



Contents lists available at ScienceDirect

Chemical Engineering Research and Design

|ChemE

journal homepage: [www.elsevier.com/locate/cherd](http://www.elsevier.com/locate/cherd)

# A method for pattern mining in multiple alarm flood sequences<sup>☆</sup>

Shiqi Lai\*, Tongwen Chen

Department of Electrical and Computer Engineering, University of Alberta, Edmonton T6G 2V4, Canada

---

## ARTICLE INFO

### Article history:

Received 20 March 2015

Received in revised form 22 May 2015

Accepted 10 June 2015

Available online xxx

### Keywords:

Alarm flood analysis

Time-stamped sequences

Multiple sequence alignment

Smith–Waterman algorithm

Industrial alarm monitoring

---

## ABSTRACT

Alarm flood is a serious hazard for industrial processes, alarm management techniques such as delay timers and dead-bands often are incapable of suppressing alarm floods because of the existence of consequence alarms. However, this problem could be handled by alarm flood analysis, which could help find the root cause, locate badly designed part in alarm systems and predict incoming alarm floods. In this paper we propose a method for pattern mining in multiple alarm flood sequences by extending a time-stamp adapted Smith–Waterman algorithm to the case with multiple sequences, making up one of the missing steps in alarm flood analysis. The technique involves the following new elements: similarity scoring functions, a dynamic programming equation, a back tracking procedure, and an alignment generation method. A dataset from an actual petrochemical plant has been used to test the effectiveness of the proposed algorithm.

© 2015 The Institution of Chemical Engineers. Published by Elsevier B.V. All rights reserved.

## 1. Introduction

Alarm systems play an important role in industrial process monitoring. Safety, quality, and efficiency of production depend heavily on alarm configurations, especially for large plants. Nowadays, advanced technologies make it much easier for a process to be monitored by a bunch of variables using new sensors and software. This convenience, however, can result in a huge number of measured variables and their corresponding configured alarms. An alarm flood, a serious hazard for industrial process monitoring, can be one of the consequences of such vast alarm configuration. During alarm floods, operators can be overwhelmed by the large amount of alarms raised in a short period of time, leading likely to improper handling of important abnormalities. EEMUA and ISA standards (EEMUA, 2013; ISA, 2009) recommend to set the threshold alarm rate for alarm floods to be 10 alarms per 10 min based on operators' normal response time (Dal Vernon et al., 1193).

Usually a large proportion of an alarm flood are nuisance alarms, of which chattering alarms form an important part.

Methods such as high density alarm plots, calculation of a chattering index, delay-timers, and dead-bands can be used to visualize, quantify, and reduce such chattering alarms (Kondaveeti et al., 2010, 2013; Izadi et al., 2009). However, by applying delay-timers and dead-bands, often one cannot totally suppresses alarms during alarm floods; the remaining alarms are mainly consequence alarms, which can be caused by three reasons: (1) process state changes such as start-up and shutdown, (2) bad alarm configurations such as redundant measurements on a single process and (3) causal relationships among measured variables. In this case, alarm flood analysis are useful to reveal correlation of alarm messages and discover possible patterns in alarm flood sequences.

The discovered patterns are helpful in alarm management. For example, the first alarm message in a pattern is usually more suspicion to be the root cause of the following alarms in the sequence. The discovered patterns can also help train the operators to handle corresponding series of alarm messages more efficiently in order to prevent the overwhelming situation during alarm floods. Some badly configured parts

\* This work was supported by an NSERC CRD project. A preliminary version of this paper was presented at the IFAC International Symposium on Advanced Control of Chemical Processes, June 7–10, 2015, Whistler, BC, Canada.

<sup>☆</sup> Corresponding author. Tel.: +1 587 938 3239.

E-mail addresses: [s.lai3@ualberta.ca](mailto:s.lai3@ualberta.ca) (S. Lai), [tchen@ualberta.ca](mailto:tchen@ualberta.ca) (T. Chen).

<http://dx.doi.org/10.1016/j.cherd.2015.06.019>

0263-8762/© 2015 The Institution of Chemical Engineers. Published by Elsevier B.V. All rights reserved.

in alarm systems can be revealed by the alarm flood patterns as well. Moreover, if a pattern database can be set up, it would become possible to match online alarm messages with the existing patterns, providing operators an early warning of incoming floods and their corresponding management strategies. As suggested in the ISA standard (ISA, 2009), potential dynamic alarm management could be predictive alarming, online alarm attribute modification, and online alarm suppression.

Causality analysis can also be applied once the patterns are obtained to help recover the connections between the corresponding tags in the pattern sequences and find the root cause. In (Bauer and Thornhill, 2008) a time delay estimation method was proposed to analyze causality by taking the time delay between variables as an evidence of causality. In (Duan et al., 2012), a direct transfer entropy was proposed to not only quantify the causality between process tags but also tell if the causality is direct or indirect. A transfer zero-entropy based causality analysis method was proposed in (Duan et al., 2014a) on the basis of 0-entropy and 0-information, removing the assumption of data stationarity. In (Duan et al., 2014b) methods for root cause diagnosis of plant-wide oscillations were summarized and compared, of which data-driven causality analysis as an important branch was reviewed. Alarm correlation analysis methods were also proposed in (Kondaveeti et al., 2010; Nishiguchi and Takai, 2010; Yang et al., 2012, 2013); these methods could help restore the connections between alarm tags directly without using process data.

