Contents lists available at SciVerse ScienceDirect

Chemical Engineering Research and Design

journal homepage: www.elsevier.com/locate/cherd

Pattern matching of alarm flood sequences by a modified Smith–Waterman algorithm

Yue Cheng[a,*], *Iman Izadi*[b], *Tongwen Chen*[a]

[a] *Department of Electrical and Computer Engineering, University of Alberta, Edmonton T6G 2V4, Canada*
[b] *Matrikon Inc., Edmonton T5J 3N4, Canada*

ABSTRACT

Alarm flooding is one of the main problems in alarm management. Alarm flood pattern analysis is helpful for root cause analysis of historical floods and for incoming flood prediction. This paper deals with a data driven method for alarm flood pattern matching. An alarm flood is represented by a time-stamped alarm sequence. A modified Smith–Waterman algorithm considering the time stamp information is proposed to calculate a similarity index of alarm floods. The effectiveness of the algorithm is validated by a case study on actual chemical process alarm data.

Keywords: Alarm flood; Time-stamped sequences; Approximate sequence matching; Smith–Waterman algorithm; Knowledge extraction

1. Introduction

In large industrial plants, alarm systems are of great importance to meet the demand of safety, quality and efficiency. Nowadays, hardware and software advances make it easy to access almost every process variable that can be measured; this provides alarm systems more process information. However, a serious problem exists in the industry: alarm flooding. Tens or hundreds of alarms may arise during a short period, which can overwhelm even experienced operators. Operators cannot analyze every alarm properly during alarm floods because of shortage of time; so they can only acknowledge the alarms and handle the abnormality based on their experience (Folmer et al., 2011). As a result, EEMUA and ISA standards (EEMUA, 2007; ISA, 2009) suggest that an operator should not receive more than six alarms per hour, and define an alarm flood as the period of more than 10 alarms per 10 min per operator.

It is shown in Henningsen and Kemmerer (1995) that a large number of unimportant alarms fall within three categories: standing alarms, repetitive alarms, and consequence alarms. For standing and repetitive alarms, such widely implemented techniques as delay-timer, deadband and shelving are suggested in EEMUA (2007) and ISA (2009). An intelligent alarm management software focusing on chattering and standing alarms was developed and successfully implemented on a refinery process (Liu et al., 2003; Srinivasan et al., 2004). During steady operating conditions, the alarm rate can be effectively reduced to a manageable level thanks to these techniques. In abnormal situations, however, it is still very difficult to get rid of alarm floods though these techniques can suppress some of the alarms. The remaining alarms are mainly consequence alarms. Compared with standing and repetitive alarms, consequence alarms are more complicated and more difficult to handle. Consequence alarms are mainly caused by abnormality propagation. Usually at the beginning of an abnormal situation, a few alarms arise, followed by many cascaded alarms, which eventually lead to an alarm flood. As a result, if the correlations among different alarms can be obtained, the information will be helpful to handle consequence alarms.

1.1. State of the art concerning alarm sequence pattern analysis

The most accurate approaches for alarm sequence pattern analysis are process knowledge based methods. The alarm sequence patterns can be identified manually. Some structural modeling tools such as signed digraphs (SDG) are helpful for

* Corresponding author.
E-mail addresses: cheng5@ualberta.ca (Y. Cheng), iman.izadi@matrikon.com (I. Izadi), tchen@ualberta.ca (T. Chen).
Received 26 July 2012; Received in revised form 21 October 2012; Accepted 1 November 2012
0263-8762/$ – see front matter © 2012 The Institution of Chemical Engineers. Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cherd.2012.11.001

abnormality propagation path searching and hence for identifying alarm sequence patterns (Umeda et al., 1980). However, these knowledge based methods are usually very time consuming to implement and require involvement of experts.

To reduce the analysis time, data mining techniques are a good choice. Pioneer work on applying sequential pattern mining to consequence alarm suppression includes Cisar et al. (2009), Hostalkova and Stluka (2010), Folmer et al. (2011), and Folmer and Vogel-Heuser (2012). In Cisar et al. (2009) and Hostalkova and Stluka (2010), the authors modified the Generalized Sequential Patterns (GSP) algorithm (Srikant and Agrawal, 1996) to search for frequent alarm sequences (patterns) in the historical alarm records. They emphasized that besides the chronological order the triggering time was also very important, so time stamps of alarms were considered to add time constraints in their algorithm. They also pointed out that a potential use of the pattern analysis result was dynamic alarm suppression. Since one alarm pattern is usually related to a specific underlying abnormality, all the alarms in a certain pattern can be temporarily suppressed when the corresponding abnormality is detected. In Folmer et al. (2011) and Folmer and Vogel-Heuser (2012), the authors set up Automatic Alarm Data Analyzer (AADA) automatons and alarm-sequence automatons to represent the frequent alarm sequences in historical alarm records. However, it was also mentioned that one of the deficiencies of the algorithm was the lack of robustness to disturbances, e.g., similar alarm sequences with only one different alarm were recognized as different alarm sequences.

1.2. *Background of sequence data mining*

Sequence data is studied in many application domains such as intrusion detection (Hofmeyr et al., 1998; Lane and Brodley, 1999), customer purchases pattern mining (Chakrabarti et al., 1998), text pattern matching/detection (Cole and Hariharan, 1998; Amir et al., 2009), biological sequences matching/detection (Smith and Waterman, 1981; Aach and Church, 2001), and so on. Since purposes and natures of input data in different applications are not the same, there are a variety of problems formulations, e.g., exact or approximate matching of sequences (Cole and Hariharan, 1998; Amir et al., 2009; Smith and Waterman, 1981; Aach and Church, 2001), frequently repeated sequential pattern-mining (Srikant and Agrawal, 1996; Mabroukeh and Ezeife, 2010), abnormal sequences detection (Chandola et al., 2012; Hofmeyr et al., 1998; Lane and Brodley, 1999), sequence mining with consideration of intelligent adversaries (Ng et al., 2010), and so on. Here we would only offer some discussion on the research topics related to the application of alarm sequences analysis.

Frequently repeated sequential pattern-mining tries to discover short subsequences that are frequently appear in a large sequential database. An exhaustive survey was provided in Mabroukeh and Ezeife (2010). In the survey, algorithms were classified into three categories: apriori-based, pattern-growth, and early-pruning algorithms. Because of the different techniques used, the algorithms have different memory and time consumptions. The pioneer work on consequence alarms suppression (Cisar et al., 2009; Hostalkova and Stluka, 2010; Folmer et al., 2011; Folmer and Vogel-Heuser, 2012) apply frequently repeated sequential pattern-mining techniques.

Anomaly detection is another extensively discussed topic. The objective of anomaly detection is to identify abnormal sequences or subsequences from normal ones. The survey paper Chandola et al. (2012) provided a structured overview of the research and categorized the techniques to three distinct frameworks: sequence-based anomaly detection, subsequence-based anomaly detection, and pattern frequency-based anomaly detection. The problem of anomaly detection is different from ours, since in our situation, all the events appear in the sequences are alarm events that indicate abnormalities. However, the problem formulation of sequence-based anomaly detection is very similar to the one of this paper. The only difference is that in anomaly detection, the training data set is a set of normal sequences, and they mainly focus on the test sequences that are not similar to any cluster of normal sequences. While in our problem, the training data set is a set of abnormal sequences, namely, alarm floods, and we mainly focus on the test sequences that may belong to one or several clusters (patterns) in the training set.

