2010

Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

Savo Kordic  
*Edith Cowan University*

Chiou Peng Lam  
*Edith Cowan University*

Jitian Xiao  
*Edith Cowan University*

Huaizhong Li

Dynamic and Advanced Data Mining for Progressing Technological Development: Innovations and Systemic Approaches

A B M Shawkat Ali
Central Queensland University, Australia

Yang Xiang
Central Queensland University, Australia
Chapter 2

Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

Savo Kordic  
_Edith Cowan University, Australia_

Chiong Peng Lam  
_Edith Cowan University, Australia_

Jitian Xiao  
_Edith Cowan University, Australia_

Huaizhong Li  
_Wenzhou University, China_

ABSTRACT

The productivity of chemical plants and petroleum refineries depends on the performance of alarm systems. Alarm history collected from distributed control systems (DCS) provides useful information about past plant alarm system performance. However, the discovery of patterns and relationships from such data can be very difficult and costly. Due to various factors such as a high volume of alarm data (especially during plant upsets), huge amounts of nuisance alarms, and very large numbers of individual alarm tags, manual identification and analysis of alarm logs is usually a labor-intensive and time-consuming task. This chapter describes a data mining approach for analyzing alarm logs in a chemical plant. The main idea of the approach is to investigate dependencies between alarms effectively by considering the temporal context and time intervals between different alarm types, and then employing a data mining technique capable of discovering patterns associated with these time intervals. A prototype has been implemented to allow an active exploration of the alarm grouping data space relevant to the tags of interest.

DOI: 10.4018/978-1-60566-908-3.ch002
INTRODUCTION

Complex industrial processes such as chemical plants and petroleum refineries produce large amounts of alarm information on a daily basis, due to the many different types of alarm that can occur in a relatively short period of time. Additionally, in the last two decades, “software alarms” were introduced in distributed control systems. These can be implemented simply by changing computer settings, which is an inexpensive process compared to installing “real alarms”. Thus many process engineers choose to add extra alarm points to the existing DCS to monitor anything about which they may be concerned. Consequently, in many emergency situations excessive numbers of inappropriate alarms are generated, making the alarm system difficult to use when it is most urgently needed. A recent example is the 2005 explosion at the BP Texas City Refinery (OSHA, 2005) which left 15 people dead. BP North America was found to be responsible for the tragedy by (BP, 2007), and was fined a record $50 million and spent more than $1 billion for the inspection and refurbishment of all main process units in the refinery.

According to Shook (2004) the typical alarm management strategy for monitoring an alarm system includes collecting all alarms from all consoles, performing analysis to identify “nuisance alarm” occurrences, assessing the original performance, and then spending a few days over the period of a month to detect and reconfigure the worst nuisance alarms. The final task is to calculate statistics based on monthly alarm occurrences in order to show the frequency of alarms. While it is possible to manually extract the information required for incident reviews or alarm rationalization, the extensive quantity and complexity of data (typically collected from more than one database) have made the analysis and decomposition a very laborious task.

It is possible to identify frequent patterns on the basis of event changes over time by using temporal windows. However, a typical chemical alarm database is characterized by a large search space with skewed frequency distribution. Furthermore, since there can be several levels of alarms in an industrial plant, the discovered patterns or associations between high frequency alarms may indicate some trivial preventive actions and not necessarily provide unexpected or useful information about the state of the chemical process, while at the same time high-priority safety alarms which have a low frequency may be discarded. In contrast, setting a low frequency threshold level uniformly for all alarm tags might not only be computationally very expensive (with thousands of generated rules) but also there could be many spurious relationships between different support level alarms.

Therefore, despite a wealth of plant information, the data mining task of finding meaningful patterns and interesting relationships in chemical databases is difficult. This chapter presents a novel approach to developing techniques and tools that support alarm rationalization in legacy systems by extracting relationships of alarm points from alarm data in a cost-effective way.

RELATED WORK

Temporal data mining (Roddick & Spiliopoulou, 2002) is concerned with the analysis of sequences of events or itemsets in large sequential databases, where records are either chronologically ordered lists of events or indexed by transaction-time, respectively. The task of temporal data mining is different from the non-temporal discovery of relationships among itemsets such as association rules (Agrawal, Imielinski, & Swami, 1993), since of particular interest in temporal data mining is the discovery of causal
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

relationships and temporal patterns and rules. Thus techniques for finding temporal patterns take time into account by observing differences in the temporal data.

In temporal data mining, the discovery process usually includes sliding time windows or time constraints. Srikant & Agrawal (1996) developed the GSP algorithm which generalizes the sequential pattern framework by including the maximum and minimum time periods between adjacent elements of the sequential patterns and allows items to be selected within a user-specified transaction-time window. The idea of Zaki (2000) was to incorporate into the mining process additional constraints such as the maximum length of a pattern, and constraints on an item’s inclusion in a sequence.