### 1.1. Current status of alarm flood pattern analysis

Patterns can be obtained either manually or automatically. By far, expert consultation is still the way on which industrial companies rely most when haunted by alarm floods. Process knowledge such as process and instrument diagrams is usually involved in the analysis. This approach can bring the most accurate results; however the procedure is of low efficiency and high cost. Sometimes operators' experience can be used to identify patterns as well; but this becomes almost impossible for industrial complex facilities involving hundreds and thousands of alarms in floods.

The difficulties encountered in manual analysis of alarm floods could be overcome by resorting to data mining techniques. In (Kordic et al., 2008), a context based segmentation was carried out and alarm messages during the segmented periods were filtered to be the patterns. The method requires to pinpoint a target tag beforehand to set the starting point of segmentation. The authors in (Ahmed et al., 2013) used first-order Markov chains to capture alarm floods and then clustered them based on the Euclidian distance between probability matrices; pairwise alignment was then carried out based on Dynamic Time Warping (DTW) inside clusters. In (Folmer and Vogel-Heuser, 2012), a pattern growth method was applied to find patterns in alarm floods; however, as mentioned by the authors, the proposed method was sensitive to disturbances so that the pattern in sequences had to be exactly the same in order to be recognized. An approach based on Generalized Sequential Patterns (GSP) was proposed in (Cisar et al., 2010) to help discover frequent sequences in alarm floods; the algorithm was robust to disturbances and allowed some extent of order changes between alarm messages if they were raised closely. The authors in (Cheng et al., 2013) modified the Smith-Waterman algorithm to align a pair of alarm floods and obtain their similarity scores; it tolerates to

disturbances and order changes as well; but is limited to pairwise usage. The algorithm proposed in this paper is an extension of (Cheng et al., 2013) to the multiple sequence case. Significant modifications in similarity scoring, dynamic programming, back tracking procedure, and alignment generation have been made to achieve the extension.

### 1.2. Background of sequence pattern analysis

Searching for patterns in sequence database has been an interesting topic for many areas, for example, frequent transaction mining in business, homology detection in biology, text matching in information, and fault detection in process control. However the approaches are usually different since the types of input data and requirements on patterns are different in different areas.

In the business area, retailers want to find the items that are usually purchased consecutively so that they can rearrange the locations of these items in their stores to facilitate sales. Approaches include Apriori-like algorithms such as (Agrawal and Srikant, 1995), which grew short frequent sequences to longer ones through multiple scans of the whole database, tree based algorithms like the FP-growth (Han et al., 2000), which did not rely on candidate generation in Apriori-like algorithms, and (Zaki, 1998) that used a vertical data format to generate patterns. Some techniques such as the use of hash tables (Shintani and Kitsuregawa, 1996) and projections (Han et al., 2000; Pei et al., 2001) were developed as well to speed up the algorithms.

In the biology area, many multiple sequence alignment algorithms were modified from the pairwise alignment methods such as (Needleman and Wunsch, 1970; Smith and Waterman, 1981; Altschul et al., 1990; Pearson and Lipman, 1988). There were basically two types of modifications: the simultaneous approach, as shown in (Johnson and Doolittle, 1986), that gave exact optimal solutions, and the progressive pairwise approach, such as (Feng and Doolittle, 1987), that gave approximate results. One popular algorithm, CLUSTAL W (Thompson et al., 1994), which was usually used for homology detection in gene sequences, was based on the second type of modification. Recently developed have been also some statistic approaches based on profile Hidden Markov Models (HMM) such as (Eddy et al., 1995). However, a suitable HMM architecture (the number of states, and how they are connected by state transitions) must usually be designed manually, as pointed out in (Eddy, 1998).

### 1.3. Contribution of our work

In this paper we propose an algorithm that extends the one in (Cheng et al., 2013) to aligning multiple alarm flood sequences case by introducing: (1) new scoring functions that are capable of describing the similarity of items in multiple time-stamped sequences, (2) a dynamic programming equation for the iterative calculation of similarity indexes of multiple alarm flood sequences, and (3) back tracking and alignment generation procedures that can be used for multiple alarm flood sequences. With this proposed algorithm, we are able to find the optimal alignment of multiple alarm flood sequences and obtain the pattern for a selected alarm flood cluster, thus making up one of the missing steps in alarm flood analysis.