In the framework of similarity-based techniques for sequence-based anomaly detection, a key point is similarity measurement. Many similarity measurement techniques are based on approximate matching. The topic of approximate matching is extensively discussed in Navarro and Raffinot (2002) and the reference therein. As defined in Navarro and Raffinot (2002), "approximate matching is modeled using a distance function that tells how similar two strings are". A first such distance function, edit distance, was proposed in Levenshtein (1965), which is the minimum number of insertions, deletions, and substitutions to make two sequences equal. To increase flexibility of setting different penalty weights on insertion, deletion and substitutions, variations of the edit distance are proposed in Smith and Waterman (1981) and Needleman and Wunsch (1970), together with corresponding dynamic programming algorithms. From then on, such fast algorithms as BLAST and FASTA (Lipman and Pearson, 1990; Altschul et al., 1990) were designed to decrease the time consumption, but the objective function, namely, the distance function, was not changed. A similar idea applied to real valued time series is dynamic time warping (Ratanamahatana and Keogh, 2005). It is proposed to measure the distance between two real valued time series. Although some researchers also tried to apply it to symbolic sequence matching (Ratanamahatana and Keogh, 2005; Aach and Church, 2001), they had to firstly convert a symbolic sequence to a meaningful real valued time series. Compared with such statistical model-based techniques as Markov model and hidden Markov model (Ge and Smyth, 2000; Yamanishi and Maruyama, 2005), the approximate matching technique is more suitable for sequences that are not very long, since it does not require large amount of data to model statistical properties of the sequences.

1.3. *Problem description and contribution of our work*

In this paper, we address the problem of consequence alarms, but our problem formulation is somewhat different from Cisar et al. (2009), Hostalkova and Stluka (2010), Folmer et al. (2011), and Folmer and Vogel-Heuser (2012). We consider the alarms during one alarm flood as one alarm flood sequence, and compare the alarm flood sequences of different floods. Similar alarm flood sequences usually relate to the same kind of underlying abnormality; so by clustering similar alarm flood sequences we can find out flood pattern candidates. Then experts can analyze these pattern candidates to determine real flood patterns and their root causes which will be shown to the operators when a new flood with high similarity to a

certain flood pattern is forthcoming. Since the purpose is clustering similar alarm sequences instead of identifying exact same sequences, the deficiency of the method provided in Folmer et al. (2011) and Folmer and Vogel-Heuser (2012) can be avoided.

The goal is very similar to the ones in abnormality diagnosis papers using process data by Johannesmeyer (1999), Singhal and Seborg (2001), and Singhal (2002), because automation on abnormality diagnosis discussed in these papers can greatly reduce the operators' workload when alarm floods occur and thus is a rational resolution for alarm flood problems. However, in our paper, we use alarm data rather than process data to measure the similarity of different abnormalities.

Approximate sequence matching techniques can be applied to calculate similarities of alarm flood sequences. However, current approximate matching techniques rarely consider time stamp information of each element in the sequence, since they are mainly designed for text pattern and biological sequences matching. Our main contribution is modifying the formulation of the Smith–Waterman algorithm by adding time stamp information, and propose a modified Smith–Waterman algorithm suitable for alarm flood pattern matching.

1.4. *Organization*

The rest of this paper is organized as follows. The alarm data generation and preprocessing, as well as the standard Smith–Waterman algorithm, are briefly introduced in Section 2. In Section 3 a modified algorithm is proposed. Validation of the modified algorithm with petrochemical plant data is presented in Section 4. Finally, concluding remarks are given in Section 5.

2. Background

2.1. *Alarm data and alarm sequence*

Alarm data is a set of text messages generated by the DCS and stored in alarm log. When a process value exceeds one of its predetermined thresholds, an alarm message is generated. Usually an alarm message contains several fields of information: time stamp, namely, the time instant when the message is generated, tag name, tag identifier, e.g., 'PVHI', 'PVLO', 'OFFNORM', and some other information such as the priority, the value of the process variable, the trip point and so on (Izadi et al., 2010). The tag name plus tag identifier reflects what type of alarm occurs, and the time stamp reflects when it occurs.

As a simple example, assume there are only two process variables in the alarm system, x_1 and x_2. For each process variable, two thresholds (an upper limit and a lower limit) are set. Under normal situations, the process variables should operate between their upper and lower limits. In other words, when their values are larger (smaller) than their upper (lower) limits, PVHI (PVLO) alarms will arise, while alarms will be cleared when the process variables return to their normal regions (between their upper and lower limits). Hence there are totally 4 types of alarms: x_1.PVHI, x_1.PVLO, x_2.PVHI, and x_2.PVLO. Fig. 1 shows the generation of these 4 types of alarms according to the process variables x_1 and x_2. At the instant that a process variable exceeds a threshold, a red bar is plotted on the corresponding alarm plot. For example, the process variable x_1 crosses the upper limit at 30 s, hence a red bar is shown at 30 s on the alarm x_1.PVHI plot. Table 1 provides the corresponding alarms in the log. This is the alarm sequence we focus on. In the alarm sequence, a troublesome issue is chattering alarms. If a single type of alarm arises and clears again and again in a short time interval, the whole alarm sequence will be dominated by it, and the useful information of sequential alarms will be submerged. An example of chattering alarms in Table 1 is the alarms x_2.PVLO arise at 67 s, 69 s, 71 s and 73 s. As a result,

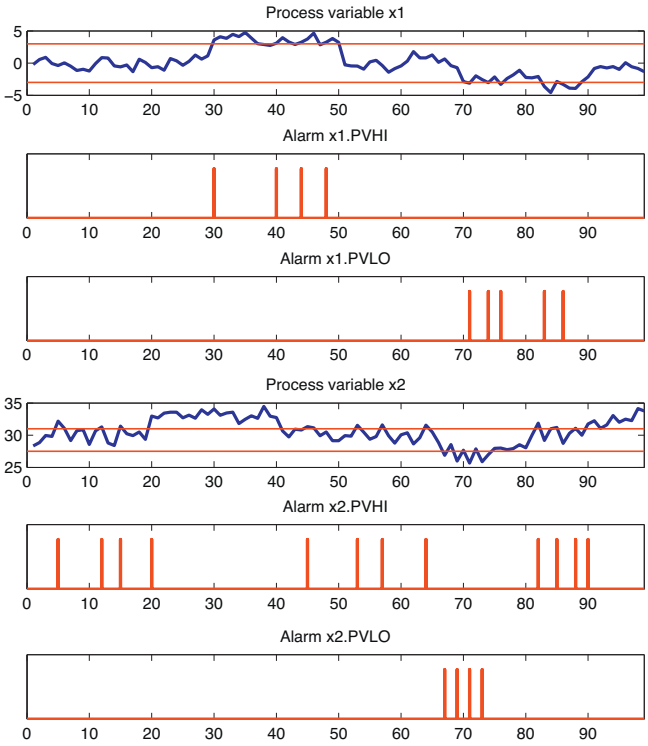

Fig. 1 – Example of alarm generation.

Table 1 – Alarm message log.

Tag name	Tag identifier	Times tamp
x_2	PVHI	5 s
x_2	PVHI	12 s
x_2	PVHI	15 s
x_2	PVHI	20 s
x_1	PVHI	30 s
x_1	PVHI	40 s
x_1	PVHI	44 s
x_2	PVHI	45 s
x_1	PVHI	48 s
x_2	PVHI	53 s
x_2	PVHI	57 s
x_2	PVHI	64 s
x_2	PVLO	67 s
x_2	PVLO	69 s
x_1	PVLO	71 s
x_2	PVLO	71 s
x_2	PVLO	73 s
x_1	PVLO	74 s
x_1	PVLO	76 s
x_2	PVHI	82 s
x_1	PVLO	83 s
x_2	PVHI	85 s
x_1	PVLO	86 s
x_2	PVHI	88 s
x_2	PVHI	90 s

the alarm sequence should be preprocessed to eliminate the chattering alarms before it is used for flood pattern analysis. A simple but effective way is to combine alarms of the same type that occur within a short time.