Over the last decade other researchers extended the sequential pattern mining framework in various ways such as considering multidimensionality and periodicity. Lu, Han, & Feng (1998) proposed the use of multidimensional inter-transaction association rules where essentially dimensional attributes such as time and location were divided into equal length intervals. In the case of cyclic association rules (Özden, Ramaswamy, & Silberschatz, 1998) the sequences were segmented into a range of equally spaced time-periods and then these were used for discovering regular cyclic variations over time. Instead of looking for full periodic patterns, Han, Dong, & Yin (1999) considered only a set of desired time periods called partial periodic patterns. Ma & Hellerstein (2001) generalized the concept of partial periodicity by taking into account time tolerances, and Cao, Cheung, & Mamoulis (2004) proposed a method for automatic discovery of frequent partial periodic patterns by using a structure called the abbreviated list table.

More relevant to our research, based on a real plant mining problem is discovery of temporal rules in telecommunications networks where data is given as a sequence of events ordered with respect to the time of alarm occurrence. One of the main difficulties when analyzing event sequences in WINEPI (Mannila, Toivonen, & Verkamo, 1995) was to specify the window size within which an episode (i.e. a partially ordered set of events) must occur. If the window is too small information will be lost or if the window is too big unrelated alarms could be included, making the process of detecting frequent episodes increasingly difficult. Basically there are three types of episodes: a serial episode which occurs in a fixed order (i.e. time-ordered events), a parallel episode which is an unordered collection of events (i.e. trivial partial order), and a composite episode which is built from a serial and a parallel episode. While the WINEPI algorithm calculates the frequency of an episode as the fraction of windows in which the episode occurs, the subsequent algorithm MINEPI (Mannila & Toivonen, 1996) directly calculates the frequency of an episode β in a given event sequence s as the number of minimal occurrences (mo) of β in the sequence s, within a given time bound. Therefore, the frequency of an episode will depend on the user-given time bound between events. Bettini, Wang, & Jajodia (1998) generalized the framework of mining temporal relationships by introducing time-interval constraints on events, and representing event structures as a rooted directed acyclic graph.

More recently, Casas-Garriga (2003) described the concept of unbounded episodes where the proposed algorithm automatically extends the window width during the mining process based on the size of the episodes being counted. Laxman, Sastry, & Unnikrishnan (2007a) introduced the non-overlapping occurrences counting method which has the advantage in comparison to overlapping methods in terms of actual space and efficiency during the discovery of episodes. Some recent work in temporal data mining also focuses on the significance of discovered episodes. For instance, Gwadera, Atallah, & Szpankowski (2005) showed that the lower and upper thresholds of statistically significant episodes can be determined by comparing the actual observed frequency with the estimated frequency generated by a Bernoulli distribution model. It is also desirable to consider the duration of important events. An application of this general idea to data from the manufacturing plants of General Motors is presented by Laxman, Sastry,
Unnikrishnan (2007b). In this chapter we focus on the analysis of alarm sequences in a chemical plant, in which not only the duration of events but also the time between events is considered. Critical to our study are the duration of activation and return alarm intervals, and the differences in the distribution of events within time-intervals. Such information is essential for the elimination of irrelevant data points in a chemical process sense.

THEORETICAL FRAMEWORK

In this section a framework that facilitates understanding of the phenomena under investigation is discussed.

Alarm Events

*Alarms* are used as a mechanism for alerting operators to take actions that would alleviate or prevent an abnormal situation. Alarm data is a discrete type of data that will be generated only if a signal exceeds its limits.

Alarm Database and Event Intervals

Alarm databases in a chemical plant consist of records of time-stamped alarm events which include activation (ALM), return (RTN) and acknowledge (ACK) event types. We assume that a possible alarm sequence could be “ALM” → “RTN”, or “ALM” → “ACK” → “RTN”, but not “ALM” → “ACK” → “ALM”. Note that the *acknowledge* type only indicates an operator action to stop the alarm going off, but no remedial action is taken, thus it is not considered in our research.

An alarm sequence can be seen as a series of event types occurring at specific times. The role of time is crucial, so a successful conceptual framework cannot rely purely on simple time points representing the instantaneous events (i.e. points at which alarm tags activate). A design that would be adequate should allow the representation of alarm events with *duration*. Since any two event types are separated in time, each interval between events can be seen as a temporal window. For simplicity and without loss of generality, let us consider only three alarms, namely, *TAG 1*, *TAG 2* and *TAG 3* in a chemical process. Alarms which are activated after the event when *TAG 1* is activated and before *TAG 1* is returned, form an *activation-return* (A-R) temporal window. The recognition that *TAG 2* and *TAG 3* for example, also occur within the (A-R) interval of *TAG 1*, implies change in both *TAG 2* and *TAG 3* over the duration of *TAG 1*. Although there may not exist both a causal and a temporal order, the main principle underlying our design is that *TAG 1* must precede *TAG 2* and *TAG 3*.

Temporal Orders and Intervals

The study design assumes that the temporal order between alarm events is preserved, and changes in alarms are manifested as disturbances until the system is returned to a normal state. Obviously, we want to investigate two questions when an alarm activation event (for example, *TAG 1* activation) occurs:
1. What are the next likely set of alarms or alarm groups which will be activated after the activation of \textit{TAG 1}?
2. Are the alarms or alarm groups identified in Question 1 associated with the activation of \textit{TAG 1}?

The answer to the first question provides the statistical prediction for the occurrence of alarm events. For example, if it is known that \textit{TAG 2} is likely to be activated after the activation of \textit{TAG 1}, and \textit{TAG 2} is a critical alarm, then necessary precautions can be taken after the event of \textit{TAG 1}'s activation to avoid a disastrous consequence.

