# Trace Alignment in Process Mining: Opportunities for Process Diagnostics

Rantham Prabhakara Jagadeesh Chandra Bose[1,2] and Wil van der Aalst[1]

[1] Department of Mathematics and Computer Science, University of Technology, Eindhoven, The Netherlands
[2] Philips Healthcare, Veenpluis 5-6, Best, The Netherlands
`j.c.b.rantham.prabhakara@tue.nl, w.m.p.v.d.aalst@tue.nl`

**Abstract.** Process mining techniques attempt to extract non-trivial knowledge and interesting insights from event logs. Process mining provides a welcome extension of the repertoire of business process analysis techniques and has been adopted in various commercial BPM systems (BPM|one, Futura Reflect, ARIS PPM, Fujitsu, etc.). Unfortunately, traditional process discovery algorithms have problems dealing with less-structured processes. The resulting models are difficult to comprehend or even misleading. Therefore, we propose a new approach based on *trace alignment*. The goal is to align traces in a way that event logs can be explored easily. Trace alignment can be used in a preprocessing phase where the event log is investigated or filtered and in later phases where detailed questions need to be answered. Hence, it complements existing process mining techniques focusing on discovery and conformance checking.

## 1 Introduction

Many of today's information systems are recording an abundance of event logs. Process mining techniques attempt to extract non-trivial knowledge and interesting insights from these event logs and to exploit these for further analysis [1]. Process mining techniques aim at discovering process, control, data, organizational and social structures from event logs. The majority of research in process mining so far has focussed on process discovery (both from a control-flow and organizational perspective). One of the challenging topics in process mining is *process diagnostics*. Process diagnostics encompasses process performance analysis, anomaly detection, diagnosis, inspection of interesting patterns and the like. Research so far in diagnosing processes is limited to exploring ways and means of analyzing process models (such as conformance checking), projecting diagnostic information on these models or in dashboard like approaches over some (performance) metrics. Diagnostics of processes at the model level is cumbersome, tedious and sometimes infeasible, especially when dealing with real-life and flexible processes. We have applied process mining in more than 100 organizations and our experiences show that processes tend to be less structured than expected. Traditional process mining algorithms have problems dealing with such unstructured processes and generate spaghetti-like process models that are hard to comprehend. Such incomprehensible models are not amenable or are found lacking to assist in process diagnostic efforts. When diagnosing processes, a business analyst is confronted with lots of interesting questions. We list some of them below:

1. *What is the most common (likely) process behavior that is executed?* Given a bag of traces from a process, it would be interesting to know which process components are essential/critical for this process. Such essential components/functions form the backbone of the process and should be conserved. Process re-design/improvement efforts should focus on improving such critical components.
2. *Where do my process instances deviate and what do they have in common?:* In practice, there is often a significant gap between what is prescribed or supposed to happen, and what actually happens. There is a need to augment process diagnostics with techniques that can assist in finding deviations by analyzing raw traces in the event logs. There are many domains/applications where this requirement is felt. Fault diagnosis, anomaly detection, diagnosis of fraudulent insurance claims are some of the applications. Given an event log containing a mix of traces where the system process functioned normally and where it malfunctioned, analyzing these traces to find deviations in malfunctioned/anomalous traces from normal traces would give cues in understanding the cause of malfunction/anomaly.
3. *Are there any common patterns of execution in my traces?:* Analyzing logs at the granularity of an individual event might not always be result yielding as one often loses the context information during such analysis. An analyst would be interested in knowing whether there are any interesting execution (behavioral) patterns in the log. The absence or presence of such patterns may indicate the cause of an anomaly (say for e.g., fraudulent insurance claim) or a security violation or a malfunction.
4. *What are the contexts in which an activity or a set of activities are executed in my event log?:* Dependencies exist between activities in a process and activity executions are expected to happen within certain contexts. There can be short-range and long-range dependencies between activities. Long-range dependencies are difficult to discover. An analyst would be interested in understanding the contexts of execution of activities and/or activity sequences.
5. *What are the process instances that share/capture a desired behavior either exactly or approximately?:* Often in diagnostics, an analyst would be interested in finding process instances that share/comply to a particular desired behavior; The desired behavior can be represented as a manifestation of some pattern of activity sequences or some complex form (combination) of these patterns. Though temporal logic approaches can assist in addressing this problem to a certain extent by discovering process instances that capture the desired behavior exactly, one might also be interested in discovering process instances that share the desired behavior approximately.
6. *Are there particular patterns (e.g., milestones, concurrent activities etc.) in my process?:* Workflow patterns refer to recurring forms/structures addressing business requirements. For example, milestones indicate specific execution points in the process model and provide a mechanism for supporting the