2.2. *Brief introduction of the Smith–Waterman algorithm*

The Smith–Waterman algorithm was first proposed in Smith and Waterman (1981). Its objective is "to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology)" (Smith and Waterman, 1981).

The Smith–Waterman algorithm is a local sequence alignment method. Before discussing the algorithm, we first introduce the concept of local alignment. Given a pair of symbolic segments, one from each of two symbolic sequences, we can equalize the length of the two segments by inserting gaps (symbol '–') in one or both of them (if the two segments have the same length, we can also choose to insert no gap). Then each symbol in one segment has a corresponding symbol in the other segment at the same position. This is called alignment. Since the two symbolic segments are two contiguous subsequences of the two symbolic sequences, respectively, the alignment on the pair of segments is called local alignment of the two sequences.

For example, consider two symbolic segments:

$X = [4\ 5\ 4\ 6\ 7]$

$Y = [6\ 6\ 7\ 4]$.

An alignment of this pair of segments is:

$X' = [4\ 5\ 4\ 6\ 7\ -]$

$Y' = [6\ -\ -\ 6\ 7\ 4]$.

The two aligned segments have the same length 6. Each symbol in aligned segment X' has a corresponding symbol in the aligned segment Y', and the two symbols compose a symbolic pair. For instance, the first symbolic pair of the aligned segments X' and Y' is (4, 6) and the second pair is (5, –).

Obviously, for one pair of segments there are a huge amount of different alignments since gaps can be added arbitrarily. Consider the example mentioned above, we can also align the pair of symbolic segments as:

$X'_1 = [4\ 5\ 4\ 6\ 7\ -\ -]$

$Y'_1 = [6\ -\ 6\ -\ -\ 7\ 4]$,

or

$X'_2 = [4\ 5\ 4\ 6\ 7]$

$Y'_2 = [6\ -\ 6\ 7\ 4]$.

As a result, a scoring system is required to differentiate good alignments from bad ones, and the similarity of the two symbolic segments should depend on the optimal alignment.

For a symbolic pair (a, b) in a pair of aligned symbolic segments, where a and b are two symbols, e.g., two discrete events, two letters, two nucleic acid bases, if neither a nor b is the gap symbol '–', a similarity score function $s(a, b) : \Sigma \times \Sigma \rightarrow \mathbb{R}$, where Σ is the alphabet of the concerned sequences, is provided. The value of the similarity score function $s(a, b)$ is positive for a match ($a = b$), and is non-positive for a mismatch ($a \neq b$). Usually, a uniform score for all matched pairs (kept at 1) and a uniform negative score μ for all mismatched pairs are chosen if there is no additional prior information on the symbols.

For a symbolic pair (a, b) including a gap symbol '–', the similarity score is always negative as a penalty of inserting a gap. The value is related to l, the number of contiguous gap symbols prior to the current gap. In Smith and Waterman (1981), if l is 0, it is called opening a gap, which suffers a heavier penalty. In the case that $l > 0$, it is called extending a gap, and a lighter penalty is used. In the case that opening a gap is not of greater significance, a uniform penalty value δ for all l is preferred, since it can simplify the algorithm. Therefore, the uniform penalty value δ is used in our paper.

The similarity score of an alignment is the summation of all the similarity scores of its symbolic pairs. The similarity score calculation of the aligned segments X' and Y' is shown below.

Rewrite the alignment:

$X' = [4\ 5\ 4\ 6\ 7\ -]$

$Y' = [6\ -\ -\ 6\ 7\ 4]$.

By using the uniform gap penalty value $\delta = -0.4$ and similarity score function

$$s(a, b) = \begin{cases} 1, & \text{if } a = b \\ -0.6, & \text{if } a \neq b \end{cases}$$

we can calculate the similarity score of this alignment as:

$$\begin{aligned} & s(4, 6) + \delta + \delta + s(6, 6) + s(7, 7) + \delta \\ = \ & -0.6 + (-0.4) + (-0.4) + 1 + 1 + (-0.4) \\ = \ & 0.2. \end{aligned}$$

The higher the similarity score is, the better the alignment is. The alignment of a segment pair that have the highest similarity score is the optimal alignment of this pair, and the corresponding similarity score is defined as the similarity index of this segment pair.

Given concepts of alignment and the similarity index of a segment pair, it is easy to formulate the local alignment problem.

Consider two symbolic sequences with arbitrary lengths:

$A = [a_1\, a_2 \cdots a_M]$

$B = [b_1\, b_2 \cdots b_N]$,

where a_m, $b_n \in \Sigma$, $m = 1, 2, \ldots, M$, $n = 1, 2, \ldots, N$. $A_{i:p}$ and $B_{j:q}$ denote segments, or called contiguous subsequences, of A and B, respectively:

$A_{i:p} = [a_i\, a_{i+1} \cdots a_p]$

$B_{j:q} = [b_j\, b_{j+1} \cdots b_q]$.

$I(A_{i:p}, B_{j:q})$ denotes the similarity index of the segment pair ($A_{i:p}$, $B_{j:q}$). The goal of the local alignment problem is searching for the optimal segment pair whose similarity index is the highest among all segment pairs. This highest similarity index is then defined as the similarity index of the two sequences A and B, which is denoted by $S(A, B)$. The only exception is that the two sequences are totally different, and all the segment pairs have a negative similarity index. In this case, $S(A, B)$ is set to be 0, which means no similarity at all. In a nut shell, the optimal local alignment of sequences A and B is the optimal alignment of the optimal segment pair of two sequences,

and the similarity index $S(A, B)$ of these two sequences can be expressed by the following equation:

$$S(A, B) = \max_{1\le i\le p\le M, 1\le j\le q\le N}(I(A_{i:p}, B_{j:q}), 0).$$

The Smith–Waterman algorithm provides us a procedure to find out the similarity index $S(A, B)$ and the corresponding local alignment.

The Smith–Waterman algorithm generates an index matrix H whose element $H_{p+1,q+1}$ denotes the maximum positive similarity index of segment pairs ending in a_p and b_q, respectively. If there is no positive similarity index, $H_{p+1,q+1} = 0$. In other words,

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

When one or both of the segments are empty, the segments pair similarity index is 0. As a result, $H_{1,q} = 0$, and $H_{p,1} = 0$ for any p and q.

$H_{p+1,q+1}$ can be recursively calculated by the following equation with a uniform gap penalty δ:

$$H_{p+1,q+1} = \max\{H_{p,q} + s(a_p, b_q), H_{p,q+1} + \delta, H_{p+1,q} + \delta, 0\}. \quad (1)$$

Eq. (1) is somewhat different from the one provided in Smith and Waterman (1981), since a uniform gap penalty is used here. In the review article Vingron and Waterman (1994), the authors described the algorithm by the same equation as Eq. (1) except the sign of δ, since in that paper δ was set a positive value and a minus sign was then used before the penalty term.

Based on equation (1), a dynamic programming algorithm can be easily developed to solve this optimization problem. The algorithm is described by the following steps:

1. Build a matrix $H \in R^{(M+1)\times(N+1)}$, and initialize the first row and column of H to be 0.
2. Calculate the other entries in matrix H from upper left to lower right by Eq. (1).
3. Find the highest value in the matrix H. This value is the similarity index of the two sequences ($S(A, B)$).
4. Go backward from this highest value until meet an entry with value 0. The path shows the optimal local alignment (detailed in the following example).