In order to answer the second question, we consider the nature of the alarm events in a chemical process. In a simplistic sense, if the activation of \textit{TAG 1} causes the activation of \textit{TAG 2}, then when \textit{TAG 1} returns, the cause for \textit{TAG 2} is eliminated. Therefore, it can be expected that \textit{TAG 2} will return shortly after \textit{TAG 1} returns. For this reason, an appropriate conceptual framework should also support representation of return events. The return event (RTN) of \textit{TAG 1} marks the beginning of a verifying window where the window \textit{time-width} is the process lag length. All alarm return events in the verifying window form the members in the verifying return-time-width interval (R-W) window which is associated with the activation time window. There must be a finite amount of time for all tags associated with the problem to return. If \textit{TAG 1} is activated again before the process lag is reached, the activation of \textit{TAG 1} marks the end of the verifying return-activation (R-A) window. Activation events in the activation-return (A-R) window are pruned if their return events do not appear in the associated verifying window.

**Event Interval Filtering**

Two strategies, incorporating chemical process related heuristics, are designed to remove spurious data points. These are:

**Return-Point Strategy (R-p)**

The rationale used here is that a dependent variable should return after the independent variable (i.e. a cause alarm) has returned. If the cause of the problem is resolved then all subsequent alarm tags related to the problem should return some time after the initial alarm tag associated with the cause has returned.

It is expected that after a transportation lag, return of the cause alarm will propagate to all associated alarms and thus cause the associated alarms to return. Thus, the relationship can be described in terms of the intersection between activation events in the activation-return (A-R) window and return events in the return-activation (R-A) or return-time-width interval (R-W) window. Formally we could write this criterion as

\[ R-p = \text{Activations (A-R)} \cap \text{Returns (R-A, or R-W)} \]

**Activation-Point Strategy (A-p)**

The heuristic used here relates to activations of alarm tags after the cause of the problem has been resolved. The rationale is that dependant variables (consequence alarms) once returned, should not re-activate after the independent variable is returned (i.e. the cause alarm is eliminated). Thus, the relationship can
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

Figure 1. Two filtering strategies

be described in terms of the difference between activation events in activation-return (A-R) windows and activation events in return-activation (R-A) or return-time-width interval (R-W) windows. Formally we could write this criterion as:

\[ A_p = \text{Activations (A-R)} \cap \text{Activations (R-A, or R-W)} \]

Two heuristic-based preprocessing strategies that incorporate domain specific concepts, and their filtering effects related to four types of alarms are shown in Figure 1. In this example, we assume that TAG 1 is a causal alarm and TAG 2 is a consequential alarm.

Specifically, we need to be concerned with the temporal order of events and represent durations that reflect possible causal dependencies. With an efficient alarm system, there should be no more than one alarm indicating an abnormal situation, as shown in case (a) of Figure 1. However, a perfect association between the variables under investigation may not exist. Operator related problems and inadequately configured alarm systems with improper setpoints (i.e. a minimum/maximum operating range) or poorly defined alarm limits or deadbands (i.e. a signal band where no action occurs) can cause repeating alarms, frequent alarms and long standing alarms.

Filtering Limitations

Return-point and Activation-point filtering strategies require some specific amount of time that is sufficient for consequential alarms to follow the cause alarm in time, in both activation-return and return-activation/time temporal ordering. The following is the list of possible filtering limitations.
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

- The duration of an activation-return (A-R) event-interval (window) should be *long enough* to capture the relationship by which *TAG 1* causes *TAG 2*. If an activation interval is short then there could be some related alarms, however, the duration of the activation (A-R interval) may not be long enough to capture all causal consequences if a causal process takes time.
- If the verifying return-activation/time-width interval is set *too short*, then the consequential alarm will not have enough time to return. If the verifying interval is set *too long*, then many events may come and go and thus affect relationships. Therefore, it is difficult to decide *how close is close enough* when giving the time width of the (R-W) window, as it depends on the size as well as the causal significance of the time intervals. Currently the proposed approach uses a fixed verifying window width for all alarm tags. Thus the determination of a verifying window width requires domain knowledge from the domain experts about the ‘process lag’ between propagating alarms. It may be a good idea to ensure that the duration of the verifying window interval for each alarm reflects the actual duration of its associated activation-return (A-R) event interval.
- When an alarm triggers other alarms, there is a ‘time lag’ while the entire event sequence finishes. If the cause alarm returns and activates again before the first group of associated alarms returns, there is a *cross-effect* between two consecutive occurrences of an associated alarm group. In order to correctly mine the alarm data using the proposed approach, a group of associated alarms should all return well before any new activation of the alarms within the group. Further research (Kordic et al., 2008) has been carried out to investigate this issue.

FORMAL BACKGROUND AND NOTATIONS

In this section we define formal concepts and notations that we will use to describe our mining methods. We follow the basic definitions introduced by Mannila, Toivonen & Verkamo (1997) in defining alarm sequences.

**Definition 1. (Time Point)**

A *time point* is an integer that represents the occurrence time of an alarm event. Let $t_a$ and $t_b$ be time points and there is a partial order between time points $t_a \leq t_b$.

