Journal of Process Control 82 (2019) 44–57

Contents lists available at ScienceDirect

Journal of Process Control

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

Accelerated multiple alarm flood sequence alignment for abnormality pattern mining

Shiqi Lai [a], Fan Yang [b,*], Tongwen Chen [a], Liang Cao [b]

[a] Department of Electrical and Computer Engineering, University of Alberta, Edmonton T6G 2V4, Canada
[b] Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 27 March 2017
Received in revised form 7 May 2019
Accepted 5 June 2019
Available online 23 August 2019

Keywords:
Alarm flood analysis
Industrial alarm monitoring
Time-stamped sequences
Multiple sequence alignment
Smith–Waterman algorithm

ABSTRACT

Alarm floods can interfere with operators and may therefore cause or aggravate industrial accidents. A novel algorithm is proposed for pattern mining in multiple alarm floods. Unlike traditional methods which either cannot deal with multiple sequences with time stamps or suffer from high computational cost, the computational complexity of this proposed algorithm is reduced significantly by introducing a generalized pairwise sequence alignment method and a progressive multiple sequence alignment approach. Two types of alignment refinement methods are developed to improve the alignment accuracy. The effectiveness of the proposed algorithm is tested using a dataset from a real chemical plant.

1. Introduction

1.1. Background and motivation

As the scale of industrial plants grows, process monitoring becomes indispensable for ensuring the safety and efficiency of operation. An alarm is defined as an audible and/or visible means of indication to the operator an equipment malfunction, process deviation, or abnormal condition requiring a responses [1]. In a typical modern control system, a Message Processor program is employed to handle the alarms which are detected by the data acquisition subsystem. An alarm message is a text message or digital message generated by the alarm system and stored in the alarm log. To be more specific, when a process value exceeds one of its predetermined thresholds, an alarm message is generated. Alarms can be easily configured on a Distributed Control Systems (DCS), which reduces the cost of alarm design and configuration, but on the other hand prolongs the life-cycle of alarm rationalization, targeted to avoid poor alarm designs. One consequence of poor alarm designs is an increase in both the number and intensity of alarm floods. Alarm floods can interfere with operators and may therefore cause or aggravate industrial accidents because even an experienced operator can be overwhelmed by tens or hundreds of alarms raised in a short period of time. Without enough time for analysis, an operator can only handle the abnormal event based on his/her experience or even take no actions, likely leading to improper executions for important abnormalities. As a result, based on operators' normal response time [2], both EEMUA, ISA and IEC standards [3,1,4] recommend to set the upper limit of alarms announced to an operator to be 6 alarms/h, and the alarm rate threshold for alarm floods to be 10 alarms/10 min per operator.

One way to reduce alarm floods is to study univariate alarms, as pointed out in [5]: "an alarm flood reduction will almost certainly require a rationalization exercise to challenge each and reduce the number of configured alarms." Usually a majority of univariate alarms are chattering alarms, which can be removed efficiently by applying delay timers or dead-bands. Techniques for designing delay timers and dead-bands have been proposed in [6–8].

Another way to reduce alarm floods is to rationalize consequential alarms. A alarm sequence pattern is usually generated when there is an abnormality propagating through physical connections or information paths in a process. This type of pattern can be obtained by applying pattern mining algorithms. The discovered pattern can be potentially used in the advanced methods mentioned in the EEMUA standard [3], such as predictive alarming, online alarm attribute modification, and online alarm suppression. Using pattern alarm sequences, causality analysis, either based on process data [9–11] or alarm data (event logs) [12], can also be conducted to investigate the root cause.

The authors in [13] summarized and categorized the existing studies on the alarm overloading problem and formulated nine fundamental research problems to be solved. In this paper, we propose an accelerated multiple sequence alignment algorithm

* Corresponding author.
E-mail addresses: slai3@ualberta.ca (S. Lai), yangfan@tsinghua.edu.cn (F. Yang), tchen@ualberta.ca (T. Chen), caoliang@mail.tsinghua.edu.cn (L. Cao).

https://doi.org/10.1016/j.jprocont.2019.06.004
0959-1524/

for pattern mining in multiple alarm floods, which potentially can be used in solving the 5th and 6th listed problems, namely, "whether there exists any nuisance alarms in the historical data, worthy of redesigning alarm generation mechanisms" and "how to design mechanisms to generate predictive alarms in order to predict upcoming critical abnormal events."

1.2. Alarm flood analysis

To give further explanation of our contribution in solving the 5th and 6th listed problems, we need to have a detailed survey on alarm flood analysis, which contains univariate alarm analysis, offline pattern mining of alarm floods, online alarm flood pattern matching, and causality analysis. Usually, a large proportion of an alarm flood are nuisance alarms, of which chattering alarms form an important part. Some methods, such as delay-timers [14], have been used to reduce such chattering alarms. However, these methods cannot totally suppress nuisance alarms during alarm floods, the majority of remaining alarms are consequential 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 is useful to reveal connections of alarm messages and discover possible patterns in alarm flood sequences.

Fig. 1 shows the procedures of alarm flood analysis. The purpose of the offline part is to set up a pattern database for offline analysis and the use of online pattern matching. 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 recommended by ISA standard is used to extract alarm floods. The periods during which the alarm rate is higher than the threshold is extracted. The extracted alarm flood sequences are then numbered and saved for further pattern analysis.

In the next step, pattern mining algorithms can be applied to automatically find a common pattern sequence for each of the clusters and save the patterns into a database for offline and online analysis. The pattern database can provide early prediction for an incoming alarm flood by matching the online alarm sequence with the patterns in the database and obtain their similarity scores; then, based on these scores, identify whether the online sequence is similar to any of the patterns in the database and thereafter predict the incoming alarm flood if a matching can be found [15]. In Fig. 1, 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 target the root cause.

1.3. Literature survey

This subsection contains a detailed literature survey on recent work for sequence pattern mining and alarm flood pattern analysis.

Expert consultation and operator experience, by far, are still the two approaches that have been used by most industrial companies when dealing with alarm floods. Expert consultation provides good and accurate results; however, without doubts, its efficiency is extremely low because of the involvement of a relatively large amount of process knowledge. The approach based on the operator experience is usually faster, but its accuracy is not guaranteed since it depends heavily on human judgments. Many pattern mining algorithms have been developed to facilitate the study of alarm floods.

Sequence pattern mining finds relevant parts in data examples that appear repeatedly. A sequential pattern is a subsequence that appears in a data set with frequency no less than a user-specified threshold. For example, a subsequence, buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a sequential pattern. In [16], an A priori-like algorithm was proposed to mine frequent patterns by combining short sequences into longer ones. The authors in [17] developed an algorithm to achieve patterns based on tree structures. In [18], a vertical data format was utilized to generate patterns, which did not require multiple scans over the database. Algorithms based on projections were proposed in [19,20], which had improvements on efficiency.

In the biology area, a very large number of pattern mining algorithms have been proposed, most of which were modifications of the pairwise sequence alignment methods [21–24]. Mainly two types of modifications exist: the exhaustive search approach, as in [25], which can guarantee global optimality, and the progressive pairwise approach, such as [26] and [27], which can approximate global optimal solutions. There is another type of algorithm based on profile Hidden Markov Models (HMM), such as [28]. However, in [29], the authors pointed out that: "a suitable HMM architecture (the number of states, and how they are connected by state transactions) must usually be designed manually."

While a large amount of pattern mining algorithms exists in the literature, most of them cannot be directly applied to find patterns in alarm and event logs, due to the special format of alarm data (every message comes with a time stamp). A variety of pattern mining methods have been developed to analyze alarm floods. The authors in [30] manually selected some alarm tags to be the target tags at first, then applied the extracted patterns as the basis for discovery of local relationships by identifying primary and consequential alarms. It is an event-based segmentation strategy which data segmentation takes place in an event-based context. In [31], the authors proposed a way to capture the relations of alarm messages in an alarm flood using first-order Markov chains; then, the Euclidean distance between the transition probability matrices of two alarm floods was used to cluster the alarm floods into groups. The authors in [32] developed a pattern growth method to obtain patterns from alarm floods. However, the authors pointed out "the proposed method was sensitive to disturbances so that the pattern in sequences had to be exactly the same in order to be recognized." A dissimilarity-based method was proposed in [33] to extract alarm sequence templates of given faults and a Needleman–Wansch based algorithm was developed to isolate alarm sequences caused by certain known faults in [34]. In [35], the authors proposed a method to re-order alarms during alarm floods by assigning their priorities values and designed an interface for a better operator support during flood scenarios.

Time information is also very important. On one hand, two sequences with the same alarms and their ordering but different time intervals in between the alarms can be caused by totally different faults. On the other hand, even the orders of some alarms in two alarm sequences are different, they still can be alarm sequence patterns. Because of this, methods have been proposed to take time intervals between alarms into consideration when studying the sequences.

For example, for two alarm sequences, take into account the time information as shown below:

$$S_1 = <(4,00:01),(5,00:02),$$
$$\textbf{(3,00:05),(2,00:06),(1,00:08),(3,00:19),}(4,00:20)>$$
$$S_2 = <(3,00:00),(2,00:01),$$
$$\textbf{(3,00:03),(2,00:04),(1,00:05),(3,00:09),}(5,00:12)>$$

where the first one of each pair is the type of alarm and the second one is the corresponding time stamp. By analyzing the time infor-

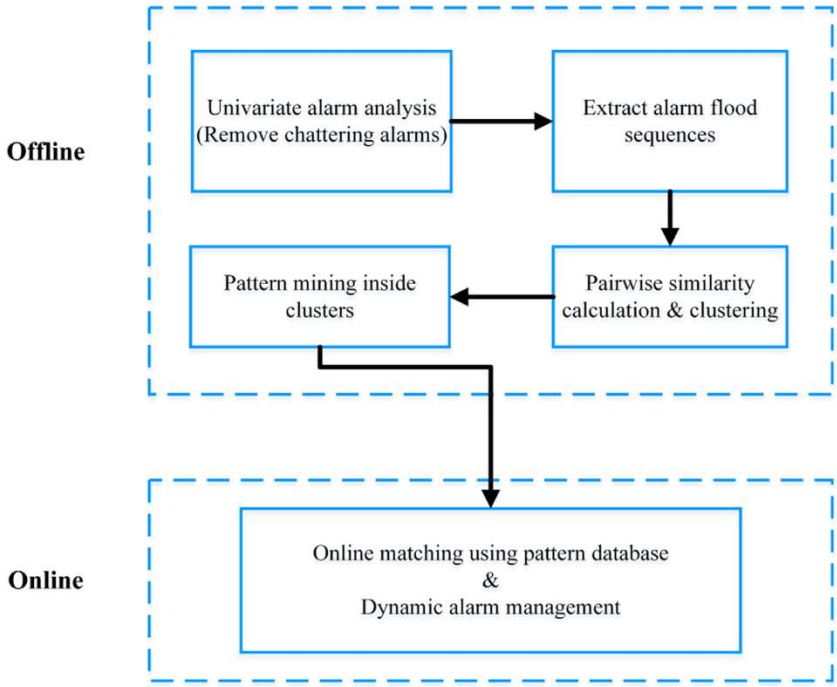

Fig. 1. Flowchart of alarm flood analysis.

mation, we may find that the bold part may not be alarm sequence pattern due to the consideration of time stamps.

