8: Sub-process-model 1

Sequences include information concerning time, as the events in sequences are ordered chronologically. Searching for sequential patterns instead of items results in more specific models. The ClaSP (Closed Sequential Patterns) algorithm was used



(a) Sub-process-model 2, filtered by the frequency of nodes

(b) Sub-process-model 2, filtered by the frequency of edges

Figure 5.9: Sub-process models based on frequent itemsets

with a minimum support value of 50%, and the results were also visualized by a heat map (Figure 5.10).

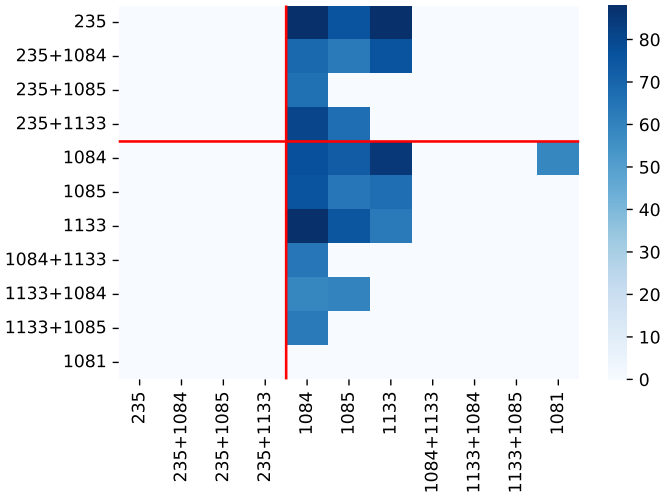
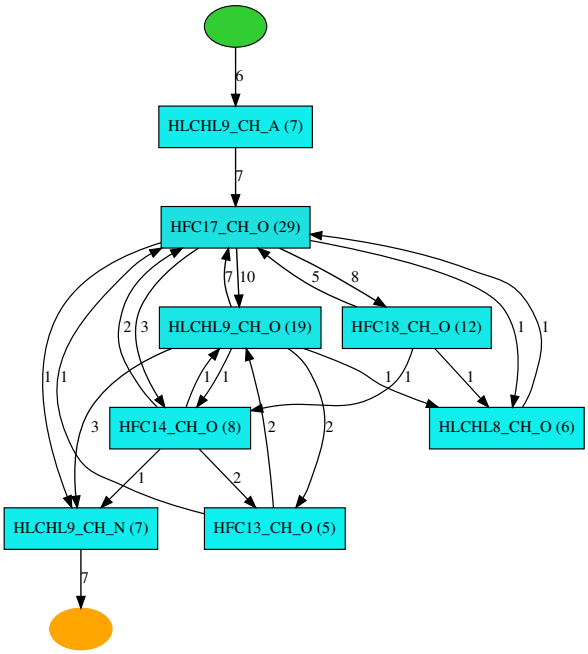


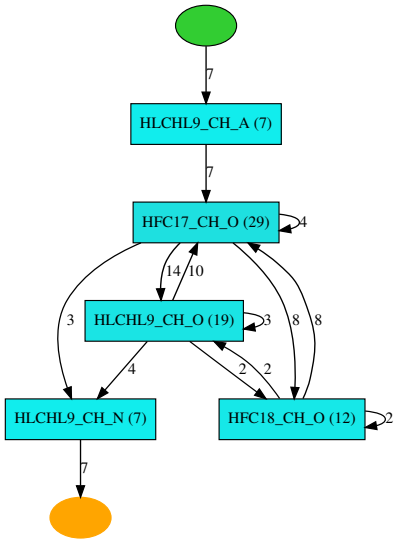
Figure 5.10: Heat map of frequent sequences

Based on this heat map, given that sequences [235,1084], [235,1133], [1084,1133], and [235+1084,1133] have high support, a strong connection exists between these three events. The first two items of the three-item-long sequences were merged (as they are sequences since the first two have to occur before the third) to visualize them in this figure. The minimum activity count was set at five, and the result was a quite compact process model (Figure 5.11a).

The final move is to remove the events regarded as infrequent or irrelevant, yielding the most specific and clear process model. In this figure, the advantages of this method are evident. Using the [235,1084,1133] sequence and deleting events that are not included in sets containing at least two items, Figure 5.11b is produced. These events are marked in Figure 5.6. It would be impossible to separate this sub-process from the original model due to the overlapping processes. If the number of



(a) Sub-process-model 3



(b) Sub-process-model 3 with event filtering

Figure 5.11: Sub-process models based on frequent sequences

transitions (edges) and occurrences of events (nodes) are checked, applying a filter to them would result in a significant loss of information. The events HFC17_CH_O and HFC18_CH_O occur in more sub-processes, as their *activity count* number is significantly higher. In Figure 5.6, event HLCHL9_CH_O appears to occur separately and does not belong to the HLCHL9_CH_O → HLCHL9_CH_N sub-process, which statement is incorrect. Without the targeted log-file partitioning, the gained process models would be much less precise.

5.3.3 Suggested performance metrics to evaluate the effectiveness of the proposed pre-processing method

An objective evaluation is essential to automate the method. This section discusses the measures to evaluate the quality of the discovered processes. Some of them are based on the relationship between the log file and the process model. Others can be calculated by handling the models as networks. These metrics have no dimensions and are numbers between 0 and 1. Their explanations are mentioned in the actual metric description.

The log file-process-model related measures are the following:

- Simplicity (Q_s): defines as how easily a human can understand the model. The numerical value can be calculated by comparing the size of the process tree with the number of activities contained [114], or as the weighted average arc degree of the model [115]. $Q_s \in [0, 1]$, a higher value indicates a more simple model.

- Replay fitness (Q_{rf}): indicates how much of the behavior in the log is admitted by the process-model. The bigger the fraction of the behaviour in the event

log that the model can replay is, the higher the fitness value the model has [116]. $Q_{rf} \in [0, 1]$, a higher value indicates a more fit model.

- Precision (Q_p): measures the deviation between the behaviour described by the log and by the process-model. Less precise models allow more behaviour that is not observed in the log-file [117]. $Q_p \in [0, 1]$, a higher value indicates a more precise model.

- Generalisation (Q_g): A model is *general*, when it represents behaviours that are not registered in the log, but are possible [115]. With other words, how often the elements of the model are used to reply the log [114]. $Q_g \in [0, 1]$, a higher value indicates a more general model.

- Soundness (Q_{so}): A model is considered *sound*, if every activity can participate in a process instance, it is ensured that the process always terminates properly and there are no dead transitions [118]. $Q_{so} \in \{False, True\}$.

Another metric can be how definitive the model is. An option is to check the ratio of those event pairs in the log that never occur in the same traces compared to the number of possible event pairs. A lower ratio indicates a more definitive process model. The result has to be deducted from one to obtain a more informative value (unless the best model equals zero, which may be confusing). This measure is denoted as the *Definitive index* ($Def \in [0, 1]$). The number of events that never co-occur can be gained from the *log skeleton* [119], denoted with E^q. If the number of events in the process model is n, then the number of possible event pairs is:

$$EP = n \times (n - 1), \tag{5.3}$$

and the Definitive index is:

$$Def = 1 - \frac{E^q}{EP}. \tag{5.4}$$

If the process models are interpreted as networks, the following network-theory metrics can be applied:

- Network density (d_i) [120]: the ratio of actual node connections to the number of all possible node connections in an **N** network. $d_i \in [0, 1]$, a higher value indicates a more complex model.

- Flow hierarchy (h) [121]: the fraction of edges that do not participate in cycles in a directed graph. $h \in [0, 1]$, a higher value indicates a more clean-cut model.

- Global efficiency ($E(G)$) [122]: the *efficiency* of a pair of nodes on a graph is the multiplicative inverse of the shortest-path distance between the nodes. The average *global efficiency* of a **G** graph is the average efficiency of all pairs of nodes. $E(G) \in [0, 1]$, a higher value indicates a more clean-cut model.

Table 5.1: Performance metrics.
FI: frequent itemset, **FSP**: frequent sequential pattern, **IER**: infrequent event removal. The results confirm the applicability of the proposed method.

Source log	Original log min/max/mean	Log partitioned by **FI** min/max/mean	Log partitioned by **FSP** min/max/mean	Log partitioned by **FSP+IER** min/max/mean
Log fitness	0.428/0.430/0.429	0.458/0.476/0.467	0.455/0.489/0.466	**0.891/0.902/0.898**
Trace fitness	0.299/0.302/0.300	0.539/0.547/0.543	0.535/0.556/0.542	**0.910/0.919/0.915**
Precision	0.460/0.469/0.463	0.358/0.380/0.368	0.337/0.399/0.366	**0.966/0.969/0.967**
Generalisation	**0.935/0.937/0.936**	0.869/0.893/0.878	0.867/0.906/0.883	0.881/0.886/0.884
Simplicity	0.455/0.455/0.455	0.463/0.474/0.472	0.463/0.486/0.474	**0.750/0.750/0.750**
Is sound	False	False	False	**True**
Definitive index	0.114/0.117/0.116	0.258/0.289/0.271	0.263/0.278/0.272	**1.000/1.000/1.000**
Network density	0.008/0.008/0.008	0.012/0.013/0.012	0.012/0.013/0.012	0.650/0.650/0.650
Flow hierarchy	0.036/0.045/0.039	0.017/0.019/0.017	0.016/0.02/0.018	**0.308/0.308/0.308**
Global efficiency	0.300/0.317/0.311	0.348/0.358/0.352	0.349/0.354/0.352	**0.850/0.850/0.850**

To test the robustness of the method, bootstrapping-like procedure has been applied where 90% of the data was sampled five times. The minimum, maximum, and mean of the metrics of different process models can be seen in Table 5.1.

The results prove the assumptions. The frequent itemset-based partitioning does not significantly increase the performance of the process model. However, it provides a more targeted representation of the process. Applying the frequent sequential pattern-based method seems essential to achieve a better performance. It can be seen in the case of *Fitness*, *Simplicity*, *Wiener index*, and *Flow hierarchy* values. Regarding the removal of infrequent events, a significant increase in *Precision*, *Definitive index*, *Global efficiency* and the *Soundness* of the discovered model is only secured by following this approach. *Generalisation* values decrease as the methods are applied, which is in line with the increase in *Precision* values. A model can simultaneously be *general* and *precise* if the behaviors described in the model are not stored in the log but are similar to those. *Network density* cannot be evaluated unambiguously, but still can provide useful information about the model.

In general, it can be stated that the suggested method is highly applicable for preprocessing log files that are not well structured or supporting process-mining tasks where knowledge of the system is insufficient. *Fitness, Precision*, and *Soundness* all increased, meaning that the gained models describe the behaviour of the real world industrial system better.

5.4 Discussion and conclusion

Complex systems may contain parallel processes. However, in most cases, event data of these processes are collected in one log file, which renders process mining a challenging task. This problem is well-known, and the solution is clear. Sub-log files have to be created to discover the targeted processes. Different tools can be used to generate these sub-log files, and the method heavily relies on the type of system to be analyzed. One potential issue can be that not every logged event is part of a "normal" process, irrelevant or unnecessary actions (originating from bad operational practices or an underqualified workforce) can be present.

The suggested method of filtering out these unnecessary events while simultaneously creating sub-logs combines frequent pattern mining and traditional process

cube operations. It was applied to a log file of an industrial Hydrofluoric Acid Alkylation plant. The resultant process models in the case study were evaluated using the introduced performance metrics. The results proved that the method is effective in partitioning log files, regrouping events for targeted process-discovery tasks, and handling the problem of parallel processes. The requirements, benefits, and limitations of the method are the following:

- Benefits
  - Sequential pattern mining and process mining use the same source. No extra preparation of the log-file is required.
  - The method is efficient from a computational demand point of view, as the size of the log-file to be processed has been significantly reduced.
  - Prior knowledge can be transferred to identify the relevant frequent patterns, facilitating iterative work. Process-relevant information can be efficiently included in process-mining.
  - The defined metrics enable an objective evaluation of the results. The evaluation step can facilitate the automation of the method.
- Requirements - Limitations
  - Although the method needs a pattern mining tool, this is not a critical issue as open-source tools are widely available.
  - A certain amount of knowledge is required to select the right pattern-mining algorithm. Suggestions are made regarding this topic.
  - Prior knowledge of the process is essential. Similarly to the *Knowledge Discovery Databases* (KDD) process-model [123], the iterative and interactive character of the method eases this issue.

One research direction to bring about further improvements is to consider the numeric values of controlled parameters. By clustering these potential values, an extra dimension can be added to the process variables, supporting a better understanding of the evolution of event sequences. Another research area is the application of network theories in the *Selection* step of the method (Figure 5.2). By considering process models as networks, network attributes can be interpreted as measurable values of sub-process models, extending the discussed performance metrics.

Chapter 6

Network-based visualisation of frequent sequences

6.1 Introduction

In this chapter I propose three visualisation methods that allow quick access to the context of the data without eliminating any relevant information - for example, relatively rare, but important events - during postprocessing. These methods are NBVFS-WG (Network-Based Visualisation of Frequent Sequences - Weighted Graph), NBVFS-CM (Network-Based Visualisation of Frequent Sequences - Confidence-Based Multidimensional scaling), and NBVFS-TM (Network-Based Visualisation of Frequent Sequences - Transaction-Based Multidimensional scaling).

The proposed method was developed in a Python environment, ensuring focused analysis through tailorability. Furthermore, specific solutions can be built on the basic principle of the method. The key idea is to interpret the connection of frequent sequences as a network, using sequence metrics as network attributes and as inputs for similarity measurement. This approach can be considered a network embedded with side information [124] and can be a powerful analysis tool that pairs with the temporal skeletonization of sequential data discussed in [125]. The characteristics of the networks provided represent the characteristics of the frequent sequences obtained from the data set. For example, the width of an edge can be proportional to the confidence value of the transition between the nodes that directly represent subsequent sequences. The comparison of the method with other existing solutions can be found in Table 6.1:

Table 6.1: Comparison of visualization techniques

Method	**Information content**	**Readability**	**Flexibility**
Individual representation[31]	+	/	+
Flow diagram[32]	+	+	-
Aggregated pattern[33]	+	+	++
Placement strategy[34]	+	+	++
Episode visualization[35]	/	/	++
NBVFS	+	+	++

The proposed method performs well with respect to the three requirements mentioned in Section 1.4.

- **Information content**: More content gives a better and more complete view of the system, but above a certain level, it causes indistinctness. Information can be the support values, the confidence values of sequences and their transitions, the relationship between connected and nonconnected sequences, like the parent-child relationship, similarity, or the group where the sequence belongs.
- **Readability**: The balance between information content and readability is a critical issue in data visualisation concepts, especially in the case of a large amount of data to process.
- **Flexibility**: From a data set, different knowledge extraction directions can be performed. Scalability and the opportunity to tailor the method to the actual task are essential. A good visualisation concept has filtering options to perform goal-oriented analysis tasks.

The application of MDS ensures that as much information as possible is kept; dimension reduction can be applied succesfully in solving high-dimensional problems, for example, to interpret large amount of association rules [126]. The network-like representation provides good readability. Finally, the open-source character gives excellent flexibility.

The contributions of this work are the following:

- the detailed process description of methods NBVFS-WG, NBVFS-CM, and NBVFS-TM is introduced, along with the theoretical background;
- the gained network-based visualizations of the frequent sequences are demonstrated and discussed;
- a summary of the proposed methods is provided with suggestions for future research directions.

The remainder of the chapter is organised as follows. First, the theoretical background and a detailed description of the proposed method are presented. Second, use cases are provided with two different data sources, along with a discussion of the results. In the end, the proposed method is summarised and future research directions are identified.

6.2 The proposed algorithms for the Network-based Visualisation of Frequent Sequences

This section discusses the proposed visualisation algorithms with a related theoretical background.

The method aims to provide an informative visualisation of the frequent patterns of sequences. The concept of the proposed visualisation process can be seen in Figure 6.1. The source is a data set where items are ordered and grouped in sequences in the form of transactions. The transactional database is processed with a sequence pattern mining algorithm, and the output is the set of frequent sequences. The frequent sequences are completed with several attributes, like confidence or parent sequence. These attributes form the basis of the visualisation process. Three ways were chosen to visualise the networks: confidence-based (NBVFS-WG and NBVFS-CM) and transaction-based (NBVFS-TM).

- **NBVFS-WN** (**N**etwork-**B**ased **V**isualization of **F**requent **S**equences - **W**eighted **N**etwork): this method uses a confidence-based adjacency matrix and the calculated metrics of the sequences. It results in a weighted network where those sequences are connected that are direct extensions of each other. The weight is the confidence value of the transition between the two connected sequences. This kind of visualisation helps to understand the conditions and consequences of occurring events.

