Preprints of the
9th International Symposium on Advanced Control of Chemical Processes
The International Federation of Automatic Control
June 7-10, 2015, Whistler, British Columbia, Canada

Fast Sequence Alignment for Comparing Industrial Alarm Floods ⋆

Wenkai Hu * **Jiandong Wang** ** **Tongwen Chen** *

* *Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, T6G 2V4*
e-mail: {wenkai, tchen}@ualberta.ca
** *College of Engineering, Peking University, Beijing, China*
e-mail: jiandong@pku.edu.cn

Abstract: An alarm flood is a commonly encountered problem in industrial alarm systems. It may distract operators from critical alarms and lead to plant failures. Thereby, addressing of alarm floods becomes an important task to achieve reliable alarm management. Some sequence alignment algorithms have been used to discover useful patterns of alarm floods from historical alarm data. In order to improve the computational efficiency of the existing algorithms, a fast sequence alignment approach is developed based on a basic local alignment search tool. It accelerates the computation using a seed-and-extend strategy. The priority information is incorporated so that the algorithm is more sensitive to alarms of higher priorities. Besides, the set based pre-matching is proposed to avoid unnecessary computation by excluding irrelevant alarm floods and alarm tags. Industrial case studies demonstrate that the proposed method significantly outperforms the existing algorithms.

Keywords: Alarm systems, alarm sequences, alarm floods.

1. INTRODUCTION

In modern automated industrial plants, alarm systems are designed to assist operators to perceive hazardous situations and maintain industrial processes within safe operating boundaries (EEMUA-191, 2007). A good alarm system must detect and warn the operators of abnormalities while not mislead, overload or distract the operators (Rothenberg, 2009). However, a commonly encountered poor function of industrial alarm systems is an alarm flood, which is defined as a situation that the annunciated alarms arise massively and exceed the operator's response capability. As a result, ANSI/ISA-18.2 (2009) and EEMUA-191 (2007) recommends 10 alarms per 10 min as a rule of thumb to identify alarm floods. The presence of alarm floods may distract operators from critical alarms and lead to serious consequences, thereby making a difficult process situation much worse.

Alarm floods could be caused by many reasons, e.g., process upsets, abnormal situations, improper alarming, and transition of operating states (Hollifield et al., 2011). In practice, alarm floods caused by changes of operating states could be detected and avoided using some advanced alarming techniques, e.g., the logic based alarming, and the state based alarming (ANSI/ISA-18.2, 2009). But those alarm floods caused by faults are difficult to manage. The objectives in studying alarm floods may include handling alarm floods effectively or even avoiding the occurrence of alarm floods. Thus, it may be necessary to find alarm floods with similar patterns in the historical alarm data so that some useful information could be discovered. For this purpose, Ahmed et al. (2013) exploited the dynamic time warping (DTW) algorithm to find the common subsequences among alarm floods. Cheng et al. (2013) modified the Smith-Waterman algorithm for the pattern matching of alarm floods and considered the time stamps in the computation of alarm flood similarities. The common drawback of the two algorithms is the high computational complexity. That is, if there are too many alarm floods or the alarm floods too lengthy, it is almost impossible to finish their computations within a tolerable time period.

This paper contributes in the following aspects. First, a fast sequence alignment algorithm is proposed for alarm floods based on a basic alignment search tool; it accelerates the computation using a seed-and-extend strategy. Second, the priority information is incorporated so that the algorithm is more sensitive to alarms of higher priorities. Third, the set based pre-matching is proposed to avoid unnecessary computation by excluding irrelevant alarm floods and alarm tags.

The rest of the paper is organized as follows. Section 2 discusses the refinement and the representation of alarm floods. Section 3 presents the mechanism of the proposed fast sequence alignment algorithm. The set based pre-matching approach is described in Section 4. Numerical examples and industrial cases are presented in Section 5 and 6, respectively. The final section concludes the paper.

⋆ This work was partially supported by an NSERC CRD project, and the National Natural Science Foundation of China under grant No. 61061130559.

2. REFINEMENT OF ALARM FLOODS

To refine alarm floods from alarm data, an exact definition involving the start and the end of an alarm flood is given by ANSI/ISA-18.2 (2009). A typical alarm flood begins at the first regular 10 min interval with the alarm rate exceeding 10 alarms per 10 min and ends when the alarm rate falls under 5 alarms per 10 min. Hence, to capture alarm floods, we need a 10 min sliding window to calculate the number of alarms, which is known as the burst rate in Hollifield et al. (2011). The burst alarm rate plot gives a good view to observe the alarm floods. An example is shown in Fig. 1, where 19 alarm floods were extracted from May 5th to May 12th, 2013 as highlighted by blue rectangles.