For two alarm sequences,

$S_3 = <(4, 00:01), (5, 00:02),$

(3,00:05),(1,00:06),(2,00:08),(3,00:13),$(4, 00:20)>$

$S_4 = <(3, 00:00), (2, 00:01),$

(3,00:05),(2,00:06),(1,00:08),(3,00:12),$(5, 00:17)>$

even the orders of some alarms in two alarm sequences are different, they may be an alarm sequence pattern due to the fact that the time interval of alarms occurrences is relatively close, the similarity of these two alarm sequences may be high.

In [36], Generalized Sequential Patterns (GSP) was applied to search for pattern alarm sequences in alarm floods; the algorithm was able to blur the orders of alarms if they are raised in quick succession and thus their order becomes less important. Algorithms for pattern matching of two sequences were developed in [37] and [38], based on the Smith-Waterman [22] and BLAST [23] algorithms, respectively. In both papers, a weighted time distance and vector were used to take the relative time between alarms into account during the alignment. In [39], the authors proposed a method for pattern mining in multiple alarm flood sequences using a sequence alignment approach. However, the computational cost of the algorithm does not scale well with the number and length of sequences to be aligned.

1.4. Contribution and organization of the paper

In this work, we propose an accelerated multiple pattern mining algorithm to find a suitable alignment of multiple alarm flood sequences and obtain the pattern for a selected alarm flood cluster. Unlike traditional methods which either cannot deal with multiple sequences with time stamps or suffer from high computational cost, the computational complexity of this proposed algorithm is reduced significantly by introducing a generalized pairwise sequence alignment method and a progressive multiple sequence alignment approach. Based on this work, a pattern database can be set up, which can provide early prediction for an incoming alarm flood by matching the online alarm sequence with the patterns in the database and predict an incoming alarm flood if a matching can be found. The discovered patterns can help train operators to handle corresponding series of alarm messages more efficiently in order to prevent the overwhelming situation during alarm floods. Some badly configured parts in alarm systems can be revealed by the alarm flood patterns as well.

The rest of the paper is organized as follows. The problem description and algorithm principle are given in section 2. In section 3, an industrial case study is provided to test both efficiency and accuracy of the proposed algorithm and comparisons are made with the exhaustive search approach in [39]. Detailed discussions on the accuracy, computational complexity, algorithm convergence, and some potential problems are carried out in section 4, followed by the conclusions in section 5.

2. Algorithm principle

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.

2.1. Problem formulation

2.1.1. Alarm system and alarm message analysis

To offer a comprehensive explanation of alarms and alarm messages, we give the figure of the alarm system dataflow and the figure of alarm message log from industry. Fig. 2 shows a typical schematic of alarm system dataflow. The basic Process Control System (BPCS) and Safety Instrumented System (SIS) are the two important components that control the process and generate alarms based on sensor measurements and predefined logics. In our work, we focus on the alarms from safety instrumented systems based on the standard IEC 62682 [4], the design and management of safety instrumented systems are excluded from this standard. Please refer to the standard IEC 61511 [40]. The panel and the Human Machine

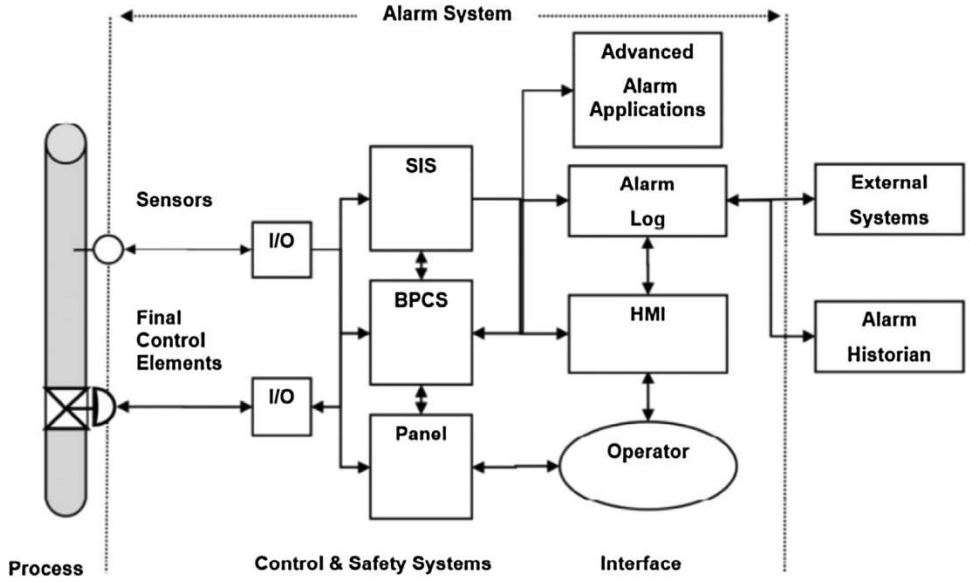

Fig. 2. Alarm system dataflow [1].

Interface (HMI) allow the operator to intervene the control of the process and view alarm logs.

In a DCS engaged plant, all monitored process variables and alarms are collected onto the DCS server, where alarms are formatted and stored in alarm logs. Generally, in alarm logs, an alarm message contains important information such as time stamp, tag name, alarm type (identifier), priority, and acknowledgment state. An alarm message may also include descriptive information such as trip value, event, and description. Time stamp indicates the occurrence time of an alarm. Tag name is a unique code of a system variable that is being monitored; it indicates the process variable that has caused the alarm. Alarm type, or identifier, is the alarm type related to the monitored variable (e.g., PVLO when the variable is under its low limit). Priority shows the importance of alarms. Acknowledgment state shows the status of an alarm, namely, Return to Normal (RTN) or Alarm (ALM).

Fig. 3 gives an example of a segment of an alarm log. It is necessary to point out that there are different kinds of alarm logs in different process or situations. Therefore, we need to give specific explanations according to different situations. The definitions in this example cannot be applied directly to other examples. In the example, a PVLO alarm was raised for tag A at 2013-09-01 0:13 because its corresponding process variable went below the trip value, which was set as 131. Then, this PVLO alarm for tag A returned to normal at 2013-09-01 0:13 since the corresponding process variable went back to its normal range. The priority of this alarm is JOURNAL, which means the lowest priority. Normally, we connect the tag name and the tag identifier of an alarm together with a dot in between while processing alarm messages, e.g., A. PVLO, and replace ALM and RTN messages with 1 and 0 for computing purposes.

2.1.2. Mathematical representation

Consider the following alarm message sequences:

$$S_1 = <(e_{11}, t_{11}), (e_{12}, t_{12}), ..., (e_{1m}, t_{1m}), ..., (e_{1M}, t_{1M})>,$$

$$S_2 = <(e_{21}, t_{21}), (e_{22}, t_{22}), ..., (e_{2n}, t_{2n}), ..., (e_{2N}, t_{2N})>,$$

$$...$$

$$S_J = <(e_{J1}, t_{J1}), (e_{J2}, t_{J2}), ..., (e_{Jo}, t_{Jo}), ..., (e_{JO}, t_{JO})>,$$

where $e_{1m}, e_{2n}, ..., e_{Jo} \in \Sigma$ (the set of alarm types) and $t_{1m}, t_{2n}, ..., t_{Jo}$ are the corresponding time stamps, respectively. $S_1, S_2, ..., S_J$ are similar sequences detected by running pairwise sequence matching algorithms such as [37]. The objective is to obtain the optimal alignment of these sequences by adding gaps (represented by "[]") and deleting unrelated alarm messages, for example:

$$<(e_{16}, t_{16}), (e_{17}, t_{17}), (e_{18}, t_{18}), [\ ], (e_{19}, t_{19})>,$$

$$<(e_{22}, t_{22}), [\ ], (e_{23}, t_{23}), (e_{24}, t_{24}), (e_{25}, t_{25})>,$$

$$...$$

$$<[\ ], (e_{J1}, t_{J1}), (e_{J2}, t_{J2}), [\ ], (e_{J3}, t_{J3})>.$$

2.2. Generalized pairwise sequence alignment

In this subsection, to explain the principles of generalized pairwise sequence alignment, three elements, including time distance and weight vectors, generalized similarity score and generalized dynamic programming procedure, will be introduced.

2.2.1. Time distance and weight vectors

Time stamps are important components of alarm messages; two alarm messages with different time intervals could indicate totally different problems in the process. In [37], it introduces the time information into the Smith-Waterman algorithm to evaluate alarm sequence alignment, which is called the modified Smith-Waterman algorithm. The Smith-Waterman algorithm is proposed for the local alignment of sequences in the field of biological informatics. 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) [22].

The Smith-Waterman algorithm is a dynamic programming algorithm, which contains two main steps. The first step is calculating the similarity score of two sequences and forming a similarity matrix or table, the similarity score of the mismatch and gap-match is negative as a penalty. The similarity score function can be defined as follows:

$$(x_i, y_i) = \begin{cases} 1, \text{if} e_{xi} = e_{yj} \\ \mu, \text{if} e_{xi} \neq e_{yj} \end{cases} \quad (1)$$

TIME	TAG	TRIP_VAL	TYPE	PRIO	DSCR	UNIT	EVENT	ACK
2013-09-01 0:13	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:13	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	134.994	RTN
2013-09-01 0:15	B	3.9	PVLO	JOURNAL	D3: ABS AREA SUMP LEVEL	Q1	3.899	ALM
2013-09-01 0:15	C		CMDDIS	HIGH	G-5B Abs Area Sump Pmp	Q1	CLOSED	ALM
2013-09-01 0:15	D		CMDDIS	HIGH	D1:ABS HOLD TNK FLSH WTR	Q1	TRANSIT	ALM
2013-09-01 0:15	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:15	D	2	CMDDIS	HIGH	D1:ABS HOLD TNK FLSH WTR	Q1	OPENED	RTN
2013-09-01 0:16	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	134.017	RTN
2013-09-01 0:16	C		CMDDIS	HIGH	G-5B Abs Area Sump Pmp	Q1	CLOSED	RTN
2013-09-01 0:18	E	37.8	PVHI	LOW	V2: HYDR CON-TROL UNIT	Q1		RTN
2013-09-01 0:19	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:20	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	136.02	RTN
2013-09-01 0:22	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM
2013-09-01 0:22	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	134.017	RTN
2013-09-01 0:25	B	3.9	PVLO	JOURNAL	D3: ABS AREA SUMP LEVEL	Q1		RTN
2013-09-01 0:26	F	1.341	PVHI	LOW	G4: PRMRY UF-PLT481/D1	Q1	1.341	ALM
2013-09-01 0:26	F	1.341	PVHI	LOW	G4: PRMRY UF-PLT481/D1	Q1	1.341	ALM
2013-09-01 0:28	A	131	PVLO	JOURNAL	G1B:RCYCLPMP CURRENT IND	ZD	130.989	ALM

Fig. 3. An example of a segment of an Alarm log.