- **NBVFS-CM** (**N**etwork-**B**ased **V**isualization of **F**requent **S**equences - **C**onfidence-based **M**DS projection): this method uses a similarity calculation, using transition confidence values for similarity measurement. The adjacency matrix must be enriched regarding the nondirectly connected sequences using transitive distance calculation. The positions of the nodes on the network are calculated with Multidimensional scaling, and the more similar the two sequences are, the closer they will be presented on the network. This kind of network representation contains more information about the relationship of sequences.

- **NBVFS-TM** (**N**etwork-**B**ased **V**isualization of **F**requent **S**equences - **T**ransaction-based **M**DS projection): the positions of the nodes are calculated with MDS identically to the NBVFS-CM method. The process is more or less the same, but the similarity calculation is based on the overlap of their common supporting transactions. This approach is closer to process mining, as transactions can be taken as traces and can provide feedback to the mining process.

6.2.1 Network visualisation of sequences

A data set must be prepared to contain all frequent sequences and all the necessary information describing the relationship between them. The goal is to visualise the connection network of the sequences by connecting the parent-child sequences, such as ϕ_i and ϕ_j, by using the attributes of the sequences as network parameters. In this

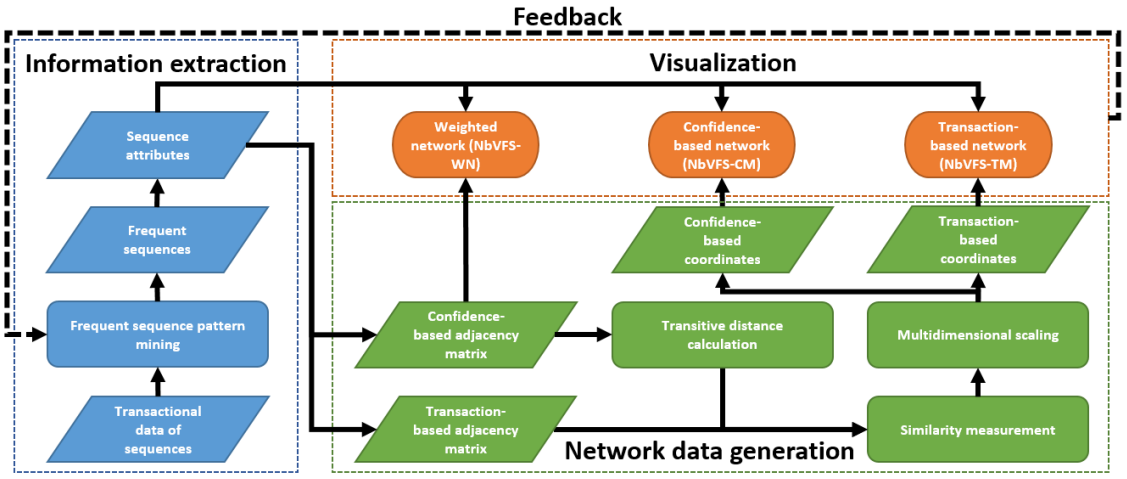


Figure 6.1: The concept of the visualization process. After the frequent sequences are produced from the data set, their attributes are calculated. These attributes are the inputs of the three visualization algorithms, NBVFS-WG, NBVFS-CM, and NBVFS-TM.

work, two visualisation methods were applied, a confidence-based and a similarity-based one.

Consider the sequences ϕ_i and ϕ_j as nodes, where ϕ_j as the extension of one event of ϕ_i, $\phi_j = (\phi_i \rightarrow e_j)$. $G(V, E)$ is a weighted directed network, where $V(G)$ is the set of nodes, $E(G)$ is the set of edges [127]. If $(\phi_i, \phi_j) \in E(G)$ (a directed edge from node ϕ_i to ϕ_j exists), the weight of the edge (ϕ_i, ϕ_j) is considered equal to the confidence value of the transition $(\phi_i \rightarrow \phi_j)$, and N is the number of nodes, then the $\phi_i\phi_j$ related value of the adjacency matrix $A = (a_{i,j})_{N\times N}$ is

$$a_{i,j} = Conf(\phi_i \rightarrow \phi_j). \tag{6.1}$$

All sequences are collected around their joint starting event in this weighted network. There are visualisation options to emphasise the attributes of the network. For example, the confidence of edges and nodes can be indicated by their width or the sequence length with colour, gaining a visually informative network. The limitation of this network is that it does not handle the connection between the different subnetworks. Its only organising principle is the direct connections between events, starting from their shared first event; the network is undirected.

The method NBVFS-CM discovers the relationship between non-directly connected sequences to create a directed network by calculating their similarity values. Let S^{ϕ_i} and S^{ϕ_j} be the sets of transactions containing ϕ_i and ϕ_j, respectively, the Jaccard similarity measurement can be used. The principle of the method is to connect the sequences with their direct extensions, weighted by the confidence value of the transition, resulting in a similarity value:

$$sim(\phi_i, \phi_j) = \frac{\mid S^{\phi_i} \cap S^{\phi_j} \mid}{\mid S^{\phi_i} \cup S^{\phi_j} \mid}. \tag{6.2}$$

Considering that ϕ_j is the one event extension of ϕ_i, we can state that $S^{\phi_j} \subseteq S^{\phi_i}$, so $\mid S^{\phi_i} \cap S^{\phi_j} \mid = \mid S^{\phi_j} \mid$ and $\mid S^{\phi_i} \cup S^{\phi_j} \mid = \mid S^{\phi_i} \mid$. This means that the similarity between two sequences is the ratio of the cardinality of their supporting transactions.

That is, the confidence of transition $(\phi_i \rightarrow \phi_j)$:

$$sim(\phi_i, \phi_j) = Conf(\phi_i \rightarrow \phi_j). \tag{6.3}$$

The adjacency matrix of the weighted network can be used to gain the similarity values directly. Still, it has to be enriched with values between nondirectly connected sequences.

It can be supposed that the 2-length and 4-length subsequences of a 5-length sequence are related. Transitive distance calculation is a proper tool for obtaining missing values. The transitive distance method is based on the similarity of nodes [128]. This method helps calculate the similarity between nodes not connected directly (non-neighbours). A sequence can be taken as the connected nodes of a network (in a directed way), where the sequences are the nodes. Consider the transition system $\phi_i \rightarrow \phi_j \rightarrow \phi_k$, where ϕ_j is the one-event extension of ϕ_i and ϕ_k is the one-event extension of ϕ_j. Although ϕ_i and ϕ_k are not connected directly, they are in relation through ϕ_j. Using equation 6.3 and the theory of transitive distance calculation,

$$sim(\phi_i, \phi_k) = sim(\phi_i, \phi_j) \times sim(\phi_j, \phi_k) = Conf(\phi_i \rightarrow \phi_j) \times Conf(\phi_j \rightarrow \phi_k). \tag{6.4}$$

This equation shows that the confidence value of the sequence transitions can be used to calculate the transitive distance-based similarity between nondirectly connected nodes. With this method, the visualisation of the network is enriched with nondirect connections, using multidimensional scaling (MDS) to define the relative position of the sequences on the network. The main principle of this approach is similar to hyperbolic embedding of networks [129], but uses different similarity measures.

In this work, *Metric (or classical)* MDS was used. MDS is a tool to visualise items in a high-dimensional feature space by mapping them to a low-dimensional data space (mainly 2D), based on the similarity of the items in the original space[130]. It discovers the hidden structure of the items by preserving similarity information, which is the pairwise distance between the items. Like other dimension reduction-based visualisation methods, MDS tries to minimise a stress function E, that is, using the square error cost.

$$E_{metricMDS} = \frac{1}{\sum_{i<j}^{N} d_{i,j}^{*2}} \sum_{i<j}^{N} (d_{i,j}^{*} - d_{i,j})^2. \tag{6.5}$$

In the proposed method, $d_{i,j} = 1 - sim(\phi_i, \phi_j)$ (the original distance between ϕ_i and ϕ_j), and $d_{i,j}^{*} = || \mathbf{y}_i - \mathbf{y}_j ||$, where $\mathbf{y}_i$ and $\mathbf{y}_j$ are the coordinates of sequences ϕ_i and ϕ_j in the visualised network. Adding the $\mathbf{Y}=[\mathbf{y}_1, ..., \mathbf{y}_i, ..., \mathbf{y}_N]$ vector of the sequence coordinates gained to the sequence attributes already collected, the network can be plotted. The visualisation options are similar to those of the weighted network. The support of the sequences can be represented by the size of the nodes, while the confidence of the succeeding sequence with the width of the edge connected to it.

The similarity of the two sequences can also be calculated based on the number of shared supporting transactions (NBVFS-TM). In this method, unlike NBVFS-CM, the similarity values cannot be obtained directly. Information about related

supporting transactions is needed, which is represented by an N × n matrix, where N=| ϕ_{freq} | and n=| $S^{\phi_{freq}}$ |, and $S^{\phi_{freq}}$ is the set of transactions that support frequent sequences. We get an adjacency matrix N × N filled with the number of common supporting transactions of the sequences. The Jaccard similarity can be calculated from this matrix, and the MDS-driven plotting of the network is identical to that of NBVFS-CM.

The three visualisation methods can be formalised as an algorithm now that the necessary theoretical background is discussed.

Algorithm 4 NBVFS

Input: T, M, N ▷ T is the set of transactions, M:{*minsup,max gap,min pattern length,max pattern length*},N is the number of frequent sequences
$\{\phi_{freq}\}, S^{\phi} \leftarrow$ FSPM(T,M)
$i \leftarrow 1$
while i ≤ N **do**
$j \leftarrow 1$
if method is *Confidence-based* **then** ▷ NBVFS-WN or NBVFS-CM
while j ≤ N **do**
$\mathbf{A} \leftarrow Conf(\phi_i \rightarrow \phi_j)$ ▷ $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{N \times N}$ is the adjacency matrix of ϕ_{freq}
if method is NBVFS-WN **then**
V(G), E(G), w ← **A** ▷ nodes (V), edges (E) and weights (w) of the network
else if method is NBVFS-CM **then**
$\mathbf{A}^* \leftarrow$ `TransDist`(**A**) ▷ Calculation of transitive distances
D = 1-$\mathbf{A}^*$ ▷ **D** is the dissimilarity matrix
Y(G) ← `MDS`(**D**) ▷ **Y** is the coordinate vector of nodes
V(G), E(G) ← **A** ▷ nodes (V) and edges (E) of the network
end if
$j \leftarrow j + 1$
end while
else if method is *Transaction-based* **then** ▷ NBVFS-TM
while j ≤ n **do** ▷ n=| $S^{\phi_{freq}}$ |
$\mathbf{A} \leftarrow \frac{|S^{\phi_i} \cap S^{\phi_j}|}{|S^{\phi_i} \cup S^{\phi_j}|}$ ▷ $\mathbf{A} = (a_{i,j}) \in \mathbb{R}^{N \times n}$ is the adjacency matrix of ϕ_{freq} and $S^{\phi_{freq}}$
D = 1-**A** ▷ **D** is the dissimilarity matrix
Y(G) ← `MDS`(**D**) ▷ **Y** is the coordinate vector of nodes
V(G), E(G) ← **A** ▷ nodes (V) and edges (E) of the network
$j \leftarrow j + 1$
end while
end if
$i \leftarrow i + 1$
end while
`plot` G ▷ NBVFS-WN: Weighted network, NBVFS-CM/TM: MDS projection
Output: G ▷ Network-Based Visualisation of Frequent Sequences

From a time complexity point of view, as the method assumes that frequent sequences have already been obtained, the computational complexity of the generation of the weighted network and the MDS projections must be discussed separately. If the number of events is $\mid I \mid$, then, in the case of the weighted network, the maximal number of edges between the $\mid I \mid$ nodes is $\mid I \mid \times(\mid I \mid -1)$, that is, the complexity. The complexity of the MDS-based representation depends mainly on the complexity of the MDS algorithm (which is equivalent to the complexity of the distance matrix), that is, $\mid I \mid^3$. The complexity of frequent sequence pattern mining is $2^{|I|}$, so in the case of large amounts of events (where the method starts to be really helpful), the exponential characteristic of the sequence pattern mining becomes dominant, so the visualisation part does not cause significant complexity increase compared to the sequence pattern mining itself.

6.3 Results and Discussion

The applicability of the methods was validated using two types of data sets. The first is a click-data set of a website that is an ideal input to demonstrate the benefits of NBVFS. The second is an alarm management log file, as this development work was motivated by alarm management. Alarm management is the set of techniques and tools that support the safe and efficient control of industrial processes [131]. In alarm management, the prediction of the probability of event propagation based on historical data is crucial. Analysing event chains that have already occurred allows action scenarios for events that occur in the future. Frequent sequence pattern mining is a widely used tool in alarm management, for example, in alarm suppression methods [25], or in grouping different alarm floods based on historical data using similarity measurement [132]. However, pattern mining alone is insufficient for deep exploration of the actual system from an event-dependence point of view. This work aims to extend the existing toolkit for event data analysis based on the proposed methods and theories.

The weighted network has many available layout options, for example, a “circle” type (the starting event is located in the centre, and the sequences form concentric circles with increasing length outward), or a “hierarchical” type (the starting event is located on top with increasing sequence length downward). The width of the edges represents the confidence of the transition of the two connected sequences. The colours represent the length of a sequence. The size of the node is proportional to the confidence value of the sequence.

The transitive distance-based network provides more information about the relationship of the sequences than a weighted network. The additional information is the relationship between those sequences that are not connected.

The developed method also allows filtering on the networks. An option available in both network types (weighted and MDS projection) is to filter on the first event of the connected sequences (one or more), enabling a targeted visualisation. Another option is to draw only those edges above a minimum confidence level, allowing the identification of the most probable event scenarios. The latter is available for MDS projections.

The first source was click data from UWV (Employee Insurance Agency), a Dutch autonomous administrative authority, downloaded from the IEEE - Task Force For Process Mining website [133]. There are the following 14 identified click actions with

their representing codes in the networks: Your last employer - 0, The dismissal - 1, Hours worked - 2, Other work/income - 3, Your employment history - 4, Send data - 5, Your personal details - 6, Other information - 7, Supplement - 8, Your situation - 9, Your income - 10, Your possessions - 11, The labour market - 12, Your personal information - 13. The source file contained data for eight months. However, to save computational time, only part of the data was used to demonstrate the operation of the methods. The algorithm used was CM-SPAM [134], as this can provide the ID-s of the supporting traces, the optional hyperparameters give good flexibility and a customising option for the pattern mining task and also provide good computational speed. The input attributes for the algorithm were the following: *minsup* 1%, *max gap* 1, *min pattern length* 1, *max pattern length* 10. Sequence pattern mining resulted in 242 frequent sequences, with lengths between 1 and 8.

The other data source was an alarm management log file from a Hydrofluoric Acid Alkylation (HFA) plant. To meet the terms and conditions of the data source, all events were coded to numbers (which is also required to run the frequent pattern searching algorithms). The plant consists of four production units and more than 400 tags (source of the signals). The distributed control system is a Honeywell product. The file contained more than 200,000 events over four months, that was filtered and reduced to save computational time. The type of events in this work was limited to two, *alarm* and *operator action*. The algorithm used was again CM-SPAM. The input attributes for the algorithm were the following: *minsup* 1%, *max gap* 2, *min pattern length* 1, *max pattern length* 10. As the algorithm needs numbers as input, the events were coded to integers with the following rules: the numbers between 100 and 499 represent *alarm*s, above 1000 *operator action*s. Pattern mining resulted in 420 frequent sequences with a length between one and six. As the number of nodes is significantly high, two frequent sequence-starting events were chosen for an in-depth analysis of the results. The operator actions '1084' and '1085' control the bottom inlet and the steam inlet of the stripper, respectively. Each step of the method was performed in the *Python* programming language.

Table 6.2: Statistics of the source data.

Data source	**Click data**	**Industrial alarm data**
Number of events	1636	28536
Number of event types	14	357
Number of traces	343	3956
Average trace length	5	7

6.3.1 NBVFS-WN

The option of filtered on the start events enables a goal-oriented analysis. This kind of visualisation helps identify frequent trigger events and has quick access to the probability of occurrence of events that depend on previous events simultaneously. The weighted network has some visible characteristics that are worth discussing. Many edges get thicker heading to the outside of the network, meaning the confidence of the transitions increases with every sequence extension. Events occurring later in

the sequence depend highly on preceding events; the network represents a higher-order Markov chain. If this phenomenon occurs in a highly branched sequence line, it indicates the most probable sequence of events, enabling targeted filtering on the network.