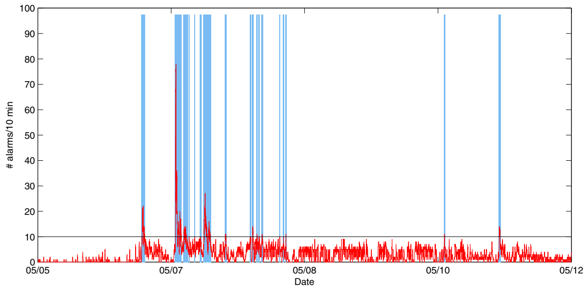

Fig. 1. Example of the burst alarm rate plot.

Essentially, an alarm flood is a sequence comprised of a series of chronologically sorted alarms. It can be expressed as

$$A =< a_1, a_2, ..., a_m > \tag{1}$$

where $< \cdot >$ indicates a sequence, m is the number of alarms in the alarm flood A, a_i denotes the i-th alarm. For the simplest case, a_i is referred to as the alarm tag

$$a_i = e_i \tag{2}$$

where e_i indicates the numerical symbol of the i-th alarm. This form was exploited by Ahmed et al. (2013) in the similarity analysis of alarm floods. If incorporating the time stamp, a_i has two dimensions (Cheng et al., 2013). Incorporating the time and priority information, a_i takes the form of three dimensions as

$$a_i = (e_i, t_i, p_i) \tag{3}$$

where t_i represents the time stamp, p_i denotes the priority level. Table 1 shows an example of an alarm flood sequence with alarms chronologically sorted. Three priorities are allocated to all alarm tags, including Low, High and Emergency (simplified as Emg.). For the convenience of computation, the alarm flood can be expressed by a numerical sequence like $< 9, 5, 6, 2, 1, 3, 4, 3, 4, 7, 4, 5 >$ with each number indicating a unique alarm.

In the analysis of alarm floods, a troublesome issue is about chattering alarms. The presence of chattering alarms conveys repeating information, leading to the false identification of alarm floods. Thus a prerequisite for alarm flood analysis is to remove chattering alarms. Recently, the detection, quantification and removal of chattering alarms have been studied by Kondaveeti et al. (2013); Wang & Chen (2013, 2014). Among various strategies for addressing chattering alarms, delay timers can be directly applied to alarm data, which makes them more suitable for the case in this paper.

Table 1. Example of an alarm flood sequence.

Alarm Tag(e_i)	Time stamp(t_i)	Priority(p_i)
T09.PVLO	10/5/2013 02:20:01	Low
T05.PVHI	10/5/2013 02:20:55	Low
T06.PVLO	10/5/2013 02:22:02	Low
T02.PVHI	10/5/2013 02:22:42	High
T01.LOLO	10/5/2013 02:23:35	High
T03.PVHI	10/5/2013 02:24:22	Low
T04.PVLO	10/5/2013 02:25:11	Low
T03.PVHI	10/5/2013 02:27:02	Low
T04.PVLO	10/5/2013 02:27:03	Low
T07.PVLO	10/5/2013 02:27:03	Emg.
T04.PVLO	10/5/2013 02:29:05	Low
T05.LOLO	10/5/2013 02:30:05	Low

3. ACCELERATED SEQUENCE ALIGNMENT

In this section, the seed-and-extend strategy is firstly described. Then, some improvements are presented, including the fully matched seeds, the priority based scoring and the time ambiguity tolerance mechanism.

3.1 Seed-and-Extend Strategy

The basic local alignment search tool (BLAST) is a fast heuristic algorithm proposed by Altschul et al. (1990, 1997). The principle of BLAST can be simplified as the seed-and-extend approach. In the seeding stage, sequences to be compared are broken into short words of a fixed size. A lookup table is built for all possible assemblies. By comparing short words rather than the whole sequences, ungapped similar segments are quickly located. In the extending stage, the segments of high similarity are extended in two directions until the score falls more than a cutoff threshold below the best score achieved. The alignment with score above a certain threshold is called high scoring segment pair (HSP). The HSP of the highest score can be treated as the longest common subsequence for the pairwise sequence alignment. Compared with the Smith-Waterman algorithm, BLAST does not need to align all elements. Instead, it quickly locates similar regions and extends them in two directions so that most search space is pruned.