**Table 1 – An alarm message log example.**

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

#### 1.4. Organization of the paper

The rest of the paper is organized as follows. We first introduce some background of alarm flood analysis in [Section 2](#) to make it clear about the problem we are dealing with. Then in [Section 3](#), we show the principal steps of the algorithm using the case of three alarm flood sequences; pseudo code of the algorithm for the three sequence case will be shown in [Section 3](#) as well. Following that, some discussions concerning the algorithm and pattern selection are provided in [Section 4](#). In [Section 5](#), algorithms for aligning three and five alarm flood sequences will be tested on datasets of an actual petrochemical plant. Finally, conclusions are given in [Section 6](#).

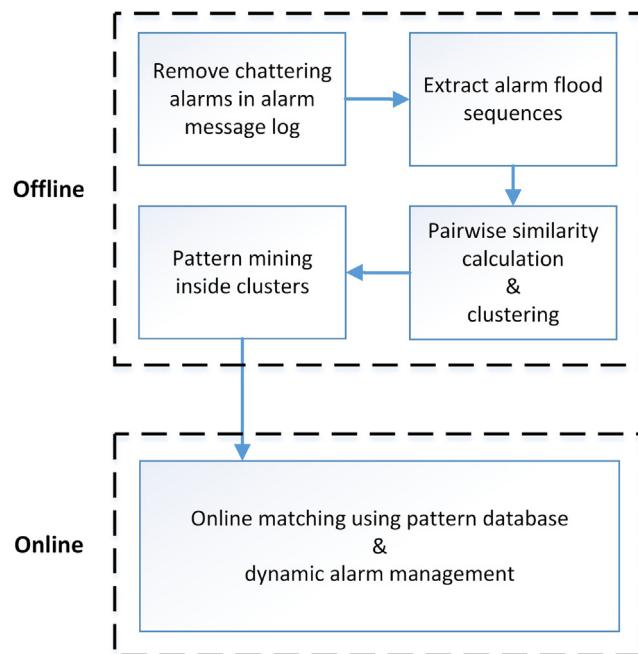
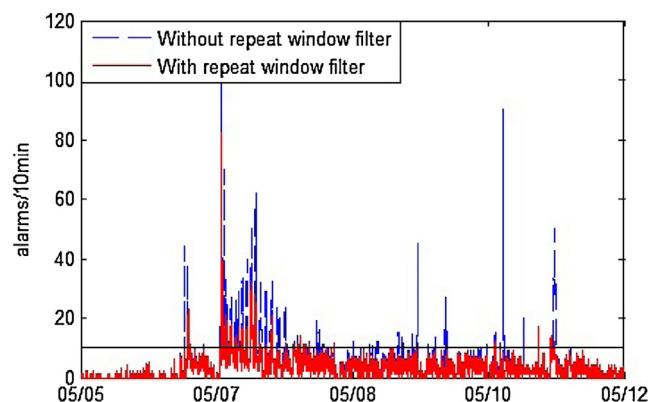
## 2. Background of alarm flood analysis

### 2.1. Alarm message log

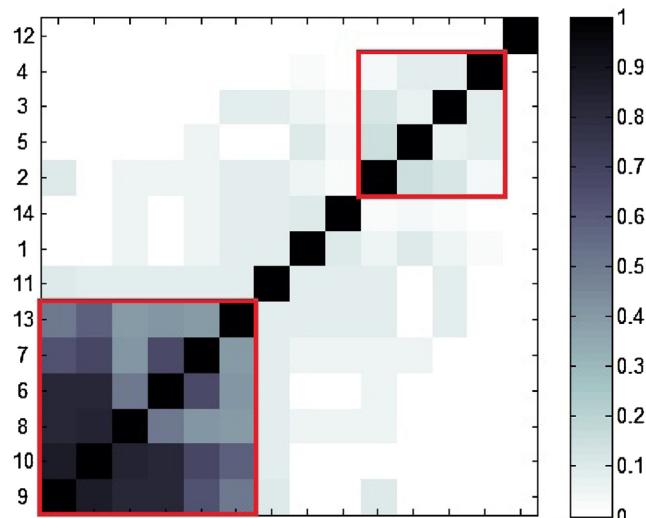
In a DCS (Distributed Control System) engaged plant, all monitored process variables are collected onto the DCS server, in which alarms are displayed and stored based on pre-set configurations. Usually an alarm message consists of at least the tag name, tag identifier, time stamp, and priority information. The tag name is unique code of a process variable that is being monitored. The tag identifier is the alarm type related to the monitored process (e.g., PVLO when the variable is under its low limit). We usually connect the tag name and a tag identifier of an alarm together with a dot in between while processing alarm messages, e.g., Tag1.PVLO. [Table 1](#) shows one example of alarm message log. Since priority information is not considered in this study, such information has been eliminated.