Weighted network created from click-data

The weighted network (Figure 6.2) was filtered for starting action *2*, namely "Hours worked". The hierarchical layout was chosen to get maximum readability from a relatively low number of nodes present in the network. The network clearly shows the most probable following actions after "Hours worked", which are "Other work/income" (3) and "Your employment history" (4), with "Supplement" (8) as the most frequent final action.

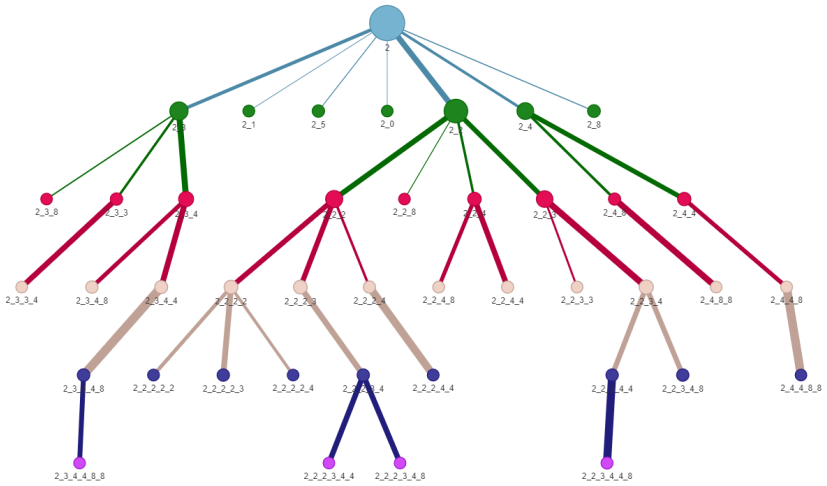


Figure 6.2: Weighted network produced from click-data, filtered on starting event "Hours worked" (2). It is followed mainly by "Other work/income" (3) and "Your employment history" (4). If 2,3 and 4 four occurs, the most frequent end event is "Supplement" (8).

This network allows to identify action rules very quickly, for example: the order of actions is always $2 \rightarrow 3 \rightarrow 4 \rightarrow 8$; if actions 2,3 and 4 occur, it is almost 100% that 8 will occur as well.

Weighted network created from alarm management data

The network produced can be seen in Figure 6.3. The most probable action to occur in addition to the operator action 1084 is 1085. The majority of sequences represented by network nodes contain these two actions. Interestingly, the most probable action scenarios also have these two actions as end events. With action 1084 occurring, an extended action sequence can be expected, and this can be clearly seen by checking the sequence routes containing transitions with high probability. By adding the actual state of the system variables when operator actions are triggered, this network can support the development of an early warning system.

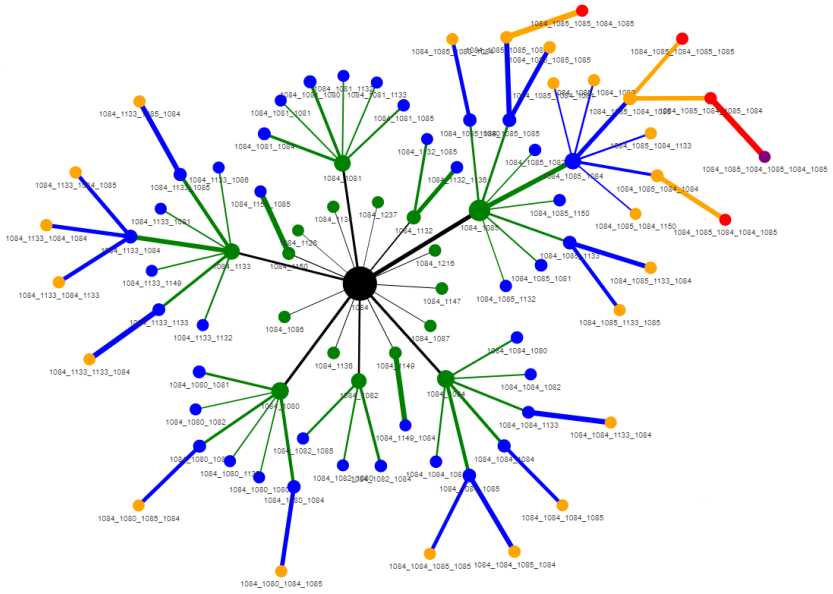


Figure 6.3: The weighted network produced from alarm log data, filtered on starting event operator action 1084. A clear view of the event propagation probability distribution is provided. It gives an informative overview of operators' intervention strategies.

6.3.2 NBVFS-CM and NBVFS-TM

The results of these two methods will be discussed in one section, since although they are both MDS projections based on similarity values, there are some differences between them.

In the NBVFS-CM generated network, the width of the edges represents the confidence of the subsequent node. This confidence value is decreasing and is heading to the periphery of the network. In the case of the transitive distance-based MDS projection(NBVFS-TM), where transition confidence values are used as similarity values, the distance between two nodes represents the confidence of that transition.

Some statements can be made for both types of visualisation. The longer an edge, the lower the confidence of that transition. In addition, the more outgoing edges a parent point has, the longer the edges. It means that the children's confidence is lower if a parent has more children, which is logical. If an event triggers more probable chain events, the probability of those chains will be reversely proportional to the number of chains. It is valid for points in the middle of the network, where events that occur in many transactions are present. Closer to the network's periphery, the points located close to each other are related by similarity. An attractive area is the "single" points. Those events are not connected, this means they are not a part of any frequent sequence, at least with the actual gap setup, but they occur with the connected ones close to them. In alarm management, these single points can be chattering alarms. If we raise the gap value, the number of isolated points and subnets will be lower; however, new single points will occur.

MDS projections of the click-data

Based on Figure 6.2, the MDS projections were filtered for starting events 2, 3, and 4. There are apparent differences between the two networks (Figures 6.4 and 6.5). The nodes in the red boxes in Figure 6.4 are far from each other, which means that their similarity value is low, at least using confidence values and transitive distance calculation. These nodes in Figure 6.5 have almost identical coordinates (practically 2_3_3_4 and 3_3_4 have the same), which means that they occur most of the time in the same transactions, which is also underlined by Figure 6.2.

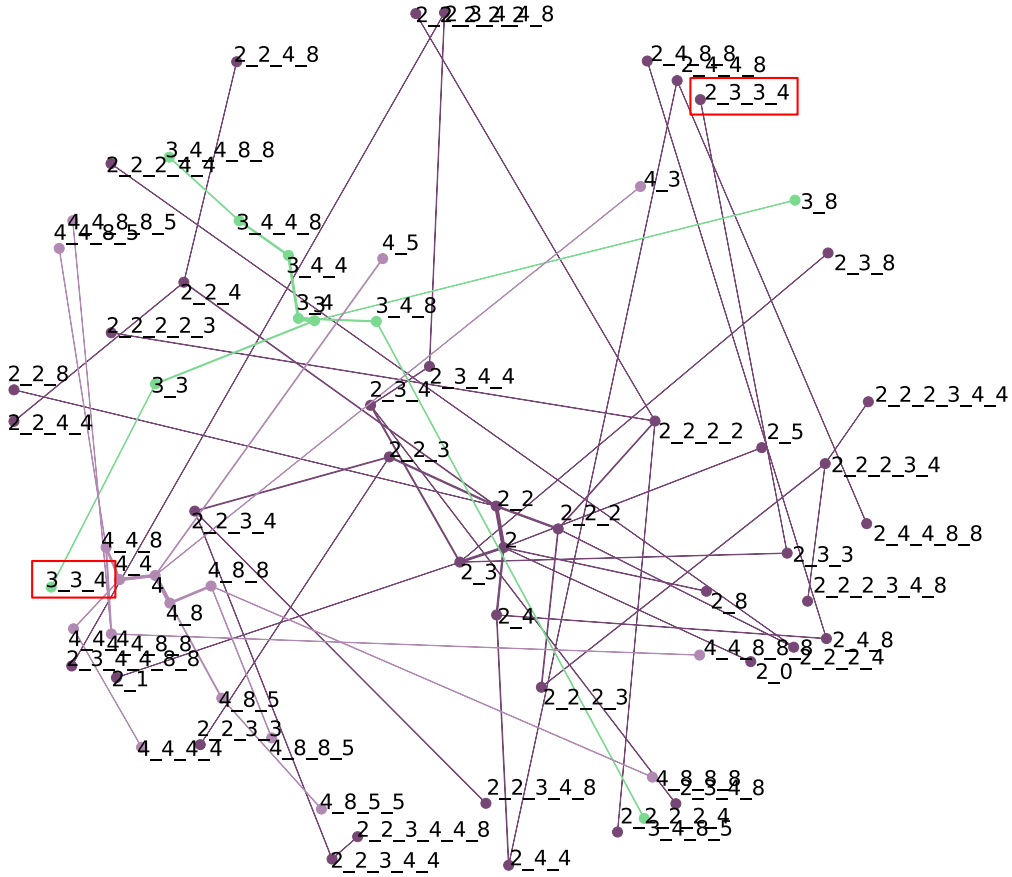


Figure 6.4: The confidence-based MDS projection of the click-data.

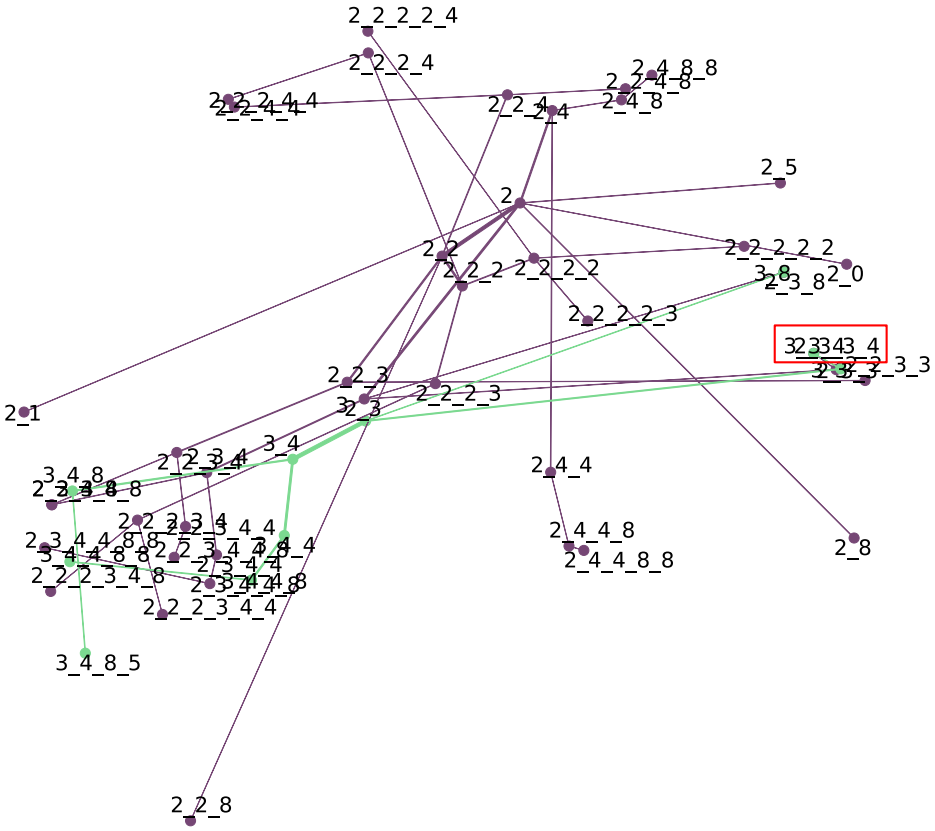


Figure 6.5: The transaction-based MDS projection of the click-data.

MDS projections of alarm management data

There is a clear difference between confidence-based networks (Figure 6.6) and transaction-based networks (Figure 6.7). The projection provided by the NBVFS-TM method has denser parts where those sequences are located, which are permutations of the same several events. This approach is closer to Process Mining. In the case of industrial log files, where the transactions are traces and these traces are generated based on timestamps, we can state that points close to each other are occurring close in time. Based on this theory, the length of the nodes represents the average time elapsed between two consecutive sequences. In this way, the traces are represented by the points located close to each other; they can be called event clusters. We can identify multi-cluster triggering events connected with clearly separated event groups, which can be helpful information in targeted event analysis and process mining tasks.

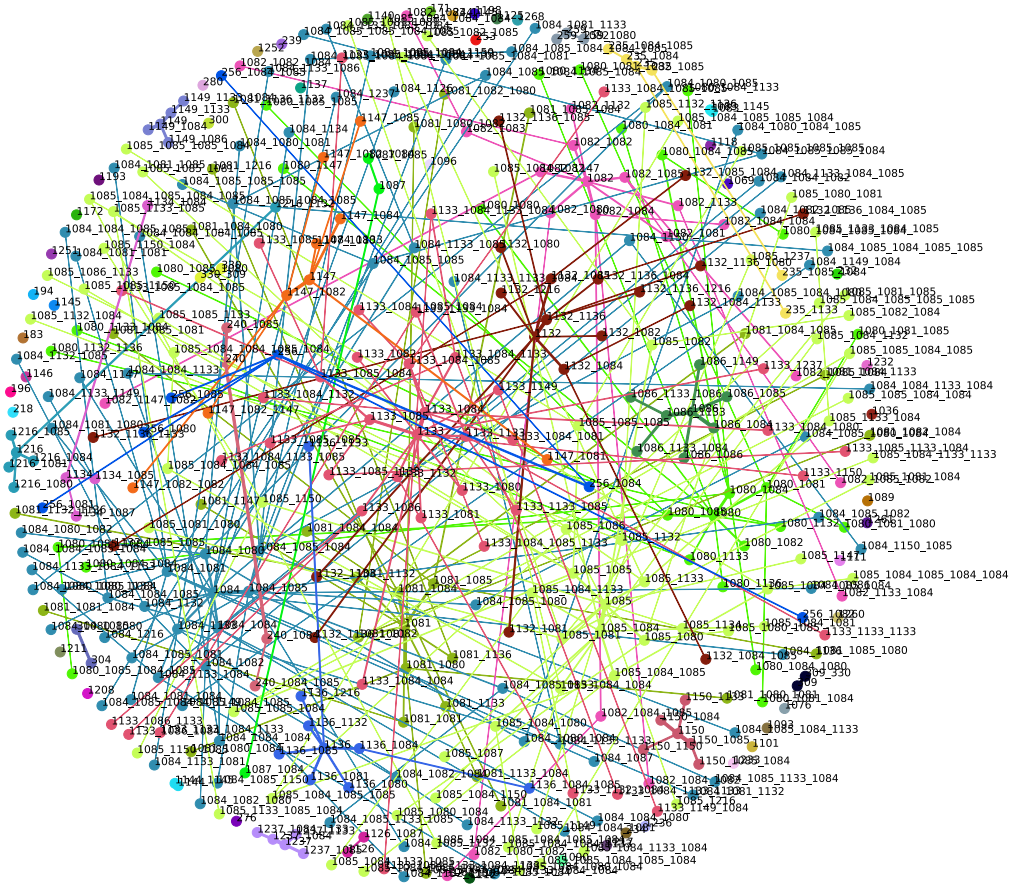


Figure 6.6: The network of sequences provided by method NBVFS-CM. It provides an enriched information visualization compared to the weighted network.