conditional execution of a task or sub-process. An analyst would be interested in discovering the presence of, and in analyzing milestone patterns in the process event log. Discovery of process models with concurrency is one of the challenging problems in process mining. The presence of concurrent activities creates different permutations of activities in the event log that adds to the complexity of discovery algorithms. Detection of the presence of concurrent activities might also help in pre-processing the logs.

In this paper, taking inspiration from biological sequence alignment [2], we propose a novel approach, called *trace alignment*, of aligning traces in an event log and show the promise of such an approach in process diagnostics addressing some of the questions enumerated above. Multiple sequence alignment is a topic of extensive research of over three decades in computational biology and still remains intriguing due to the intricate challenges it poses. *There are significant challenges in adopting them to trace alignment.* We highlight some of the challenges in this paper and believe that this will *open a new area of research within process mining.* Figure 1 illustrates the traditional dotted chart analysis and the proposed trace alignment[1]. It is apparent that the proposed approach of trace alignment uncovers common execution patterns and deviations in the log yielding better insights for analysis.

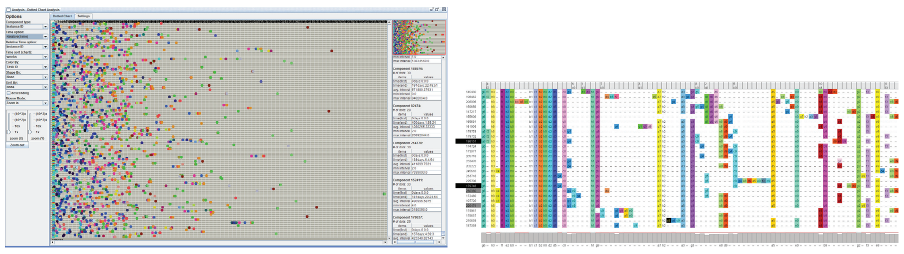


**Fig. 1.** Comparison of Dotted Chart Analysis and Trace Alignment

The remainder of this paper is organized as follows. In Section 2, we introduce the notations used in the paper. Section 3 introduces the concept of trace alignment and discusses the techniques for finding alignments. In Section 4, we propose a framework for finding alignments over a set of traces. In Section 5, we present and discuss the results of trace alignment on a synthetic log and a real-life log and show how trace alignment can assist in gaining better insights for process diagnostics. We discuss related work in Section 6. Finally, Section 7 concludes the paper.

## 2 Notations

– Let $\Sigma$ denote the set of activities. $|\Sigma|$ is the number of activities.

[1] In the dotted chart, a dot represents an activity and the x-axis represents time. In trace alignment, the x-axis represents the alignment position. y-axis represents trace indices for both the dotted chart and trace alignment.