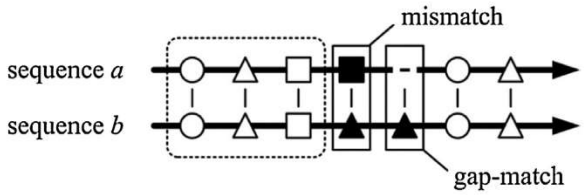

Fig. 4. Smith–Waterman algorithm.

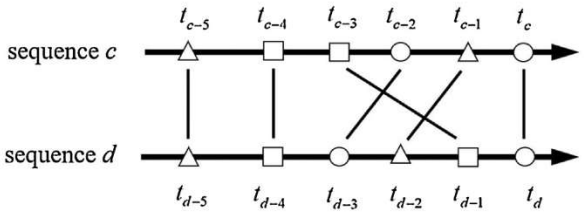

Fig. 5. Modified Smith–Waterman algorithm.

The element of similarity matrix H can be recursively calculated by the following equation:

$$H_{i+1,j+1} = \max\left\{H_{i,j} + s\left(x_i, y_j\right), H_{i,j+1} + \delta, H_{i+1,j} + \delta, 0\right\} \tag{2}$$

where δ is uniform gap penalty. The second step is backward path search based on similarity table. Go backward from this highest value until meet an entry with value 0 (Figs. 4 and 5).

For the modified Smith-Waterman algorithm in [37], it introduces the time information in sequences, which can be reflected by the change of similarity score function:

$$s((e_a, t_a), (e_b, t_b)) = \max_{1\le k\le K}\left[w_a^k \times w_b^k\right](1-\mu) + \mu \tag{3}$$

where w is the time weight vector, w is a time weighting function with respect to the time distance. The time distance vector for an alarm message (e_m, t_m) is defined as:

$$\begin{aligned} \mathbf{d}_m &= [d_m^1, d_m^2, ..., d_m^k, ..., d_m^K] \\ d_m^k &= |t_m - t_k|, \end{aligned} \tag{4}$$

where K is the total number of alarm messages in the sequence. Thus, d_m^k gives the absolute time distance between (e_m, t_m) and (e_k, t_k) in the sequence. The time weight vector for (e_m, t_m) is defined as:

$$\begin{aligned} \mathbf{w}_m &= [w_m^1, w_m^2, ..., w_m^k, ..., w_m^K] \\ &= [f(d_m^1), f(d_m^2), ..., f(d_m^k), ..., f(d_m^K)], \end{aligned} \tag{5}$$

where $f(\cdot): \mathbb{R} \to \mathbb{R}$ is the weighting function. A scaled Gaussian function is selected in this paper:

$$f(x) = e^{-x^2/2\sigma^2}, \tag{6}$$

where σ is the standard deviation, controlling how much weight to be put on the close-by alarm messages to blur their orders in the alignment. The function also normalizes the real time distance to [0, 1]; a large output indicates a short time distance. Correspondingly, for modified Smith-Waterman algorithm, the element of similarity matrix H can be recursively calculated by the following equation with a uniform gap penalty δ:

$$H_{i+1,j+1} = \max\left\{H_{i,j} + s\left((e_i, t_i), \left(e_j, t_j\right)\right), H_{i,j+1} + \delta, H_{i+1,j} + \delta, 0\right\} \tag{7}$$

$$
\begin{array}{l}
 & T_{A1} & T_{A2} & T_{A3} & & T_{Ap} & & T_{AP} & \\
A_1 = < & (e^1_{A1}, t^1_{A1}) & (e^2_{A1}, t^2_{A1}) & (e^3_{A1}, t^3_{A1}) & \cdots & (e^p_{A1}, t^p_{A1}) & \cdots & (e^P_{A1}, t^P_{A1}) & > \\
A_2 = < & (e^1_{A2}, t^1_{A2}) & [\,] & (e^3_{A2}, t^3_{A2}) & \cdots & [\,] & \cdots & (e^P_{A2}, t^P_{A2}) & > \\
\vdots & & & & & & & & \\
A_M = < & [\,] & (e^2_{AM}, t^2_{AM}) & [\,] & \cdots & (e^p_{AM}, t^p_{AM}) & \cdots & (e^P_{AM}, t^P_{AM}) & > \\
 & & & & & & & & \\
B_1 = < & (e^1_{B1}, t^1_{B1}) & [\,] & (e^3_{B1}, t^3_{B1}) & \cdots & (e^q_{B1}, t^q_{B1}) & \cdots & (e^Q_{B1}, t^Q_{B1}) & > \\
B_2 = < & (e^1_{B2}, t^1_{B2}) & [\,] & (e^3_{B2}, t^3_{B2}) & \cdots & (e^q_{B2}, t^q_{B2}) & \cdots & (e^Q_{B2}, t^Q_{B2}) & > \\
\vdots & & & & & & & & \\
B_N = < & [\,] & (e^2_{BN}, t^2_{BN}) & [\,] & \cdots & (e^q_{BN}, t^q_{BN}) & \cdots & (e^Q_{BN}, t^Q_{BN}) & > \\
 & T_{B1} & T_{B2} & T_{B3} & & T_{Bq} & & T_{BQ} &
\end{array}
$$

Fig. 6. An example of tuples in two alarm sequence alignments.

2.2.2. Worked example of time distance and weight vectors

Here is a demonstration of how the time distance and weight vectors are calculated for an alarm sequence. Consider the following alarm sequence

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

where there are 5 alarm messages and 4 alarm types. The time distance vector $\mathbf{d}_1$ for message (2, 2) is calculated as: $d_1^1 = 2 - 2 = 0, d_1^2 = 3 - 2 = 1, d_1^3 = 3.5 - 2 = 1.5, d_1^4 = 5 - 2 = 3$, and $d_1^5 = 5.2 - 2 = 3.2$. The time distance matrix formed by vectors $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{d}_3$, $\mathbf{d}_4$, and $\mathbf{d}_5$ is

$$[\mathbf{d}_1^T, \mathbf{d}_2^T, \mathbf{d}_3^T, \mathbf{d}_4^T, \mathbf{d}_5^T] = \begin{bmatrix} 0.0 & 1.0 & 1.5 & 3.0 & 3.2 \\ 1.0 & 0.0 & 0.5 & 2.0 & 2.2 \\ 1.5 & 0.5 & 0.0 & 1.5 & 1.7 \\ 3.0 & 2.0 & 1.5 & 0.0 & 0.2 \\ 3.2 & 2.2 & 1.7 & 0.2 & 0.0 \end{bmatrix}.$$

By applying the weighting function (with $\sigma = 1$) on the time distance matrix, the time weight matrix is obtained:

$$[\mathbf{w}_1^T, \mathbf{w}_2^T, \mathbf{w}_3^T, \mathbf{w}_4^T, \mathbf{w}_5^T] = \begin{bmatrix} 1.00 & 0.61 & 0.32 & 0.01 & 0.01 \\ 0.61 & 1.00 & 0.88 & 0.14 & 0.09 \\ 0.32 & 0.88 & 1.00 & 0.32 & 0.24 \\ 0.01 & 0.14 & 0.32 & 1.00 & 0.98 \\ 0.01 & 0.09 & 0.24 & 0.98 & 1.00 \end{bmatrix}.$$

2.2.3. Generalized similarity score

In [37], the authors proposed a way to calculate the similarity score between two alarm messages in two alarm sequences. Here we generalize it to calculate the similarity score between two tuples in two alarm sequence alignments. The definition of a tuple is illustrated by the example in Fig. 6. In the figure, there are two alarm sequence alignments ($A_1, A_2, \ldots, A_M$ and $B_1, B_2, \ldots, B_N$); the tuples ($T_{A1}, T_{A2}, \ldots, T_{AP}$ and $T_{B1}, T_{B2}, \ldots, T_{BQ}$) are formed by alarm messages or gaps located at the same position of the corresponding alignment. The lengths of the alignments are P and Q, respectively. Notice the lengths of $A_1, A_2, \ldots, A_M$ are always the same because they are from the same alignment, while P and Q may not be the same value.

The formula for calculating the similarity score $\mathbb{S}(T_{Ap}, T_{Bq})$ between the tuples T_{Ap} and T_{Bq} is as follows:

$$\mathbb{S}(T_{Ap}, T_{Bq}) = \mu + \frac{1-\mu}{MN} \times \sum_{i=1}^{M} \sum_{j=1}^{N} \mathcal{S}((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj})), \tag{8}$$

where $\mathcal{S}((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj}))$ is the similarity score between two alarm messages and is obtained by

$$\begin{aligned} &\mathcal{S}((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj})) \\ &= \max\{\ s((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj})), \\ &\qquad s((e^q_{Bj}, t^q_{Bj}), (e^p_{Ai}, t^p_{Ai}))\}, \end{aligned} \tag{9}$$

where

$$\begin{aligned} &s((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj})) \\ &= \begin{cases} \max\limits_{1 \le k \le P} \{w^k_{Ai,p} : e^k_{Ai} = e^q_{Bj}\}, & \text{if the set isn't empty} \\ & \text{and } \left(e^q_{Bj}, t^q_{Bj}\right) \text{ is not a gap} \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{10}$$

$$\begin{aligned} &s((e^q_{Bj}, t^q_{Bj}), (e^p_{Ai}, t^p_{Ai})) \\ &= \begin{cases} \max\limits_{1 \le k \le Q} \{w^k_{Bj,q} : e^k_{Bj} = e^p_{Ai}\}, & \text{if the set isn't empty} \\ & \text{and } \left(e^p_{Ai}, t^p_{Ai}\right) \text{ is not a gap} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{11}$$

The negative parameter μ in Eq. (8) is the mismatch penalty. The similarity score $\mathbb{S}(T_{Ap}, T_{Bq})$ between the two tuples T_{Ap} and T_{Bq} is obtained by averaging all the pairwise similarity scores of the alarms in the two tuples. The value of $\mathbb{S}(T_{Ap}, T_{Bq})$ is always within $[\mu, 1]$; a larger value means a stronger similarity. The similarity score $\mathcal{S}((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj}))$ between two alarm messages (e^p_{Ai}, t^p_{Ai}) and (e^q_{Bj}, t^q_{Bj}) is achieved by selecting the larger value between

$$
\begin{array}{lcccccccc}
 & & T_{A1} & & T_{A2} & & T_{A3} & & T_{A4} & & T_{A5} \\
A_1 & = < & (2,2) & & (1,3) & & (3,3.5) & & (1,5) & & (4,5.2) & > \\
A_2 & = < & (2,22) & & (1,22.5) & & (3,24) & & (1,25) & & (4,26) & > \\
 & & & & & & & & & & & \\
B_1 & = < & (2,12) & & (5,13) & & (3,14) & & (1,14.5) & & (4,15) & > \\
B_2 & = < & (2,52.5) & & (1,53) & & (3,54.5) & & [\,] & & (4,55) & > \\
 & & T_{B1} & & T_{B2} & & T_{B3} & & T_{B4} & & T_{B5} &
\end{array}
$$

Fig. 7. A numerical example for the calculation of the generalized similarity scores.