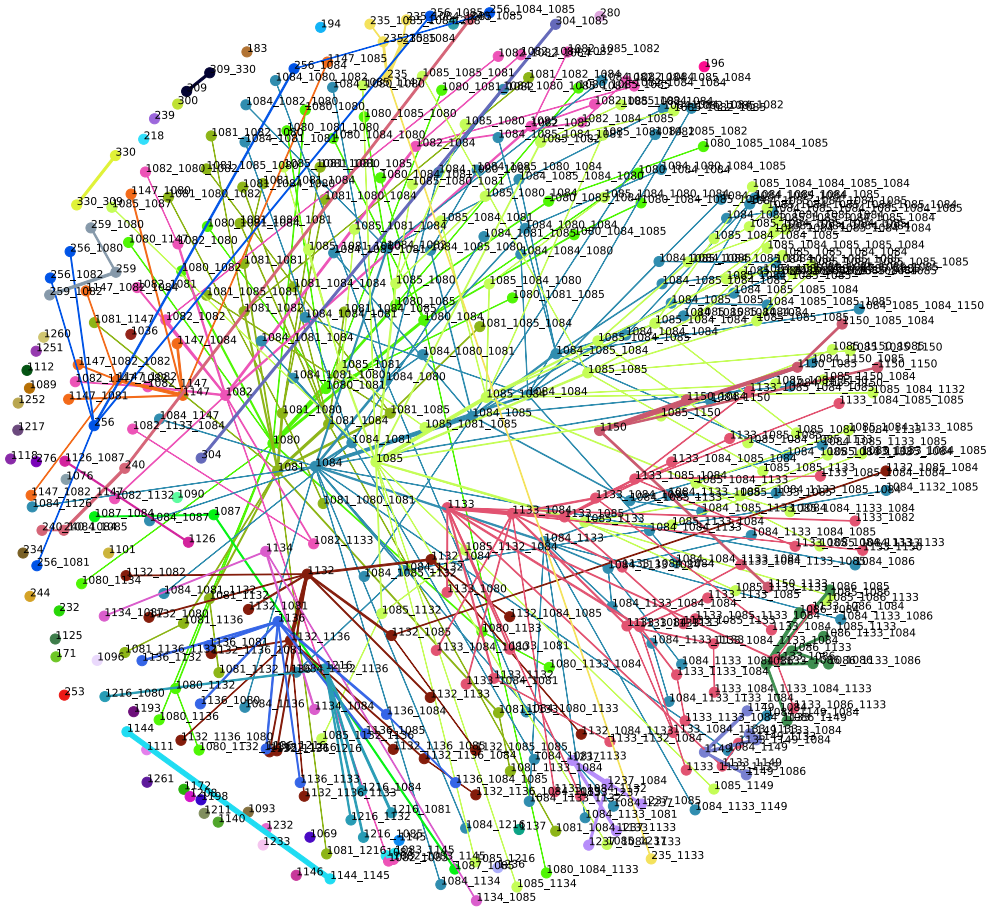


Figure 6.7: The network of sequences provided by method NBVFS-TM. There are clear groups separated from each other, and the sequences within occur in the same transactions.

6.4 Conclusion

The analysis of frequent sequences is an essential technique for the extraction of knowledge from data-based event systems. There are many available methods to explore the connection between sequences of events, and one section is devoted to the visualisation of them. Every visualisation technique has advantages and disadvantages, and the critical factor is finding the balance between information content and readability. Besides these two essential requirements, flexibility is also important and an excellent visualisation-based tool is applicable for diverse analysis tasks. This work aimed to extend this toolkit by applying a network-based approach to frequent sequence pattern mining, seeking to fulfil the mentioned requirements as well as possible.

The confidence values of the frequent sequences can be considered similarity values in the case of direct successor sequences. The adjacency matrix can be enriched with transitive distance calculation, gaining similarity values of nondirectly connected sequences. The basis of similarity can be the overlap of the supporting transactions of the sequences, and this approach provides a different view of the relationship of the sequences. A network can be created using Multidimensional scaling, where the distances between the nodes represent their similarity.

The NBVFS substeps are similar to three main groups of association rule (AR) visualisation techniques, which are one step away from the sequences. The end products of the proposed method, the networks, stand for the network-based view of AR, proving that NBVFS offers a wide range of built-in visualisation options (this work focused on the network-based ones). The three types of networks provide different types of knowledge extraction. The weighted network identifies frequent trigger events by representing support and confidence values and the length of sequences on the same network. The network provided by NBVFS-CM is an extension of the weighted network, adding the similarity of sequences as a new dimension. The resulting network of the NBVFS-TM method is based on a slightly different approach. Using the overlap of supporting transactions is one step closer to process mining. The visible groups of sequences on the network represent traces. This method can be applied to validate trace generation rules or help define them if the event database is not well structured and labelled. Last, but not least, as an open-source solution, other types of targeted visualisation are also possible. The source code will be shared on GitHub to make it available to the broad scientific community.

The potential in similarity measurement is that it can be combined with other measures. This network-like visualisation of time-series-type event databases has many possible applications. The introduced method is a goal-oriented analysis tool to extract useful knowledge from the event database by visualising the event sequences. Due to its interactive character, it can successfully support iterative data analysis tasks. For example, it enables the adjustment of the frequency sequence pattern mining parameters.

The proposed visualisation method allows for quick recognition of relevant event chains and key events with their most important attributes, such as support and confidence. This kind of information interpretation can speed up the parameter identification step of the machine learning model building. It can also validate the trace rules and other parameters used in process discovery tasks with respect to process mining. It has to be customised for the exact purpose of the task.

Although the methods are already valuable additions to sequence visualisation techniques, there are clear directions for future research. One option is the segmentation of the network. It would give a more targeted interpretation of the event relationships using network attributes like size or density. Combining similarity measurements and their aggregation with other measures is also promising.

Chapter 7

Machine Learning-supported designing of Human-Machine Interfaces

7.1 Introduction

It is already recognised that HMIs should be adaptable and easy to use [135]. Distributed Control Systems (DCS) and Supervisory Control And Data Acquisition (SCADA) systems provide a large amount of data stored in data warehouses, supporting the development of a data-driven analysis layer with machine learning (ML) functionality for the management of process safety, and artificial intelligence will become an essential part of smart factories [136]. Although there are solutions for data-driven optimisation of industrial HMIs [135], systems containing parallel and overlapping processes require a more sophisticated method. Operator behaviour models were explored from the operation log [137] or based on semi-Markovian models [138]. The mentioned techniques are essential to develop an adaptive and interactive HMI, but are not sufficient. The idea of developing function-based HMI layouts had been discussed before [139], but the methodology is still based on broad system knowledge and the P&IDs. Although state-of-the-art HMI devices are already available [140] for the "greenfield" development of HMI, it is worth optimising existing systems, as "brownfield" investments offer a reasonable solution for modernisation [141]. These brownfield investments can be executed cost effectively with the use of open source and web-based software, providing high flexibility and customisation. The proposed method shares this approach, which is also aligned with the actual trend of using the Industrial Internet of Things philosophy in HP-HMI development tasks [142].

The key idea is that typical process event scenarios can be gained by using frequent sequential pattern mining, which is a useful technique to analyse industrial events, for example, to deal with alarm floods [25]. Scenarios enable the creation of goal-orientated log files from which, with the help of process mining tools, an integrated process operator behaviour model can be produced. Process descriptions support the appropriate labelling of the data necessary to obtain precise, informative, and comparable models. The principle of the iterative method developed is to use the combination of proper frequent pattern and process mining techniques. The proposed HMI displays and functionalities are based on the combined interpretation

of P&I diagrams, process deviation models based on typical event/alarm sequences, and process control behaviour models based on typical operator responsive action sequences.

The theoretical background of the method is discussed in Section 7.2. In Section 7.3, an application example is presented in a case study, and the results and considerations with respect to the application of the method are discussed. Section 7.4 contains the conclusion and possible future research directions.

7.2 Machine Learning-supported HMI optimisation

Pattern mining-based filtering is more targeted than post-filtering methods in process mining tools, resulting in more focused process models. The proposed method can effectively support the process mining of any system where events are stored.

In this work, the applicability is demonstrated on an alarm management system. The concept discussed in the rest of the chapter will be alarm-related. From the log file in which the alarms (alarm data), operator and display actions (HMI data) are stored together, key events are identified through frequent pattern mining tools. If these kinds of event are stored in separate log files, they have to be merged. Patterns can be itemsets or sequences of events. These key events frequently occur with each other, and the set of these events will be the basis for log-file filtering. The filtered log file is the input of the process mining and the output is a typical process pattern. Different filtered log files can be created from the group of critical events, resulting in different process patterns. Once typical process patterns are available, with the help of existing system knowledge, situation-related action patterns are obtained, indicating amendments in the actual HMI layout. Each new pattern enriches the knowledge of the system. The concept of the proposed method can be seen in Figure 7.1.

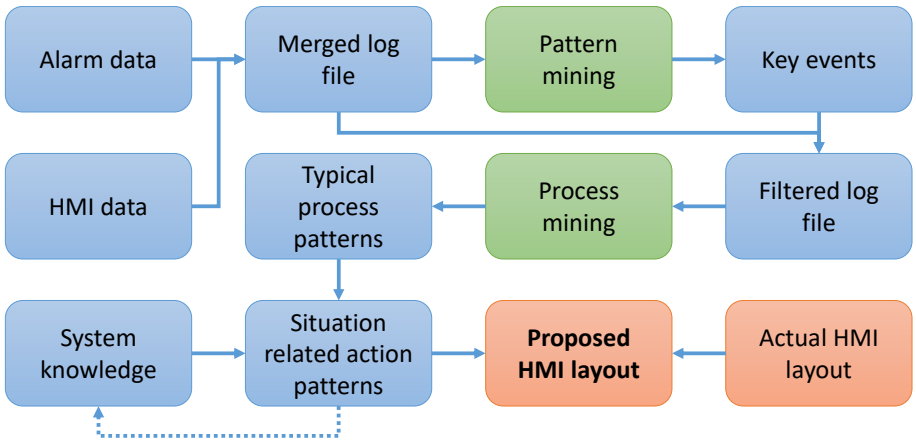


Figure 7.1: The proposed ML-supported HMI optimization method with alarm management-related interpretation.

An essential condition for the development of HMI supported by ML is a properly designed log file [82]. For example, a process event in the log file can be an alarm, an adjustment of a process variable, a measured process value signal, or an alarm acknowledgement. HMI displays are drawn based on P&I diagrams, and one process

event may belong to different subprocesses, so some controlled process variables are placed in more displays, as they can affect several parts of the plant. Based on the stored events, process models can be gained by using process mining algorithms, such as *Heuristic miner* [54].

The design goal of a High Performance HMI (HP-HMI) can be to minimise the number of displays and the number of information on the displays, and to have supportive process-orientated display layouts. The applicability of the proposed method is highly dependent on the quality of the stored data in terms of proper labelling. If many subprocesses are related to more plant units, the creation of overview-like displays is recommended, containing only the relevant process values and controlled process variables.

7.2.1 Identification of the workflows as frequent event patterns

The basis of the process exploration tasks is an L log, represented as a set of events (E) stored in the order of their occurrences. In alarm management, events can be alarms, operator actions, process variable changes, etc. These events are considered states of the system formed by $< PV, a >$ tuples. PV is a process variable and a is the attribute indicating the state of the variable. For example, $E_j =< oil\ temperature, high >$.

An $L = \{T_1, \ldots, T_n\}$ log contains $T_k = \{E_1, \ldots, E_p\}$ traces, where $k = [1, n]$ and $p = [1, z]$. A trace is a subset of all $E = \{E_1, \ldots, E_z\}$ events within the log file that describe a sequence of process control activities. Considering that traces are formed of events in a given sequence, the most frequent traces can be considered as process patterns or workflows. Therefore, events assigned to a workflow are also a subset of all events.

$$WF_j = \{E_1, \ldots, E_k\} \subseteq E. \tag{7.1}$$

From another perspective, a workflow is a valid frequent sequential pattern and can be defined using frequent sequential pattern mining. A sequential pattern can be considered frequent if it exceeds a predefined *minimum support* value, that is the number of traces where the pattern is present and is a hyperparameter of the method. Besides the *minimum support* value, *confidence* can be considered as another hyperparameter, that is, the strength of connection of the event patterns. An *s-length* workflow, e.g. sequence can be split into s subsequences, $WF_j = \{WF_j^1 \rightarrow ... \rightarrow WF_j^i \rightarrow ... \rightarrow WF_j^s\}$. The confidence can be calculated as follows:

$$conf(WF_j^i) = \begin{cases} \frac{sup(WF_j^i)}{sup(WF_j^{i-1})} \times conf(WF_j^{i-1}) & s \geq i > 1 \\ 1 & i = 1 \end{cases}, \tag{7.2}$$

where $sup(WF_j^i)$ is the number of traces in L, where WF_j^i occurs.

Each confidence value of the transitions can be interesting, not only the confidence value of the whole event chain. Consider a sequence of events, where the last event has great negative consequences. If the transition confidence of any parent-child event pair is low, the overall confidence of the whole workflow will be low as well (equation 7.2). If this low transition confidence value is at the beginning of the

workflow, and all subsequent transitions have high confidence, then the unwanted last event will have a high probability of occurrence after a certain event sequence occurred. Knowing these sequence confidences, critical event scenario milestones can be identified that support the creation of event forecast models [143].

Frequent sequential patterns support the goal-orientated creation of log files that describe the target process. The process model can be created using process mining techniques on the created log. The selection of the pattern mining technique depends on the complexity of the actual system, for example, the number of parallel processes.

7.2.2 Connection between workflows and displays

Elements of an HMI display are visualisations of the process variables and their conditions; e.g. they describe the actual state of the system. On the basis of this notation, a D display can be considered a set of events, that is, a subset of all events (similarly to workflows). A set of displays is assigned to a workflow.

$$\begin{gathered} D_m = \{E_1, \ldots, E_n\} \subseteq E, \\ D(WF_j) = \{D_c, ..., D_m\} \rightarrow WF_j. \end{gathered} \tag{7.3}$$

This means that an element on a display may belong to more workflows, and a workflow may be represented on more displays. The number of elements in a given display cannot exceed a threshold C due to the cognitive limits of the operators with respect to signal perception [144]. The goal is to minimise the number of displays allocated to a workflow to ensure that the operator does not have to switch too much between different displays when solving a frequently occurring problem, this way C can be also a hyperparameter of the method.

$$\begin{gathered} |D_m| < C, \\ min|D(WF_j)|. \end{gathered} \tag{7.4}$$

7.2.3 Workflows, displays and their elements as a community

Another way to define the ideal event-display order is to apply the community theory [145]. Consider a $WF_j|j = [1, w]$ workflow as a graph, where the $WF_j = \{E_1, \ldots, E_z\}$ events are the nodes, and the sequence of the events indicates the edges between them, resulting in a directed graph. Due to the confidence values of the event transitions, the graph is weighted as well. The $D(WF_j) = \{D_1, ..., D_m\}$ set of displays containing events related to WF_j are subgraphs of WF_j. If more displays are aligned to a workflow, the order of the used displays form a sequence as well; the graph of the displays is also directed and weighted. In a plant, there are w workflows in parallel. $D^{WF} = \bigcup_{i=1}^{w} D(WF_i)$ is the set of all events on all displays related to all workflows.

If the node connections of D^{WF} are marked with $V(D^{WF})$, then the minimum goal is to have displays that form a *weak community*, and the problem to solve is

$$\frac{V^{ext}(D^{WF})}{V^{int}(D^{WF})} < 1, \tag{7.5}$$

where $V^{int}(D^{WF})$ is the number of connections for nodes within the displays, and $V^{ext}(D^{WF})$ is the number of connections for nodes between the displays.

The best solution is to have displays, where the nodes within each display form a *strong community*. If the number of displays is m, then the optimisation problem is

$$min \sum_{i=1}^{m} \frac{V^{ext}(D_i)}{V^{int}(D_i)}, \tag{7.6}$$

e.g., the intersection of the displays must be minimal. Obviously, in real life the zero intersection between displays is not realistic, but can be approximated. Equation 7.6 also indicates that the optimisation is done on the whole system, not on each workflow-display pair individually. An idealistic state can be seen in Figure 7.2, which can be reached with the help of a community detection algorithm. It is very important to note that in industrial control systems, communities overlap. This problem must be addressed by the detection method, as in [146] and [147].

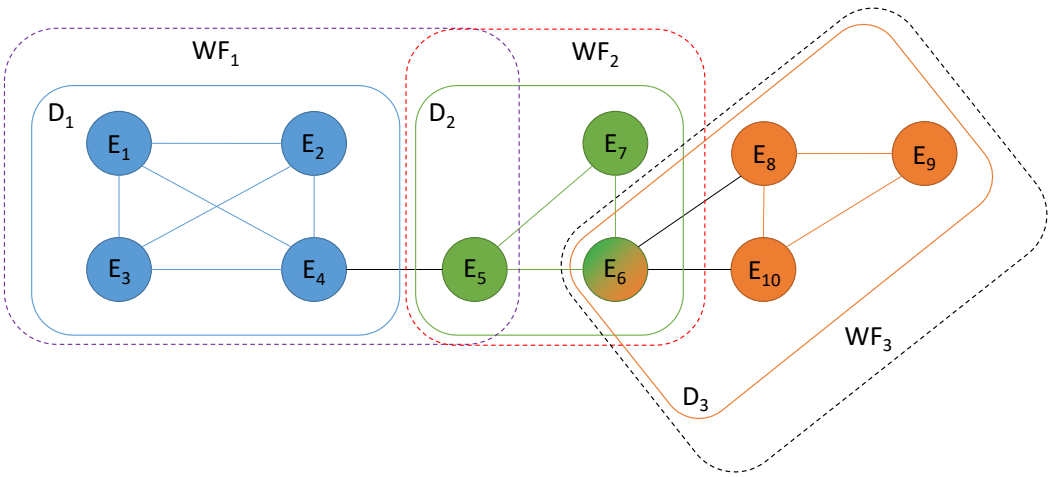


Figure 7.2: An optimised example community of events (E), displays (D, continuous lines) and workflows (WF, dashed lines).