As an example consider two sequences $A = [1\ 2\ 1\ 4\ 1]$ and $B = [1\ 3\ 2\ 1\ 1\ 2\ 3]$. The penalty of a gap is $\delta = -0.4$; the match score is 1; and the penalty for mismatching is $\mu = -0.6$. The matrix H and the optimal local alignment are given by Fig. 2. The calculation of H is straightforward. For instance, the calculation of $H_{6,6}$ is based on the values of $H_{5,5}$, $H_{5,6}$ and $H_{6,5}$. Since the fifth symbols in both sequences are '1', we should calculate the values of $H_{5,5} + 1$, $H_{5,6} + \delta$ and $H_{6,5} + \delta$. They are 3.2, 1.6, and 1.4, respectively. So the value of $H_{6,6}$ should be the largest positive value among them, namely, 3.2.

The backward path search is based on the H matrix calculation. Since $H_{6,6}$ has the largest value in the matrix, the path search should begin at this entry. Since the value of $H_{6,6}$ is $H_{5,5} + 1$, we should go back from $H_{6,6}$ to $H_{5,5}$ instead of $H_{5,6}$ or $H_{6,5}$. Then we find that the value of $H_{5,5}$ is calculated from $H_{4,5}$, so we should go back to $H_{4,5}$. Continuously do this until we meet the first 0 entry at $H_{1,1}$. Now the whole path is obtained. The entries of the path are underscored in Fig. 2.

Then we go forward the path to complete the optimal local alignment. For each diagonal move, add a corresponding symbol in both aligned segments. For instant, the path start at $H_{1,1}$ and moves to $H_{2,2}$; so the first symbol in the aligned segments should be a_1 and b_1, namely, '1' and '1', respectively. For each horizontal move, add a corresponding symbol in the aligned segment of B, and add a gap in the aligned segment of A. For instant, the second move in the path is a horizontal one, so the second symbol in the aligned segment of B is b_2, namely, '1', while the second symbol in the aligned segment of A is a gap. For each vertical move, add a corresponding symbol in the aligned segment of A and a gap in the aligned segment of B. Following these rules, we finally reach the optimal local alignment: [1 − 2 1 4 1] and [1 3 2 1 − 1].

	Δ	1	3	2	1	1	2	3
Δ	0	0	0	0	0	0	0	0
1	0	1	0.6	0.2	1	1	0.6	0.2
2	0	0.6	0.4	1.6	1.2	0.8	2	1.6
1	0	1	0.6	1.2	2.6	2.2	1.8	1.4
4	0	0.6	0.4	0.8	2.2	2	1.6	1.2
1	0	1	0.6	0.4	1.8	3.2	2.8	2.4

(a) Similarity table H

$A_{1:5}$:	1		–		2	1		4		1		
Score:	1	+	−0.4	+	1	+	1	+	−0.4	+	1	= 3.2
$B_{1:5}$:	1		3		2	1		–		1		

(b) Optimal local alignment

Fig. 2 – (a and b) Local alignment result.

3. Modified Smith–Waterman algorithm for alarm flood pattern matching

The alarm flood pattern matching problem is to calculate a certain similarity index between alarm floods, and determine which floods should be classified in one group (pattern) based on the similarity index. The common segment of alarm floods in a pattern may be used as a symptom of this kind of alarm floods to determine whether a incoming flood belongs to that pattern.

If we use a unique symbol to denote each alarm type, finding the similarity index and common segment of alarm flood sequences is very similar to the goal of approximate matching. Particularly, a good sequence matching algorithm for alarm sequences should have the following properties:

1. Tolerant to some irrelevant alarms occurring in one or both of the alarm sequences.
2. Somewhat tolerant to ambiguity of order.

The reason why the first property is required is obvious. Standard approximate matching algorithms considering gap and mismatch can achieve this requirement very well. As for the second property, it is possible that several strongly connected alarms arise almost simultaneously, but the order of them in alarm sequences varies from time to time. Moreover, because of random detection delay, one alarm may arise after its cascaded alarms. As a result, when the time stamps of two alarms are close, the order of these two alarms is not so important. In other words, we should make the order of these alarms vague.

In the approximate matching area, a similar topic is swap. A few papers focus on this topic such as Dombb et al. (2010) and Lipsky et al. (2010). However they discuss the sequences

without time stamps. In Cisar et al. (2009) and Hostalkova and Stluka (2010), the authors considered time constraints. However the pattern-growth methods are for frequent sequence mining, but not for approximate pattern matching. Moreover, hard time constraints instead of soft ones are concerned in those methods. Thus those methods are not suitable for our problem. As a result, it is necessary to propose a modified approximate matching algorithm that is suitable for alarm flood pattern matching.

We assume that the alphabet of alarm types is

$$\Sigma = 1, 2, \ldots, K.$$

The size of the alphabet is K. This assumption is only for convenience (without loss of generality) since we can map any finite alphabet to this one. A time-stamped alarm sequence is defined as follows:

$$A = a_1\, a_2 \cdots a_M$$

$$a_m = (e_m, t_m),\ m = 1, 2, \ldots, M,$$

where e_m is the alarm type, namely an integer from 1 to K, and t_m is the time stamp of the alarm a_m.

The essential idea of the modification on the Smith–Waterman algorithm for the time-stamped alarm sequence is to redefine the similarity score of a symbolic pair $s(a, b)$ to a similarity score of a time-stamped alarm pair. In order to explain this new similarity score clearly, two new concepts are introduced: 'time distance vector' and 'time weight vector'.

A time distance vector is defined for each alarm in an alarm sequence. The time distance vector for the mth alarm a_m is as follows:

$$\mathbf{d}_m = [d_m^1\, d_m^2 \cdots d_m^K]^T, \quad \text{for } k = 1, 2, \ldots, K.$$

$$d_m^k = \begin{cases} \min\limits_{1\le i\le M} \left\{ |t_m - t_i| : e_i = k \right\}, & \text{if the set is not empty} \\ \infty, & \text{otherwise,} \end{cases} \tag{2}$$

An entry d_m^k in the time distance vector $\mathbf{d}_m$ carries the information of the time gap between the mth alarm and the nearest alarm on the time axis with alarm type k. If there is no type k alarm in the alarm sequence, namely $e_i \neq k$ for all $i \in [1, M]$, the time gap is ∞. Obviously, since the mth alarm has the alarm type e_m, $d_m^{e_m}$ is zero.

Then we define a time weight vector for each alarm in an alarm sequence. The time weight vector for the mth alarm is as follows:

$$\begin{aligned} \mathbf{w}_m &= [w_m^1\, w_m^2 \cdots w_m^K]^T \\ &= [f(d_m^1) f(d_m^2) \cdots f(d_m^K)]^T, \end{aligned} \tag{3}$$

where $f(\cdot) : \mathbb{R} \mapsto \mathbb{R}$ is a time weighting function with respect to the time distance d_m^k. A function that satisfies the following conditions can be used as $f(\cdot)$:

1. Monotonically decreasing on the positive axis.
2. $f(0) = 1, f(\infty) = 0$.

As a result, the e_mth entry of the time weight vector $\mathbf{w}_m$ is 1, since $d_m^{e_m}$ is zero. If one type of alarm, say the jth type, arises very close to the mth alarm on the time axis, the jth entry in $\mathbf{w}_m$ has a large value almost 1. On the other hand, if one type of alarm does not exist in a neighborhood of the mth alarm on the time axis, it will get a low weight in vector $\mathbf{w}_m$, and the weight will go to zero if the time gap approaches ∞. In the following example and case study provided in this paper, when we do the sequences matching on a pair of time-stamped alarm sequences, we choose the scaled Gaussian function

$$f(x) = e^{-x^2/2\sigma^2} \tag{4}$$

as the time weighting function for the first sequence, and for the $f(\cdot)$ of the second sequence,

$$f(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{if } x \neq 0 \end{cases}. \tag{5}$$

The reasons why we choose two different time weighting functions for the two sequences are twofold. Firstly, the time difference is a relative but not absolute concept; so only blurring the order of one sequence actually can affect both sequences. Secondly, if we use the scaled Gaussian function for both sequences, one matching pair may be counted more than once when several alarms closely arise, which leads to a false high similarity index.