### 2.2. Alarm flood analysis procedures

[Fig. 1](#) shows the procedures of alarm flood analysis. One of the purposes of the offline part is to set up a pattern database for the use of online matching. The discovered patterns are also useful for some other potential applications described in [Section 1](#). During the offline part, data preprocessing is carried out first to remove chattering alarms before the extraction of alarm floods. Usually an off-delay timer is used since it will not introduce detection delays when alarms are raised. After preprocessing, an alarm rate threshold of 10 alarms per 10 min, as suggested by EEMUA and ISA standards ([EEMUA, 2013; ISA, 2009](#)), is used to extract alarm floods. The periods during which the alarm rate is higher than the threshold are extracted. The preprocessed alarm message log during each extracted alarm flood period is then numbered and saved for further pattern analysis. [Fig. 2](#) shows an example of data preprocessing and alarm flood extraction, in which an off-delay timer of 40 s has been applied to remove chattering alarms and a threshold of 10 alarms per 10 min has been set for alarm flood extraction. Then the algorithm in ([Cheng et al., 2013](#)) is applied to align each pair of the 14 extracted alarm floods and obtain their

**Fig. 1 – Flowchart of alarm flood analysis.****Fig. 2 – Example of alarm flood extraction.**

similarity scores. Based on those scores, clustering can be carried out to group the closely related flood sequences together, as shown in [Fig. 3](#). In the next step, the algorithm proposed in this paper can be applied to find the common

**Fig. 3 – Example of clustering 14 alarm floods based on pairwise similarity scores.**

pattern sequence for each of the clusters and save the useful patterns into a database for the use of online analysis.

### 3. Principle of the new algorithm

Our intended problem is to find the optimal alignment for a cluster of alarm floods so that based on this alignment an alarm sequence pattern can be easily found. However, the algorithm proposed in (Chen et al., 2013) is limited to pairwise alignment of flood sequences. In the following part, we will introduce an algorithm that extends the algorithm in (Chen et al., 2013) to the alignment of three alarm flood sequences. The idea for aligning more sequences will be similar, but more complex. Three new elements concerning similarity scoring functions, calculation of a similarity index cuboid and a back tracking procedure will be introduced.

#### 3.1. Problem description

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

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

where  $e_{1m}, e_{2n}, e_{3o} \in \Sigma$ , and  $\Sigma = \{1, 2, \dots, K\}$  is the set of different alarm types in the three sequences;  $t_{1m}, t_{2n}$  and  $t_{3o}$  are corresponding time stamps. The aim of the algorithm is to find the optimal local alignment (with deletions and inserted gaps '[]') of the three sequences, for example,

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

#### 3.2. Time distance and weight vectors

The original Smith-Waterman algorithm can help find optimal alignment between two sequences without time stamps. In order to adjust it to our problem, where time stamps are included in the sequences, a "time distance vector" and a "time weight vector" are defined, same as in (Chen et al., 2013). A time distance vector for an alarm message  $(e_m, t_m)$  is defined as:

$$\begin{aligned} \mathbf{d}_m &= [d_m^1, d_m^2, \dots, d_m^k, \dots, d_m^K], \\ d_m^k &= \begin{cases} \min_{1 \leq i \leq M} \{|t_m - t_i| : e_i = k\}, & \text{if the set is not empty} \\ \infty, & \text{otherwise.} \end{cases} \end{aligned} \quad (1)$$

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

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

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

$$f_1(x) = e^{-x^2/2\sigma^2}, \quad (3)$$

$$f_2(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{if } x \neq 0, \end{cases} \quad (4)$$

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

$$\langle (2, 2), (1, 3), (3, 3.5), (1, 5), (4, 5.2) \rangle$$

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

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

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

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

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

#### 3.3. Calculation of similarity scores

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

The 2-way similarity scoring function is

$$S((e_a, t_a), (e_b, t_b)) = \max\{S_0((e_a, t_a), (e_b, t_b)), S_0((e_b, t_b), (e_a, t_a))\} \times (1 - \mu) + \mu, \quad (5)$$

where,

$$S_0((e_a, t_a), (e_b, t_b)) = \max_{1 \leq k \leq K} [w_a^k \times w_b^k]. \quad (6)$$

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

The 3-way similarity scoring function is

$$S((e_a, t_a), (e_b, t_b), (e_c, t_c)) = S_0((e_a, t_a), (e_b, t_b), (e_c, t_c)) \times (1 - 2\mu) + 2\mu, \quad (7)$$

where

$$S_0((e_a, t_a), (e_b, t_b), (e_c, t_c)) = \max \left\{ \frac{S_0((e_b, t_b), (e_a, t_a)) + S_0((e_c, t_c), (e_a, t_a))}{2}, \frac{S_0((e_a, t_a), (e_b, t_b)) + S_0((e_c, t_c), (e_b, t_b))}{2}, \frac{S_0((e_a, t_a), (e_c, t_c)) + S_0((e_b, t_b), (e_c, t_c))}{2} \right\}. \quad (8)$$