However, components in alarm floods are more diverse than the biological cases, making it difficult to build, save and look up a table for all short words. Hence, the traditional seeding approach in BLAST is not suitable for alarm floods. To solve the problem, a new seeding approach is developed. The principle is to find all successively matched pairs and keep those of high scores as seeds. A successively matched pair Z can be indexed by its positions in two sequences A, B as

$$Z = (x, y, l) \tag{4}$$

where x and y indicate the start positions in sequence A and B respectively, l represents the length of Z. A successively matched pair Z has two properties. First, all alarms indexed by Z in two sequences should be identical, i.e. $a_{x+i} = b_{y+i}$ for $i = 0, 1, ..., l-1$. Second, all alarms indexed by Z have successive positions in both sequences. To select seeds from the successively matched pairs, similarity scores should be calculated. The score of Z is formulated as

$$z = \sum_{i=0}^{l-1} s(a_{x+i}, b_{y+i}) \tag{5}$$

where $s(a, b)$ represents the similarity score between alarm a and alarm b. The scoring strategy will be discussed later in Sub-section 3.2. Matched pairs having top scores are kept as seeds while other pairs are discarded. Seeds and their corresponding scores are kept as $\tilde{Z}(k)$ and $H_s(\tilde{Z}(k))$ respectively, where $k = 1, ..., K$, and K is the number of seeds.

Once some seeds are found in the seeding stage, the extension approach will be triggered to find the maximum alignment. Considering there are many deletions or insertions in alarm floods, the gapped extension which is based on the gap penalty scheme is utilized (Altschul et al., 1997). Based on the k-th seed $\tilde{Z}(k)$, the gapped extension should be made in two directions. In each direction, a score matrix H is calculated based on the subsequences A_s of length m and B_s of length n to be compared. For all $i = 2, 3, ..., m+1$ and $j = 2, 3, ..., n+1$, $H_{i,j}$ is calculated as

$$H_{i,j} = \max\{H_{i-1,j-1} + s(a_i, b_j), H_{i,j-1} + \delta, H_{i-1,j} + \delta, 0\} \tag{6}$$

where δ indicates the gap penalty, $s(a_i, b_j)$ denotes the score between alarm a_i and alarm b_j, $a_i \in A_s$ and $b_j \in B_s$. For all $i = 2, 3, ..., m + 1$ and $j = 1$, $H_{i,1}$ is calculated as

$$H_{i,1} = \max\{H_{i-1,1} + \delta, 0\} \tag{7}$$

For all $i = 1$ and $j = 2, 3, ..., n + 1$, $H_{1,j}$ is calculated as

$$H_{1,j} = \max\{H_{1,j-1} + \delta, 0\} \tag{8}$$

In the gapped extension, a critical procedure is the stop of extension. In the iterative computation, the extension to $H_{i,j}$ stops at $H_{i,j}$ if

$$\begin{cases} H_{i-1,j-1} < H_{\max} - T & \text{for } i \geq 2 \text{ and } j \geq 2 \\ H_{i,j-1} < H_{\max} - T & \text{for } i \geq 1 \text{ and } j \geq 2 \\ H_{i-1,j} < H_{\max} - T & \text{for } i \geq 2 \text{ and } j \geq 1 \end{cases} \tag{9}$$

where $H_{\max}$ denotes the best score achieved before the extension proceeds to $H_{i,j}$, T represents the cutoff threshold. To avoid the case that $H_s(\tilde{Z}(k)) < T$, the initial score of H should be $H_{1,1} = H_s(\tilde{Z}(k)) + T$. The gapped extension is proceeded iteratively with the increasing of i and j. To present the alignment, the backtracking starts at the cell of the highest score and proceeds backward until $H_{1,1}$, yielding the best local alignment. The description of the backtracking can be found in Smith & Waterman (1981); Cheng et al. (2013). After the extension, the score for the final alignment based on seed $\tilde{Z}(k)$ is

$$S(\tilde{Z}(k)) = H_l + H_r - H_s(\tilde{Z}(k)) - 2T \tag{10}$$

where H_l and H_r denote the left extension and right extension respectively. Since several seeds could be found, the alignments achieved could be different. The one of the highest score will be treated as the best alignment. The corresponding score is the best score between sequence A and B and is represented by $S(A, B)$.