Now we redefine the similarity score $s(a, b)$ for a time-stamped alarm pair $((e_a, t_a), (e_b, t_b))$. In the classical Smith–Waterman algorithm, the similarity score function is a two-value function. The only possible values are the match score 1 and the mismatch penalty μ. In the modified algorithm, we define the similarity score function value as a linear combination of the match score and mismatch penalty, and the linear combination ratio is according to the time weight vectors $\mathbf{w}_a$ and $\mathbf{w}_b$ of the two alarms:

$$s((e_a, t_a), (e_b, t_b)) = \max_{1\le k\le K} [w_a^k \times w_b^k](1 - \mu) + \mu. \tag{6}$$

Substituting the redefined similarity weight for the original one in equation (1), the Smith–Waterman algorithm is modified to a time-stamped sequence approximate matching algorithm that is suitable for alarm flood pattern matching.

The modified algorithm has the following properties:

1. The largest similarity index of any two sequences with lengths M and N, respectively, is $\min(M, N)$.
2. If two sequences with the same length M have identical alarm sequences, the similarity index is M, regardless of time stamps.
3. If the scaled Gaussian function is used as the time weighting function, a larger σ leads to a larger or equal similarity index. If $\sigma = 0$, the modified algorithm reduces to standard Smith–Waterman algorithm.
4. The similarity calculation is not commutative, i.e., $S(A, B) \neq S(B, A)$.

According to the first property, we can normalize the similarity index to be between 0 and 1. The second property states that if the alarm sequences are the same, the normalized similarity index is 1. Property 3 gives some hints on how to choose the time weighting function. The fourth property means that the calculated similarity index of sequence A to sequence B may not equal to that of sequence B to sequence A. To fix this problem, we choose the greater one as the similarity index of the pair of sequences.

Δ	(1,0)	(3,1)	(2,4)	(1,4.2)	(4,7)	(3,9)	(2,12.5)
Δ	0	0	0	0	0	0	0	0
(4,0)	0	0	0	0	0	1	0.6	0.2
(1,3)	0	1	0.6	0.2	1	0.6	1.370	0.970
(3,5)	0	0.6	2	1.6	1.2	0.8	1.6	1.290
(1,7.5)	0	1	1.6	2.951	2.6	2.2	1.8	2.551
(2,8)	0	0.951	1.2	2.6	3.902	3.502	3.102	2.8
(3,13)	0	0.551	1.951	2.2	3.502	3.302	4.502	4.102
(1,20)	0	1	1.551	1.8	3.2	2.902	4.102	3.902

(a) Similarity table H

$A_{2:6}$:	(1,3)		(3,5)		(1,7.5)		(2,8)	–	(3,13)		
Score:	1	+	1	+	0.951	+	0.951	+ −0.4	+ 1	=	4.502
$B_{1:6}$:	(1,0)		(3,1)		(2,4)		(1,4.2)	(4,7)	(3,9)		

(b) Optimal local alignment

Fig. 3 – Local alignment result by modified Smith–Waterman algorithm.

At the end of this section, we provide a simple example to illustrate the modified algorithm. Consider two time-stamped alarm sequences A and B:

$$A = (4, 0),\ (1, 3),\ (3, 5),\ (1, 7.5),\ (2, 8),\ (3, 13),\ (1, 20)$$
$$B = (1, 0),\ (3, 1),\ (2, 4),\ (1, 4.2),\ (4, 7),\ (3, 9),\ (2, 12.5).$$

The alphabet is $\Sigma = \{1, 2, 3, 4\}$ with the size $K = 4$; and the lengths of both sequences are 7, i.e., $M = 7$. For the alarm sequence A we need to calculate 7 time distance vectors since there are 7 alarms, and each vector should consist of 4 entries since the size of the alphabet is 4. Since the alarm type of the first alarm is $e_1 = 4$, the fourth entry of d_1 is 0. The nearest alarm with alarm type 1 occurs at time instant 3, so $d_1^1 = 3 - 0 = 3$. Similarly, $d_1^2 = 8 - 0 = 0$ and $d_1^3 = 5 - 0 = 0$. We can complete all the 7 time distance vectors in the same way to reach the following result:

$$[\mathbf{d}_1 \quad \mathbf{d}_2 \quad \cdots \quad \mathbf{d}_7] = \begin{bmatrix} 3 & 0 & 2 & 0 & 0.5 & 5.5 & 0 \\ 8 & 5 & 3 & 0.5 & 0 & 5 & 12 \\ 5 & 2 & 0 & 2.5 & 3 & 0 & 7 \\ 0 & 3 & 5 & 7.5 & 8 & 13 & 20 \end{bmatrix}.$$

Element-wisely calculate the time weighting function (4) of the distance vectors, the time weight vectors are obtained:

$$[\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_7] = \begin{bmatrix} 0.325 & 1.000 & 0.607 & 1.000 & 0.969 & 0.023 & 1.000 \\ 0.000 & 0.044 & 0.325 & 0.969 & 1.000 & 0.044 & 0.000 \\ 0.044 & 0.607 & 1.000 & 0.458 & 0.325 & 1.000 & 0.002 \\ 1.000 & 0.325 & 0.044 & 0.001 & 0.000 & 0.000 & 0.000 \end{bmatrix}.$$

Setting the algorithm parameters $\delta = -0.4$, $\mu = -0.6$, the matrix H for similarity index of alarm sequences A to B and the optimal local alignment is shown in Fig. 3. The calculation is very similar to the classical Smith–Waterman algorithm that we introduced in Section 2.2. The only difference is the calculation of similarity score. For instance, when we calculate the value of $H_{5,4}$, we need to get the similarity score of the 4th alarm in sequence A and the 3rd alarm in sequence B. We know that the time weight vector of the 4th alarm in sequence A is $\mathbf{w}_4 = [1\ 0.969\ 0.458\ 0.001]^T$. For the second sequence B, the time weight vector of the 3rd alarm is $[0\ 1\ 0\ 0]^T$ using the time weighting function (5). Do the element-wise multiplication of these two vectors and pick up the largest element, which is 0.969, we can calculate the similarity score as: $0.969 \times (1 + 0.6) - 0.6 = 0.951$. Then the value of $H_{5,4}$ can be calculated by the following equation:

$$H_{5,4} = \max\{H_{4,3} + 0.951, H_{4,4} - 0.4, H_{5,3} - 0.4, 0\} = 2.951.$$

Since the time difference of the 4th and 5th alarms in sequence A is small, our algorithm can swap them to match the 3rd and 4th alarms in sequence B as shown in Fig. 3(b). While the algorithm will not do the same for the 5th and 6th alarms in sequence A to match the last two alarms in sequence B, because the time gap of these two alarms is large. Then we repeat this procedure to calculate the matrix H for the similarity index of B to A which is shown in Fig. 4.

Δ	(4,0)	(1,3)	(3,5)	(1,7.5)	(2,8)	(3,13)	(1,20)
Δ	0	0	0	0	0	0	0	0
(1,0)	0	0	1	0.812	1	0.6	0.812	1
(3,1)	0	0	0.812	2	1.624	1.224	1.6	1.624
(2,4)	0	0	0.992	1.6	2.992	2.624	2.224	2.592
(1,4.2)	0	0.001	1	1.2	2.6	3.984	3.584	3.224
(4,7)	0	1	0.6	1.370	2.2	3.584	4.355	3.955
(3,9)	0	0.6	0.490	1.6	1.8	3.184	4.584	4.184
(2,12.5)	0	0.2	0.090	1.2	1.4	2.8	4.184	3.984

Fig. 4 – B to A similarity table H.