$s((e^p_{Ai}, t^p_{Ai}), (e^q_{Bj}, t^q_{Bj}))$ and $s((e^q_{Bj}, t^q_{Bj}), (e^p_{Ai}, t^p_{Ai}))$, as the commutative law does not hold. When both (e^p_{Ai}, t^p_{Ai}) and (e^q_{Bj}, t^q_{Bj}) are gaps, their score will be assigned as 0. Note that the pairwise similarity score formula in [37] is a special case of the generalized similarity score calculation proposed in this paper; when the two alignments are reduced to two alarm sequences (e.g., $M = N = 1$), the two formulae will give exactly the same output.

2.2.4. Worked example of generalized similarity score

Consider the example of two alarm sequence alignments in Fig. 7, where $M = N = 2$ and the lengths of the two sequence alignments are both 5. The time weight matrices of A_1, A_2, B_1, and B_2 are obtained as $\mathbf{W}_{A_1}$, $\mathbf{W}_{A_2}$, $\mathbf{W}_{B_1}$, and $\mathbf{W}_{B_2}$, given $\sigma = 1$. The fourth row and column of $\mathbf{W}_{B_2}$ are empty since the fourth element in B_2 is a gap. Let $\mu = -1$, the similarity score for the pair of tuples T_{A4} and T_{B4} is calculated as

$$\mathbb{S}(T_{A4}, T_{B4}) = -1 + \frac{1+1}{2\times 2} \times \sum_{i=1}^{2}\sum_{j=1}^{2} \mathcal{S}((e^4_{Ai}, t^4_{Ai}), (e^4_{Bj}, t^4_{Bj})) = -0.12,$$

where

$$\mathcal{S}((e^4_{A1}, t^4_{A1}), (e^4_{B1}, t^4_{B1})) = \max\{w^3_{A_1,4}, w^3_{B_1,4}\} = 0.88,$$

$$\mathcal{S}((e^4_{A2}, t^4_{A2}), (e^4_{B1}, t^4_{B1})) = \max\{w^3_{A_2,4}, w^3_{B_1,4}\} = 0.88,$$

$$\mathcal{S}((e^4_{A1}, t^4_{A1}), (e^4_{B2}, t^4_{B2})) = \max\{0, 0\} = 0,$$

and

$$\mathcal{S}((e^4_{A2}, t^4_{A2}), (e^4_{B2}, t^4_{B2})) = \max\{0, 0\} = 0.$$

Intuitively, this negative similarity score also makes sense since there is a gap in tuple T_{B4} and the type of the two alarm messages in tuple T_{A4} is "1" while the type of the alarm message in tuple T_{B4} is "3". The reason why this similarity score is −0.12 instead of the minimum value −1 is because there exists an alarm message (1, 14) in tuple T_{B3} that was raised slightly ahead of (3, 14.5) and its alarm type matches the alarm messages in tuple T_{A4}.

$$\mathbf{W}_{A_1} = [\mathbf{w}^T_{A_1,1}, \mathbf{w}^T_{A_1,2}, \mathbf{w}^T_{A_1,3}, \mathbf{w}^T_{A_1,4}, \mathbf{w}^T_{A_1,5}] = \begin{bmatrix} 1.00 & 0.61 & 0.32 & 0.01 & 0.01 \\ 0.61 & 1.00 & 0.88 & 0.14 & 0.09 \\ 0.32 & 0.88 & 1.00 & 0.32 & 0.24 \\ 0.01 & 0.14 & 0.32 & 1.00 & 0.98 \\ 0.01 & 0.09 & 0.24 & 0.98 & 1.00 \end{bmatrix}$$

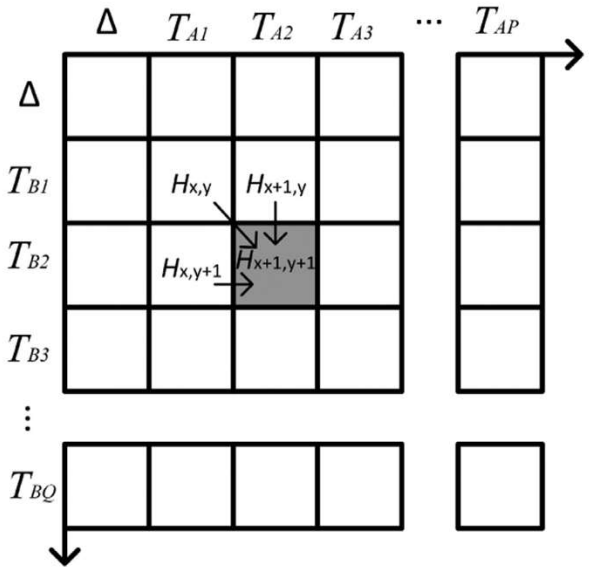

Fig. 8. An illustrative example on how to obtain the generalized dynamic programming matrix.

$$\mathbf{W}_{A_2} = [\mathbf{w}^T_{A_2,1}, \mathbf{w}^T_{A_2,2}, \mathbf{w}^T_{A_2,3}, \mathbf{w}^T_{A_2,4}, \mathbf{w}^T_{A_2,5}] = \begin{bmatrix} 1.00 & 0.88 & 0.14 & 0.01 & 0 \\ 0.88 & 1.00 & 0.32 & 0.04 & 0 \\ 0.14 & 0.32 & 1.00 & 0.61 & 0.14 \\ 0.01 & 0.04 & 0.61 & 1.00 & 0.61 \\ 0 & 0 & 0.14 & 0.61 & 1.00 \end{bmatrix}$$

$$\mathbf{W}_{B_1} = [\mathbf{w}^T_{B_1,1}, \mathbf{w}^T_{B_1,2}, \mathbf{w}^T_{B_1,3}, \mathbf{w}^T_{B_1,4}, \mathbf{w}^T_{B_1,5}] = \begin{bmatrix} 1.00 & 0.61 & 0.14 & 0.04 & 0.01 \\ 0.61 & 1.00 & 0.61 & 0.32 & 0.14 \\ 0.14 & 0.61 & 1.00 & 0.88 & 0.61 \\ 0.04 & 0.32 & 0.88 & 1.00 & 0.88 \\ 0.01 & 0.14 & 0.61 & 0.88 & 1.00 \end{bmatrix}$$

$$\mathbf{W}_{B_2} = [\mathbf{w}^T_{B_2,1}, \mathbf{w}^T_{B_2,2}, \mathbf{w}^T_{B_2,3}, \mathbf{w}^T_{B_2,4}, \mathbf{w}^T_{B_2,5}] = \begin{bmatrix} 1.00 & 0.88 & 0.14 & [\,] & 0.01 \\ 0.88 & 1.00 & 0.32 & [\,] & 0.04 \\ 0.14 & 0.32 & 1.00 & [\,] & 0.61 \\ [\,] & [\,] & [\,] & [\,] & [\,] \\ 0.01 & 0.04 & 0.61 & [\,] & 1.00 \end{bmatrix}$$

2.2.5. Generalized dynamic programming procedure

Similarly, the dynamic programming procedure proposed in [37] for aligning two alarm sequences is generalized to aligning two alarm sequence alignments.

Fig. 8 illustrates the way to achieve the generalized dynamic programming matrix from the tuples' point of view. To begin with, the first row and column of the matrix are filled in with zeros. Then, iteratively fill the rest of cells of the matrix with $H_{x,y}$ obtained from

$$\begin{aligned} H_{x+1,y+1} &= \max_{1\le i\le x, 1\le j\le y} (I(T_{A,i:x}, T_{B,j:y}), 0) \\ &= \max\{ H_{x,y} + S(T_{A,x+1}, T_{B,y+1}), \\ &\qquad H_{x,y+1} + \delta, \quad H_{x+1,y} + \delta, \quad 0\}, \end{aligned} \tag{12}$$

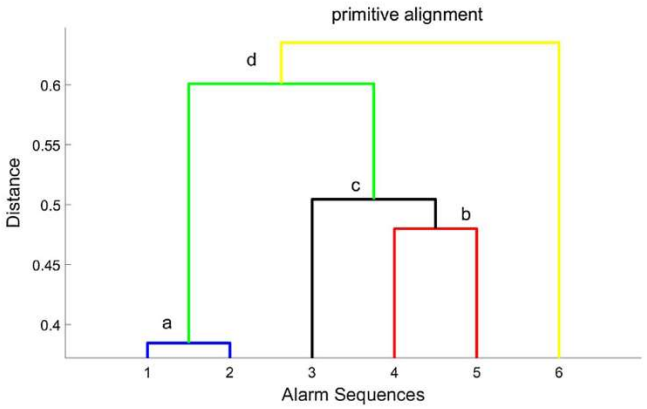

Fig. 9. An example of dendrogram and the progressive multiple sequence alignment procedure based on it.

where the negative parameter δ is the gap penalty, and the similarity index $I(T_{A,i:x}, T_{B,j:y})$ is the alignment score of the segment pair $(T_{A,i:x}, T_{B,j:y})$.

Once the whole dynamic programming matrix is achieved, the optimal alignment of the two alignments can be generated based on back-tracking. During back-tracking, the position of the maximum value in the dynamic programming matrix is located at first. Then, the algorithm tracks backwards to obtain the path along which the maximum score is generated. At last, one forward pass through the back tracking path gives the optimal alignment.

The difference between the generalized and the standard dynamic programming procedures is that the former one treats the "sequence" from a "tuple" point of view, which allows the "sequence" to be either an alarm sequence or an alignment of multiple alarm sequences. Thus, the standard dynamic programming procedure can be seen as a special case of the generalized one.

2.3. Progressive multiple sequence alignment method

The computational cost of the exhaustive search approach in [39] grows exponentially with the number of sequences and their lengths, making the algorithm easily overwhelmed by real data. To improve the efficiency, we propose a progressive multiple sequence alignment method based on the generalized dynamic programming to find the optimal sequence alignment.

In the first step of our algorithm, the modified Smith-Waterman is applied to calculate all the pairwise similarity scores between the alarm sequences. Then, a dendrogram is built based on these pairwise similarity scores using UPGMA (Unweighed Pair Group Method with Arithmetic Mean). Fig. 9 shows an example of a dendrogram, in which there are 6 alarm sequences and the height of a link indicates the dissimilarity between two connected sequences.