The time complexity of the method consists of three maximum parts: pattern mining (O_{pa}), process mining (O_{pr}), and optionally network visualisation (O_{ne}). If $\mid E \mid$ denotes the number of different events stored in the L log (with a size of $\mid L \mid$), and N is the number of traces with sizes of $\mid T_i \mid$ (considering sequential pattern mining):

$$O_{pa} = O(\sum_{i=1}^{N} |T_i|) + O(|L| \times |E|) + O(2^{|E|-1}) \times O(|L|),$$
$$O_{pr} = O(|L|, |E|^2). \tag{7.7}$$

In a network-like representation of the sequence of events, the nodes are the events (E) and their transitions are the edges. To visualise the process, a weighted network can be applied, where the weight of the edges is proportional to the number of transitions between the nodes. The complexity is the maximal number of edges between the $\mid E \mid$ nodes:

$$O_{ne} = O(\mid E \mid \times(\mid E \mid -1)) \tag{7.8}$$

Considering that industrial control systems contain a large number of elements, e.g. events, the complexity of the method increases exponentially with $\mid E \mid$. However, it should be noted that the method is useful in the planning phase, in this way the speed of applicability is not that important, and the specific system has a huge effect on the complexity.

7.3 Application of the method on a real world industrial system

To validate the effectiveness of the method, the chosen place of application was a real-life Hydrofluoric Acid Alkylation plant. The schematic drawing of the plant can be seen in Figure 7.3.

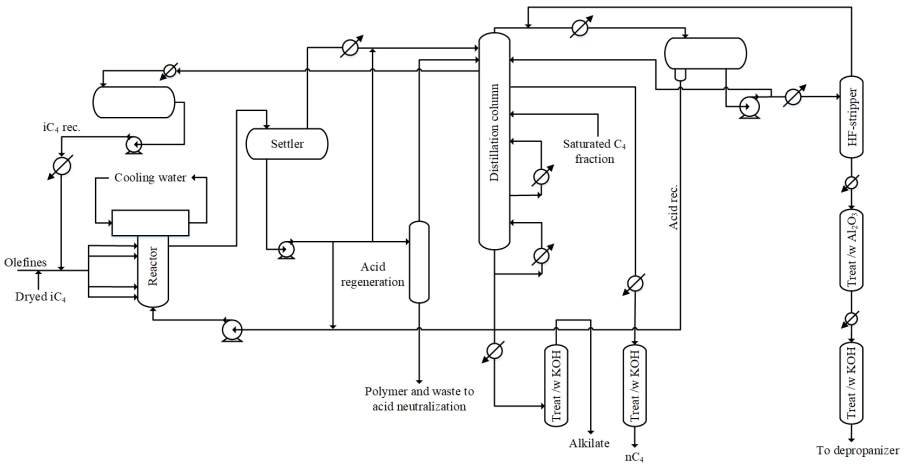


Figure 7.3: The schematic drawing of the plant.

The method has modest computational requirements. The computer used had an Intel (R) Core (TM) i7-7700K CPU @ 4.20GHz with 64GB RAM and Windows 10 pro. With this configuration, the algorithms used had running times in the range of minutes.

7.3.1 Problem description

The processed log file contained more than 200.000 events over a month. After filtering out events that have incomplete data regarding relevant properties, approximately 90.000 events remained. After processing the remaining events, one was identified as a chattering alarm; it was also removed, resulting in a log with around 20000 events, distributed among 276 tags. The trace rule threshold was set at 120 seconds (the minimum time interval without any logged event) on the basis of industrial experience. Based on the results of frequent sequential pattern mining, the log file was partitioned, defining a specific subprocess. A network-like representation of the frequent sequential patterns found can help select the ones to filter the log file. All steps of the proposed method (Figure 7.1) were carried out without deep knowledge or experience related to the plant to ensure objectivity and avoid manipulation during the exploration of the process. The results were evaluated with the plant staff.

7.3.2 Results

The resulting process model can be seen in Figures 7.4 and 7.5. The mining algorithm used in the process was *Heuristic miner*, the minimum activity count was set at 30. This threshold depends on the event-occurrence distribution and has to be specified case by case. *Heuristic miner* is a well-known and widely used process mining algorithm, offering a good balance between the usual benchmarks of process models: simplicity, generality, fitness, and precision.

The frequent sequential pattern mining algorithm was selected from the *SPMF* [148] library, which offers a wide range of pattern mining algorithms. The algorithm used was CM-SPAM [134], the optional hyperparameters give good flexibility and a customising option for the pattern mining task and also provide good computational speed. Table 7.1 shows the effect of minimum support on algorithm performance. It is clear that increasing this hyperparameter causes a significant change in the results.

Table 7.1: Performance results of sequential pattern mining. The performance factor is the quotient of the number of found patterns and the elapsed time.

Minimum support	**Found sequential patterns**	**Elapsed time [ms]**	**Performance factor**
0.2	6777245	312271	21.7
0.3	65601	4278	15.3
0.4	3058	322	9.5
0.5	405	101	4.0
0.6	99	72	1.4

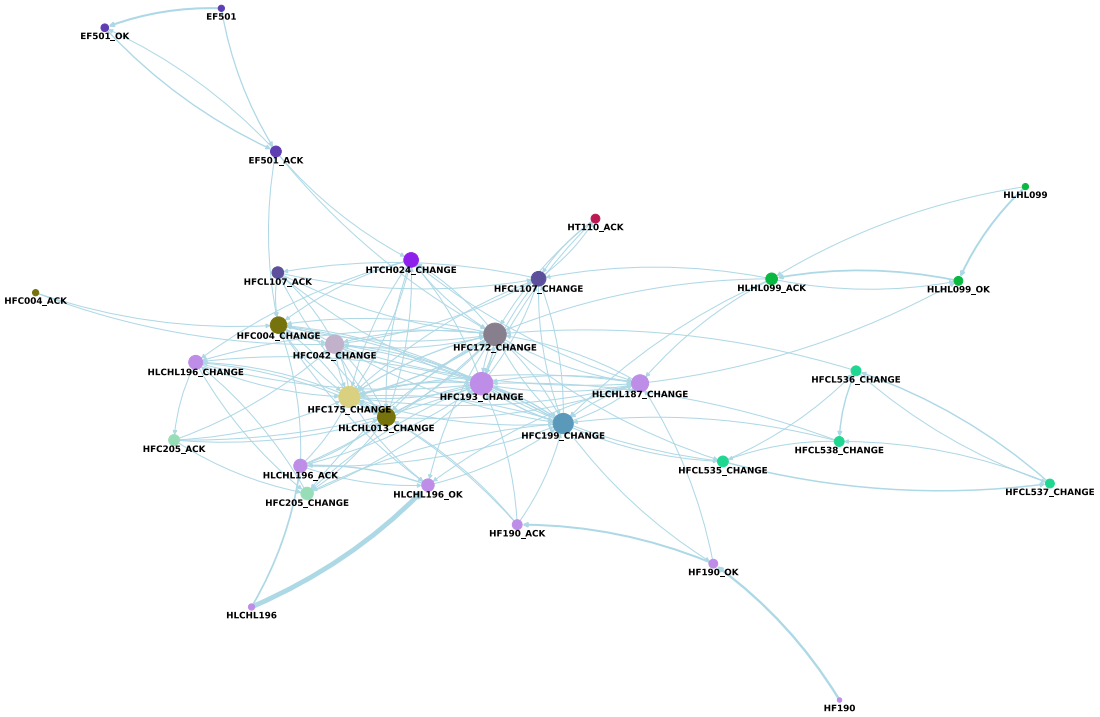


Figure 7.4: A specific process model of the plant. Colours mark the display where the event is represented, the size of the nodes refers to the cardinality of the event. The structure of the node names is XX_YY, where XX denotes the the ID of the event and YY is the type of the event (in the case of an alarm signal, no type can be seen).

Another performance check option is to change the number of processed traces. Table 7.2 shows that increasing the number of traces does not cause a significant computational demand. As discussed in complexity analysis, the dominant factor is related to the pattern mining step, so it can be stated that the method is capable of handling large log files without excessive computational demand.

Table 7.2: Performance results of sequential pattern mining for a fixed minimum support value of 0.6, changing the number of processed traces.

Number of traces	Number of events	Found sequential patterns	Elapsed time [ms]	Performance factor
495	21277	99	72	1.4
982	50870	31	83	0.4
1474	62496	27	88	0.4
1970	70404	26	113	0.2
2464	84824	25	112	0.2

Based on the data provided by the mining algorithm, a network-like representation of the event chains can be created. This kind of visualisation offers the placement of several process model attributes on the graph, which is an advantage over the built-in visualisation of the Heuristic miner. These attributes can be the

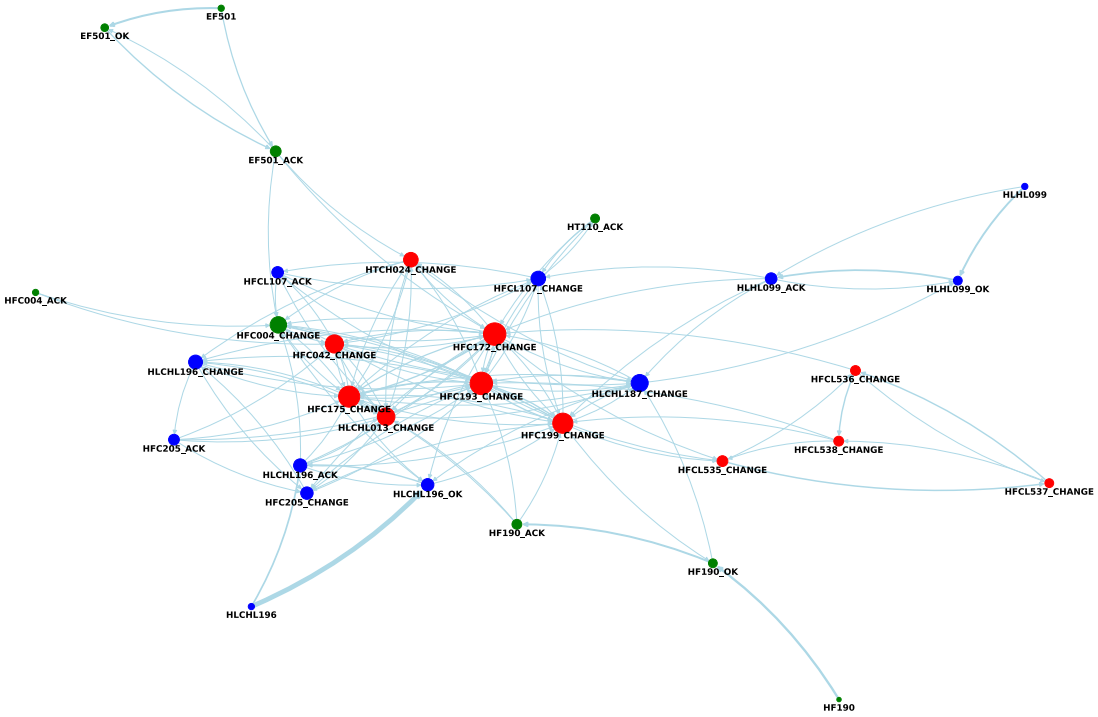


Figure 7.5: A specific process model of the plant. The colours represent the workflow clusters provided by the built in clustering algorithm of pm4py. The structure of the node names is XX_YY, where XX denotes the the ID of the event and YY is the type of the event (in the case of an alarm signal, no type can be seen.)

occurrence rate of an event, the transition frequency, or the previously defined cluster of the event. In Figure 7.4, the clusters are the actual displays where the events are presented. In Figure 7.5, the clusters are defined by the built-in clustering algorithm of the used process mining python package (pm4py), on the basis of the similarity of activity resources [149]. The number of clusters dropped to 3 from 22, and the calculated community coefficient (7.6) is 127 compared to 133 (which is not a significant difference, but should be evaluated considering the change in the number of clusters), meaning that 3 workflows are dispersed among 22 displays. The results indicate that the actual display layouts need to be reconsidered. Based on the process model and the existing available HMI displays, a new synoptic overview display layout (Figure 7.6) was proposed.

Obviously, the results and the proposed new layout have to be evaluated together with the plant operators, their knowledge and experience cannot be ignored in this kind of development task. They can identify irrelevant signals or the need for additional elements on the new displays. Every deviation from "real life" indicates problems with the process model gained. Reasons can be various; for example, the structure and data labelling of the log file is not appropriate, or the alarm management system is not adjusted appropriately. However, these deviations can also originate from poor operator practices and the method also supports the analysis of process control effectiveness. A usability test has to be performed as well. Usability inspection methods were discussed in [150], *Formal usability inspections* and *Pluralistic walkthroughs* are especially recommended.

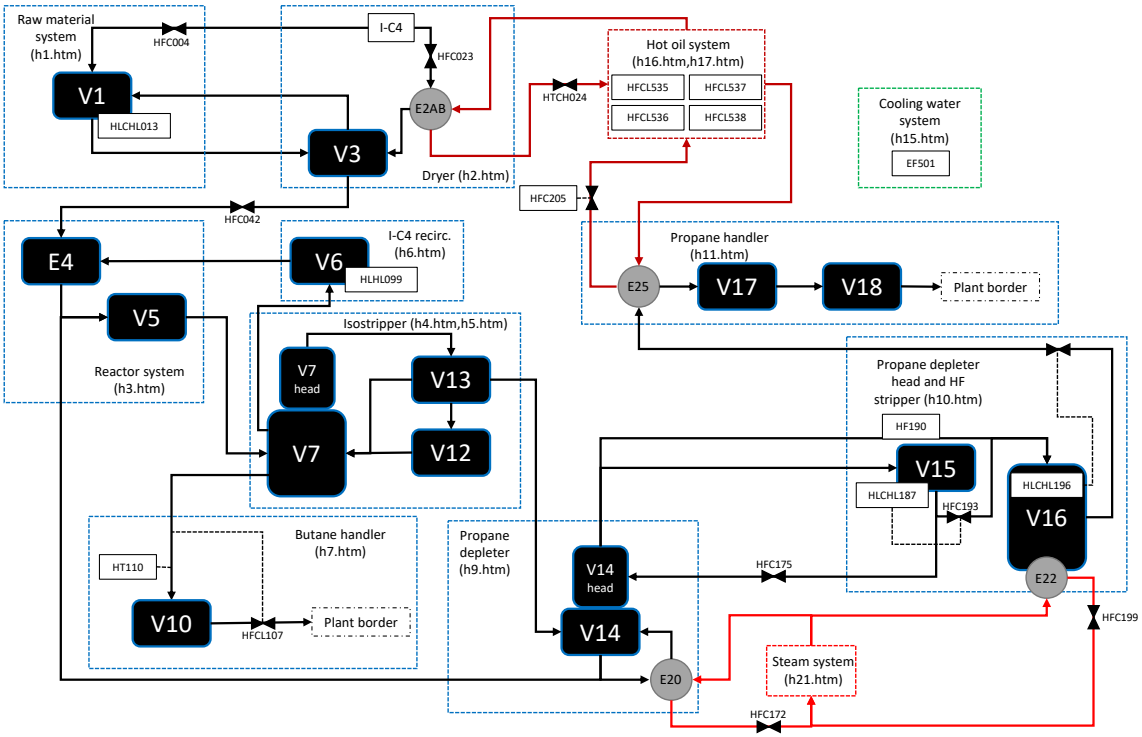


Figure 7.6: The proposed synoptic HMI display. Blue dashed-line boxes represent the actual displays. Elements put between two displays are located on both. The figure is not an exact display, it only represents the proposed content.

7.3.3 Discussion

The study has shown that the method is useful in alarm management and in the optimisation of industrial control systems. Applicability can be broadened, but some limitations must be considered.

One potential issue can arise if one event is allocated to more displays. In this case, a decision has to be made, whether the display cluster will be a union of the touched displays, or the event is paired with only one display. Theoretically, it could be analysed to see the display where the interactions are made with the event the most frequently. In practise, the log file barely indicates this information. One solution is to create "union" displays, which is not an ideal solution, this "error" must be considered during the evaluation of the results. The chosen approach depends on the number of multi-display events. If this number is high, the sequence of events forms a multidimensional network, where the definition of multidimensional edges enables clustering of events [151]. If their number is relatively low, the issue can be handled with experience and knowledge of the actual system.

Visualisation is also a key part of the method. If the number of nodes and edges is high, the readability of the network can quickly worsen. Using open source Python solutions, this issue can be solved, for example, by changing the layout algorithm to find the most transparent view of the network.