**Definition 2. (Time Interval)**

A *time interval* is a contiguous sequence with a range of time points such that

$[t_a, t_b] \equiv \{t: t_a \leq t \leq t_b\}$

The duration of the interval $w = |t_b - t_a|$ is the *width* of the time interval.

**Definition 3. (Alarm Sequence)**

Given a class of event types $T$, an *alarm* is a pair of terms $(a, t)$ where $a \in T$ and $t$ is the *occurrence time* represented as an integer. An *alarm sequence* $S$ is an ordered collection of alarms defined as $S =$
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

\{(a_1,t_1), (a_2, t_2), \ldots, (a_n, t_n)\}, such that \(a_i \in T\) for all \(i=1,2,\ldots,n\), and \(t_1 \leq t_{i+1}\) for all \(i=1,\ldots,n-1\).

Example 1

In our research we consider alarm activation and alarm return knowledge. While activations are represented as positive integers, returns are represented as negative integers. An example of an alarm sequence is

\[ S = \{(1, 5), (2, 12), (-1, 19), (-2, 23), (1, 30), (2, 32), (-1, 35)\} \]

Notice that a pair of terms (alarm tag and occurrence time), have been recorded for each event that occurred in the time interval [5, 35]. For instance, the first member of the sequence (1, 5) indicates that alarm TAG 1 is activated at the occurrence time = 5, and the third element of the sequence (-1, 19), indicates that alarm TAG 1 is returned at occurrence time = 19.

Definition 4. (Activation-Return Interval Window)

An activation-return (A-R) window \(W_{A-R}\) is a subsequence of an entire event sequence \(S\) with respect to an event interval \(W_{A-R}=(S, t_{act}, t_{ret})\). It consists of the alarm pairs \((a, t)\) from sequence \(S\), where \(a \in T\) and \(t_{ret} \geq t_{act}\). Intuitively, activation-return windows are constructed by using the activation instances of a target tag, where the activation event marks the beginning of an interval window and the return event of the target alarm indicates the end of a window.

Definition 5. (Return-Activation/Width Interval Window)

A return-activation/width (R-A/W) window \(W_{R-A/W}\) is a subsequence of an entire event sequence \(S\) with respect to an event interval \(W_{R-A/W}=(S, t_{ret}, t_{act})\) or \(W_{R-w}=(S, t_{ret}, t_w)\). It consists of the alarm pairs \((a, t)\) from sequence \(S\), where \(a \in T\) and \(t_{act} \geq t_{ret}\) or \(t_{w} > t_{ret}\). Intuitively, verifying return-activation/width windows will be formed either if an alarm tag is re-activated or the user given maximal width \(w\) of the window is reached.

Definition 6. (Interval Event-Set)

An interval event-set \(\Phi\) is a partially ordered set of alarm types \(a_1 \leq a_2 \leq \ldots \leq a_n\) containing activation events pruned with either the Return-point (R-p) or the Activation-point (A-p) filtering strategy. Interval event-sets do not contain duplicate event types.

Definition 7. (Parallel Episode Frequency)

Let \(C\) be a collection of interval event-sets with respect to all event intervals of a specific tag of interest \(a_k\). Let \(E\) be a set of the distinct alarm tags \(a_1, a_2, \ldots, a_n\) such that \(E \subseteq T\). An interval event-set \(\Phi\) is said to contain \(E\) if and only if \(E \subseteq \Phi\), and \(\Phi \in C\). We define the frequency (\(E\)) of an unordered (parallel) episode as the number of interval event-sets which contain all the members in \(E\). Given a threshold minimum support for the minimal frequency, the set \(E\) is frequent if frequency (\(E\)) \(\geq\) minimum support.
THE PROPOSED MINING PROCESS

The overall algorithm involving three phases is shown below. Essentially, it relies on a context-based segmentation strategy, and incorporates some frequent itemset mining techniques.

Algorithm

The pseudo-code for the main algorithm is shown below.

**Input:** sequence of historical alarm event logs  
**Output:** frequent episode

(Phase 1): Data Generation and Preparation
Extract the relevant information associated with alarm tags from simulation event log-file or alarm database and put into an appropriate format

(Phase 2): Data Segmentation and Filtering
FOR ALL target alarm tags in the event log file
   Extract the $W_{A-R(i)}$ and the $W_{R-A/W(i)}$ sets of transactions
   FOR EACH alarm tag
      Do filtering based on Return-point (R-p) strategy or Activation-point (A-p) strategy
   END FOR
END FOR

(Phase 3): Discovery of Interesting Patterns
DO “frequent itemset mining” to obtain a set of co-occurring alarm tags associated with each tag of interest

Phase 1: Data Generation and Preparation

The strategy in our experiments was to use a systematic and data-driven approach achieved by increasing model/data complexity in order to evaluate and refine the developed techniques. The proposed approach was evaluated initially using simulated data produced from a Matlab (MathWorks, 2009) model of the Vinyl Acetate chemical process, and then using real chemical plant data with more than 100 distinct alarm tags.