- $\Sigma^+$ is the set of all non-empty finite sequences of activities from $\Sigma$. $T \in \Sigma^+$ is a trace over $\Sigma$. $|T|$ denotes the length of trace $T$.
- The set of all $n$-length sequences over the alphabet $\Sigma$ is denoted by $\Sigma^n$. A trace of length $n$ is denoted as $T^n$ i.e., $T^n \in \Sigma^n$, and $|T^n| = n$.
- The ordered sequence of activities in $T^n$ is denoted as $T(1)T(2)T(3)\ldots T(n)$ where $T(k)$ represents the $k^{th}$ activity in the trace.
- $T^{n-1}$ denotes the $n-1$ length prefix of $T^n$. In other words $T^n = T^{n-1}T(n)$.
- An event log, $\mathcal{L}$, corresponds to a multi-set (or bag) of traces from $\Sigma^+$.

## 3 Trace Alignment

In this section, we formally define what trace alignment is and discuss techniques for finding optimal alignments.

**Definition 1.** *Trace alignment over a set of traces $\mathbb{T} = \{T_1, T_2, \ldots, T_n\}$ is defined as a mapping of the set of traces in $\mathbb{T}$ to another set of traces $\overline{\mathbb{T}} = \{\overline{T_1}, \overline{T_2}, \ldots, \overline{T_n}\}$ where each $\overline{T_i} \in (\Sigma \cup \{-\})^+$ for $1 \leq i \leq n$ and*

- $|\overline{T_1}| = |\overline{T_2}| = \ldots = |\overline{T_n}| = m$,
- $\overline{T_i}$ *by removing all "$-$" gap symbols is equal to* $T_i$,
- $\nexists k, 1 \leq k \leq m$ *such that* $\forall_{1 \leq i \leq n}$, $\overline{T_i}(k) = -$

$m$ in the definition above is the length of the alignment. An alignment over a set of traces can be represented by a rectangular matrix $\mathcal{A} = \{a_{ij}\} (1 \leq i \leq n, 1 \leq j \leq m)$ over $\Sigma' = \Sigma \cup \{-\}$ where $-$ denotes a gap. The third condition in the definition above implies that no column in $\mathcal{A}$ contains only gaps ($-$). It is imperative to note that there can be many possible alignments for a given set of traces and that the length of the alignment, $m$, satisfies the relation $l_{max} \leq m \leq l_{sum}$ where $l_{max}$ is the maximum length of the traces in $\mathbb{T}$ and $l_{sum}$ is the sum of lengths of all traces in $\mathbb{T}$.

### 3.1 Pairwise Trace Alignment

Before we get into the details of aligning a set of traces, let us first consider a special case of trace alignment, where the number of traces to align is 2. Aligning a pair of traces is referred to as *pair-wise* trace alignment. Let us consider the example of aligning the two traces $T_1$ = `abcac` and $T_2$ = `acacad`. Figure 2 depicts three variants of aligning the two traces. In fact, the number of possible alignments for two traces of length $l$ is $\approx (1 + \sqrt{2})^{2l+1} l^{-1/2}$ [2], e.g., for two traces of length 100, the number of possible alignments is approximately $10^{77}$. Therefore, it is infeasible to enumerate all possible alignments even for moderate values of $l$. Moreover, not all of these alignments would be interesting. In order to compute "best" alignments, we need a means of associating a score to an alignment.

Alignment between a pair of traces, $T_1$ and $T_2$ can be considered as a transformation of the trace $T_1$ to $T_2$ or viceversa through a set of editing operations

| | (i) | | (ii) | | (iii) |
|---|---|---|---|---|---|
| $\overline{T_1}$ | a b c a c - - | $\overline{T_1}$ | a b c a c - | $\overline{T_1}$ | a b c a c - - - - - - |
| $\overline{T_2}$ | a - c a c a d | $\overline{T_2}$ | a c a c a d | $\overline{T_2}$ | - - - - - a c a c a d |

**Fig. 2.** An Example of Pair-wise Trace Alignments

applied to one of the traces iteratively. The traces are said to be aligned after the transformation, and can be represented as a rectangular matrix as mentioned earlier. Assuming that $\overline{T_1}$ is written over $\overline{T_2}$ in the alignment (as in Figure 2), the following edit operations are defined for any column $j$ in the alignment:

- the activity pair $(a, b)$, $a, b \in \Sigma$, denotes a substitution of activity $a$ in $T_1$ with activity $b$ in $T_2$,
- the activity pair $(a, -)$ denotes the deletion of activity $a$ in $T_1$, and
- the activity pair $(-, b)$ denotes the insertion of activity $b$ in $T_1$.