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

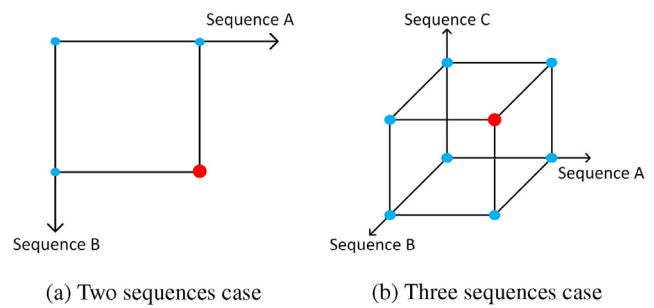
### 3.4. Dynamic programming

In the case of two alarm flood sequences, shown in Fig. 4(a), the score (red dot) in each cell of the similarity index matrix is obtained from three candidates (small blue dots). As each sequence holds one dimension in the alignment space, there is a similarity index cuboid instead of a matrix if the number of sequences to be aligned grows from two to three. Thus for the case of three alarm flood sequences, as shown in Fig. 4(b), in each step of dynamic programming, the score (big red dot) is obtained from seven candidates (small blue dots). Eq. (9) gives the way to calculate similarity index during each step:

$$\begin{aligned} H_{m+1,n+1,o+1} &= \max_{1 \leq i \leq M, 1 \leq j \leq N, 1 \leq g \leq O} I(A_{i:m}, B_{j:n}, C_{g:o}), 0 \\ &= \max \{ H_{m+1,n+1,o} + 2\delta, H_{m+1,n,o+1} + 2\delta, H_{m,n+1,o+1} + 2\delta, \\ &\quad H_{m,n,o+1} + \delta + S((e_{m+1}, t_{m+1}), (e_{n+1}, t_{n+1})), \\ &\quad H_{m,n+1,o} + \delta + S((e_{m+1}, t_{m+1}), (e_{o+1}, t_{o+1})), \\ &\quad H_{m+1,n,o} + \delta + S((e_{n+1}, t_{n+1}), (e_{o+1}, t_{o+1})), \\ &\quad H_{m,n,o} + S((e_{m+1}, t_{m+1}), (e_{n+1}, t_{n+1}), (e_{o+1}, t_{o+1})), \\ &\quad 0 \}, \end{aligned} \quad (9)$$

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

Back tracking is carried out on the three dimensional space as well, as shown in Fig. 5. First, the position of the largest similarity index is found (noted by big red dot in Fig. 5). Then



**Fig. 4 – Illustration of similarity score calculation for two and three sequences cases.**

it follows the arrow which points to the ancestor from whom the current score is obtained, until the path reaches the place with a zero similarity index.

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

### 3.5. An illustration example

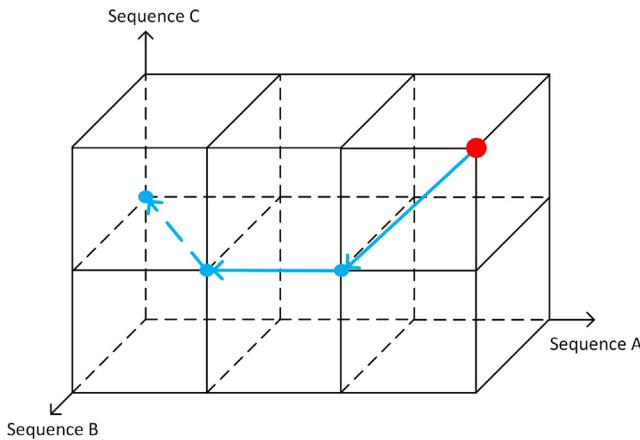
Here we provide a simple example of aligning three alarm flood sequences to demonstrate the calculation of similarity scores and the dynamic programming procedures. Pseudo codes are presented in Algorithm 1. For cases with more than three sequences, the idea will be the same. Note that the back tracking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

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

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

with  $\Sigma = \{1, 2, 3, 4, 5\}$ . We first show the calculation of 2-way and 3-way similarity scores for the ternary made up of the first messages in the three sequences, namely,  $(3, 1)$ ,  $(3, 2)$  and  $(3, 4)$ . Time weight vectors for  $(3, 1)$  in sequence A are  $[0, 0.84, 1, 0, 0]_{f_1}$  and  $[0, 0, 1, 0, 0]_{f_2}$ ; similarly, those for  $(3, 2)$  in sequence B are  $[0, 0, 1, 0.05, 0.14]_{f_1}$  and  $[0, 0, 1, 0, 0]_{f_2}$ , and those for  $(3, 4)$  in sequence C are  $[0, 0, 1, 0, 0]_{f_1}$  and  $[0, 0, 1, 0, 0]_{f_2}$ .  $S_0((3, 1), (3, 2)) = \max \{0 \times 0, 0.84 \times 0, 1 \times 1, 0 \times 0, 0 \times 0\} = 1$  and  $S_0((3, 2), (3, 1)) = \max \{0 \times 0, 0 \times 0, 1 \times 1, 0.05 \times 0, 0.14 \times 0\} = 1$ , thus  $S((3, 1), (3, 2)) = \max \{1, 1\} \times (1 - \mu) + \mu = 1$ . Similarly we can obtain  $S((3, 1), (3, 4)) = 1$  and  $S((3, 2), (3, 4)) = 1$ . With all the 2-way scores, the value of the 3-way score  $S((3, 1), (3, 2), (3, 4))$  can be calculated to be 1. We can also manually check this result. Since the alarm types of the three messages are the same, they should get a matching score.