Generation of Simulated Alarms Using the Vinyl Acetate Matlab Model

The Vinyl Acetate model can be used to accurately simulate a Vinyl Acetate process. However, unlike a real Vinyl Acetate plant, the Matlab model does not have alarm monitors built in and hence it cannot produce alarms which would be caused by setpoint changes or disturbances in the process. Figure 2 shows the simulated alarm monitors (AM), the associated alarm tags, and the monitored measurement variables.
In order to simulate alarms using the Matlab model, it was necessary to perform simulations twice. The first simulation was performed to obtain the normal measurement outputs of the Vinyl Acetate model under normal operating conditions. Then the second simulation was performed to obtain the disturbed measurement outputs of the model under disturbance. The difference between normal measurement outputs and the disturbed measurement outputs was used to generate discrete alarm data associated with the injection of disturbances. For simplicity, it was assumed that a simulated alarm is activated if the following condition is satisfied:

\[ \text{Abs} \left( \frac{D_m - N_m}{N_m} \right) \geq S_{\text{am}} \]

where \( D_m \) is the disturbed output magnitude, \( N_m \) is the normal output magnitude and \( S_{\text{am}} \) is the sensitivity of the simulated alarm monitor. Note that the simulated alarm will return to normal if the above condition is not satisfied. The signal detection sensitivity for all alarm monitors was set to be equal to 0.0005.
Simulated Data Pre-Processing

The Vinyl data consists of records of alarm logs which characterize the actual state of the plant at particular points in time, representing the status of 27 alarm monitors. Using a simple algorithm, the event log file is processed to produce formatted alarm data as shown in Figure 3. Note that the symbol (A) stands for activated and (R) for returned. Also note that Figure 3 shows a simple record representing only the first 8 alarm monitors.

The approach captures the first time instance an alarm tag goes into activation and the time instance the alarm tag is returned. For example, in the column associated with TAG 1 in Figure 3, it shows that TAG 1 was activated at 1:32 and returned at 1:35, thus the duration of this tag being in an activation state was 3 minutes. TAG 1 is re-activated at 1:37. Similarly, TAG 6 was activated at 1:33 and it returned at 1:36. The collection of alarm activations/returns and time-stamps associated with a defined set of alarm tag identifiers as shown in Figure 3 forms an alarm sequence. The starting point of the activation-return (A-R) window is defined as the time the tag of interest went into activation and the ending point is where it went into the corresponding return state. For example, if the tag of interest is TAG 1 in Figure 3, the event interval windows are defined as $W_{A-R(1)}$, from 1:32 to 1:35 and $W_{A-R(2)}$, from 1:37 to 1:39. A second type of window (denoted as $W_{R-A/R-w}$) used here is defined by a fixed user-defined duration after a tag of interest moves back into a return state (after being activated). This user-defined duration may be shortened if the tag of interest is reactivated. Again, using a user-defined window width of 3 minutes and TAG 1 as an example, its $W_{R-A/R-w(1)}$ should start from 1:35 and end 3 minutes later at 1:38, but as TAG 1 is re-activated again at 1:37 this window will terminate at the point TAG 1 is re-activated (i.e. a duration of 2 minutes instead of 3 minutes).

Figure 3. An example of an alarm sequence
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

Real Plant Data Pre-Processing

Since different vendors of modern control systems generate their own format for messages, the format of plant alarm data is both software and hardware dependent. As a minimum requirement, the alarm event log consists of records with fields that store information such as a unique identifier for each alarm tag, time/date, alarm priorities, alarm settings and the possible states of an alarm tag [ALM, RTN]. When an alarm tag is in an activation [ALM] state, it implies that the value of the associated process variable is outside its normal operational setting, and when this value returns to the normal operational setting, the alarm tag is then in a return [RTN] state. It is important to check an event log file carefully because real plant data tends to be inconsistent, with errors, missing values, outliers, and duplicate values. The data pre-processing steps are summarized as follows:

• **Removal of irrelevant records** - records in the alarm databases (or alarm log files) are irrelevant to the pattern discovery aim if the data fields have no [ALM] or [RTN] values. For example, all the records in the database containing [ACK] values.

• **Removal of irrelevant attributes (database fields)** – each extracted event has only three important attributes – alarm tag identifier, its state, and the time of the event occurrence.

• **Removal of redundant data** - the same events may be recorded two or more times. Examples of such values are multiple [ALMs] with identical alarm stamps but with only one [RTN].

• **Removal of outliers** that fall outside the activation-return and return-activation boundaries. Examples of such values are [RTN] with no [ALM] at the beginning of the file, and [ALM] with no [RTN] at the end of the file, due to inaccurate sampling.

• **Data substitution** – finally, an alarm sequence is created where each event has an associated time of occurrence given as an integer number, and also activation and return event types are represented as positive and negative integers, respectively.

**Phase 2: Data Segmentation and Filtering**

This phase extracts a sequence of alarm events from the alarm event log database (file) of a chemical plant. Since we consider both activation and return events, it is possible to segment the entire alarm sequence by using the activation instances of a target tag, where the activation event marks the beginning of a sliding widow and the return event of the target alarm indicates the end of a window. Thus if the total number of activations of a specified alarm tag is n, then the whole alarm sequence can be segmented into n windows of $W_{A,R}$. Similarly the whole alarm sequence can also be segmented into n windows of type $W_{R,A/R-w}$ if there are n returns associated with the tag of interest. Obviously, this segmentation is an event-based extraction within a clear contextual meaning. At each point in time, the window is shifted along to the next activation instance of the tag of interest. Each of the n windows of $W_{A,R}$ captures data that indicates which other alarm tags also went into activation while the alarm tag of interest was in activation. On the other hand, each of the n windows of $W_{R,A/R-w}$ captures data that show which other alarm tags also returned within a user-defined window after the tag of interest returned.
The Segmentation and Filtering Algorithm

**Problem:** suppose we are given a sequence \( s = (s, T_s, T_e) \) where \( T_s \) is the starting time and \( T_e \) is the ending time, an integer \( a_k \) represents a specified target alarm tag, and a window width \( w \). Find all *interval event-sets* with respect to the target alarm tag.