It is important to note that insertion and deletion operations are complementary in that an insertion in one trace can be considered as a deletion in another trace. Henceforth, we refer to insertion and deletion operations as *indel* operation. *indels* should be sensitive to the context in which the operations are performed. For example, it is ok to have an activity `fread` after `fopen` but not after `fclose`. Hence, we consider the *indel* operation as `indelRightGivenLeft` which indicates the insertion of an activity to the right of another activity. A score function needs to be defined for the substitution and indel operations. The substitution score is a function $S : \Sigma \times \Sigma \rightarrow \Re$ where $S(a, b)$ denotes the score for substitution of activity $a$ with activity $b$ for all $a, b \in \Sigma$. The `indelRightGivenLeft` score is a function $\mathcal{I}_l : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \rightarrow \Re$ where $\mathcal{I}_l(a, b)$ denotes the score for inserting activity $a$ given that the left activity is $b$ for all $a, b \in \Sigma$. $\mathcal{I}_l(a, -) = \mathcal{I}_l(-, a) = \mathcal{I}_l(-, -) = 0$ for all $a \in \Sigma$. Given $S$ and $\mathcal{I}_l$, *the score of a pair-wise alignment can be defined as the sum of the scores of the edit operations across all columns in the alignment.* In other words, if $\overline{T_1}$ and $\overline{T_2}$ are the aligned traces of $T_1$ and $T_2$, and the alignment is of length $m$, then:

$$Score(\overline{T_1}, \overline{T_2}) = \sum_{j=1}^{m} e_j$$

where

$$e_j = \begin{cases} S(a, b) & \text{if } \overline{T_1}(j) = a \text{ and } \overline{T_2}(j) = b \\ \mathcal{I}_l(a, b) & \begin{cases} \text{if } \overline{T_1}(j) = a, \overline{T_1}(j-1) = b \text{ and } \overline{T_2}(j) = - \text{ or} \\ \text{if } \overline{T_1}(j) = -, \overline{T_2}(j) = a \text{ and } \overline{T_2}(j-1) = b \end{cases} \end{cases}$$

$\overline{T_1}(0) = \overline{T_2}(0) = -$. Assuming a simple scoring function where a substitution of activity pair $(a, b)$ is associated with a score of 1 if $a = b$ and a score of $-1$ otherwise, and an indel scoring function, $\mathcal{I}_l(a, b) = -1$, for all $a, b \in \Sigma$, the alignments enumerated in Figure 2 have the scores 1, $-4$ and $-9$ respectively. A "best" alignment can be considered to be the one with the maximum score. It is imperative to note that the best scoring alignment is sensitive to the substitution and indel score functions.

**How to Compute Alignments.** Needleman and Wunsch [3] have proposed a dynamic programming algorithm for finding the optimal alignment between two amino acid sequences. The basic idea is to build up an optimal alignment using previous solutions for optimal alignments of smaller subsequences. Let $T_1$ and $T_2$ be two traces. A matrix $F$ indexed by $i$ and $j$, is constructed where the value $F(i,j)$ is the score of the best alignment between the prefix $T_1^i$ of $T_1$ and the prefix $T_2^j$ of $T_2$. $F(i,j)$ is constructed recursively by initializing $F(0,0) = 0$ and then proceeding to fill the matrix from top left to bottom right. It is possible to calculate $F(i,j)$ if $F(i-1,j-1)$, $F(i-1,j)$ and $F(i,j-1)$ are known. There are three possible ways that the best score $F(i,j)$ of an alignment up to $T_1^i$ and $T_2^j$ could be obtained: $T_1(i)$ could be aligned to $T_2(j)$, in which case $F(i,j) = F(i-1,j-1) + S(T_1(i), T_2(j))$; or $T_1(i)$ is aligned to a gap, in which case $F(i,j) = F(i-1,j) + \mathcal{I}_l(T_1(i), T_1(i-1))$; or $T_2(j)$ is aligned to a gap, in which case $F(i,j) = F(i,j-1) + \mathcal{I}_l(T_2(j), T_2(j-1))$. The best score up to $(i,j)$ will be the largest of these three options. In other words, we have