The number of new displays depends on the attributes of the network gained. In the case study, there were 3 workflows, but only one new display was recommended, because the number of nodes was relatively low. If the number of nodes (events) exceeds a recommended threshold with regard to the human cognitive limits, e.g. the maximal information put on a display, the workflow has to be split. The number of

splits (displays) has to be optimised considering the dynamics of the system that can be visualised on the network graph, for example, with node sizes and edge thickness. The proposed method allows many information visualisation options, which support a quick understanding of the system mechanism.

The main function of the proposed method, the process-related grouping of HMI elements, allows the development of dynamic HP-HMI screens. As the number of monitors in the control room is limited, it is recommended to display as much situation-aware information as possible. Figure 7.7 demonstrates a possible working principle of a fictive plant with six monitors in the control room. Again, the aim of the figure is to suggest the content of the screens. The actual displays should be enriched, for example, with trend charts, the visualisation and design should follow the suggestions of the ISA101 standard.

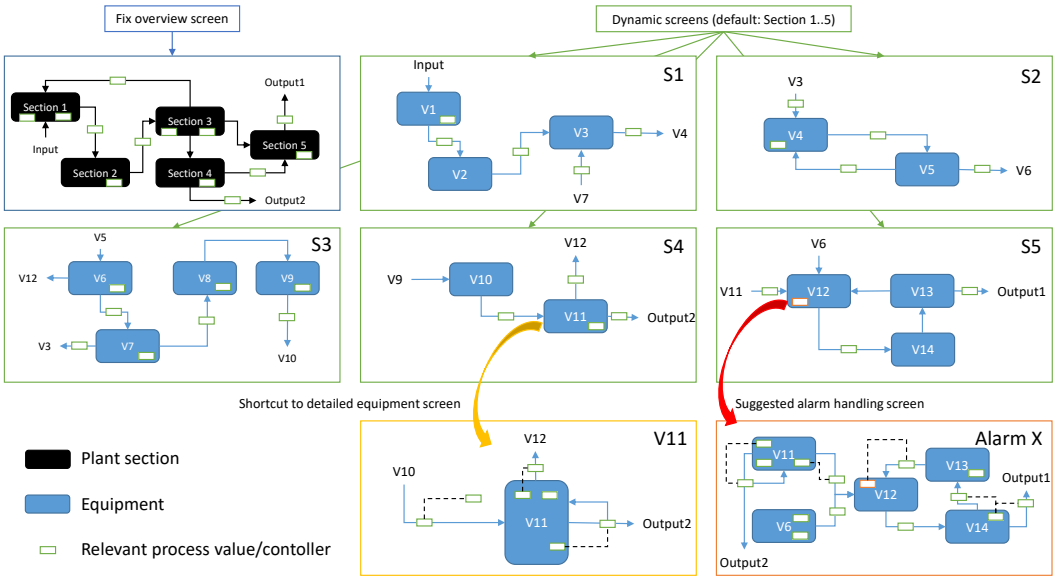


Figure 7.7: The fictive plant has five sections containing 14 equipments. The displayed content of the sections is defined by using the proposed method. The suggested alarm handling screen is created based on processed historical data.

7.4 Conclusion

In this work, a Machine Learning-supported HMI designing method was proposed. The method aims to support the application of the *Navigation and Layout* principle of the ISA 101 standard, namely the *grouping related elements together* aspect. Alarm signals, operator actions, and display actions from a hydrofluoric acid alkylation plant were analysed. The log file was partitioned based on frequent sequential patterns of the stored events. Process control models were explored with process mining tools. A network-like visualisation of a typical process enabled a comparison of the number of workflows and the number of displays where the workflows are represented. A new, higher-level display was recommended on the basis of the models gained and the existing structure of the HMI displays. Even without deep knowledge of the system, the presented method proved effective. However, manual work is

needed to evaluate the displays, which results in subjective decisions. The method can be upgraded to define objective KPIs that support solving the minimisation problem.

In addition to data-driven HMI optimisation, there are other potential benefits to using the extracted knowledge. By comparing the actual online process data with formerly defined event scenarios, a semi- or fully automated online HAZOP analysis tool can be built. The gained process patterns can be compared to a reference model that allows the evaluation of the work quality of the operators and supports the development of the Operator Training System (OTS).

Another potential future research direction is the implementation of machine learning in SCADA systems, which is an actual topic in the industry, as more and more solutions will be assisted by artificial intelligence. The proposed method is capable of processing acquired data and finding patterns among the event chains that lead to failures and the response actions of the operators. Using ML solutions enables the continuous dynamic and adaptive development of industrial process control systems.

The topic of trend charts can also be an interesting area. With the help of ML-supported solutions, process-variable dependencies can be gained from historical data, adding a predictive function to the system. For example, if the operator calls a trend chart of a specific process variable, an intelligent system can offer other trend charts, which may be useful in the actual control activity.

It is important to emphasise that ML-supported process control solutions do not aim to replace the human workforce. Using these methods and techniques shall result in safer and more efficient process operations by supporting decision-making with more process-aware human machine interfaces.

Chapter 8

Conclusion

This research aims to explore potential data and process science tools that support the analysis and modelling of process-related events in industrial control systems. The appearance of the Industry 4.0 approach has raised the importance of stored process data with the increasing need for deep analysis of processes. With a reach literature in the field of continuous data handling, the topic of discrete process event analysis and modelling with process mining techniques has room for improvement. The goal of my work was to collect the available techniques and develop new supporting methods to process industrial log files, with alarm management in the focus.

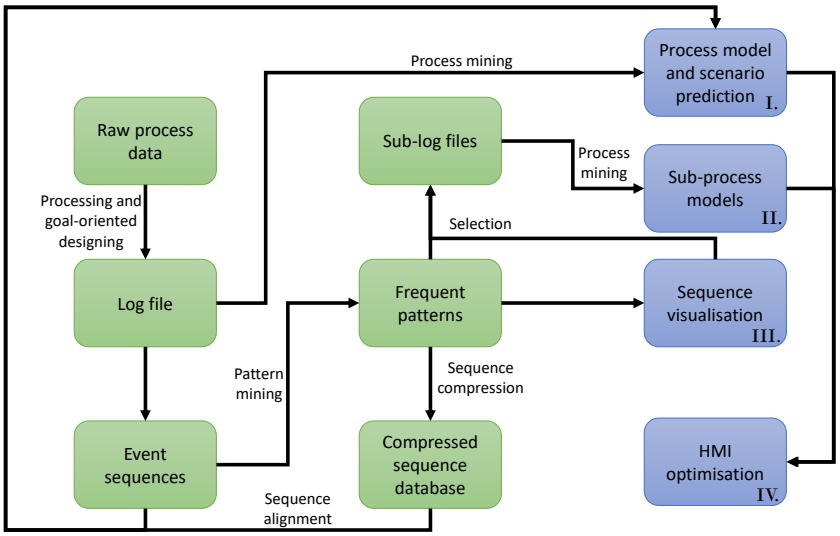


Figure 8.1: Roadmap of the dissertation. The input of process mining is the stored raw data, that has to be formatted in a standardized way. The standard format allows to gain process models and frequent event patterns as well, supporting an event scenario prediction method. Frequent event patterns also can be used to generate sub-process models, a network-like visualisation of the patterns supports the selection step. The mentioned techniques form a machine learning-based method to optimise human-machine interfaces.

The collection and storage of industrial data became easy and inexpensive at the beginning of the twenty-first century. Adding the exponential increase in computational speed resulted in a process-mining breakthrough. The input of process mining is a properly designed log file that contains all relevant information about the stored events. In addition to the mandatory event properties (event ID, timestamp,

case ID), the log file can be enriched with specific data (resource, organisational role, cost, etc.) enabling targeted or hierarchical process mining resulting in specific ***process model***s. A properly designed log file has other benefits too, for example, frequent sequential event patterns can be gained to ***predict*** event ***scenarios***. Frequent event sequences can be stored in a database using sequence compression. If actual events are transformed into sequences, they can be compared with the database with the help of sequence alignment. Based on the similarity of the actual sequence to the stored ones, a probability event prediction can be made.

One potential issue with industrial log files may be that parallel processes are stored in the same log, resulting in biased and hard-to-understand process models. By assuming that the specific processes contain typical event groups, frequent pattern mining techniques can help to identify the sub-logs of the sub-processes, supporting the generation of ***sub-process models***. The challenge in this kind of log file partitioning is to choose the right frequent patterns. In the case of huge logs, an excessive number of patterns will be found that are hard to understand. There are several methods for sequential pattern post-processing; one of them is visualisation. By handling the frequent event sequences as elements of a network, an information-rich visualisation can be made. With the developed method, the pattern selection step in creating sublog files can be faster and easier.

In addition to the aspect of the targeted exploration of processes by creating sublog files, the gained models can be used to develop or ***optimise*** level two or three Human Machine Interface (***HMI***) displays. Typically, HMI displays are drawn based on the Piping and Instrumentation Diagrams (PIDs), which is good for level one displays, but on higher-level displays, a different approach may be necessary. To support the work of operators, that is, to develop human-centric HMI, process-aware layouts are one solution. The results of process mining combined with network theory can identify the groups of actors that are essential parts of a given operation, and shall be put on the same display.

8.1 Experimental tools and methods

During my research, based on a comprehensive literature overview, I have collected the existing data and process mining techniques and identified their shortcomings with regard to industrial process analysis tasks using distributed control system (DCS) data. To answer those shortcomings, new process analysis support tools were developed:

- a targeted event log transformation method to have log files in a standardised format,
- a sequence alignment-based event prediction method,
- a frequent pattern-based log file partitioning method,
- a similarity- and network theory-based sequence visualisation method,
- and a machine learning-supported HMI optimisation method.

The log data used were from two sources: DCS data from different parts of the oil refinery plant of MOL in Százhalombatta; and click data from a publicly available data repository.

The data processing and analysis steps were performed partially in ProM. To ensure that the developed methods can be used in a flexible way with respect to the source data, the whole framework was put into Python programming language. Python environment offers a wide range of solutions, log transformation to XES (eXtendable Event Stream) standard, the SPMF library for frequent pattern mining, the pm4py library for process mining, and other libraries for visualisation tasks (graphviz, networkx). The developed method and python package can be used for other discrete data analysis projects, independently from the data source.

8.2 Theses

Thesis I.

I have proved that with goal-orientated trace identification and log file design, process mining is a suitable tool for analysing, discovering and predicting industrial processes. [1,2]

The exploration of industrial processes is to get information from related logged events. Related events in log files form traces. Trace identification is essential to explore industrial processes. However, it is not a trivial task, as usually the logged data is not labelled from this point of view. There are several methods to group events; one way is to define a minimum time interval between two events. Structuring and labelling the log file is also inevitable. Proper data labelling enables goal-oriented and hierarchical process mining, and every aspect of the processes can be interpreted. In alarm management the alarms, operator, and return to normal actions can be evaluated together and separately as well, showing different layers of the same process.

A well-designed log file allows to gain frequent sequential patterns containing useful information about process evolvement. If the most frequent event streams are collected in a compressed sequence database, comparing them with actual event sequences by using sequence alignment, the most probable event scenario can be predicted.

The developed trace identification and log file designing method was applied in the alarm management rationalisation project of an industrial hydrofluoric acid alkylation plant. I have identified the key events and generated different log files for different process discovery purposes. The resultant log files were analysed with process mining tools. The project demonstrated that with the help of process mining, alarm signals could be rationalised; therefore, the work of the operators will become safer, as well as more effective and, last but not least, the workload can be decreased. The sequnce compression and alignment method has been examined and characterized using real-life data originating from a delayed coker, and its usability and limitations have been determined. The results show that the method has a very effective pattern mining capability that extended with the sequence alignment method can recognize an operational state just after a few typical alarms and match it with historical patterns in less than a second. High-confidence predictions could be obtained easily.

Thesis II.

I have developed a frequent sequential pattern-based log file partitioning method to gain specific models of subprocesses stored in the same log file. [3]

The suggested method of filtering out these unnecessary events while simultaneously creating sub-logs combines frequent pattern mining and traditional process cube operations. Frequent itemset and sequential pattern mining were applied on the original log file, and the resultant itemsets and sequences were the basis of the log file partitioning. If the structure of the log file is proper (Thesis I.), a hierarchical partitioning is possible as well.

The method was applied to a log file of an industrial Hydrofluoric Acid Alkylation plant. The resultant process models in the case study were evaluated using the performance metrics introduced. The results proved that the method is effective in partitioning log files, regrouping events for targeted process discovery tasks, and handling the problem of parallel processes. The benefits, requirements, and limitations of the method were identified and are the following:

- Benefits
  - Sequential pattern mining and process mining use the same source. No extra preparation of the log-file is required.
  - The method is efficient from a computational demand point of view, as the size of the log-file to be processed has been significantly reduced.
  - Prior knowledge can be transferred to identify the relevant frequent patterns, facilitating iterative work. Process-relevant information can be efficiently included in process-mining.
  - The defined metrics enable an objective evaluation of the results. The evaluation step can facilitate the automation of the method.
- Requirements - Limitations
  - Although the method needs a pattern mining tool, this is not a critical issue as open-source tools are widely available.
  - A certain amount of knowledge is required to select the right pattern-mining algorithm. Suggestions are made regarding this topic.
  - Prior knowledge of the process is essential. Similarly to the *Knowledge Discovery Databases* (KDD) process-model, the iterative and interactive character of the method eases this issue.

Thesis III.

I have developed an attribute-based network-like visualisation of frequent sequences (NBVFS) that supports quick understanding of event scenarios. [4]

The developed method transforms frequent event sequences to ego-networks that contain sequences having a common starting event, which is placed in the centre of the network. As every sequence may have different event expansions, the branches result in a tree structure. The key point in the method is how to place the nodes (the n-long sequences) within the network. I used the frequency (the *support*) of the sequences and their similarity to each other. Three visualisation methods were developed:

- **NBVFS-WN** (**N**etwork-**B**ased **V**isualization of **F**requent **S**equences - **W**eighted **N**etwork): this method uses a confidence-based adjacency matrix and the calculated metrics of the sequences. It results in a weighted network where those sequences are connected that are direct extensions of each other. The weight is the confidence value of the transition between the two connected sequences. This kind of visualisation helps to understand the conditions and consequences of occurring events.

- **NBVFS-CM** (**N**etwork-**B**ased **V**isualization of **F**requent **S**equences - **C**onfidence-based **M**DS projection): this method uses a similarity calculation, using transition confidence values for similarity measurement. The adjacency matrix must be enriched regarding the nondirectly connected sequences using transitive distance calculation. The positions of the nodes on the network are calculated with Multidimensional scaling, and the more similar the two sequences are, the closer they will be presented on the network. This kind of network representation contains more information about the relationship of sequences.

- **NBVFS-TM** (**N**etwork-**B**ased **V**isualization of **F**requent **S**equences - **T**ransaction-based **M**DS projection): the positions of the nodes are calculated with MDS identically to the NBVFS-CM method. The process is more or less the same, but the similarity calculation is based on the overlap of their common supporting transactions. This approach is closer to process mining, as transactions can be taken as traces and can provide feedback to the mining process.

This network-like visualisation of time-series-type event databases has many possible applications. The introduced method is a goal-oriented analysis tool to extract useful knowledge from the event database by visualising the event sequences. Due to its interactive character, it can successfully support iterative data analysis tasks. For example, it enables the adjustment of the frequent sequence pattern mining parameters. The proposed visualisation method allows for quick recognition of relevant event chains and key events with their most important attributes, such as support and confidence. This kind of information interpretation besides the support in cognitive information processing, can speed up the parameter identification step of machine learning model building, for example. It can also validate trace rules and other parameters used in process discovery tasks with respect to process mining. It has to be customised for the exact purpose of the task.

Thesis IV.

I have proposed a machine learning-based method to develop human-centric and process-aware HMI displays. [5]

The developed method aims to support the application of the *Navigation and Layout* principle of the ISA 101 standard, namely the *grouping related elements together* aspect. The principle of the iterative method developed is to use the combination of proper frequent pattern and process mining techniques. The proposed HMI displays and functionalities are based on the combined interpretation of P&I diagrams, process deviation models based on typical event/alarm sequences, and process control behaviour models based on typical responsive operator action sequences.

Alarm signals, operator actions, and display actions from a hydrofluoric acid alkylation plant were analysed. The log file was partitioned based on frequent sequential patterns of the stored events. Process control models were explored with process mining tools. A network-like visualisation of a typical process enabled a comparison of the number of workflows and the number of displays where the workflows are represented. A new, higher-level display was recommended based on the models gained and the existing structure of the HMI displays.

Even without a deep understanding of the system, the presented method proved effective. However, manual work is needed to evaluate the displays, resulting in subjective decisions. The method can be upgraded to define objective key performance indicators that support solving the minimisation problem.