The pseudo-code for the segmentation and R-p filtering is shown below.

**Input:** a sequence \( s \), a window width \( w \), and a specific tag of interest \( a_k \)

**Output:** the collection \( C \) of interval event-sets with respect to tag of interest

```plaintext
/* initialisation*/
A := {∅}, R := {∅}, C := {∅};
k := 0;

FOR each \( a_k \) in \( s \)
   A ← subset of activation segment \( (W_{A-R}[k]) \);
   R ← subset of return segment \( (W_{R-A/R-w}[k], w) \);
   C[k] ← intersection \( (A ∩ R) \);
   k := k + 1;
END FOR

Output C
```

The main idea of the algorithm is the following: during each shift of the window \( W_{A-R}(k) + W_{R-A/R-w}(k) \), we recognize all unique activations and return events and place them into sets \( (A \text{ and } R) \). Next, we store the intersection of two sets \( A ∩ R \) into a collection of event-sets \( C \). In the recognition phase the total time spent going through the loop will be linear since there are \( O(n) \) shifts of the window. The body of the for-loop consists of four assignment statements, and thus the time taken to execute the body once is \( O(1) \). Therefore, the time spent by the algorithm is \( O(n) + O(1) \). As the lower-order term can be neglected, the time complexity is \( O(n) \) unless there is only one occurrence of \( a_k \). In this case \( k = 1 \), and consequently the time complexity is \( O(1) \).

**Phase 3: Discovery of Interesting Patterns**

At the end of the first two phases the unstructured data has been partitioned into a cluster of information on the basis of its relation to the activation and return of an alarm tag of interest.

*Figure 4. An example of processing interval event-sets associated with TAG 3 using the Return-point strategy and a window width of 30 minutes*
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

Example 2

Figure 4 shows an example of processing event-sets associated with TAG 3 using the R-p filtering strategy and a window width of 30 minutes.

The collection of resulting event-sets in column 3 of Figure 4 is used in the next step to find a data driven frequency threshold value that can be used subsequently for finding patterns of interest.

CASE STUDY

To evaluate the proposed method two simulated data sets were generated with varying complexity and fault durations.

Case 1: Simulated Fault Data Set 1

The duration of the first simulation was 10080 minutes which represents one week’s operation of the Vinyl Acetate process. The measurement outputs were monitored and sampled at a frequency of 1 sample in one minute. The sampled measurement outputs were then streamed into a data file. After the completion of the simulation, the data file contained 27 columns and 10080 rows, each column representing a measurement output, and each row representing one minute intervals in the simulation. The second simulation is performed to simulate the response of the Vinyl Acetate process under disturbances. Frequent disturbances of one type, namely the loss of %O2 in the Reactor Inlet (associated with alarm TAG 1), were injected into the Vinyl Acetate process to induce a response in measurement outputs. Each fault was injected 10 times and lasted for different durations. Similar to the normal simulation, the duration of the second simulation was also 10080 minutes. Again, the measurement outputs were monitored and sampled at a frequency of 1 sample in one minute. The sampled measurement outputs were then streamed into the second data file. Using the normal process output stored in the first data file and the disturbed process output stored in the second data file, it is straightforward to obtain the changes in the process measurements which were caused by the injected disturbances using the formula given in the section “Generation of Simulated Alarms using the Vinyl Acetate Matlab Model”.

An implementation of the FP-growth (Han et al., 2000) algorithm was used to output all frequent patterns associated with each alarm tag of interest. Firstly, we wanted to demonstrate the effectiveness of the filtering, and thus we set the minimum frequency very low at 1 occurrence. Figure 5 illustrates the effect of the A-point and R-point filtering on simulated dataset 1 with respect to faults associated with alarm TAG 1. In this experiment we compared the quantities of patterns discovered without filtering (activations), with the quantities of patterns discovered using a verifying group and filtering.

Figure 5 shows that overall the numbers of frequent patterns dropped dramatically in the case of R-point filtering. The number of frequent patterns ranged from 799795 for the activations with no filtering, down to 255 for the activations with R-point filtering.