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + S(T_1(i), T_2(j)), \\ F(i-1,j) + \mathcal{I}_l(T_1(i), T_1(i-1)), \\ F(i,j-1) + \mathcal{I}_l(T_2(j), T_2(j-1)). \end{cases} \tag{1}$$

The values along the top row (when $i = 0$) and left column (when $j = 0$) need to be handled as follows. The values $F(i,0)$ represent alignments of a prefix of $T_1$ to all gaps in $T_2$. So, we can define $F(1,0) = 0$ and for $i > 1, F(i,0) = F(i-1,0) + \mathcal{I}_l(T_1(i), T_1(i-1))$. Similarly, we can define $F(0,j)$. The value in the bottom right cell of the matrix, $F(|T_1|, |T_2|)$, is the best score for an alignment of $T_1$ and $T_2$. To find the alignment itself, we must find the path of choices from (1) that led to this best score, i.e., we move from the current cell $(i,j)$ to one of the cells $(i-1,j-1)$, $(i-1,j)$ or $(i,j-1)$ from which the value $F(i,j)$ was derived. While doing so, we add a pair of symbols onto the front of the alignment: $T_1(i)$ and $T_2(j)$ if the step was to $(i-1,j-1)$, $T_1(i)$ and the gap symbol '$-$' if the step was to $(i-1,j)$, or '$-$' and $T_2(j)$ if the step was to $(i,j-1)$. At the end we will reach the start of the matrix, $i = j = 0$. The above procedure, called traceback, will retrieve only one of the alignments that gives the best score; there can be cases where multiple options of (1) are equal. In these cases, an arbitrary choice is made. The set of all possible alignments for the best score can be enumerated by using graph traversal techniques.

### 3.2 Multiple Trace Alignment

Having discussed the alignment of two traces, let us move on to the alignment of a set of traces. One of the most popular scoring mechanisms for multiple sequence alignment of genomic sequences is the *sum-of-pairs* (SP) method. We adopt the sum-of-pairs method for trace alignment as well. Let $\overline{T_j}$ and $\overline{T_k}$ be two distinct rows extracted from a multiple trace alignment $\mathcal{A}$ (over a set of set of $n$ traces), and let $Score(\overline{T_j}, \overline{T_k})$ be the alignment score calculated in the same way as ordinary pairwise alignment of $T_j$ and $T_k$, then the SP score of a multiple trace alignment $\mathcal{A}$ is defined as

$$Score_{SP}(\mathcal{A}) = \sum_{1 \leq j \leq k \leq n} Score(\overline{T_j}, \overline{T_k})$$

It is possible to generalize the pairwise dynamic programming alignment approach to the alignment of $n$ traces. However, it is impractical for more than a few traces. Assuming that the traces are all of roughly the same length $l$, the space complexity of the multidimensional dynamic programming algorithm is $\mathcal{O}(l^n)$ and the time complexity is $\mathcal{O}(2^n l^n)$ [4]. Multiple sequence alignment that maximizes the SP score was shown to be NP-complete [5].