After similarity scores are obtained, dynamic programming can be carried out based on Eq. (9). Two slices of the obtained



**Fig. 5 – An example of back tracking of a case with three alarm flood sequences.**

similarity cuboid is shown in Fig. 6. Since the values in the slice  $H_{0:6,0:4,0}$  are all 0, they are not shown in the figure in order to save space.

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

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

(3,4)	$\Delta$	(3,2)	(5,3)	(4,3.2)	(1,6)
$\Delta$	0	0	0	0	0
(3,1)	0	1.00	0.60	0.60	0.60
(2,1.3)	0	0.75	0.27	0.27	0.27
(3,3.5)	0	<u>1.00</u>	0.60	0.60	0.60
(4,5)	0	0.60	<u>1.56</u>	1.20	0.40
(5,5.1)	0	0.60	1.20	<u>2.12</u>	1.32
(1,10)	0	0.60	0.40	1.32	2.72

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

(1,9)	$\Delta$	(3,2)	(5,3)	(4,3.2)	(1,6)
$\Delta$	0	0	0	0	0
(3,1)	0	0.60	0	0	1.20
(2,1.3)	0	0.27	0	0	0.87
(3,3.5)	0	0.60	0	0	1.20
(4,5)	0	0	1.16	0.60	1.80
(5,5.1)	0	0	0.60	1.72	2.72
(1,10)	0	1.20	1.80	2.72	<u>3.12</u>

**Fig. 6 – Similarity index cuboid and back tracking of the example.**

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

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

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

#### Algorithm 1: Aligning three alarm flood sequences

**input** : 3 alarm flood sequences, variance of Gaussian function  $\sigma$ , gap penalty  $\delta$  and miss-match penalty  $\mu$

**output**: Optimal local alignment of three alarm flood sequences

```

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

```

#### 4. Industrial case study

A dataset from an actual chemical process has been used to test the effectiveness of the proposed algorithm. Equipments used in the process include pumps, compressors, furnaces, and filters. 300 seconds' off-delay timers were applied to remove chattering alarms. Such a long time was chosen for off-delay-timers in order to prevent obtaining the patterns formed by repeating alarms. 359 alarm flood sequences were

**Table 2 – General descriptions of the dataset**

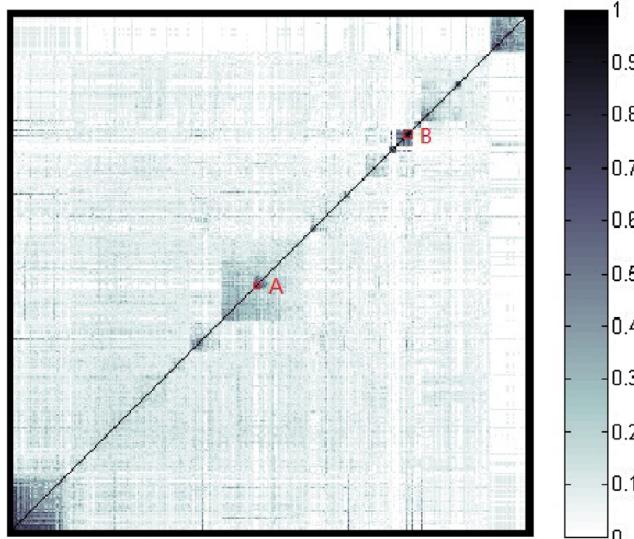
Description	Number
Total time period	336 days
Total number of tags	1502
Total number of alarms	109,393
Average alarm rate	14/h
Highest peak alarm rate	334/10 min
Number of alarm floods	359
Average length of alarm floods	39

extracted based on the ISA standard, which is 10 alarms per 10 min. General descriptions of the dataset with off-delay timers applied and the extracted alarm floods can be found in [Table 2](#). The pairwise pattern matching algorithm proposed in ([Cheng et al., 2013](#)) was applied to calculate pairwise similarity scores of the extracted alarm floods. Clustering based on the obtained similarity scores was carried out thereafter; result is shown in [Fig. 7](#). Both horizontal and vertical axes of the chessboard in the figure are formed by the indices of alarm floods. Each dot in the figure represents the similarity score between a corresponding pair of alarm flood sequences; the darker the color of a dot is, the higher the similarity score is. Based on the clustering result, pattern mining could be carried out manually by comparing the corresponding sequences; but it would be time consuming and the accuracy would not be guaranteed. In the following part, we will carry out the proposed algorithm respectively on two clusters of three and five alarm flood sequences.

#### 4.1. Pattern mining in three alarm flood sequences

Three alarm floods, of lengths 13, 16, and 12 respectively, in cluster A, shown in [Fig. 7](#), were selected to be the testing sequences. The proposed algorithm for pattern mining in three sequences was applied with parameters set as:  $\sigma = 0.2$ ,  $\mu = -1$ , and  $\delta = -1$ . On a 64 bit Windows PC with Intel(R) Core(TM) i7-4770 3.40GHz CPU and 24.0 GB memory, the algorithm took only 0.9 s to finish and the result is shown in [Table 3](#).