Next, guided by the dendrogram, the primitive alignment of the sequences is obtained by progressively aligning all the sequences, from the most similar pair to the most dissimilar ones. In the example shown in Fig. 9, the link connecting sequences 1 and 2 is the lowest one, indicating them to be the most similar sequence pair. Thus, these two sequences are aligned first and thus their alignment a is obtained. Similarly, sequences 4 and 5 are aligned and thus alignment b is obtained. Then, since there is a connection between sequence 3 and alignment b, the generalized pairwise sequence alignment method is conducted to get alignment c by aligning sequence 3 and alignment b. By iteratively aligning the sequences based on the dendrogram, the primitive alignment of all the 6 alarm sequences can be achieved.

Compared to the exhaustive search approach in [39], which conducts exhaustive search to find the exact optimal alignment, the proposed method approximates the optimal alignment by progressively aligning all the sequences. This new approach greatly improves the efficiency of the algorithm; as the cost, however, the global optimum is no longer guaranteed.

2.4. Iterative alignment refinement methods

Two iterative alignment refinement methods are developed to improve the accuracy of the primitive alignment generated by the progressive sequence alignment approach.

2.4.1. Leave-one-out alignment refinement

The first one is Leave-one-out alignment refinement. Similar to the leave-one-out cross validation, which leaves one dataset out for testing and use the rest for training, during each iteration of the leave-one-out alignment refinement, one sequence in the alignment is replaced by its original sequence and then be re-aligned with the rest of sequences in the alignment (if any, common gaps in the rest of the alignment sequences should be removed) using the generalized pairwise sequence alignment method. After re-aligned, the new alignment score is compared with the old one before the current refinement iteration, and the alignment with the higher score will be kept for the next iteration. Iteratively repeat this procedure until the alignment score converges.

Fig. 10a shows an illustration of one iteration of the leave-one-out refinement; respectively, the solid lines and "[]" represent alarm sequences and gaps in the alignments. In the example, alignment 1 is left out and replaced by its original alarm sequence, as indicated by the red bold line. Then the common gaps in the rest of the alignments (2, 3, 4, and 5) are removed before they are re-aligned with the original alarm sequence 1.

Note the convergence is guaranteed by the leave-one-out refinement method. However, it is not guaranteed to converge to the global optimum. Details of convergence will be provided in the discussion section.

2.4.2. Random-division alignment refinement

The second is Random-division alignment refinement. Compared to the leave-one-out approach, it has a relatively lower convergence speed, but in some cases, it could still improve the alignment result even when the leave-one-out alignment converges. The detailed reason will be provided in the discussion section.

As illustrated in Fig. 10b, the primitive alignment is randomly divided into two groups (1, 2, 4, and 3, 5). Then, if any, the common gaps in each of the two groups of sequences should be removed, as shown in the example. Next, the two groups of sequences are re-aligned using the proposed generalized pairwise sequence alignment method. After re-aligned, the new alignment score is compared with the old one before the current refinement iteration, and the alignment with the higher score will be kept for the next iteration. Iteratively repeat this procedure until the upper limit number of iterations is reached.

Similar to the leave-one-out refinement, the convergence is guaranteed for the random-division approach when no upper limit iteration number is set. However, the convergence is hard to be identified without knowing the optimal alignment score beforehand, also the convergence value is not guaranteed to be global optimum. Moreover, since no sequence in the alignment will be replaced by its original sequence during the refinement, the primitive alignment becomes much more critical; and the deleted alarm messages in the sequence of an alignment will never be brought back.

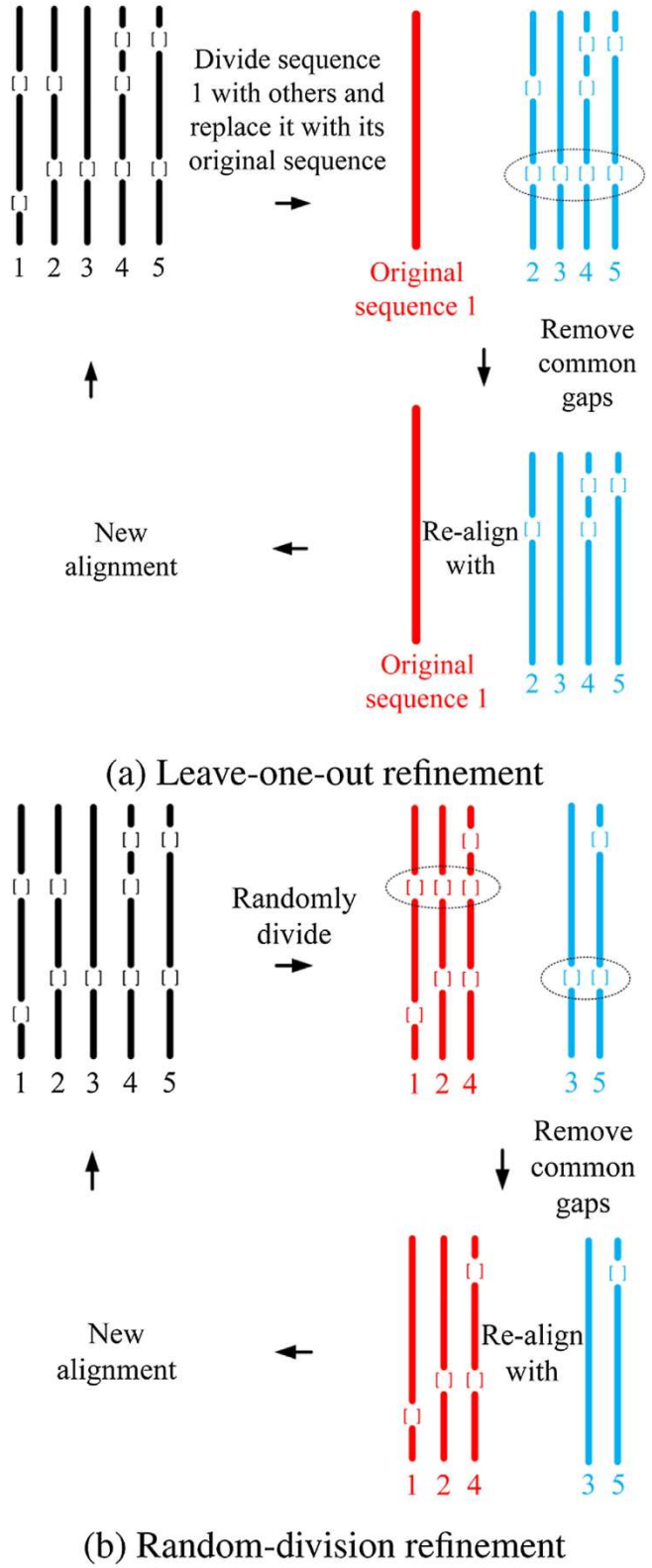

Fig. 10. Illustrations of the two types of alignment refinement methods.

3. Industrial case study

The proposed algorithm has been tested on an industrial dataset from Suncor Energy Inc. The dataset comes from a plant that is already in production but need rationalization to deal with chattering alarms and alarm floods. Off-delay timers of 300 seconds have been applied as a preprocessing step at the beginning of the analysis to remove chattering alarms. The reason for using off-delay rather than on-delay is because it does not introduce any delay when raising an alarm. Alarm floods were extracted based on the ISA standard: 10 alarms per 10 minutes per operator. General descriptions of the dataset are shown in Table 1.

Table 1
Statistics 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

Table 2
Information of the alarm sequences in each selected cluster

Cluster	A	B	C	D
Number of alarm floods	19	12	9	17
Average sequence length	26.6	14.6	31.6	20.1
Longest sequence length	46	23	64	36
Shortest sequence length	15	10	14	10
Pattern length	14	4	11	2

The flowchart of the algorithm is shown in Fig. 11. The parameters were chosen as: $\sigma = 0.2$, $\mu = -1$, $\delta = -1$, and the upper limit iteration number for random-division refinement is set as 50. All the tests were carried out on a 64 bit Windows PC with Intel i7-4770 3.40GHz CPU and 24.0 GB memory.

Pairwise similarity scores were calculated using the method in [37] and the UPGMA clustering was conducted based on the obtained similarity scores thereafter. Similar to the proposed approach, the method in [37] finds the similarity score between a pair of sequences by searching for their best alignment. It is able to find the exact optimal alignment; however, it can only be applied on a pair of sequences rather than multiple ones. The clustering result is shown in Fig. 12, where each pixel represents the similarity score between two corresponding alarm flood sequences; darker color means a higher similarity. Four groups of alarm floods were selected to be the test dataset, denoted as clusters A, B, C, and D. The reason for choosing them as the test datasets was because they each contains sufficient number of alarm floods for testing, and a pattern of alarm sequence exists in each of the clusters. Detailed information about the four selected clusters can be found in Table 2.

Both accuracy and efficiency tests have been conducted to compare the proposed algorithm with the exhaustive search approach in [39]. Same parameters were set for both algorithms.

3.1. Accuracy tests

For each of the four clusters, 3 sequences were randomly selected, and the two algorithms were applied respectively to find the best alignment. Since the exhaustive search approach guarantees the global optimum, the accuracy of the proposed algorithm can be evaluated by the alignment score difference between the two algorithms. Repeat the whole procedure for 50 times and obtain the average alignment score difference for each cluster; the result is shown by the green line in Fig. 13.

From the left to the right, the four points on the green line indicates the average alignment score difference between the two algorithms in clusters B, D, A, and C respectively (in the ascendant order of the average sequence length in each cluster). Similarly, the average alignment score difference between the two algorithms was obtained when 4 and 5 sequences were randomly selected

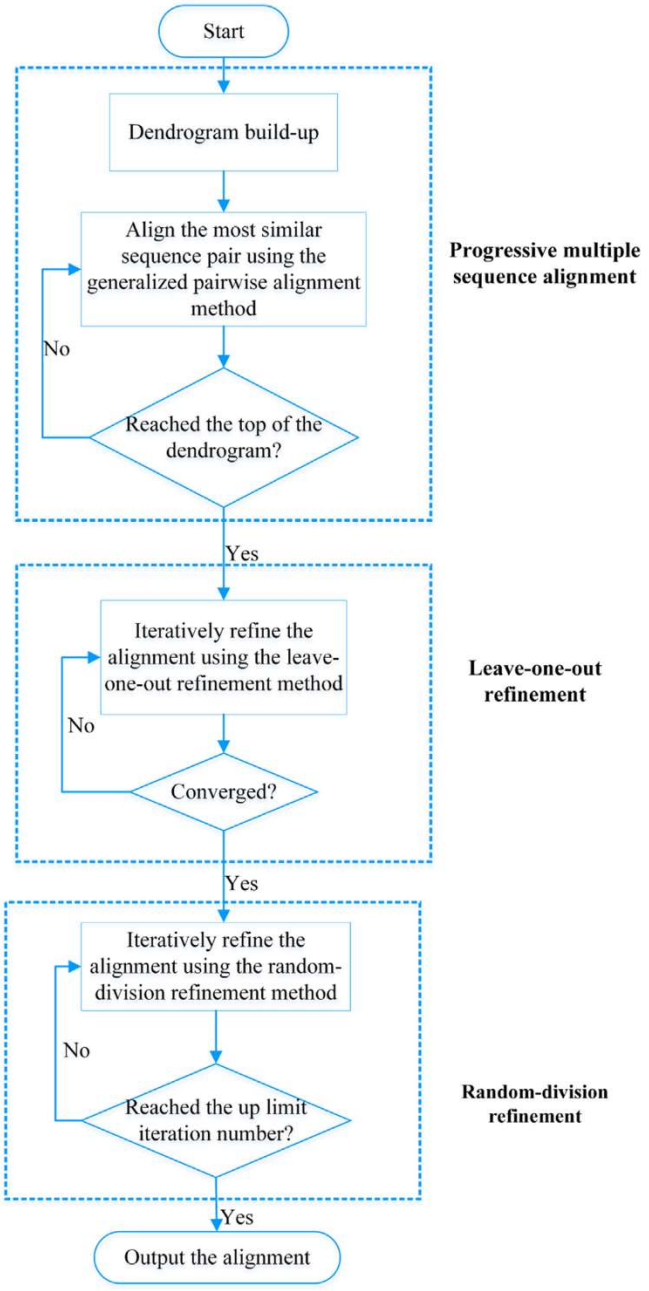

Fig. 11. Flowchart of the proposed algorithm.

from each of the four clusters, presented by the red and blue lines respectively.