It can be found that in this example, the similarity indices of A to B and B to A are close to each other; and the optimal local alignments are the same. Usually this is the case. Finally, we obtain the normalized similarity index of the two sequences:

$$S(A, B) = \frac{\max(4.502, 4.584)}{\min(7, 7)} = 0.655.$$

4. Case study

The proposed method is applied to data from a hydrocracking unit of a typical refinery process consists of equipments like furnaces, pumps, compressors, separation drums, etc. In the alarm data over a period of 9 months, there are totally 506 alarm tags and each alarm tag may have several different alarm identifiers such as 'PVLO', 'PVHI', 'PVLL' or 'PVHH'. As a result, the alarm data includes more than 1000 types of alarms. Thanks to the good basic alarm rationalization performance that process control engineers have done, the process has a good normal alarm statistics, namely under 6 alarms/h. The main concern is alarm floods when some abnormalities or events take place. Our pattern matching technique is suitable for this kind of alarm event data, since it is easy for us to identify the alarm flood periods simply by counting the alarm frequency in the event data. According to the definitions of alarm flood in EEMUA (2007) and ISA (2009), we identified 39 alarm floods over this period. The lengths of the 39 alarm flood sequences vary from 10 to 297 alarms.

We first preprocess the 39 alarm flood sequences by merging the alarms with the same alarm type and occurring within 5 s (5 s off-delay timer) to eliminate chattering alarms. The rationality and effect of the preprocessing is thoroughly discussed in Hostalkova and Stluka (2010). Then the proposed algorithm is applied on each pair of preprocessed alarm flood sequences to calculate the normalized similarity index. Finally we obtain a 39×39 similarity matrix which is show in Fig. 5 in a color map format. The results are obtained when we set

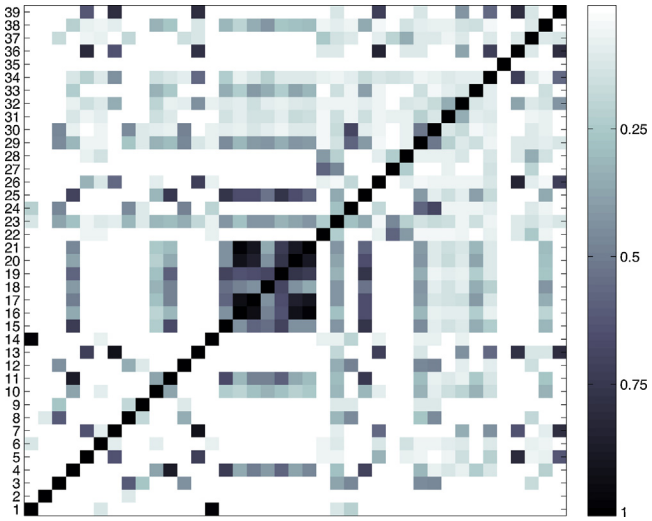

Fig. 5 – Similarity color map. (For the interpretation of colour in the artwork, the reader is referred to the web version of the article.)

$\delta = -0.4$, $\mu = -0.6$, and $\sigma^2 = 4$. In the color map, the darker the color the more similar the two sequences are. All the diagonal entries in the color map are black, i.e., have the value 1. It is expected, since properties 1 and 2 of the proposed algorithm guarantee that the normalized similarity index of two identical alarm sequences is always 1. We can also find some pairs with high similarity (normalized similarity index is greater than 0.6) such as (1, 14), (26,36), (24,30) and so on. It is confirmed by the site engineers that the found similar pairs are reasonable.

Figs. 6–8 show the alignment of several pairs of alarm sequences, which can give us a clearer picture of the algorithm. For the first pair (floods 1 and 14), no swap is needed to get the optimal alignment. In this case, the modified algorithm works similar as the standard algorithm. In the second pair (floods 26 and 36), the modified algorithm finds that if we shift the alarm '001.PVHI' by 2 s, a better local alignment can be obtained, which can increase the similarity of the two alarm flood sequences. For the third pair (floods 24 and 30), a lot of alarms arise almost simultaneously. In this case, the order of the sequence is of little significance, therefore the original Smith–Waterman algorithm cannot handle it. But the modified algorithm can discover the similarity between the two sequences since it is insensitive to the sequence, to some extent.

14	
002.PVHI	14:08:48
028.OFFNORM	14:17:06
029.OFFNORM	14:18:48
030.OFFNORM	14:20:18
030.OFFNORM	14:27:34
031.OFFNORM	14:28:24
032.OFFNORM	14:29:05
033.OFFNORM	14:29:29
034.OFFNORM	14:30:03
035.OFFNORM	14:30:45
036.OFFNORM	14:31:10
037.OFFNORM	14:31:45
038.OFFNORM	14:32:15
039.OFFNORM	14:37:23
040.OFFNORM	14:37:57
041.OFFNORM	14:38:18

1	
028.OFFNORM	08:49:22
029.OFFNORM	08:50:15
031.OFFNORM	08:50:36
032.OFFNORM	08:50:56
033.OFFNORM	08:51:14
034.OFFNORM	08:51:25
035.OFFNORM	08:53:11
036.OFFNORM	08:53:23
037.OFFNORM	08:53:40
038.OFFNORM	08:54:18

Fig. 6 – Alarm flood sequences alignment (floods 1 and 14).

36	
057.PVHI'	16:06:56
058.OFFNORM	16:08:03
059.OFFNORM	16:08:05
061.OFFNORM	16:08:05
062.OFFNORM	16:08:06
001.PVHI	16:09:03
063.PVLO	16:09:04
066.OFFNORM	16:09:05
065.OFFNORM	16:09:05
064.OFFNORM	16:09:05
001.PVLO	16:13:40

26	
495.PVLO	16:23:45
001.PVHI	16:25:51
057.PVHI	16:28:04
058.OFFNORM	16:28:15
059.OFFNORM	16:28:17
061.OFFNORM	16:28:17
062.OFFNORM	16:28:18
063.PVLO	16:29:16
066.OFFNORM	16:29:17
065.OFFNORM	16:29:17
064.OFFNORM	16:29:17
001.PVHI	16:31:01
307.PVHI	16:31:17
307.PVHH	16:31:17
308.OFFNORM	16:31:21
233.PVLO	16:31:23
496.PVHI	16:31:24
373.OFFNORM	16:31:28
043.PVLO	16:31:33
⋮	

Fig. 7 – Alarm flood sequences alignment (floods 26 and 36).