3.2 Priority Based Scoring

Both of the scores in (5) and (6) are based on $s(a_i, b_j)$. To achieve better alignment result, a priority based scoring scheme is proposed to assign $s(a_i, b_j)$. Priority levels are settings assigned to indicate the relative importance of alarms, e.g., the seriousness of consequences and the allowable response time (ANSI/ISA-18.2, 2009). For most cases, three or four priorities are available and often labeled by different names as shown in Table 2. Alarms are not identical in importance. Higher priorities are rarely configured and annunciated, but more important in indicating abnormalities. By contrast, lower priorities are associated with less severe consequences and assigned to most alarms.

Table 2. Alarm priorities in alarm systems.

Priority Level	List 1	List 2	List 3
Priority 1 (p_1)	Emergency	Emergency	Critical
Priority 2 (p_2)	High	High	Warning
Priority 3 (p_3)	Medium	Low	Advisory
Priority 4 (p_4)	Low		

In the sequence alignment of alarm floods, scaled similarity scores should be used, so that the algorithm will be more sensitive to alarms of higher priorities. For a descending priority list from Priority 1 to Priority L, the match scores $\{\phi(p_1), \phi(p_2), \cdots, \phi(p_L)\}$ are assigned. The combination of matched scores, mismatch score and gap penalty should satisfy the following inequalities

$$\begin{cases} \phi(p_i) > 0 \\ \phi(p_i) > \phi(p_{i+1}), i = 1, 2, ..., L-1 \\ \eta < 2\delta < 0 \end{cases} \tag{11}$$

where η indicates the mismatch score, L denotes the number of priority levels. A smaller index i indicates a higher priority level. The second inequality guarantees that the algorithm is more sensitive to alarms of higher priorities. The third inequality makes the extension to be a gapped strategy. With a given absolute mismatch score $|\eta|$, higher match scores $\phi(p_i)$ will lead to weaker but longer alignments, while higher match scores will find shorter alignment of high similarity. An example of scores for L priorities is recommended in Table 3.

Table 3. Example of scaled similarity scores.

Item	Symbol	Score
Match for p_i	$\phi(p_i)$	$3 + 1.5(L - i)$
Mismatch	η	-2.5
Gap penalty	δ	-1

3.3 Time Ambiguity Tolerance

Inspired by the fact that some strongly connected alarms arise almost simultaneously, time information was firstly exploited to alleviate the ambiguity of orders as indicated in (Cheng et al., 2013). The same principle is adopted so that the proposed algorithm tolerates the order of alarms occurring almost simultaneously. Firstly, the time intervals between the i-th alarm and all other alarms in alarm flood A can be formulated as

$$\mathbf{d}_i = [d_{i1}\ d_{i2}\ \cdots\ d_{im}]^T \tag{12}$$

where $d_{ik} = |t_i - t_k|, k = 1, \cdots, m$, t_i and t_k indicates the time stamps of the i-th and the k-th alarm, $d_{ik} = 0$ if $k = i$. The weight vector for the i-th position in A is formulated as

$$\mathbf{w}_i = [w_{i1}\ w_{i2}\ \cdots\ w_{im}]^T \tag{13}$$

where $w_{ik} = f(d_{ik})$, $f(\cdot)$ is a weighting function with respect to d_{ik}. To tolerate the order of alarms with short time intervals, $f(\cdot)$ should satisfy two conditions (Cheng et al., 2013). First, $f(\cdot)$ is monotonically decreasing on the positive axis. Second, $f(0) = 1$ and $f(\infty) = 0$. The scaled Gaussian function is a good choice, i.e.,

$$f(x) = e^{-\frac{x^2}{2\sigma^2}} \tag{14}$$

where σ is the standard deviation of the Gaussian function.