We adopted the progressive alignment approach for trace alignment. The basic idea of progressive alignment is to iteratively construct a succession of pairwise alignments. Alignment is allowed between a pair of traces, a trace and an alignment and between alignments. The selection of traces for alignment at each iteration is based on their similarity. Traces that are most similar to each other are aligned first. Once similar traces have been aligned, align the resulting clusters of traces against each other. A guide tree is built to assist this process. We use the agglomerative hierarchical clustering algorithm (AHC) for generating this tree. We can use either distance metrics such as Euclidean distance or similarity measures for clustering. The choice of AHC is due to the fact that it produces the tree naturally as a dendrogram while the tree has to be constructed subsequently if other clustering algorithms such as $k$-means is used.

Figure 3 illustrates an example of the progressive alignment strategy. In this example, we consider 5 traces. A guide tree is generated using AHC. Based on the guide tree, the traces $T_2$ and $T_3$ would first be aligned using pairwise trace alignment. Next traces $T_4$ and $T_5$ would be aligned using pairwise trace alignment. Subsequently, trace $T_1$ is aligned with the alignment obtained from $T_2$ and $T_3$. Finally the two alignments obtained from the set of traces $\{T_1, T_2, T_3\}$ and $\{T_4, T_5\}$ are aligned.

While aligning an alignment $\mathcal{A}$, with another alignment $\mathcal{B}$, (1) is modified as

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + \overline{S}(C^i_{\mathcal{A}}, C^j_{\mathcal{B}}), \\ F(i-1, j) + \overline{\mathcal{I}_l}(C^i_{\mathcal{A}}, C^{i-1}_{\mathcal{A}}), \\ F(i, j-1) + \overline{\mathcal{I}_l}(C^j_{\mathcal{B}}, C^{j-1}_{\mathcal{B}}). \end{cases} \tag{2}$$

where $\overline{S}(C^i_{\mathcal{A}}, C^j_{\mathcal{B}})$ denotes the score of substituting column $i$ of alignment $\mathcal{A}$ with column $j$ of alignment $\mathcal{B}$ and is defined as

$$\overline{S}(C^i_{\mathcal{A}}, C^j_{\mathcal{B}}) = \sum_{\forall a,b \in \Sigma} n^i_{\mathcal{A}}(a).n^j_{\mathcal{B}}(b).S(a,b) \tag{3}$$

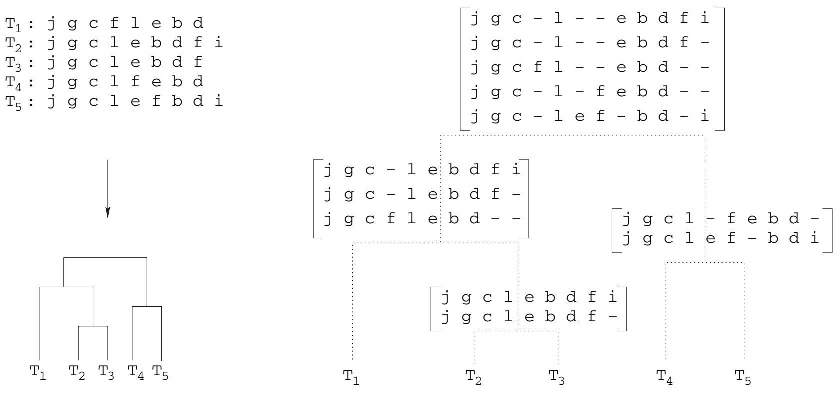


**Fig. 3.** Example of Progressive Alignment Approach for Multiple Trace Alignment

where $n^i_{\mathcal{X}}(a)$ denotes the frequency (count) of activity $a$ in column $i$ of alignment $\mathcal{X}$. $\overline{\mathcal{I}_l}(C^i_{\mathcal{A}}, C^{i-1}_{\mathcal{A}})$ denotes the score of inserting column $i$ in alignment $\mathcal{A}$ given that its left column is $i-1$ and is defined as

$$\overline{\mathcal{I}_l}(C^i_{\mathcal{A}}, C^{i-1}_{\mathcal{A}}) = \sum_{\forall a,b \in \Sigma} f^i_{\mathcal{A}}(a,b).\mathcal{I}_l(a,b) \tag{4}$$