Next, we examined the validity of the discovered patterns. For the simulated dataset 1, we set the minimum frequency a bit higher - equal to 8 occurrences - to find more statistically significant (frequent) rules relating to 10 injected faults. In our research we are only interested in finding primary and consequential alarms, and therefore in rules that have a specific target tag appearing in the antecedent. The idea of mining maximal frequent itemsets (Bayardo, 1998; Burdick, Calimlim, & Gehrke, 2005;
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

Figure 5. The effect of the A-point and R-point filtering on simulated dataset 1

Gouda & Zaki, 2001) is to use a very concise set of frequent itemsets called maximal frequent itemsets. For a given maximal frequent event set \( E = a_1, a_2, \ldots, a_m, m \geq 2 \), the consequent \( Y \) will be the event set \( E - a_i \). Thus the confidence of composite episode \( confidence(a_i \Rightarrow Y) \) is calculated as \( \frac{frequency(a_i \cup Y)}{frequency(a_i)} \). Calculating the confidences of episodes is not difficult since the alarm sequence is segmented into sets of windows associated with each primary alarm tag.

Figure 6 shows rules with respect to the first ten alarm tags. In terms of checking whether the group of associated alarms is correct, the results sets associated with fault TAG 1 (loss of %O2) can be checked against the Vinyl Acetate process flowsheet chart illustrated in Kordic et al. (2008). Based on the Vinyl Acetate flowsheet analysis and the alarm status display, the loss of Oxygen feed (TAG 1) in the reactor could significantly affect temperature change in the Reactor (TAG 7), Heat Exchanger (TAG 6), Separator (TAG 9), and Vaporizer Level (TAG 4).

Case 2: Simulated Fault Data Set 2

Frequent disturbances were injected into the Vinyl Acetate process to induce a response in measurement outputs. Only one type of disturbance was introduced in this simulation, namely the loss of the fresh HAc feed stream (associated with alarm TAG 3), with each fault lasting over different durations. The fault was injected 10 times. The measurement outputs were monitored and sampled at a frequency of 1 sample in five seconds. After the completion of the simulation, the data file contained 27 columns and 60481 rows, each column representing a measurement output, and each row representing a five second interval in the simulation.

Similarly to the effect of the A-point and R-point filtering on simulated dataset 1, the numbers of frequent patterns dropped dramatically in the case of both A-point and R-point filtering. This time the number of frequent patterns ranged from 2069255 for the activations with no filtering, down to only 11 for the activations with R-point filtering. We have also developed a prototype data mining tool that reduces the effort needed for pre-processing, data segmentation and analysis of alarm log files. As shown in Figure 7, a graphical user interface was created to allow user interaction.
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

Figure 6. An example of processing selected comparative results for simulated dataset 1 with respect to minimum frequency = 8 occurrences, minimum confidence = 40%, R-point filtering and verifying window width w = 900 seconds

<table>
<thead>
<tr>
<th>TAG</th>
<th>Frequent Episodes</th>
<th>frequency</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2 (fault)</td>
<td>1 =&gt; (6 7 9 4)</td>
<td>10</td>
<td>10/10 = 100%</td>
</tr>
<tr>
<td>Press</td>
<td>2 =&gt; () below minimum frequency</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HAc-L</td>
<td>3 =&gt; () below minimum frequency</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Vap-L</td>
<td>4 =&gt; () below minimum frequency</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Vap-P</td>
<td>5 =&gt; () below minimum frequency</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pre-T</td>
<td>6 =&gt; () below minimum frequency</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RCT-T</td>
<td>7 =&gt; (12)</td>
<td>12</td>
<td>12/29 = 41%</td>
</tr>
<tr>
<td>Sep-L</td>
<td>8 =&gt; () below minimum frequency</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sep-T</td>
<td>9 =&gt; () below minimum frequency</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sep-V</td>
<td>FIXED</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Case 3: Chemical Plant Data Set

To evaluate the efficiency of mined results we also used real petrochemical plant data with more than 100 distinct alarm tags. Due to the confidentiality agreement involved, we will only present some of the advantages of our approach with respect to this dataset. The chemical plant data sample was given as a ‘dump file’ containing records of alarm logs taken from an Oracle database. These alarm records represented the actual state of the plant over a period of 12 days. An example of the plant data after irrelevant data fields were removed is given below:

23/03/2005 ALM 08:55:49 TAGB1924

23/03/2005 RTN 08:55:54 TAGB1925

…

Please note that in the above example, and wherever else ‘real plant data’ is shown, tag labels have been changed to protect the confidentiality of the data provided by the industrial partner.

Choosing the right support threshold is critical in terms of finding interesting patterns. There were 116 alarm tags, and only 7 alarm tags had a frequency greater than 100 occurrences. 87.1% of the events (i.e. the first 101 alarm tags) have a frequency that is less than 50 occurrences (2.7%). A common misconception is that the higher the value associated with support then the greater will be the degree of interestingness (McGarry, 2005) to the user. For example a pattern with a support value of 90% may be considered better than one with a support value of 30%. Thus a user may arbitrarily determine a high
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

support value for finding patterns of interest from a data set, and then if no patterns are found, it may lead to the misconception that there are no interesting patterns in the data set. The concept here is that “the story is in the data”. For example, if the occurrence of a group of co-related tags in the data set has a frequency of 10% then that is the value to use, rather than selecting an arbitrary value. For this reason, our approach is to set a sufficient threshold for each alarm tag independently, prior to the data mining phase.

We have also developed a prototype data mining tool to aid in the analysis of an alarm tag of interest. As shown in Figure 8, a graphical user interface was created to speed up the process and to provide an intuitive means of interpreting the results - especially for databases with a large number of alarm tags. The example shown in Figure 8 describes the temporal relationships between alarms with respect to target alarm 5 representing \textit{TAG T08UA976}.