To weight the similarity score $s(a_i, b_j)$ for $a_i = (e_i^a, t_i^a, p_i^a)$ and $b_j = (e_j^b, t_j^b, p_j^b)$ in (6), the weighting function should only be applied to one sequence in case of one matched pair counted for more than once. If $f(\cdot)$ is applied to A, then all alarms identical to b_j are indicated by a binary vector as

$$\mathbf{v}_i = [v_{i1}\ v_{i2}\ \cdots\ v_{im}]^T \tag{15}$$

where $v_{ik} = 1$ if $e_k^a = e_j^b$, $v_{ik} = 0$ if $e_k^a \neq e_j^b$. Then the best score between a_i and b_j can be formulated as

$$s_1(a_i, b_j) = \max_{1\leq k\leq m}[w_{ik} \times v_{ik}](\phi(p_j^b) - \eta) + \eta \tag{16}$$

Since this is an asymmetric calculation, the best score could be different if $f(\cdot)$ is applied to B. The score between a_i and b_j becomes

$$s_2(a_i, b_j) = \max_{1\leq k\leq n}[w_{jk} \times v_{jk}](\phi(p_i^a) - \eta) + \eta \tag{17}$$

where w_{jk} indicates the weight for the k-th position in B, v_{jk} denotes whether $e_k^b = e_i^a$. Based on (16) and (17), the best score between a_i and b_j can be formulated with a symmetric computation as

$$s(a_i, b_j) = \max(s_1(a_i, b_j), s_2(a_i, b_j)) \tag{18}$$

It is easy to notice the following facts: if $e_i^a = e_j^b$, then $s(a_i, b_j) = \phi(p_i^a) = \phi(p_j^b)$; if $e_i^a \neq e_j^b$ and $e_k^a \neq e_j^b$ for all $k = 1, \cdots, m$, and $e_k^b \neq e_i^a$ for all $k = 1, \cdots, n$, then $s(a_i, b_j) = \eta$; if $e_i^a \neq e_j^b$ but we can find an alarm a_k closely occurring with a_i and $e_k^a = e_j^b$, then $s(a_i, b_j)$ will be close to $\phi(p_j^b)$; otherwise, $s(a_i, b_j)$ will be close to η.

4. SET BASED PRE-MATCHING

In alarm systems, a large number of alarms are configured to monitor different processes. Thus many alarm floods are different because faults may happen in different areas, raising different groups of alarms. Therefore, a set based pre-matching approach is necessary. To find common alarms for alarm floods A and B, a matrix is formulated as

$$R(A,B) = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix} \tag{19}$$

where

$$r_{ij} = \begin{cases} 1 & e_i^a = e_j^b \\ 0 & e_i^a \neq e_j^b \end{cases} \tag{20}$$

where e_i^a and e_j^b are the alarm tags of $a_i \in A$ and $b_j \in B$. All matched pairs are indicated by 1 in the matrix $R(A, B)$. If $R(A, B)$ is a null matrix, we say that there is no common alarms between alarm floods A and B, and thus we do not need to proceed to sequence alignment. Meanwhile, a set based similarity score is formulated by incorporating the priority information and the alarm occurrence times as

$$S_{set}(A,B) = \frac{\sum_{i=1}^{\tilde{m}}(\phi(p_i^{\tilde{a}}))\sum_{i=1}^{\tilde{n}}(\phi(p_i^{\tilde{b}}))}{\sum_{i=1}^{m}(\phi(p_i^{a}))\sum_{i=1}^{n}(\phi(p_i^{b}))} \tag{21}$$

where $\tilde{m}$ and $\tilde{n}$ are numbers of common alarms included in A and B, respectively; $\tilde{a}$ and $\tilde{b}$ indicate the common alarms in A and B. This similarity measure has the following properties: first, $0 \leq S_{set}(A, B) \leq 1$; second, $S_{set}(A, B) = S_{set}(B, A)$; third, $S_{set}(A, A) \geq S_{set}(A, B)$; fourth, it does not require order or time information. If A and B have no common alarms, $S_{set}(A, B) = 0$. Using the set based pre-matching approach, we save a lot of time in measuring those irrelevant alarm floods. Especially for the case to compare one alarm flood with a database, many alarm floods having no common alarms can be excluded.