Even though the cluster is formed by short alarm floods, manually comparing the three sequences would still be time consuming. However, using the proposed algorithm, the alignments can be obtained accurately and almost immediately.

**Fig. 7 – Clustering result of extracted alarm floods.**

Moreover, by taking a closer look at the time stamps of the alignments, one can notice that these three alarm floods were raised during October, December, and February, respectively, which means the pattern found from those alignments could be a regular one. Moreover, priorities (not listed here in order to save space) of many alarms in the three sequences had been configured as "High". Thus, the pattern obtained from the alignments could be valuable for predictive alarming and operator training.

#### 4.2. Pattern mining in five alarm flood sequences

Setting the parameters to be:  $\sigma = 0.2$ ,  $\mu = -1$ , and  $\delta = -1$ , the proposed algorithm for pattern mining in five sequences was applied on cluster B, which was formed by five alarm flood sequences. Lengths of the five sequences are 20, 49, 122, 25, and 56 respectively. On a 64 bit Windows PC with Intel(R) Core(TM) i7-4770 3.40GHz CPU and 24.0 GB memory, the algorithm took 982 seconds to complete and the result is shown in [Table 4](#).

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

## 5. Discussion

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

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

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

**Table 3 – Alignment result of the three sequences in cluster A**

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

Moreover, the three parameters used in the algorithm, the variance of the scaled Gaussian function, the gap and mismatch penalties need to be tuned. As mentioned in (Cheng et al., 2013), users need to adjust those parameters based on their requirements. The variance of the scaled Gaussian function will influence the time span within which the algorithm treats the orders of the alarms to be less important. The bigger this value is, the broader the time span is. For example, if this value would go to infinity then the algorithm would treat all the alarms to be raised simultaneously and ignore the time stamp information. On the contrary, if it was set to be zero, the algorithm would require the orders of alarm messages to be exactly the same for each aligned sequence. Gap and miss-match penalties determine the tolerance of the algorithm to gaps and mismatch terms. Usually the alignment would contain more alarm messages if we increase these two parameters; but we would expect more gaps and miss-match terms in the alignment. The flexibilities make the algorithm more adjustable to meet different requirements, but at the same time make it hard for users to tune the algorithm.

One advantage of the proposed algorithm is that it can provide exact optimal solutions through dynamic

programming, since there is not any approximate steps, which have been adopted in progressively pairwise approaches in the bioinformatics area, as described in Section 1. This advantage is quite valuable for our problem because alarms play a critical role in industrial process monitoring and we want to keep this information as accurate as possible. Moreover, since this part is done offline, accuracy should be given the first priority.

We have not compared the proposed algorithm with others since till now there are no other algorithm solving the same problem. Other methods either do not consider time stamps, such as CLUSTAL W (Thompson et al., 1994), or are not applicable to multiple sequences, like the algorithm in (Cheng et al., 2013).

Regarding to the pattern selection, our way is somewhat different from the one used in the bioinformatics area, where the consensus part of the alignment is usually selected as the pattern. However in our case, we tolerate some extent of order changes in alignment: some valuable patterns could be left out if we only select the consensus part. For this reason, we usually choose the alignment sequence that contains the least number of gaps as the pattern for the whole cluster. Experts may be involved in this part to help the pattern selection. Another

**Table 4 – Alignment result of the five sequences in cluster B.**

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

difference is that for our case, time stamps should be included in the selected patterns as well, since they carry the information that describes relative positions of alarm messages in the time domain.

## 6. Conclusions

In this paper we proposed an algorithm to find the optimal alignment of multiple alarm flood sequences, and obtained a common pattern of them thereafter. The algorithm is an extension of the algorithm in (Cheng et al., 2013) to the case of multiple sequences. It makes up one of the missing steps in our current alarm flood analysis. A lot of analysis and management such as root cause analysis, dynamic alarm suppression, and analysis on potential problems in alarm systems can be carried out based on the results of this proposed algorithm.

In the future, pruning techniques to reduce the algorithm complexity, methods to facilitate coding for cases with higher number of sequences, analysis on parameter tuning, and better pattern selection methods may be studied. We will also focus on developing algorithms to match online alarm messages with a given pattern database to provide early warnings to operators for incoming alarm floods.

## Acknowledgement

The authors benefited from discussions with John MacGowan and Aris Espejo of Syncrude Canada Limited.