A very noticeable spike can be found in all the three lines in Fig. 13, showing that the average alignment score difference between the two algorithms was large for the tests in cluster D. The reason for this is because the pattern length in cluster D was too short (only 2). Thus, the alignment can be easily influenced by the similar terms among the sequences that are not related to the pattern. Also, since the pattern length is too short, the percentage of alignment score difference caused by a small defect in the alignment would be amplified. In addition to the big spike caused by cluster D, the accuracy of the proposed algorithm is high, with no more than 10% difference compared to the exhaustive search approach. Fig. 13 also shows that there is no evident relation between the accuracy of the proposed algorithm and number of sequences to be aligned or the average sequence length.

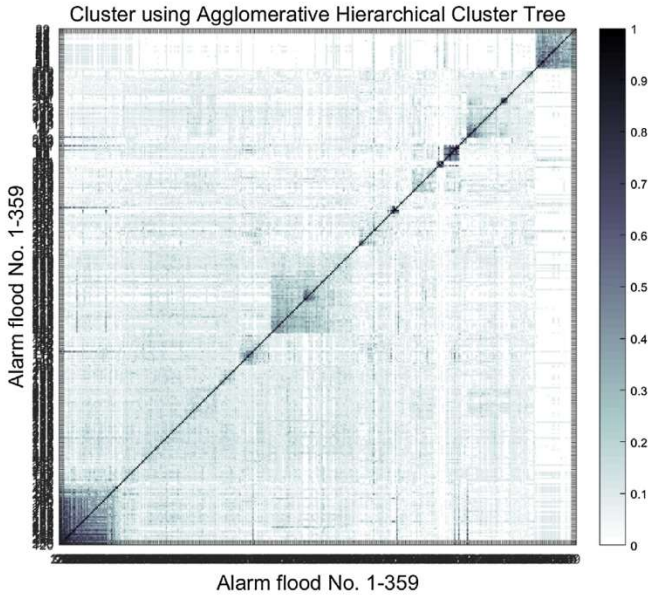

Fig. 12. Clustering result of the extracted alarm floods and the four selected clusters.

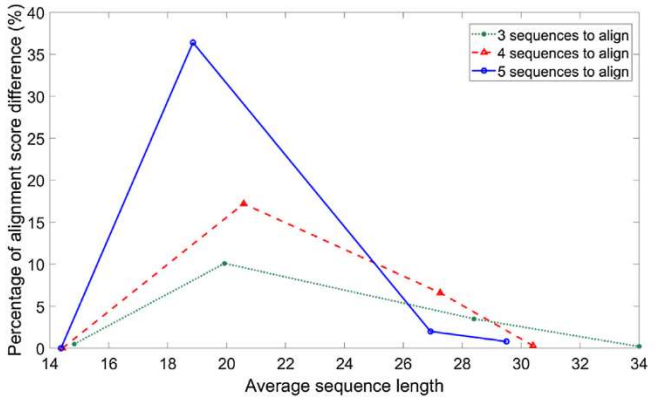

Fig. 13. Percentage of alignment score difference between the proposed algorithm and the exhaustive search approach.

3.2. *Efficiency tests*

Execution times of both algorithms during the accuracy tests were recorded to compare their efficiency.

3.2.1. *Efficiency comparison of two algorithms*

Fig. 14 shows the efficiency of the two algorithms as the number of sequences to be aligned grows. The execution times of both algorithms increases when the number of sequences grows; however, the growth rate of the proposed algorithm is much smaller than that of the exhaustive search approach. When the number of sequences reached 5, the proposed algorithm was almost 10,000 times faster than the exhaustive search approach.

Fig. 15 indicates there is a positive relation between the computational cost of the exhaustive search approach and the average sequence length, but this trend is not evident for the proposed algorithm. The reason is because the pattern length, as numbered on the figure, influences the execution time of the proposed algorithm as well. Detailed explanation will be provided in the discussion section.

3.2.2. *Execution time analysis of the proposed algorithm*

More efficiency tests were conducted to cover the testing region that had not been touched by the accuracy tests. Since the exhaus-

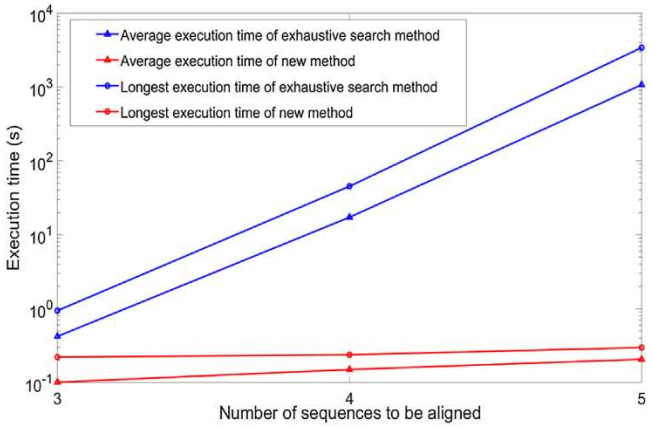

Fig. 14. Efficiency comparison of the two algorithms when the number of sequences increases.

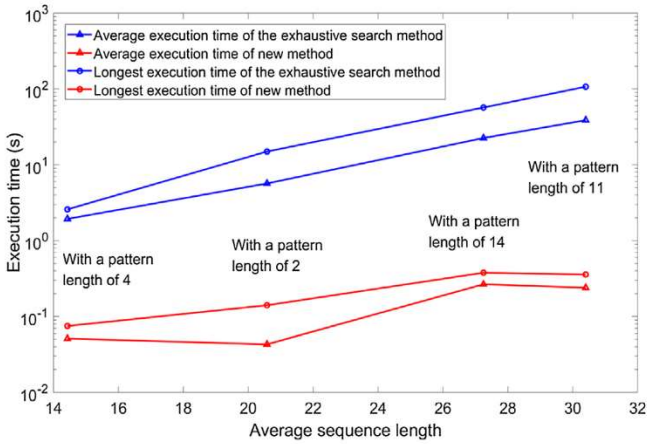

Fig. 15. Efficiency comparison of the two algorithms when the average sequence length increases.

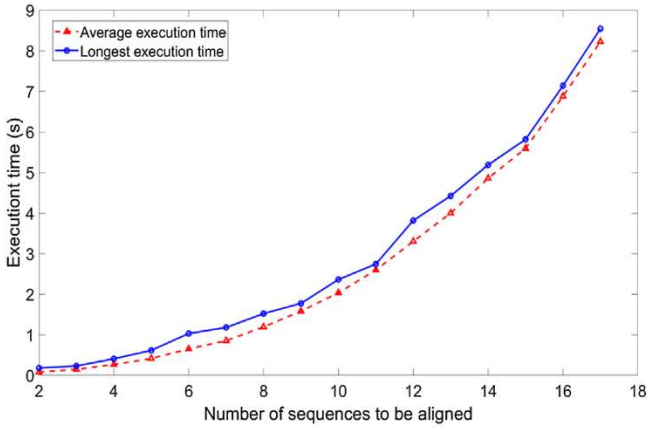

Fig. 16. Execution time of the proposed algorithm when the number of sequences grows.

tive search approach could hardly finish in most of the tests, only the result of the proposed algorithm is shown.

Fig. 16 shows the execution time of the proposed algorithm when the number of sequences to be aligned increases. The tests were conducted by randomly selecting sequences from cluster A (average sequence length 26.6); 50 repeated tests were carried out for each number of sequences. The result reveals a quadratic trend in the computational cost as the number of sequences to be aligned

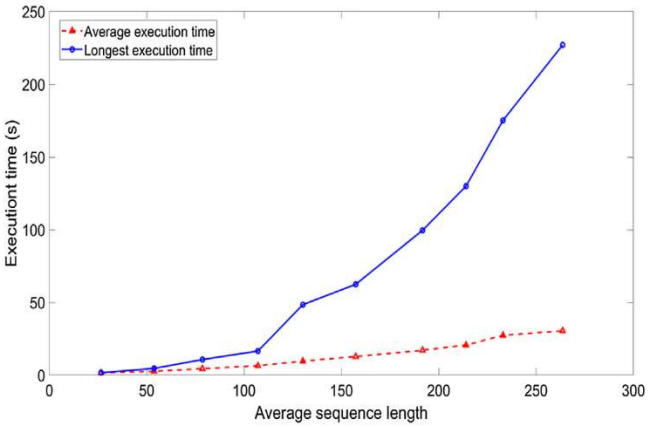

Fig. 17. Execution time of the proposed algorithm to align 10 sequences as the average sequence length grows.

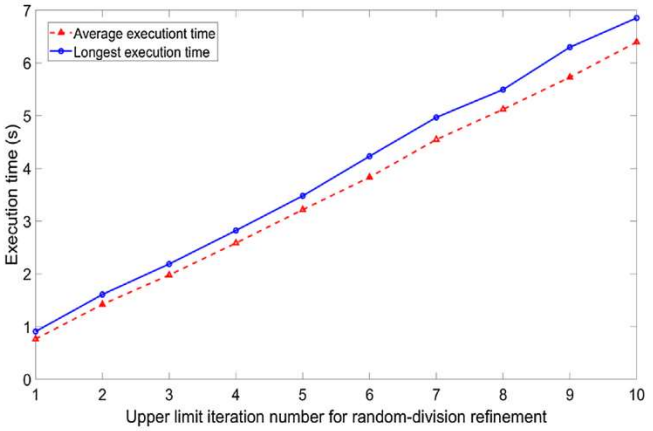

Fig. 18. Execution time of the algorithm with respect to the upper limit iteration number for the random-division refinement.

grows. Even when the sequence number exceeds 250, as shown in Fig. 17, the execution time of the proposed algorithm was still within 1 minute.