Furthermore, for most cases, two alarm floods to be compared may contain many different alarm tags. It is meaningless and extravagant to align these irrelevant alarms. Such kind of superfluous alignment can be totally avoided by directly aligning these irrelevant alarms with gaps. Hence, a common alarm indexing approach is proposed here, i.e., two shorter sequences $\tilde{A}$ and $\tilde{B}$ can be indexed from A and B by recording the positions of identical alarms. The new sequences have the following properties:

(1) $e \in \tilde{A} \cap e \in \tilde{B}$ for $\forall e \in A \cap e \in B$;
(2) $e \notin \tilde{A} \cap e \notin \tilde{B}$ for $\forall e \notin A \cup e \notin B$;
(3) $< a_i, a_j > \subseteq A$ (or $< b_i, b_j > \subseteq B$) if $< a_i, a_j > \subseteq \tilde{A}$ (or $< b_i, b_j > \subseteq \tilde{B}$) for $\forall i < j$;
(4) $S_{set}(\tilde{A}, \tilde{B}) = 1$;

The first and the second property guarantee that all common alarms in A and B will be inherited by $\tilde{A}$ and $\tilde{B}$. The third property means that any ordered alarms in $\tilde{A}$ or $\tilde{B}$ have the same order in the original sequences A or B. The final property is obvious since all the unique alarm tags in $\tilde{A}$ and $\tilde{B}$ are the same. Using the common alarm indexing approach, all the irrelevant alarms are excluded. Thereby, the search space in the sequence alignment can be reduced by $O(mn - \tilde{m}\tilde{n})$.

5. NUMERICAL EXAMPLE

To illustrate how the proposed method works, two alarm flood sequences A and B are given in Table 4. Both of them contain 12 alarms sorted by their time stamps (the date information is omitted in the table). Three priority options are allocated to all alarm tags. First, using the set based pre-matching, identical tags 1, 2, 3, 4, 5, 6, 7 and 9 are found between A and B. By excluding the irrelevant tag 8, two subsequences $\tilde{A}$ and $\tilde{B}$ are indexed as $\tilde{A} = A$ and $\tilde{B} =< 9, 5, 2, 1, 3, 4, 7, 3, 4, 9, 6 >$. Furthermore, the comparison proceeds to the sequence alignment. Successively matched pairs can be indexed from $\tilde{A}$ and $\tilde{B}$. Taking matched pairs of top 2 scores as seeds, we have $\tilde{Z}(1) = (4, 3, 4)$ with $H_s(\tilde{Z}(1)) = 15$, as well as $\tilde{Z}(2) = (8, 5, 3)$ with $H_s(\tilde{Z}(2)) = 12$.

In the extending stage, the extension threshold is set as $T = 2|\delta| = 2$, i.e. the alignments with two gaps are permitted. Considering that both A and B contain some simultaneously raised alarms, the time ambiguity tolerance

Table 4. Alarm floods A and B with time and priority information.

Sequence A			Sequence B		
e_i	t_i	p_i	e_i	t_i	p_i
9	02:20:01	Low	9	11:00:01	Low
5	02:20:55	Low	5	11:01:11	Low
6	02:22:02	Low	2	11:01:59	High
2	02:22:42	High	1	11:02:18	High
1	02:23:35	High	3	11:03:11	Low
3	02:24:22	Low	4	11:04:01	Low
4	02:25:11	Low	7	11:06:32	Emg.
3	02:27:02	Low	3	11:06:33	Low
4	02:27:03	Low	4	11:06:33	Low
7	02:27:03	Emg.	8	11:07:16	Low
4	02:29:05	Low	9	11:08:45	Low
5	02:30:05	Low	6	11:09:56	Low

scheme is adopted. The standard deviation of the Gaussian kernel is set as $\sigma = 2$. To extend $\tilde{Z}(1) = (4,3,4)$ to the right direction, we have subsequences $\tilde{A}_r =< 3,4,7,4,5 >$ and $\tilde{B}_r =< 7,3,4,9,6 >$. The score matrix H for $\tilde{A}_r$ and $\tilde{B}_r$ is generated as in Table 5.

Table 5. Matrix H in the right extension for seed $Z(1) = (4,3,4)$.

		7	3	4	9	6
	$\underline{17}$	16	0	0	0	0
3	16	$\underline{19.3537}$	19	0	0	0
4	15	18.3537	$\underline{22.3537}$	22	0	0
7	14	21	23.3549	$\underline{27.3549}$	26.3549	25.3549
4	0	20	24	26.3549	25.3549	24.3549
5	0	19	23	25.3549	24.3549	23.3549