A-R Durations

1186,1379,1518,1644,1439,1653,2058,11822,719,290,1333,886,3104,424, 760,914,813,960,844,892, 960,1043,735,632,781,789,888,837,942,885,910, 1656,1256,1017,1342,423,694,708,1371,1182,943,

Figure 7. A graphical user interface was created to allow user interaction
The aim of this procedure is to extract relevant information from vast amounts of data in order to indicate a threshold that characterizes the context of tag-dependent data. The level of skewness could be calculated with respect to minor interval changes in the curve’s slope. The number of times the consequential alarm occurred within the windows of the primary tag and the rate of change in slope interval are factors taken into account to provide the user with guidance in selecting the confidence threshold value. Note that we expected that an extreme skewness to the right may indicate a chattering alarm.

Threshold relevant to data context of Alarm 5
Minimum support = 12 occurrences
Serial episodes
5 = > (82) confidence = 37%, 20 occurrences
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

5 => (74) confidence = 35%, 19 occurrences
Composite Episode
5 => (82, 74) confidence = 29%, 16 occurrences

DESCRIPTION
Alarm: 5
Alarm 74
Activations: 35, Returns: 35, Priority: LOW Tag: TAGRI1049, REACTOR INLET
Alarm 82
Activations: 29, Returns: 29, Priority: LOW Tag: TAGRI1050, REACTOR INLET

FUTURE RESEARCH DIRECTIONS

The proposed approach has been evaluated in an extensive study using simulated data, and then applied to real chemical plant data with more than 100 distinct alarm tags. Our preliminary experiments showed the effectiveness and usefulness of the proposed approach. However, although the experiments and analysis with simulated data demonstrated a useful means of dealing with alarm faults, the computer simulation we used may not have shown the full complexities of a sophisticated, real alarm system. Future work will focus on extending the approach by considering an interestingness measure that can address this issue.

We can assume that some alarms are not independent. That is, there could be a root cause alarm which triggers the activation of many alarms in a chemical process. These “consequential” alarms may return before the root cause returns due to local feedback in the process, or due to improper alarm settings. However, by definition an interval event set Φ does not contain duplicate events (only the first incidence of an event is recorded) although in an activation-return window there may be multiple occurrences of any consequential alarm tag. If this alarm tag does not occur frequently elsewhere in the alarm sequence, it is probably a good indication that the alarm is a consequence of the primary cause. Based on the objective statistical strength of discovered patterns we can find which alarms occur most often in the data context of the particular tag of interest. For instance, given an event sequence $s$ and the set of all windows $\Delta (s, win)$ with respect to target TAG A, if we know that the subsequent TAG B occurs 20 times in $\Delta (s, win)$ and only 25 times in the entire sequence $s$, this may indicate that the connection between tag A and B is significant, since $20/25 = 0.8$. On the other hand, if the value is low, say 0.1, this value may indicate events that are not very dependent. In other words, by comparing the total number of occurrences of a particular alarm tag (in the data context of the primary tag) against its frequency in the entire sequence, we can find the degree of dependency for this particular tag. However, more rigorous testing is required to determine the applicability of this measure.

CONCLUSION

To summarize, in this chapter we have presented an approach for analyzing large alarm databases. A crucial element in this investigation is to determine intervals related to co-occurrences of the particular alarm tag of interest. The approach combines a context-based segmentation strategy with a data mining
Patterns Relevant to the Temporal Data-Context of an Alarm of Interest

technique to address the problem of discovery of interesting patterns from historical alarm event logs. The discovered groups of co-occurring alarms can be used to support alarm rationalization by identifying redundant alarms and system bad actors. We showed the efficiency of our filtering strategies and the relevance of discovered patterns. We illustrated how instead of selecting an arbitrary value for all alarms in an alarm sequence, we could set a sufficient threshold for each alarm tag independently. The approach is more cost effective for identifying primary and consequential alarms than any manual alarm analysis of event logs, which is very costly in terms of time and labour.

REFERENCES


Patterns Relevant to the Temporal Data-Context of an Alarm of Interest


**Patterns Relevant to the Temporal Data-Context of an Alarm of Interest**


**KEY TERMS AND DEFINITIONS**

- **Data Mining:** is usually defined as the process of extracting and analyzing previously unknown, hidden patterns and relationships from data that has been collected.
- **Data Preprocessing:** is the process that includes various procedures such as data cleaning, data reduction, and data filtering to transform the raw data into a suitable form for data mining.
- **Case Study:** is an intensive study that is carried out in order to investigate various strategies and prove the efficiency of the proposed tools and techniques.
- **Frequent Episodes:** is a data mining framework for discovery of temporal rules in telecommunications networks. Basically there are three types of episodes: a serial episode which occurs in a fixed order, a parallel episode which is an unordered collection of events, and a composite episode which is built from a serial and a parallel episode.
- **Domain Knowledge:** is the knowledge which is specific to a domain such as knowledge obtained from experts in the field.
- **Simulated Data:** A large volume of alarm data was generated using a computer model of the plant that simulates the behavior of a chemical process.
- **Simulation Model:** A simulation model such as the Matlab model of the Vinyl Acetate plant was designed to closely represent the behavior of the chemical plant.