Fig. 17 reveals the relationship between execution time of the algorithm and average sequence length. The tests were carried out by applying the proposed algorithm to align 10 alarm sequences randomly selected from cluster A. In order to increase the sequence length in the each test, the sequences were extended by duplicating and connecting themselves with their duplicates. Each test was repeated 50 times and both the average and the longest execution time were recorded. The result shows the longest execution time grows quite fast with the growth of average sequence length; however, the average execution time grows much slower and is always less than 50 seconds. The reason is because the sequence length in cluster A varies from 15 to 46 and thus the sequence growth rates were quite different when the sequences were extended by duplicating themselves.

Fig. 18 shows the influence of the upper limit of iteration number for the random-division refinement on the execution time of the algorithm. The results were obtained by aligning 10 sequences randomly selected from cluster A, with different upper limits of iteration numbers for the random-division refinement in the algorithm. Each test was repeated 50 times and both the average and the longest execution times were recorded. The result shows a linear increasing trend of the execution time of the algorithm when the upper limit increases, as is expected. The detailed reason will be explained in the discussion section.

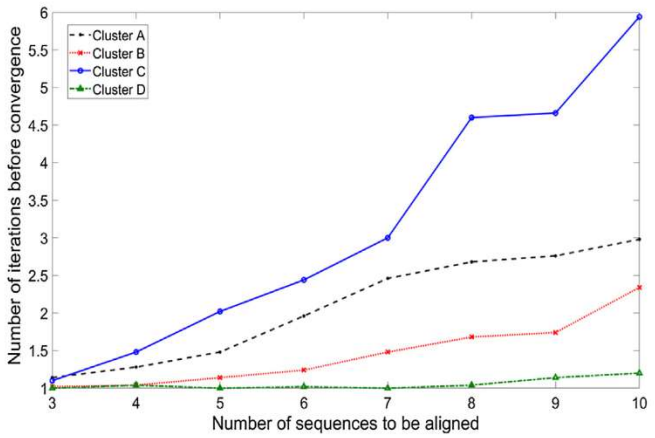

Fig. 19. The number of refinement iterations needed before reaching the convergence when the number of sequences to be aligned increases.

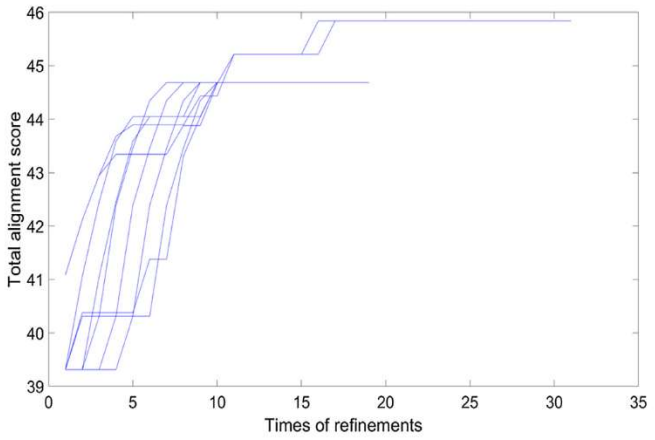

Fig. 20. An example of convergence curves when the order of sequences to be left out for the leave-one-out refinement method varies.

3.3. Convergence tests

3.3.1. Convergence analysis of the leave-one-out refinement

Since the convergence speed of the random-division refinement is much slower than the leave-one-out refinement and can hardly be identified, only the convergence speed of the leave-one-out refinement was tested. Fig. 19 shows the refinement iteration number needed before the convergence was reached by leave-one-out refinement when aligning different numbers of sequences in each cluster. As the result shows, more iterations were needed when the number of sequences to be aligned was large. However, there is no clear relationship between the refinement iteration number needed for convergence and the average sequence length, as the descending order of average sequence length in each cluster is $C > A > D > B$.

When applying the leave-one-out refinement method, for each iteration we always select the sequence by their orders in the primitive alignment. However, will the order we select the sequences to be left out make a difference in the final alignment given by the leave-one-out refinement method? The answer is yes, as the result in Fig. 20 shows, where 10 different orders were tried to improve a certain primitive alignment using the leave-one-out refinement method. The result shows the order we use to select the sequences will influence not only the convergence value but also the speed for the leave-one-out refinement method.

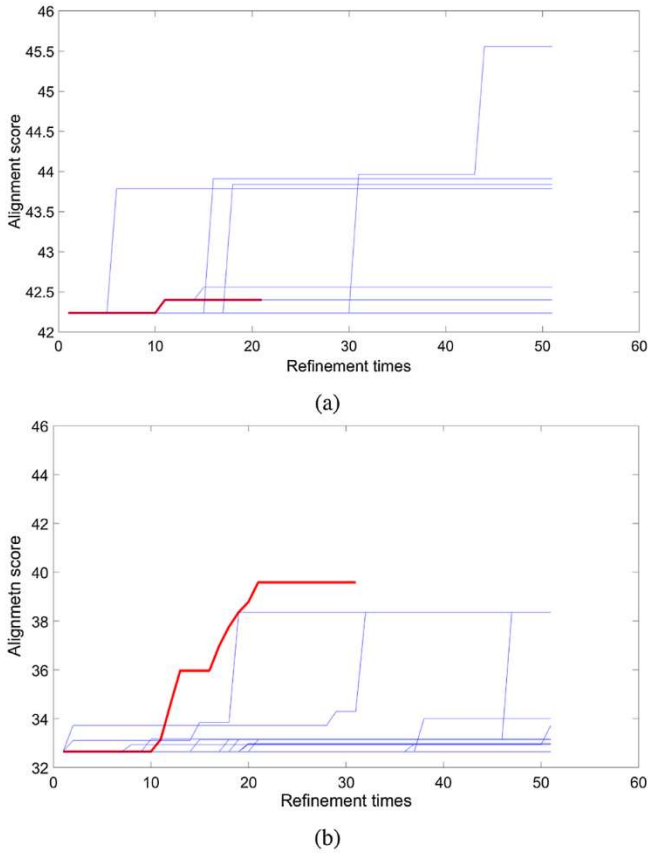

Fig. 21. Examples of the optimums achieved by the two types of refinement methods: (a) when the random-division refinement method was better; (b) when the leave-one-out refinement method was better.

3.3.2. Comparison of two refinement algorithms

The two refinement methods were also compared by evaluating the amount of improvement they can make on a given primitive alignment. Since for a certain primitive alignment, the leave-one-out refinement always has the same convergence trail, while the random-division method doesn't, the final alignment score of the leave-one-out refinement was compared with the results from 30 runs of the random-division refinement.

Two cases where the two methods achieved a higher alignment score compared with each other are shown in Fig. 21. The red bold lines represent the converging trails of the leave-one-out method, while the blue lines show the trails of the random division method. In case (a), the leave-one-out refinement was stuck in a local optimum and outperformed by most of the runs of the random division refinement. However, in case (b), the leave-one-out refinement was better than all runs using the random-division refinement. The reason for this was because the leave-one-out refinement method could recover from the incorrectly deleted part for the primitive alignment, but the random-division refinement method could not. Thus, during the procedure of forming the primitive alignment, if a part of the sequence that should appear in the optimal alignment was deleted incorrectly because of the approximation, then we should expect it to be recovered by the leave-one-out refinement method rather than the random-division refinement.

4. Discussions

4.1. Accuracy discussion

Compared to the exhaustive search approach in [39], the proposed algorithm does not guarantee the global optimum. However,

as shown in the industrial case study section, the accuracy of the proposed algorithm is acceptable (within 10% range) when the pattern length is not too small (larger than 3). For the cases where the pattern sequence is too short, the algorithm may not perform well since the alignment can be easily influenced by the similar terms among the sequences that are not related to the pattern, and the improvement on the primitive alignment that could be made by the two refinement methods will be very limited as well.

We need to point out that there is a trade-off between algorithm's efficiency and accuracy. Our algorithm does not guarantee to find the optimal solution like the method in [39] does. However, based on our tests, the accuracy of the proposed algorithm is still suitable for applications, averaging 5% difference when compared to the results given by the method in [39]. The efficiency of the proposed algorithm is higher than the method in [39] by several orders of magnitude. Thus, given a large scale dataset, the proposed algorithm is able to complete fast with usable results while the method in [39] may stuck for hours or even days.

4.2. Computational complexity discussion

The computational complexity of the proposed algorithm is composed of five parts: (1) calculation of time weight matrices $\mathcal{O}(\sum_{i=1}^{N} L_i^2)$, where N is the sequence number and L_i is the sequence length; (2) formation of the dendrogram $\mathcal{O}(N^3)$; (3) generation of the primitive alignment using progressive multiple sequence alignment method $\mathcal{O}(\sum_{j=1}^{N-1} A_{j1}A_{j2})$, where A_{j1} and A_{j2} are the lengths of the two sequences (can be either a sequence or an alignment) for the generalized pairwise sequence alignment; (4) leave-one-out refinement $\mathcal{O}(\sum_{k=1}^{I_1} A_{k1}A_{k2})$, where I_1 is the number of iterations before convergence is reached, and A_{k1} and A_{k2} are the lengths of the two sequences (can be either a sequence or an alignment) for the generalized pairwise sequence alignment; and (5) random-division refinement $\mathcal{O}(\sum_{t=1}^{I_2} A_{t1}A_{t2})$, where I_2 is the upper limit iteration number, and A_{t1} and A_{t2} are the lengths of the two sequences (can be either a sequence or an alignment) for the generalized pairwise sequence alignment. Thus the whole computational complexity is

$$\mathcal{O}\left(\sum_{i=1}^{N} L_i^2 + N^3 + \sum_{j=1}^{N-1} A_{j1}A_{j2} + \sum_{k=1}^{I_1} A_{k1}A_{k2} + \sum_{t=1}^{I_2} A_{t1}A_{t2}\right).$$

Based on the computational complexity analysis, the efficiency of the proposed algorithm is determined by many factors, among which the number of sequences, sequence length, and alignment length are the three main elements. The highest order of sequence or pattern length in the computational complexity is only quadratic for the proposed algorithm. Notice the computational complexity of the exhaustive search approach in [39] is

$$\mathcal{O}((\sum_{k=1}^{N} L_k) \times (\prod_{k=1}^{N} L_k)),$$

which increases exponentially with the sequence number and length. Thus, the proposed algorithm achieves a significant improvement in efficiency, which has also been confirmed in the industrial case study section.

4.3. Convergence discussion

The convergence can be guaranteed by both refinement methods. The proof is simple and does not require any mathematical derivations. Since in every refinement iteration, the new alignment score will be compared with the old one, and the alignment with a higher score will be fed to the next iteration. Thus, the alignment score is monotonically increasing, while there must be an upper bound for the alignment score of certain sequences. Therefore, the convergence is guaranteed for both of the refinement methods. The leave-one-out refinement method converges when no more improvement can be made to the alignment for a full round of iterations (every sequence has been left out once). However, the convergence of the random-division refinement can hardly be identified, as there does not exist a full round of iterations. Thus, an upper limit number of iterations should be configured for the random-division refinement method.