Backtracking the search path, the best alignment in the right extension is found and indicated by underscored values in Table 5. In the left extension, the alignment is achieved in the same manner. Combining the extension in two directions, the maximum alignment is achieved as

$$\begin{array}{l} \tilde{A}: 9\ 5\ 6\ 2\ 1\ 3\ 4\ \underline{3}\ \underline{4}\ \underline{7} \\ \quad\ \, |\ |\ \ \ \ |\ |\ |\ |\ |\ |\ | \\ \tilde{B}: 9\ 5\ -\ 2\ 1\ 3\ 4\ \underline{7}\ \underline{3}\ \underline{4} \end{array} \tag{22}$$

where the tags underscored are not exactly matched in the corresponding positions but detected as matched pairs due to their close time stamps. For the extension based on $\tilde{Z}(1)$, the maximum scores in the left extension and the right extension are $H_l = 22$ and $H_r = 27.3549$. Thus, the final score is $S(\tilde{Z}(1)) = 22 + 27.3549 - 15 - 2 \times 2 = 30.3549$. Fig. 2 illustrates the extension based on the seed $\tilde{Z}(1)$. Green squares indicate matched alarm pairs while grey squares represent the entries where the extension proceeds. White squares denotes the matrix cells not being proceeded in the computation. It is obvious that the extension avoids most search space so that the computation complexity drops drastically to a low level.

In the same manner, the seed $\tilde{Z}(2) = (8,5,3)$ is extended to achieve the maximum alignment as

$$\begin{array}{l} \tilde{A}: 9\ 5\ 6\ 2\ 1\ 3\ 4\ 3\ 4\ 7\ \underline{4}\ - \\ \quad\ \, |\ |\ \ \ \ |\ |\ \ \ \ \ \ \ \ |\ |\ |\ | \\ \tilde{B}: 9\ 5\ -\ 2\ 1\ -\ -\ 3\ 4\ 7\ 3\ \underline{4} \end{array} \tag{23}$$

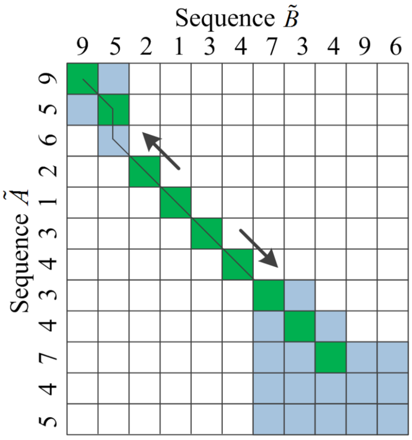

Fig. 2. Graph example of the gapped extension.

The last matched pair is found at the 11th matched position rather than the 12th position in (23). That is because the 8th tag and the 9th tag in B have the same time stamp; then the order for them should be irrelevant. The final score for the alignment based on $\tilde{Z}(2)$ is $S(\tilde{Z}(2)) = 26+17-12-2\times 2 = 27$. $S(\tilde{Z}(1)) > S(\tilde{Z}(2))$. Thus in this case, (22) presents the best alignment between A and B.

6. INDUSTRIAL CASE STUDY

To illustrate the fast sequence alignment algorithm, industrial case studies are presented. Historical alarm data with 1858 unique alarms and 3 priority levels was extracted from an operating industrial chemical plant. With chattering alarms removed, 699 alarm floods were extracted and reserved in a sequence database $\mathbb{B}$. The average length of alarm floods is 89.65. 10 alarm floods $A_i (i = 1,...,10)$ of different lengths as shown in Table 6 are prepared to query the database $\mathbb{B}$. Using the set based pre-matching approach, some relevant object alarm floods are kept for further sequence alignment while others are discarded. The number of object sequences kept for each query alarm flood A_i is listed as the third column in Table 6. Then, both of the modified Smith-Waterman algorithm (Cheng et al., 2013) and the proposed fast sequence alignment algorithm are applied to query the sequence database. The priority based scoring strategy in Table 3 is adopted by both approaches. The simulation platform has a 3.3GHz CPU, 4G RAM and 64bit operation system. Simulation results are shown in Fig. 3. $\bar{S}(A,B)$ indicates the averaged best alignment score between A and object sequences in $\mathbb{B}$; $\bar{C}(A,B)$ denotes the averaged number of matched pairs; $\bar{t}(A,B)$ denotes the averaged computation time.