Finally, we cluster the floods using single-linkage method (Wu et al., 2010). This clustering method is also applied in alarm data correlation analysis to cluster related types of alarms (Kondaveeti et al., 2010; Yang et al., 2011). Fig. 9 shows the clustered similarity color map. Setting the threshold at 0.6, the groups we obtain from the 39 floods are: (16, 21, 17, 20), (15, 19, 25), (4, 11), (5, 7, 13, 39), (26, 36), (24, 30), (1, 14). If we decrease the threshold, the first three groups and the 4th and 5th groups merge. After clustering, we can obtain the characteristic subsequence(s) of a flood group according to the optimal alignment of alarm flood sequence pairs in the group. For example, for the first group (16, 21, 17, 20), we can find the following characteristic subsequence: ['046.PVHI',

24	
247.PVLO	05:33:36
245.PVLO	05:33:36
249.OFFNORM	05:33:36
303.PVHI	05:33:36
289.OFFNORM	05:33:36
166.PVLO	05:33:36
246.OFFNORM	05:33:36
248.PVLO	05:33:36
267.OFFNORM	05:33:36
261.OFFNORM	05:33:36
250.PVLO	05:33:36
175.OFFNORM	05:33:36
305.OFFNORM	05:33:36
029.OFFNORM	05:33:36
030.OFFNORM	05:33:36
263.OFFNORM	05:33:36
028.OFFNORM	05:33:36
167.OFFNORM	05:33:36
168.OFFNORM	05:33:36
176.OFFNORM	05:33:36
177.OFFNORM	05:33:36
258.OFFNORM	05:33:36
260.OFFNORM	05:33:36
277.SET MAN	05:33:36
283.OFFNORM	05:33:36
292.OFFNORM	05:33:36
240.OFFNORM	05:33:36
169.OFFNORM	05:33:36
411.OFFNORM	05:33:36
170.OFFNORM	05:33:37
178.OFFNORM	05:33:37

30	
002.PVHI	17:13:25
411.OFFNORM	17:17:38
116.BADPV	17:17:59
93.PVLO	17:20:56
116.BADPV	17:24:50
290.OFFNORM	17:27:20
114.BADPV	17:27:20
247.PVLO	17:27:20
245.PVLO	17:27:20
249.OFFNORM	17:27:20
303.PVHI	17:27:20
246.OFFNORM	17:27:20
248.PVLO	17:27:20
261.OFFNORM	17:27:20
250.PVLO	17:27:20
175.OFFNORM	17:27:20
030.OFFNORM	17:27:20
263.OFFNORM'	17:27:20
167.OFFNORM	17:27:20
168.OFFNORM	17:27:20
170.OFFNORM	17:27:20
176.OFFNORM	17:27:20
177.OFFNORM	17:27:20
178.OFFNORM	17:27:20
258.OFFNORM	17:27:20
260.OFFNORM	17:27:20
283.OFFNORM	17:27:20
259.OFFNORM	17:27:20
262.OFFNORM	17:27:20
413.BADPV	17:27:21
166.BADPV	17:32:15
166.BADPV	17:34:21
⋮	

Fig. 8 – Alarm flood sequences alignment (floods 24 and 30).

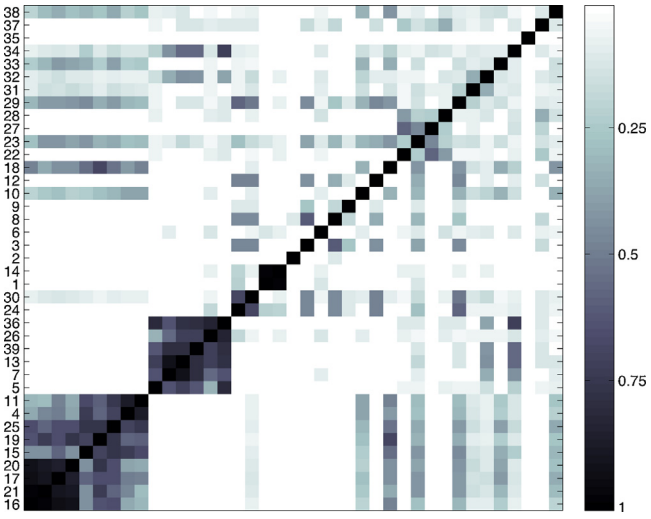

Fig. 9 – Clustered similarity color map. (For the interpretation of colour in the artwork, the reader is referred to the web version of the article.)

'046.BADPV', '047.BADPV', ('048.OFFNORM' and '047.PVHI'), ('046.PVHI' and '046.BADPV')].

We have also tried the classical Smith–Waterman algorithm on the same alarm data set to make a comparison. It is easy to prove that the similarity indices obtained by the modified algorithm are no smaller than the ones calculated by the original algorithm, since the optimal alignment of the original algorithm always falls in the feasible region (but not necessary the optimal one) of the modified one.

In the case study, there is no obvious discrepancy on the similarity indices between the modified and the original Smith–Waterman algorithms for the alarm sequences whose chronological orders are very clear, e.g., floods 1 and 14 shown in Fig. 6. More than 90% alarm sequence pairs in our case study are in this case. However, the similarity indices of the remaining ones using the modified algorithm increase obviously. For example, the similarity index of floods 24 and 30 shown in Fig. 8 decreases from 0.63 to 0.46 if the modified algorithm is replaced by the original one. Another typical example is floods 7 and 13 as shown in Fig. 10. The similarity index generated by the modified algorithm is pretty high (0.9), while reduced to 0.68 by the original algorithm. The reason of this reduction is the segments marked by red boxes in Fig. 10. The time intervals of these three segments are very short, so the orders of the alarms affect the original algorithm much more than the modified algorithm. In this group, a low-flow alarm leads to a low pressure. Then the low pressure affects three

7	
264.OFFNORM	23:19:27
027.PVLO	23:20:42
060.PVLO	23:20:42
059.OFFNORM	23:21:14
061.OFFNORM	23:21:15
062.OFFNORM	23:21:17
063.PVLO	23:22:15
066.OFFNORM	23:22:16
065.OFFNORM	23:22:16
064.OFFNORM	23:22:16
057.PVLO	23:24:04
027.PVHI	23:29:43
264.OFFNORM	23:30:43

13	
027.PVLO	11:33:57
059.OFFNORM	11:34:06
061.OFFNORM	11:34:08
062.OFFNORM	11:34:09
063.PVLO	11:35:07
064.OFFNORM	11:35:07
066.OFFNORM	11:35:08
065.OFFNORM	11:35:08
057.PVLO	11:37:17
001.PVLO	11:37:42

Fig. 10 – Alarm flood sequences alignment (floods 7 and 13).

sequential pressure variables to be abnormal almost simultaneously. We can totally find 9 more pairs of alarm sequences whose similarity indices are greater than 0.6 by the modified algorithm.

The modified algorithm can find more similar floods than the classical one, since the modified similarity indices are always no less than the original one. This is the main advantage of the modified algorithm: decreasing the probability of false negative. It can get more similar flood pairs and discover similarities that the original algorithm cannot find, such as floods 1 and 14, and floods 7 and 13. Meanwhile it may also be a disadvantage, since it may introduce more false positive similarities. Since the goal of our work is to construct a pool of similar flood group candidates to aid the engineers or experts, emphasizing more on low false negative rate is reasonable.

Moreover, the modified algorithm introduces a new control parameter: the variance of the scaled Gaussian weighting function. The modified algorithm converges to the classical one when the variance approaches 0. When the variance is very large, the order information is totally discarded, and the algorithm only counts the common alarms in the two floods. As a result, this control parameter affects the matching result. Choose other weighting functions instead of the Gaussian function may further impact the result. Although it makes the algorithm more flexible, it also increase the difficulty to handle the algorithm. Weighting function selection and its relationship to the distribution of detection delay are still open questions of the modified algorithm.

Another main disadvantage of the modified algorithm is its heavier computational burden. The calculation of classical Smith–Waterman similarity indices of all the flood pairs in this case study finished in 0.3 s using Matlab, but it takes about half a minute to finish all the calculation for the modified algorithm. The main reason is the calculation of similarity score. In the original algorithm, only one Boolean calculation is required, but in the modified algorithm, K multiplication operations, where K is the size of the alphabet, and one maximum operation are necessary. As a result, the computational complexity for one flood pair is $O(MNK)$ (the classical SW algorithm's complexity is $O(MN)$), where M and N are the length of the two floods, respectively. When there are Q floods in the historical data, the similarity indices of $Q(Q-1)/2$ flood pairs are required, so the total computational burden is $O(MNKQ^2)$. However, the processing time is acceptable since this analysis is off-line.