8.3 Future application of the results

Although the developed methods were successfully applied in alarm and process analysis tasks, there are additional application areas and potential future research directions. The developed methods can be used independently in other research areas, for example, in EDU-mining.

By comparing the actual online process data with formerly defined event scenarios, a semi- or fully automated online HAZOP analysis tool can be built. The gained process patterns can be compared to a reference model that allows the evaluation of the work quality of the operators and supports the development of the Operator Training System (OTS).

Another potential future research direction is the implementation of machine learning in SCADA systems, which is an actual topic in the industry, as more and more solutions will be assisted by artificial intelligence. The proposed method is capable of processing acquired data and finding patterns among the event chains that lead to failures and the response actions of the operators. Using ML solutions enables the continuous dynamic and adaptive development of industrial process control systems.

The topic of trend charts can also be an interesting area. With the help of ML-supported solutions, process-variable dependencies can be gained from historical data, adding a predictive function to the system. For example, if the operator calls a trend chart of a specific process variable, an intelligent system can offer other trend charts, which may be useful in the actual control activity.

Publications

1. L. Bántay, G. Dörgő, F. Tandari and J. Abonyi, "Simultaneous Process Mining of Process Events and Operator Actions for Alarm Management", *Complexity, Frontiers in Data-Driven Methods for Understanding, Prediction, and Control of Complex Systems*, Vol. 2022, 2022.

2. L. Bántay and J. Abonyi, "Frequent pattern mining-based log file partition for process mining", *Engineering Applications of Artificial Intelligence*, Vol 123., Part A, 2023.

3. L. Bántay, N. Sas, G. Dörgő and J. Abonyi, "Sequence Compression and Alignment-Based Process Alarm Prediction", *Industrial & Engineering Chemistry Research*, Vol. 62(27), 10577-10586, 2023.

4. L. Bántay and J. Abonyi, "Machine Learning-Supported Designing of Human–Machine Interfaces", *Applied Sciences*, *New Insights into Human-Computer Interaction*, Vol. 14(4), 2024.

5. L. Bántay and J. Abonyi, "Network-based visualisation of frequent sequences", *PLOS ONE*, Vol. 19(5), 2024.

Bibliography

[1] Wil Van Der Aalst. *Discovery, Conformance and Enhancement of Business Processes*. Springer: Berlin, Heidelberg, Germany, 2011.

[2] Maryam Amiri, Leyli Mohammad-Khanli, and Raffaela Mirandola. "A sequential pattern mining model for application workload prediction in cloud environment". In: *Journal of Network and Computer Applications* 105 (2018), pp. 21–62.

[3] János Abonyi, Richárd Károly, and Gyula Dörgö. "Event-tree based sequence mining using LSTM deep-learning model". In: *Complexity* 2021 (2021), pp. 1–24.

[4] G. Dorgo and J. Abonyi. "Sequence Mining Based Alarm Suppression". In: *IEEE Access* 6 (2018), pp. 15365–15379.

[5] Gy. Dorgo et al. "Learning operation strategies from alarm management systems by temporal pattern mining and deep learning". In: *28th European Symposium on Computer Aided Process Engineering (ESCAPE28)*. Ed. by A. Friedl et al. Vol. 43. 2018, pp. 1003–1008.

[6] Wenkai Hu et al. "Process Discovery of Operator Actions in Response to Univariate Alarms**This work was supported by the Natural Sciences and Engineering Research Council of Canada via the CRD program." In: *IFAC-PapersOnLine* 49.7 (2016). 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016, pp. 1026–1031. ISSN: 2405-8963.

[7] D. S. Kim, H. Shinbo, and H. Yokota. "An alarm correlation algorithm for network management based on root cause analysis". In: *13th International Conference on Advanced Communication Technology (ICACT2011)*. Feb. 2011, pp. 1233–1238.

[8] Naseeb Ahmed Adnan, Iman Izadi, and Tongwen Chen. "On expected detection delays for alarm systems with deadbands and delay-timers". In: *Journal of Process Control* 21.9 (2011), pp. 1318–1331. ISSN: 0959-1524.

[9] Hao Zang, Fan Yang, and Dexian Huang. "Design and Analysis of Improved Alarm Delay-Timers". In: *IFAC-PapersOnLine* 48.8 (2015). 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015, pp. 669–674. ISSN: 2405-8963.

[10] Iman Izadi et al. "A Framework for Optimal Design of Alarm Systems". In: *IFAC Proceedings Volumes* 42.8 (2009). 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, pp. 651–656. ISSN: 1474-6670.

[11] Wenkai Hu, Jiandong Wang, and Tongwen Chen. "A new method to detect and quantify correlated alarms with occurrence delays". In: *Computers & Chemical Engineering* 80 (2015), pp. 189–198. ISSN: 0098-1354.

[12] Shiqi Lai and Tongwen Chen. "A method for pattern mining in multiple alarm flood sequences". In: *Chemical Engineering Research and Design* 117 (2017), pp. 831–839. ISSN: 0263-8762.

[13] Gyula Dorgo, Peter Pigler, and Janos Abonyi. "Understanding the importance of process alarms based on the analysis of deep recurrent neural networks trained for fault isolation". In: *Journal of Chemometrics* 32.4 (2018), e3006. ISSN: 0886-9383.

[14] Gyula Dorgo, Ahmet Palazoglu, and Janos Abonyi. "Decision trees for informative process alarm definition and alarm-based fault classification". In: *Process Safety and Environmental Protection* 149 (2021), pp. 312–324. ISSN: 0957-5820.

[15] G. Dorgo and J. Abonyi. "Sequence Mining Based Alarm Suppression". In: *IEEE Access* 6 (2018), pp. 15365–15379. ISSN: 2169-3536.

[16] Wenkai Hu et al. "Process Discovery of Operator Actions in Response to Univariate Alarms". In: *IFAC-PapersOnLine* 49.7 (2016), pp. 1026–1031. ISSN: 2405-8963.

[17] Hao Zang, Fan Yang, and Dexian Huang. "Design and Analysis of Improved Alarm Delay-Timers". In: *IFAC-PapersOnLine* 48.8 (2015), pp. 669–674. ISSN: 2405-8963.

[18] Naseeb Ahmed Adnan, Iman Izadi, and Tongwen Chen. "On expected detection delays for alarm systems with deadbands and delay-timers". In: *Journal of Process Control* 21.9 (2011), pp. 1318–1331. ISSN: 0959-1524.

[19] Iman Izadi et al. "A Framework for Optimal Design of Alarm Systems". In: *IFAC Proceedings Volumes* 42.8 (2009), pp. 651–656. ISSN: 1474-6670.

[20] P. Duan et al. "Direct Causality Detection via the Transfer Entropy Approach". In: *IEEE Transactions on Control Systems Technology* 21.6 (2013), pp. 2052–2066. ISSN: 1558-0865.

[21] F. Yang et al. "Improved correlation analysis and visualization of industrial alarm data". In: *ISA Transactions* 51.4 (2012), pp. 499–506. ISSN: 0019-0578.

[22] M. Fani Sani, S.J. van Zelst, and W.M.P. van der Aalst. "Applying sequence mining for outlier detection in process mining". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11230 LNCS (2018), pp. 98–116.

[23] JW Vasquez Capacho et al. "Alarm management via temporal pattern learning". In: *Engineering Applications of Artificial Intelligence* 65 (2017), pp. 506–516.

[24] Sylvie Charbonnier, Nabil Bouchair, and Philippe Gayet. "Fault template extraction to assist operators during industrial alarm floods". In: *Engineering Applications of Artificial Intelligence* 50 (2016), pp. 32–44.

[25] Gyula Dorgo and Janos Abonyi. "Sequence mining based alarm suppression". In: *IEEE Access* 6 (2018), pp. 15365–15379.

[26] Bart Baesens, Stijn Viaene, and Jan Vanthienen. "Post-processing of association rules". In: *Prace Naukowe Akademii Ekonomicznej we Wrocławiu* 850 (2000), pp. 159–173.

[27] Andreas Theissler et al. "ConfusionVis: Comparative evaluation and selection of multi-class classifiers based on confusion matrices". In: *Knowledge-Based Systems* 247 (2022), p. 108651.

[28] Omar A Alzubi et al. "An optimal pruning algorithm of classifier ensembles: dynamic programming approach". In: *Neural Computing and Applications* 32 (2020), pp. 16091–16107.

[29] Yoones A Sekhavat and Orland Hoeber. "Visualizing association rules using linked matrix, graph, and detail views". In: (2013).

[30] Wolfgang Jentner and Daniel A Keim. *Visualization and visual analytic techniques for patterns*. Springer, 2019.

[31] Bram CM Cappers and Jarke J van Wijk. "Exploring multivariate event sequences using rules, aggregations, and selections". In: *IEEE transactions on visualization and computer graphics* 24.1 (2017), pp. 532–541.

[32] Zhicheng Liu et al. "Coreflow: Extracting and visualizing branching patterns from event sequences". In: *Computer Graphics Forum*. Vol. 36. 3. Wiley Online Library. 2017, pp. 527–538.

[33] Yuanzhe Chen, Panpan Xu, and Liu Ren. "Sequence synopsis: Optimize visual summary of temporal event data". In: *IEEE transactions on visualization and computer graphics* 24.1 (2017), pp. 45–55.

[34] Wolfgang Jentner et al. "Making machine intelligence less scary for criminal analysts: reflections on designing a visual comparative case analysis tool". In: *The Visual Computer* 34 (2018), pp. 1225–1241.

[35] Wolfgang Jentner et al. "Feature alignment for the analysis of verbatim text transcripts". In: *EuroVA 2017: EuroVis Workshop on Visual Analytics*. 2017, pp. 13–18.

[36] Hanan R Alnjar. "Data visualization metrics between theoretic view and real implementations: A review". In: *DYSONA-Applied Science* 1.2 (2020), pp. 43–50.

[37] Katerina Vrotsou and Aida Nordman. "Exploratory visual sequence mining based on pattern-growth". In: *IEEE transactions on visualization and computer graphics* 25.8 (2018), pp. 2597–2610.

[38] Sarah Higginson et al. "Diagramming social practice theory: An interdisciplinary experiment exploring practices as networks". In: *Indoor and Built Environment* 24.7 (2015), pp. 950–969.

[39] Anton Yeshchenko and Jan Mendling. "A survey of approaches for event sequence analysis and visualization". In: *Information Systems* (2023), p. 102283.

[40] Cedric Krause et al. "Visually Abstracting Event Sequences as Double Trees Enriched with Category-Based Comparison". In: *Computer Graphics Forum*. Wiley Online Library. 2023.

[41] László Bántay and János Abonyi. "Frequent pattern mining-based log file partition for process mining". In: *Engineering Applications of Artificial Intelligence* 123 (2023), p. 106221.

[42] Siti Amira Kamizi. "UI/UX of Human-Machine Interface for Industrial Application: Review and Preleminary Design". In: *Proceedings of the FCSIT UNIMAS FYP Symposium, Malaysia*. 2021, pp. 28–30.

[43] Ziqiu Kang, Cagatay Catal, and Bedir Tekinerdogan. "Machine learning applications in production lines: A systematic literature review". In: *Computers & Industrial Engineering* 149 (2020), p. 106773. ISSN: 0360-8352.

[44] Trevor Kistan, Alessandro Gardi, and Roberto Sabatini. "Machine learning and cognitive ergonomics in air traffic management: Recent developments and considerations for certification". In: *Aerospace* 5.4 (2018), p. 103.

[45] Francisco Pérez-Reynoso et al. "Pattern Recognition of EMG Signals by Machine Learning for the Control of a Manipulator Robot". In: *Sensors* 22.9 (2022), p. 3424.

[46] Ossama Rashad, Omneya Attallah, and Iman Morsi. "A smart PLC-SCADA framework for monitoring petroleum products terminals in industry 4.0 via machine learning". In: *Measurement and Control* 55.7-8 (2022), pp. 830–848.

[47] Etienne Valette, Hind Bril El-Haouzi, and Guillaume Demesure. "Industry 5.0 and its technologies: A systematic literature review upon the human place into IoT- and CPS-based industrial systems". In: *Computers & Industrial Engineering* 184 (2023), p. 109426. ISSN: 0360-8352.

[48] Dimitris Mourtzis, John Angelopoulos, and Nikos Panopoulos. "The Future of the Human-Machine Interface (HMI) in Society 5.0". In: *Future Internet* 15.5 (2023). ISSN: 1999-5903.

[49] Bill Hollifield. "A High Performance HMI: Better Graphics for Operations Effectiveness". In: *2012 Water/Wastewater and Automation Controls Symposium*. 2012.

[50] American National Standards Institute. *ANSI-ISA-101.01-2015, Human Machine Interfaces for Process Automation Systems*. ISA, 2015.

[51] José Kadir Febrer-Hernández and José Hernández-Palancar. "Sequential pattern mining algorithms review". In: *Intelligent Data Analysis* 16.3 (2012), pp. 451–466.

[52] Gyula Dorgo et al. "Learning operation strategies from alarm management systems by temporal pattern mining and deep learning". In: *28th European Symposium on Computer Aided Process Engineering*. Ed. by Anton Friedl et al. Vol. 43. Computer Aided Chemical Engineering. Elsevier, 2018, pp. 1003–1008.

[53] T. Le and B. Vo. "The lattice-based approaches for mining association rules: a review". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 6.4 (2016), pp. 140–151.

[54] Wil van der Aalst. "Data Science in Action". In: *Process Mining: Data Science in Action*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 3–23. ISBN: 978-3-662-49851-4.

[55] Julian Theis et al. “Improving the In-Hospital Mortality Prediction of Diabetes ICU Patients Using a Process Mining/Deep Learning Architecture”. In: *IEEE Journal of Biomedical and Health Informatics* 26.1 (2022), pp. 388–399.

[56] Pierluigi Zerbino, Alessandro Stefanini, and Davide Aloini. “Process Science in Action: A Literature Review on Process Mining in Business Management”. In: *Technological Forecasting and Social Change* 172 (2021), p. 121021. ISSN: 0040-1625.

[57] Michael Werner, Michael Wiese, and Annalouise Maas. “Embedding process mining into financial statement audits”. In: *International Journal of Accounting Information Systems* 41 (2021), p. 100514. ISSN: 1467-0895.

[58] Rebeca Cerezo et al. “Process mining for self-regulated learning assessment in e-learning”. In: *Journal of Computing in Higher Education* 32 (2020), pp. 74–88.

[59] Volodymyr Leno et al. “Robotic Process Mining: Vision and Challenges”. In: *Business & Information Systems Engineering* 63 (2021), pp. 301–314.

[60] C.D.S. Garcia et al. “Process mining techniques and applications – A systematic mapping study”. In: *Expert Systems with Applications* 133 (2019). cited By 22, pp. 260–295.

[61] R. E. Kondo, E. de F. R. Loures, and E. A. P. Santos. “Process mining for alarm rationalization and fault patterns identification”. In: *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*. Sept. 2012, pp. 1–4.

[62] Ricardo Eiji Kondo et al. “Alarm Rationalization Based on Process Mining Techniques”. In: *Advanced Materials Research* 1061-1062 (Dec. 2014), pp. 1258–1265. ISSN: 1662-8985.

[63] Wil M.P. van der Aalst. “A practitioner’s guide to process mining: Limitations of the directly-follows graph”. In: *Procedia Computer Science* 164 (2019). CENTERIS 2019 - International Conference on ENTERprise Information Systems / ProjMAN 2019 - International Conference on Project MANagement / HCist 2019 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2019, pp. 321–328. ISSN: 1877-0509.

[64] J. R. Taylor. “Automated HAZOP revisited”. English. In: *Process Safety and Environmental Protection* 111 (2017), pp. 635–651.

[65] H. M. W. Verbeek et al. “XES, XESame, and ProM 6”. English. In: *Information Systems Evolution (CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers)*. Ed. by P. Soffer and E. Proper. Lecture Notes in Business Information Processing. Germany: Springer, 2011, pp. 60–75. ISBN: 978-3-642-17721-7.

[66] Boudewijn F. van Dongen and Wil M. P. van der Aalst. “EMiT: A Process Mining Tool”. In: *Applications and Theory of Petri Nets 2004*. Ed. by Jordi Cortadella and Wolfgang Reisig. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 454–463. ISBN: 978-3-540-27793-4.

[67] John Lee, Ian Cameron, and Maureen Hassall. "Improving process safety: What roles for Digitalization and Industry 4.0?" In: *Process Safety and Environmental Protection* 132 (2019), pp. 325–339. ISSN: 0957-5820.