By comparing the values of $\bar{S}(A,B)$ and $\bar{C}(A,B)$, it is obvious that the proposed method achieves much better results than the modified Smith-Waterman algorithm. The reason is that the common alarm indexing approach adopted by the proposed method excludes many irrelevant alarms. Hence more identical alarms can be found. Comparing the average computation time, the merit of the proposed method is obvious. Table 6 presents the total query time for each query alarm flood A_i using the two approaches. Compared with the modified Smith-

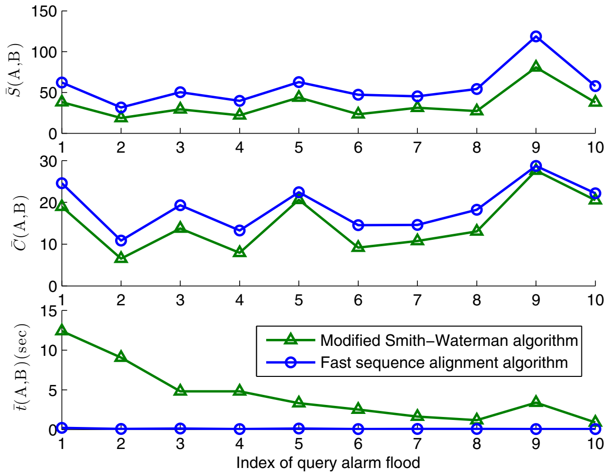

Fig. 3. Comparison for ten query alarm floods using two approaches.

Waterman algorithm, the proposed method is 18 to 350 times faster.

Table 6. Results of querying $\mathbb{B}$ for $A_i(i = 1, ..., 10)$. Length is referred to as the sequence length of each query alarm flood. $\mathcal{N}$ denotes the number of relevant object sequences found from $\mathbb{B}$ using the set based pre-matching. $\mathcal{T}_1$, $\mathcal{T}_2$ indicate the total computation time of the proposed algorithm and the modified Smith-Waterman algorithm respectively.

Flood ID	Length	$\mathcal{N}$	$\mathcal{T}_1$ (sec)	$\mathcal{T}_2$ (sec)
1	2272	630	62.8673	7725.8088
2	1866	625	15.8472	5585.8118
3	1178	681	36.1924	3235.4337
4	973	573	12.4679	2717.1148
5	732	554	39.4404	1813.6680
6	589	558	15.6735	1386.0295
7	388	541	16.6796	874.8470
8	282	502	21.6731	586.8778
9	185	55	2.1984	179.4622
10	111	164	7.7631	140.8826

7. CONCLUDING REMARKS

To find root causes and assist the prediction of alarm floods, the sequence alignment algorithms have been used to discover useful information from alarm data. In view of the low computational efficiency of existing algorithms, a fast sequence alignment method is developed; it accelerates the computation using a seed-and-extend strategy. Specialized for alarm floods, the priority and time information are adopted. Accordingly, the new method is more sensitive to alarms of higher priorities and has the ability to forget orders of alarms occurring almost simultaneously. Meanwhile, the set based matching avoids unnecessary computation by excluding all irrelevant alarm floods and alarm tags. Industrial case studies show that the proposed method is much faster than the modified Smith-Waterman algorithm. For industrial applications, the proposed method can help in identifying alarm floods caused by common faults. Thereby, operators will be able to make early decisions and prevent further incidents based on the similar alarm sequences.

REFERENCES

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215, 403-410.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25, 3389-3402.

Ahmed, K., Izadi, I., Chen, T., Joe, D., & Burton, T. (2013). Similarity analysis of industrial alarm flood data. *IEEE Transactions on Automation Science and Engineering*, 10, 452-457.

Cheng, Y., Izadi, I., & Chen, T. (2013). Pattern matching of alarm flood sequences by a modified Smith-Waterman algorithm. *Chemical Engineering Research and Design*, 91, 1085-1094.

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

Hollifield, B.R., Habibi, E. & ,Pinto, J.(2011). *Alarm Management: A Comprehensive Guide.* Research Traingle Park, NC: ISA.

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.

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. *Chemical Engineering Research and Design*, 91, 2550-2558.

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48, 443-453.

Rothenberg, D.H. (2009). *Alarm Management for Process Control: A Best-Practice Guide for Design, Implementation, and Use of Industrial Alarm Systems.* NewYork: Momentum Press.

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147, 195-197.

Wang, J., & Chen, T. (2013). An online method for detection and reduction of chattering alarms due to oscillation. *Computers and Chemical Engineering*, 54, 140-150.

Wang, J., & Chen, T. (2014). An online method to remove chattering and repeating alarms based on alarm durations and intervals. *Computers and Chemical Engineering*, 67, 43-52.