5. Concluding remarks

In this paper, a modified Smith–Waterman algorithm is proposed to obtain a similarity index of alarm flood sequences and cluster similar floods. The modified algorithm utilizes the time stamp information of the sequence to blur the order of alarms when they occur close to each other. Alarm data from an actual chemical process is used for validation.

The accuracy analysis, namely the false positive and false negative rates analysis, will be considered in the future. Especially, the correlation between the accuracy and the time weighting function will be discussed. The next step is to apply the obtained result on on-line smart alarm management system design. Based on the clustered historical flood patterns, engineers or experts can make a thoroughly analysis, such as the root cause, criticality, and suggested reaction, for each pattern, which will be recorded in the alarm management system.

Then by calculating the similarity indices between an incoming alarm flood and the clustered historical alarms, which pattern(s) the new flood likely belongs to can be determined. Then the operator will be informed of the pattern matching result, and be provided the useful information stored in the system of the matched pattern(s).

Acknowledgements

This work was supported by NSERC. We would like to thank Kabir Ahmed for his help in preprocessing the alarm data used in the industrial case study.

References

Aach, J., Church, G., 2001. Aligning gene expression time series with time warping algorithms. Bioinformatics 17, 495–508.

Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D., 1990. Basic local alignment search tool. J. Mol. Biol. 215, 403–410.

Amir, A., Aumann, Y., G., B., Levy, A., Lipsky, O., Porat, E., Skiena, S., Vishne, U., 2009. Pattern matching with address errors: rearrangement distances. J. Comput. Syst. Sci. 75, 359–370.

Chakrabarti, S., Sarawagi, S., Dom, B., 1998. Mining surprising pattern using temporal description length. In: Proceedings of 24th International Conference on Very Large Databases, New York, USA, pp. 606–617.

Chandola, V., Banerjee, A., Kumar, V., 2012. Anomaly detection for discrete sequences: a survey. IEEE Trans. Knowl. Data Eng. 24, 823–839.

Cisar, P., Hostalkova, E., Stluka, P., 2009. Data mining techniques for alarm rationalization. In: 19th European Symposium on Computer Aided Process Engineering, Cracow, Poland, pp. 1457–1462.

Cole, R., Hariharan, R., 1998. Approximate string matching: a faster simpler algorithm. In: Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, pp. 463–472.

Dombb, Y., Lipsky, O., Porat, B., Porat, E., Tsur, A., 2010. The approximate swap and mismatch edit distance. Theor. Comput. Sci. 411, 3814–3822.

EEMUA, 2007. Alarm Systems: A Guide to Design, Management and Procurement. EEMUA Publication 191, 2nd ed. Engineering Equipment and Materials Users' Association, London.

Folmer, J., Pantforder, D., Vogel-Heuser, B., 2011. An analytical flood reduction to reduce operator's workload. In: HCII'11 Proceedings of the 14th International Conference on Human-computer Interaction: Users and Applications, volume Part IV, pp. 297–306.

Folmer, J., Vogel-Heuser, B., 2012. Computing dependent industrial alarms for alarm flood reduction. In: 9th International Multi-Conference on Systems, Signal and Devices, pp. 1–6.

Ge, X., Smyth, P., 2000. Deformable Markov model templates for time-series pattern matching. In: Proceedings of the 6th ACM SIGKDD, pp. 81–90.

Henningsen, A., Kemmerer, J.P., 1995. Intelligent alarm handling in cement plants. IEEE Ind. Appl. Mag. 1, 9–15.

Hofmeyr, S., Forrest, S., Somayaji, A., 1998. Intrusion detection using sequences of system calls. J. Comput. Security 6, 151–180.

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

ISA, 2009. Management of Alarm Systems for the Process Industries. ANSI/ISA-18.2-2009, 2nd ed. The International Society of Automation, Research Triangle Park.

Izadi, I., Shah, S., Chen, T., 2010. Effective resource utilization for alarm management. In: Proceedings of 49th IEEE Conference on Decision and Control, Atlanta, USA, pp. 6803–6808.

Johannesmeyer, M.C., 1999. Abnormal situation analysis using pattern recognition techniques and historical data. Master's thesis, Santa Barbara, CA.

Kondaveeti, S., Izadi, I., Shah, S., Black, T., 2010. Graphical representation of industrial alarm data. In: Proceedings of 11th IFAC symposium on Analysis, Design and Evaluation of Human–Machine Systems, Valenciennes, France.

Lane, T., Brodley, C., 1999. Temporal sequence learning and data reduction for anomaly detection. ACM Trans. Inform. Syst. Security 2, 295–331.

Levenshtein, V., 1965. Binary codes capable of correcting spurious insertions and deletions of ones. Problems Inform. Transm. 1, 8–17.

Lipman, D., Pearson, W., 1990. Rapid and sensitive protein similarity searches. Science 227, 1435–1441.

Lipsky, O., Porat, B., Porat, E., Shalom, B.R., Tzur, A., 2010. String matching with up to k swaps and mismatches. Inform. Comput. 208, 1020–1030.

Liu, J., Lim, K.W., Ho, W.K., Tan, K.C., Srinivasan, R., Tay, A., 2003. The intelligent alarm management system. IEEE Softw. 20, 66–71.

Mabroukeh, N., Ezeife, C., 2010. A Taxonomy of Sequential Pattern Mining Algorithms. ACM Computing Surveys, 43.

Navarro, G., Raffinot, M., 2002. Flexible Pattern Matching in Strings. Cambridge University Press.

Needleman, S., Wunsch, C., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453.

Ng, B., Meyers, C., Boakye, K., Nitao, J., 2010. Towards applying interactive POMDPs to real-world adversary modeling. In: Proccedings of the 22 Innovative Applications of Artificial Intelligence Conference, pp. 1814–1820.

Ratanamahatana, C., Keogh, E., 2005. Three myths about dynamic time warping. In: Proceedings of SIAM International Conference on Data Mining, pp. 506–510.

Singhal, A., 2002. Pattern matching in multivariate time-series data. PhD thesis, Santa Barbara, CA.

Singhal, A., Seborg, D.E., 2001. Matching pattern from historical data using PCA and distance similarity factors. In: Proceedings of American Control Conference, Arlington, VA, pp. 1759–1764.

Smith, T., Waterman, M., 1981. Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197.

Srikant, R., Agrawal, R., 1996. Mining sequential patterns: generalizations and performance improvements. In: 5th International Conference of Extending Database Technology, pp. 3–17.

Srinivasan, R., Liu, J., Lim, K.W., Tan, K.C., Ho, W.K., 2004. Intelligent alarm management in a petroleum refinery. Hydrocarbon Process. 83, 47–53.

Umeda, T., Kuriyama, T., O'Shima, E., Matsuyama, H., 1980. A graphical approach to cause and effect analysis of chemical processing systems. Chem. Eng. Sci. 35, 2379–2388.

Vingron, M., Waterman, M., 1994. Sequence alignment and penalty choice review of concepts, case studies and implications. J. Mol. Biol. 235, 1–12.

Wu, H., Tien, Y., Chen, C., 2010. Gap: a graphical environment for matrix visualization and cluster analysis. Comput. Stat. Data Anal. 54, 767–778.

Yamanishi, K., Maruyama, Y., 2005. Dynamic syslog mining for network failure monitoring. In: Proceedings of 11th ACM SIGKDD, pp. 499–508.

Yang, F., Shah, S., Xiao, D., Chen, T., 2011. Improved correlation analysis and visualization for industrial alarm data. In: 18th IFAC World Congress, Milano, Italy.