[68] Saul B. Needleman and Christian D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins". In: *Journal of Molecular Biology* 48.3 (1970), pp. 443–453. ISSN: 0022-2836.

[69] Peter Grünwald. *The Minimum Description Length Principle*. 2007, pp. 26–29. ISBN: 9780262256292.

[70] Hoang Thanh Lam et al. "Mining Compressing Sequential Patterns". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 7.1 (2014), pp. 34–52.

[71] Nikolaj Tatti and Jilles Vreeken. "The long and the short of it: summarising event sequences with serial episodes". In: Association for Computing Machinery, 2012, pp. 462–470.

[72] Jilles Vreeken, Matthijs Van Leeuwen, and Arno Siebes. "Krimp: mining itemsets that compress". In: *Data Mining and Knowledge Discovery* 23 (2011), pp. 169–214.

[73] Bill R Hollifield and Eddie Habibi. *Alarm management: Seven effective methods for optimum performance*. Isa, 2007, pp. 92–96.

[74] Jiandong Wang and Tongwen Chen. "Main causes of long-standing alarms and their removal by dynamic state-based alarm systems". In: *Journal of Loss Prevention in the Process Industries* 43 (2016), pp. 106–119.

[75] Kai R Wang. "A Data-Driven Method to Discover Association Rules of Mode-Dependent Alarms". In: (2021), pp. 4–6.

[76] Richard J Simonson et al. "Impact of alarm management and automation on abnormal operations: A human-in-the-loop simulation study". In: *Applied Ergonomics* 100 (2022), p. 103670.

[77] Marcel F. Hinss, Anke M. Brock, and Raphaëlle N. Roy. "Cognitive effects of prolonged continuous human-machine interaction: The case for mental state-based adaptive interfaces". In: *Frontiers in Neuroergonomics* 3 (2022), pp. 3–4. ISSN: 2673-6195.

[78] Taek Kyu Kim et al. "Deep-learning-based alarm system for accident diagnosis and reactor state classification with probability value". In: *Annals of Nuclear Energy* 133 (2019), pp. 723–731.

[79] K. Ahmed et al. "Similarity Analysis of Industrial Alarm Flood Data". In: *IEEE Transactions on Automation Science and Engineering* 10.2 (2013), pp. 452–457. ISSN: 1558-3783.

[80] Yue Cheng, Iman Izadi, and Tongwen Chen. "Pattern matching of alarm flood sequences by a modified Smith–Waterman algorithm". In: *Chemical Engineering Research and Design* 91.6 (2013), pp. 1085–1094. ISSN: 0263-8762.

[81] Wenkai Hu, Jiandong Wang, and Tongwen Chen. "Fast Sequence Alignment for Comparing Industrial Alarm Floods. This work was partially supported by an NSERC CRD project, and the National Natural Science Foundation of China under grant No. 61061130559". In: *IFAC-PapersOnLine* 48.8 (2015), pp. 647–652. ISSN: 2405-8963.

[82] László Bántay et al. "Simultaneous Process Mining of Process Events and Operator Actions for Alarm Management". In: *Complexity* 2022 (2022).

[83] Mörchen and Fabian. "Unsupervised pattern mining from symbolic temporal data". In: *SIGKDD Explor. Newsl.* 9 (2007), p. 41.

[84] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. "Krimp: mining itemsets that compress". In: *Data Mining and Knowledge Discovery* 23.1 (2011), pp. 169–214. ISSN: 1573-756X.

[85] Matthijs Leeuwen, Jilles Vreeken, and Arno Siebes. "Identifying the Components". In: *Data Mining and Knowledge Discovery* 19 (2009), pp. 176–193. ISSN: 978-3-642-04179-2.

[86] Matthijs van Leeuwen and Arno Siebes. "StreamKrimp: Detecting Change in Data Streams". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Walter Daelemans, Bart Goethals, and Katharina Morik. Springer Berlin Heidelberg, pp. 672–687. ISBN: 978-3-540-87479-9.

[87] Vladimir Likic. "The Needleman-Wunsch algorithm for sequence alignment". In: *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne* (2008), pp. 1–46.

[88] Sean R Eddy. "What is dynamic programming?" In: *Nature biotechnology* 22.7 (2004), pp. 909–910.

[89] Sandeep R. Kondaveeti et al. "Quantification of alarm chatter based on run length distributions". In: *Chemical Engineering Research and Design* 91.12 (2013), pp. 2550–2558. ISSN: 0263-8762.

[90] D. Buddaraju. *Performance of Control Room Operators in Alarm Management*. Louisiana State University, 2011, pp. 8–12.

[91] Philippe Fournier-Viger et al. "The SPMF Open-Source Data Mining Library Version 2". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Bettina Berendt et al. Cham: Springer International Publishing, 2016, pp. 36–40. ISBN: 978-3-319-46131-1.

[92] Heidy M Marin-Castro and Edgar Tello-Leal. "Event log preprocessing for process mining: a review". In: *Applied Sciences* 11.22 (2021), p. 10556.

[93] A. Bolt and W.M.P. Van Der Aalst. "Multidimensional process mining using process cubes". In: *Lecture Notes in Business Information Processing* 214 (2015), pp. 102–116.

[94] A. Seeliger, T. Nolle, and M. Mühlhäuser. "Finding structure in the unstructured: Hybrid feature set clustering for process discovery". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11080 LNCS (2018), pp. 288–304.

[95] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. "Process mining with the heuristics miner-algorithm". In: *Technische Universiteit Eindhoven, Tech. Rep. WP* 166 (2006), pp. 1–34.

[96] Wil Van Der Aalst. *Process mining: data science in action*. Vol. 2. Springer: Berlin, Heidelberg, Germany, 2016.

[97] T. Vogelgesang, S. Rinderle-Ma, and H.-J. Appelrath. "A framework for interactive multidimensional process mining". In: *Lecture Notes in Business Information Processing* 281 (2017), pp. 23–35.

[98] Qun-Xiong Zhu et al. "Pattern Mining of Alarm Flood Sequences Using an Improved PrefixSpan Algorithm with Tolerance to Short-Term Order Ambiguity". In: *Industrial & Engineering Chemistry Research* 60.11 (2021), pp. 4375–4384.

[99] Prabhu Paulraj and Anbazhagan Neelamegam. "Improving business intelligence based on frequent itemsets using k-means clustering algorithm". In: *Networks and Communications (NetCom2013)*. Springer, 2014, pp. 243–254.

[100] Javier de San Pedro and Jordi Cortadella. "Mining structured Petri nets for the visualization of process behavior". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa (Italy), Apr 4-6*. 2016, pp. 839–846.

[101] Iq Reviessay Pulshashi et al. "Slice and Connect: Tri-Dimensional Process Discovery with Case Study of Port Logistics Process". In: *Procedia Computer Science* 72 (2015), pp. 461–468.

[102] Libor Juhaňák, Jiří Zounek, and Lucie Rohlíková. "Using process mining to analyze students' quiz-taking behavior patterns in a learning management system". In: *Computers in Human Behavior* 92 (2019), pp. 496–506. ISSN: 0747-5632.

[103] Alberto Guastalla et al. "Workshift Scheduling Using Optimization and Process Mining Techniques: An Application in Healthcare". In: *2022 Winter Simulation Conference (WSC)*. 2022, pp. 1116–1127.

[104] Raffaele Conforti, Marcello La Rosa, and Arthur HM ter Hofstede. "Filtering out infrequent behavior from business process event logs". In: *IEEE Transactions on Knowledge and Data Engineering* 29.2 (2016), pp. 300–314.

[105] Sebastiaan J van Zelst et al. "Filtering spurious events from event streams of business processes". In: *International Conference on Advanced Information Systems Engineering, Tallinn (Estonia), June 11-15*. 2018, pp. 35–52.

[106] Niek Tax et al. "Generating time-based label refinements to discover more precise process models". In: *Journal of Ambient Intelligence and Smart Environments* 11.2 (2019), pp. 165–182.

[107] Dominik Andreas Fischer et al. "Enhancing event log quality: Detecting and quantifying timestamp imperfections". In: *International Conference on Business Process Management, Sevilla (Spain), September 13-18*. 2020, pp. 309–326.

[108] Jochen De Weerdt et al. "Active trace clustering for improved process discovery". In: *IEEE Transactions on Knowledge and Data Engineering* 25.12 (2013), pp. 2708–2720.

[109] Phuong Nguyen et al. “Process trace clustering: A heterogeneous information network approach”. In: *Proceedings of the 2016 SIAM International Conference on Data Mining, Florida (USA), May 5-7*. 2016, pp. 279–287.

[110] David Chapela-Campa, Manuel Mucientes, and Manuel Lama. “Discovering infrequent behavioral patterns in process models”. In: *International Conference on Business Process Management, Barcelona (Spain), September 10-15*. 2017, pp. 324–340.

[111] RP Jagadeesh Chandra Bose and Wil MP Van der Aalst. “Abstractions in process mining: A taxonomy of patterns”. In: *International conference on business process management, Ulm (Germany), September 7-10*. 2009, pp. 159–175.

[112] B.R. Mehta and Y.J. Reddy. “Chapter 21 - Alarm management systems”. In: *Industrial Process Automation Systems*. Ed. by B.R. Mehta and Y.J. Reddy. Oxford: Butterworth-Heinemann, 2015, pp. 569–582. ISBN: 978-0-12-800939-0.

[113] Gyula Dorgo et al. “Quality vs. quantity of alarm messages-How to measure the performance of an alarm system”. In: *Chemical Engineering Research and Design* 173 (2021), pp. 63–80.

[114] Joos CAM Buijs, Boudewijn F van Dongen, and Wil MP van der Aalst. “Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity”. In: *International Journal of Cooperative Information Systems* 23.01 (2014), p. 1440001.

[115] Fabian Rojas Blum. “Metrics in process discovery”. In: *Tech. Rep. Technical Report TR/DCC-2015-6, Computer Science Dept., University of Chile* (2015).

[116] Alessandro Berti and Wil MP van der Aalst. “Reviving Token-based Replay: Increasing Speed While Improving Diagnostics.” In: *ATAED@ Petri Nets/ACSD*. 2019, pp. 87–103.

[117] Jorge Munoz-Gama and Josep Carmona. “A fresh look at precision in process conformance”. In: *International Conference on Business Process Management, Hoboken (USA/NJ), September 13-16*. 2010, pp. 211–226.

[118] Wil MP Van der Aalst. “The application of Petri nets to workflow management”. In: *Journal of circuits, systems, and computers* 8.01 (1998), pp. 21–66.

[119] HMW Verbeek and R Medeiros de Carvalho. “Log skeletons: A classification approach to process discovery”. In: *arXiv preprint arXiv:1806.08247* (2018).

[120] Peter V Marsden. “The reliability of network density and composition measures”. In: *Social Networks* 15.4 (1993), pp. 399–421.

[121] Jianxi Luo and Christopher L Magee. “Detecting evolving patterns of self-organizing networks by flow hierarchy measurement”. In: *Complexity* 16.6 (2011), pp. 53–61.

[122] Vito Latora and Massimo Marchiori. “Efficient Behavior of Small-World Networks”. In: *Phys. Rev. Lett.* 87 (19 Oct. 2001), p. 198701.

[123] Ana Rocío Cárdenas Maita et al. “A systematic mapping study of process mining”. In: *Enterprise Information Systems* 12.5 (2018), pp. 505–549.

[124] Peng Cui et al. “A survey on network embedding”. In: *IEEE transactions on knowledge and data engineering* 31.5 (2018), pp. 833–852.

[125] Chuanren Liu et al. “Temporal Skeletonization on Sequential Data: Patterns, Categorization, and Visualization”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.1 (2015), pp. 211–223.

[126] Peter Matyus et al. “Visualization of Fuzzy Association Rules Representing High-Dimensional Problems”. In: *11th IPMU International Conference*. 2006, pp. 2–7.

[127] RB Bapat, D Kalita, and S Pati. “On weighted directed graphs”. In: *Linear Algebra and its Applications* 436.1 (2012), pp. 99–111.

[128] Xiaodi Huang and Wei Lai. “Clustering graphs for visualization via node similarities”. In: *Journal of Visual Languages & Computing* 17.3 (2006), pp. 225–253.

[129] Bianka Kovács and Gergely Palla. “Model-independent embedding of directed networks into Euclidean and hyperbolic spaces”. In: *Communications Physics* 6.1 (2023), p. 28.

[130] Ágnes Vathy-Fogarassy and János Abonyi. *Graph-based clustering and data visualization algorithms*. Vol. 13. Springer, 2013.

[131] Engineering Equipment et al. *Alarm Systems: A Guide to Design, Management and Procurement*. EEMUA publication. E E M U A (Engineering Equipment & Materials Users Association), 2015. ISBN: 9780859311557.

[132] Kabir Ahmed et al. “Similarity analysis of industrial alarm flood data”. In: *IEEE Transactions on Automation Science and Engineering* 10.2 (2013), pp. 452–457.

[133] M. Dees and Boudewijn van Dongen. *BPI Challenge 2016: Clicks NOT Logged In*. 2016. URL: `https://data.4tu.nl/articles/dataset/BPI_Challenge_2016_Clicks_NOT_Logged_In/12708596/1`.

[134] Philippe Fournier-Viger et al. “Fast vertical mining of sequential patterns using co-occurrence information”. In: *Advances in Knowledge Discovery and Data Mining: 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I 18*. Springer. 2014, pp. 40–52.

[135] Daniel Reguera-Bakhache et al. “Data-driven industrial human-machine interface temporal adaptation for process optimization”. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. IEEE. 2020, pp. 518–525.

[136] Jesus Anselmo Fortoul-Diaz et al. “A Smart Factory Architecture Based on Industry 4.0 Technologies: Open-Source Software Implementation”. In: *IEEE Access* (2023).

[137] Guoqiang Sun et al. “OL-JCMSR: A Joint Coding Monitoring Strategy Recommendation Model Based on Operation Log”. In: *Mathematics* 10.13 (2022), p. 2292.

[138] Chiu-Hsiang Lin et al. “Human-robot collaboration empowered by hidden semi-Markov model for operator behaviour prediction in a smart assembly system”. In: *Journal of Manufacturing Systems* 62 (2022), pp. 317–333.

[139] Leon Urbas, Michael Obst, and Markus Stöss. “Formal models for high performance HMI engineering”. In: *IFAC Proceedings Volumes* 45.2 (2012), pp. 854–859.

[140] Raphael Hägle et al. “A Methodology for the Systematic Selection of Human-Machine Interface Device Types in Production Machinery Development”. In: *Procedia CIRP* 119 (2023). The 33rd CIRP Design Conference, pp. 975–980. ISSN: 2212-8271.

[141] Tuan-Anh Tran et al. “Retrofitting-Based Development of Brownfield Industry 4.0 and Industry 5.0 Solutions”. In: *IEEE Access* 10 (2022), pp. 64348–64374.

[142] Francisco Javier Folgado, Isaías González, and Antonio José Calderón. “Data acquisition and monitoring system framed in Industrial Internet of Things for PEM hydrogen generators”. In: *Internet of Things* 22 (2023), p. 100795. ISSN: 2542-6605.

[143] László Bántay et al. “Sequence Compression and Alignment-Based Process Alarm Prediction”. In: *Industrial & Engineering Chemistry Research* 62.27 (2023), pp. 10577–10586.

[144] Valeria Villani et al. “A general methodology for adapting industrial HMIs to human operators”. In: *IEEE Transactions on Automation Science and Engineering* 18.1 (2019), pp. 164–175.

[145] Albert-László Barabási. “Network science”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1987 (2013), p. 20120375.

[146] Oleksandr Shchur and Stephan Günnemann. “Overlapping community detection with graph neural networks”. In: *arXiv preprint arXiv:1909.12201* (2019).

[147] Xiangyi Teng, Jing Liu, and Mingming Li. “Overlapping Community Detection in Directed and Undirected Attributed Networks Using a Multiobjective Evolutionary Algorithm”. In: *IEEE Transactions on Cybernetics* 51.1 (2021), pp. 138–150.

[148] P. Fournier-Viger et al. “The SPMF Open-Source Data Mining Library Version 2.” In: *Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016) Part III*. Springer. 2016, pp. 36–40.

[149] Wil MP Van Der Aalst, Hajo A Reijers, and Minseok Song. “Discovering social networks from event logs”. In: *Computer Supported Cooperative Work (CSCW)* 14 (2005), pp. 549–593.

[150] Jakob Nielsen. “Usability inspection methods”. In: *Conference companion on Human factors in computing systems*. 1994, pp. 413–414.

[151] László Gadár and János Abonyi. “Frequent pattern mining in multidimensional organizational networks”. In: *Scientific reports* 9.1 (2019), p. 3322.

List of Figures

List of Tables