Convergence speeds of the two refinement methods are worthy studying too. The leave-one-out refinement method usually, but not always, converges faster than the random-division method. The reason is because for both of the refinement methods, there are two main steps: separate the alignment into two bunches of sequences, and re-align them. The leave-one-out refinement method has a certain order to separate the alignment, and only one sequence is separated from the rest during one iteration. While the random-division refinement method does not; thus there is a much bigger pool of ways of separating the sequences in the alignment that the random-division refinement method can select from.

Regarding the convergence value, both refinement methods cannot guarantee the global optimum, and neither of them can always achieve a better alignment score than the other. However, by combining them together we could have a better chance to reach the global optimum, because each method improves the alignment from a different perspective. The leave-one-out refinement method can recover the part of the sequence that has been incorrectly deleted during the generation of the primitive alignment because in each iteration, one sequence in the alignment is replaced by its original sequence and re-aligned with the rest of the sequences in the alignment. However, the random-division refinement method cannot do so because no separated sequences will be replaced by their original ones. Even though it cannot recover the incorrectly deleted sequence part, it still has an advantage over the leave-one-out refinement method — a much bigger pool of ways of separating the sequences in the alignment to choose from, which means more chances to improve the alignment by re-arranging the gaps and matchings. Thus, by applying the leave-one-out refinement followed by the random-division refinement method, we will be able to first recover the incorrectly deleted sequence parts, and then modify the ways of placing gaps and matchings in the existing alignment to achieve a better alignment score. This is why in the algorithm flowchart shown in Fig. 11, the leave-one-out refinement is applied before the random-division refinement.

4.4. Parameter tuning discussion

Parameter tuning can be an obstacle for applying the proposed algorithm. As mentioned in [37], the variance of the Gaussian function σ, the gap and mismatch penalties δ and μ need to be tuned to meet users' requirements. The variance of the Gaussian function σ controls the size of the time span within which the algorithm blurs the occurrence orders between the alarm messages. When the value of σ goes to infinity, the algorithm totally ignores the orders of alarms in the alignment and simply counts the alarm occurrences. On the contrary, if $\sigma = 0$, the orders of alarms have to be exactly the same in the two sequences in order to get a match. The gap and mis-match penalties, σ and μ, determine the algorithm's tolerance to including gaps and mis-match terms in alignments. When these two parameters get larger, the algorithm places more tolerance on the irrelevant alarms raised within a pattern sequence. Besides the ones in [37], a new parameter is introduced in the proposed algorithm: the upper limit number of iterations I_2 for the random-division refinement. With a higher I_2, there will be a

greater chance that the global optimum is reached, but at the cost of higher computational cost.

During the generation of the primitive alignment, the sequences are progressively aligned by the descending order of their pairwise similarity. The reason for choosing this order is because once an alignment is obtained, the gaps and the matchings between the two sequences are fixed and will be propagated into the primitive alignment. Thus, by aligning the most similar sequence pair first and leaving the most divergent pair to the last would improve the correctness of placing gaps and matchings in the alignment.

5. Conclusions

An accelerated multiple sequence alignment algorithm for pattern mining in multiple alarm sequences has been proposed. A progressive multiple sequence alignment mechanism has been introduced to accelerate the generation of the primitive alignment. Two types of refinement methods have been developed to improve the primitive alignment. A dataset from a real chemical plant has been used to test the effectiveness of the proposed algorithm and compare it with the exhaustive search approach in [39]. The results show the proposed algorithm has a significant improvement in efficiency with a small accuracy cost. Based on this work, we can provide early prediction for an incoming alarm flood by matching an online alarm sequence with a pattern database and conducting similarity calculation. The discovered patterns are also helpful in alarm management and causality analysis. In the future, techniques to improve the accuracy and methods for parameter tuning may be studied.

Acknowledgments

This work was supported by NSERC and the National Natural Science Foundation of China (Grant No. 61433001 and 61873142). We would like to thank Suncor for providing the data in the industrial case study.

References

[1] ISA 18.2, Management of Alarm Systems for the Process Industries, The International Society of Automation, Research Triangle Park, 2016.
[2] C.R. Dal Vernon, J.L. Downs, D. Bayn, 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, SAGE Publications, 2004, pp. 1189–1193.
[3] EEMUA, Alarm Systems — A Guide to Design, Management and Procurement, Engineering Equipment and Materials Users Association, London, 2013.
[4] IEC 62682, Management of Alarm Systems for the Process Industries, International Electrotechnical Commission, 2014.
[5] C. Timms, Hazards equal trips or alarms or both, Process Saf. Environ. Protect. 87 (1) (2009) 3–13.
[6] S.R. Kondaveeti, I. Izadi, S.L. Shah, T. Chen, On the use of delay timers and latches for efficient alarm design, in: 19th Mediterranean Conference on Control & Automation, IEEE, 2011, pp. 970–975.
[7] S.R. Kondaveeti, I. Izadi, S.L. Shah, D.S. Shook, R. Kadali, T. Chen, Quantification of alarm chatter based on run length distributions, Chem. Eng. Res. Design 91 (12) (2013) 2550–2558.
[8] E. Naghoosi, I. Izadi, T. Chen, A study on the relation between alarm deadbands and optimal alarm limits, in: Proceedings of 2011 American Control Conference, IEEE, 2011, pp. 3627–3632.
[9] F. Yang, S. Shah, D. Xiao, Signed directed graph based modeling and its validation from process knowledge and process data, Int. J. Appl. Math. Comput. Sci. 22 (1) (2012) 41–53.
[10] P. Duan, F. Yang, S.L. Shah, T. Chen, Direct causality detection via the transfer entropy approach, IEEE Trans. Control Syst. Technol. 21 (6) (2013) 2052–2066.
[11] P. Duan, F. Yang, S.L. Shah, T. Chen, Transfer zero-entropy and its application for capturing cause and effect relationship between variables, IEEE Trans. Control Syst. Technol. 23 (3) (2015) 855–867.
[12] F. Yang, S.L. Shah, D. Xiao, T. Chen, Improved correlation analysis and visualization of industrial alarm data, ISA Trans. 51 (4) (2012) 499–506.
[13] J. Wang, F. Yang, T. Chen, S.L. Shah, An overview of industrial alarm systems: main causes for alarm overloading, research status, and open problems, IEEE Trans. Autom. Sci. Eng. 13 (2) (2016) 1045–1061.
[14] N. Ahmed Adnan, I. Izadi, T. Chen, On expected detectiondelays for alarm systems with deadbands and delay-timers, J. Process Control 21 (9) (2011) 1318–1331.
[15] S. Lai, F. Yang, T. Chen, Online pattern matching and prediction of incoming alarm floods, J. Process Control 56 (2017) 69–78.
[16] R. Agrawal, R. Srikant, Mining sequential patterns, in: Proceedings of the 11th International Conference on Data Engineering, IEEE, 1995, pp. 3–14.
[17] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, ACM SIGMOD Record, no. 2 (2000) 1–12.
[18] M.J. Zaki, Efficient enumeration of frequent sequences, in: Proceedings of the seventh International Conference on Information and Knowledge Management, ACM, 1998, pp. 68–75.
[19] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu, Freespan: frequent pattern-projected sequential pattern mining, in: Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining, ACM, 2000, pp. 355–359.
[20] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu, PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth, in: Proceedings of 29th International Conference on Data Engineering, IEEE, 2001, pp. 215–224.
[21] S.B. Needleman, C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, J. Mol. Biol. 48 (3) (1970) 443–453.
[22] T.F. Smith, M.S. Waterman, Identification of common molecular subsequences, J. Mol. Biol. 147 (1) (1981) 195–197.
[23] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, Basic local alignment search tool, J. Mol. Biol. 215 (3) (1990) 403–410.
[24] W.R. Pearson, D.J. Lipman, Improved tools for biological sequence comparison, Proc. Natl. Acad. Sci. 85 (8) (1988) 2444–2448.
[25] M.S. Johnson, R.F. Doolittle, A method for the simultaneous alignment of three or more amino acid sequences, J. Mol. Evol. 23 (3) (1986) 267–278.
[26] D.-F. Feng, R.F. Doolittle, Progressive sequence alignment as a prerequisitetto correct phylogenetic trees, J. Mol. Evol. 25 (4) (1987) 351–360.
[27] J.D. Thompson, D.G. Higgins, T.J. Gibson, 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) (1994) 4673–4680.
[28] S.R. Eddy, Multiple alignment using hidden Markov models, Proceedings of International Conference on Intelligent Systems for Molecular Biology (1995) 114–120.
[29] S.R. Eddy, Profile hidden Markov models, Bioinformatics 14 (9) (1998) 755–763.
[30] S. Kordic, P. Lam, J. Xiao, H. Li, Analysis of alarm sequences in a chemical plant, in: Lecture Notes in Computer Science, Springer, 2008, pp. 135–146.
[31] K. Ahmed, I. Izadi, T. Chen, D. Joe, T. Burton, Similarity analysis of industrial alarm flood data, IEEE Trans. Automat. Sci. Eng. 10 (2) (2013) 452–457.
[32] J. Folmer, B. Vogel-Heuser, Computing dependent industrial alarms for alarm flood reduction, in: Proceedings of 9th International Multi-Conference on Systems, Signals and Devices, IEEE, 2012, pp. 1–6.
[33] S. Charbonnier, N. Bouchair, P. Gayet, Fault template extraction to assist operators during industrial alarm floods, Eng. Appl. Artif. Intel. 50 (2016) 32–44.
[34] S. Charbonnier, N. Bouchair, P. Gayet, A weighted dissimilarity index to isolate faults during alarm floods, Control Eng. Pract. 45 (2015) 110–122.
[35] E.N. Satuf, E. Kaszkurewicz, R. Schirru, M.C.M.M. de Campos, Situation awareness measurement of an ecological interface designed to operator support during alarm floods, Int. J. Ind. Ergon. 53 (2016) 179–192.
[36] P. Cisar, E. Hostalkova, P. Stluka, Alarm rationalization support via correlation analysis of alarm history, in: Proceedings of 19th International Congress of Chemical and Process Engineering, Prague, Czech Republic, 2010, pp. 1–6.
[37] Y. Cheng, I. Izadi, T. Chen, Pattern matching of alarm flood sequences by a modified Smith-Waterman algorithm, Chem. Eng. Res. Des. 91 (6) (2013) 1085–1094.
[38] W. Hu, J. Wang, T. Chen, A local alignment approach to similarity analysis of industrial alarm flood sequences, Control Eng. Pract. 55 (2016) 13–25.
[39] S. Lai, T. Chen, A method for pattern mining in multiple alarm flood sequences, Chem. Eng. Res. Des. 117 (2017) 831–839.
[40] IEC 61511, Functional Safety: Safety Instrumented Systems for the Process Industry Sector, International Electrotechnical Commission, 2015.