## References

- Agrawal, R., Srikant, R., 1995. Mining sequential patterns. In: *Proceedings of the Eleventh International Conference on Data Engineering*. IEEE, pp. 3–14.
- Ahmed, K., Izadi, I., Chen, T., Joe, D., Burton, T., 2013. Similarity analysis of industrial alarm flood data. *IEEE Trans. Autom. Sci. Eng.* 10 (2), 452–457.
- Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J., 1990. Basic local alignment search tool. *J. Mol. Biol.* 215 (3), 403–410.
- Bauer, M., Thornhill, N.F., 2008. A practical method for identifying the propagation path of plant-wide disturbances. *J. Process Control* 18 (7), 707–719.
- Cheng, Y., Izadi, I., Chen, T., 2013. Pattern matching of alarm flood sequences by a modified smith-waterman algorithm. *Chem. Eng. Res. Des.* 91 (6), 1085–1094.
- Cisar, P., 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.
- Dal Vernon, C.R., Downs, J.L., Bayn, D., 2004. Human performance models for response to alarm notifications in the process industries: an industrial case study. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 48, no. 10. SAGE Publications, pp. 1189–1193.
- Duan, P., Yang, F., Chen, T., Shah, S.L., 2012. Detection of direct causality based on process data. In: *American Control Conference (ACC)*. IEEE, pp. 3522–3527.
- Duan, P., Yang, F., Shah, S.L., Chen, T., 2014a. Transfer zero-entropy and its application for capturing cause and effect relationship between variables. *IEEE Trans. Control Syst. Technol.* 23 (3), 1, <http://dx.doi.org/10.1109/TCST.2014.2345095>.
- Duan, P., Chen, T., Shah, S.L., Yang, F., 2014b. Methods for root cause diagnosis of plant-wide oscillations. *AIChE J.* 60 (6), 2019–2034.
- Eddy, S.R., et al., 1995. Multiple alignment using hidden markov models. In: *ISMB*, pp. 114–120.
- Eddy, S.R., 1998. Profile hidden markov models. *Bioinformatics* 14 (9), 755–763.
- EEMUA, 2013. *Alarm Systems – A Guide to Design, Management and Procurement*.
- Feng, D.-F., Doolittle, R.F., 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.* 25 (4), 351–360.
- Folmer, J., Vogel-Heuser, B., 2012. Computing dependent industrial alarms for alarm flood reduction. In: *9th International Multi-Conference on Systems, Signals and Devices (SSD)*. IEEE, pp. 1–6.
- Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.-C., 2000. Freespan: frequent pattern-projected sequential pattern mining. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, pp. 355–359.
- Han, J., Pei, J., Yin, Y., 2000. Mining frequent patterns without candidate generation. In: *ACM SIGMOD Record*, vol. 29, no. 2. ACM, pp. 1–12.
- ISA, 2009. *Management of Alarm Systems for the Process Industries*.
- Izadi, I., Shah, S.L., Shook, D.S., Kondaveeti, S.R., Chen, T., 2009. A framework for optimal design of alarm systems. In: *in Proceedings of 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pp. 651–656.
- Johnson, M.S., Doolittle, R.F., 1986. A method for the simultaneous alignment of three or more amino acid sequences. *J. Mol. Evol.* 23 (3), 267–278.
- Kondaveeti, S.R., Izadi, I., Shah, S.L., Black, T., 2010. Graphical representation of industrial alarm data. *Anal. Des. Eval. Human-Machine Syst.* 11 (1), 181–186.
- Kondaveeti, S.R., Izadi, I., Shah, S.L., Shook, D.S., Kadali, R., Chen, T., 2013. Quantification of alarm chatter based on run length distributions. *Chem. Eng. Res. Des.* 91 (12), 2550–2558.
- Kordic, S., Lam, P., Xiao, J., Li, H., 2008. *Analysis of Alarm Sequences in a Chemical Plant*. Springer.
- Needleman, S.B., Wunsch, C.D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48 (3), 443–453.
- Nishiguchi, J., Takai, T., 2010. Ipl2 and 3 performance improvement method for process safety using event correlation analysis. *Comp. Chem. Eng.* 34 (12), 2007–2013.
- Pearson, W.R., Lipman, D.J., 1988. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. U.S.A.* 85 (8), 2444–2448.
- Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.-C., 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, p. 0215, <http://dx.doi.org/10.1109/ICDE.2001.914830>.
- Shintani, T., Kitsuregawa, M., 1996. Hash based parallel algorithms for mining association rules. In: *Fourth International Conference on Parallel and Distributed Information Systems*. IEEE, pp. 19–30.
- Smith, T.F., Waterman, M.S., 1981. Identification of common molecular subsequences. *J. Mol. Biol.* 147 (1), 195–197.
- Thompson, J.D., Higgins, D.G., Gibson, T.J., 1994. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 22 (22), 4673–4680.
- Yang, F., Shah, S.L., Xiao, D., Chen, T., 2012. Improved correlation analysis and visualization of industrial alarm data. *ISA transactions* 51 (4), 499–506.
- Yang, Z., Wang, J., Chen, T., 2013. Detection of correlated alarms based on similarity coefficients of binary data. *IEEE Trans. Autom. Sci. Eng.* 10 (4), 1014–1025.
- Zaki, M.J., 1998. Efficient enumeration of frequent sequences. In: *Proceedings of the Seventh International Conference on Information and Knowledge Management*. ACM, pp. 68–75.