where $f^i_{\mathcal{A}}(a,b)$ is the frequency of activity $a$ in column $i$ of alignment $\mathcal{A}$ given that its neighboring activity is $b$ in column $i-1$. The procedure for finding the "best" alignment is similar to that of pairwise alignment. Note that the guide tree enables the visualization of alignments for different subsets of the traces. The alignment at the root of the tree corresponds to the alignment of all the traces in the event log whereas an alignment at any internal node of the guide tree depicts the alignment corresponding to the traces constituting the leaves of the sub-tree at the node. It is often the case that event logs contain traces capturing different execution behavior of a process and clustering assists in grouping together a coherent set of traces.

## 4 Framework for Trace Alignment

We propose the framework depicted in Figure 4 for trace alignment. The framework identifies the following parts:

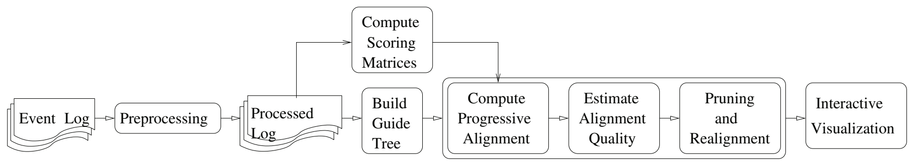


**Fig. 4.** Framework for Multiple Trace Alignment

– *Preprocessing:* Preprocessing involves steps such as removal of outliers, removal of loop-constructs, and encoding of log into character streams. The detection and removal of outliers is critical for obtaining interesting alignments.
– *Compute Scoring Matrices:* As discussed in Section 3, alignments are sensitive to the substitution and indel score functions, $S$ and $\mathcal{I}_l$ respectively. We use the approach presented in [6] for deriving the substitution and indel score functions from the event log.
– *Build Guide Tree:* A guide tree assists in progressive alignment of multiple traces as illustrated in Figure 3. We use the agglomerative hierarchical clustering (AHC) for building the guide tree. However, other approaches such as neighbor joining [7] can be used.
– *Estimate the Quality of Alignment:* Progressive alignment being a heuristics-based approach, the alignment that is obtained need not be optimal. Further any error in alignment done in early stages of progressive alignment cannot be undone. Hence it is essential to estimate the quality of an alignment. In this work, we adopt a metric based on the information score as a means for assessing the quality of an alignment. The information score of a column in an alignment is defined as $1-E/E_{max}$, where $E$ is the entropy of activities in the column[2] and $E_{max}$ is the maximum entropy which is equal to $\log_2(|\Sigma|+1)$.
– *Pruning and Realignment:* Construction of multiple trace alignment is a very complex problem, and most heuristic algorithms usually fail to generate an optimal alignment. Disturbances in an alignment can creep in from many sources thereby making the final alignment far from optimal. Disturbances here refer to the misplacement of gaps in an alignment. Efficient techniques for pruning and realigning alignments need to be supported. We will discuss more about this later in this section.
– *Interactive Visualization:* Apart from just pictorially depicting the alignment it is desirable to have additional interactive features for the analysts to explore into the patterns and the alignments uncovered. Features such as editing an alignment, sorting and/or filtering alignment columns based on activities of interest would all lead to gaining further insights into the execution of processes.

Though the definition of what constitutes an outlier is left open, in the current exploration, we have adopted one simple definition of outliers based on the length of the traces. It could be the case that in an event log there are certain process instances whose lengths deviate a lot from the average trace length in the log, e.g., one of the real life event logs that we analyzed had an average trace length of 47 activities (across 223 traces) while there were 5 traces with lengths above 250. Since an alignment is at least as long as the maximum trace length, such outlier traces in the log can lead to an alignment with too many gap symbols. Hence the removal of such traces is important. Note that the definition of outliers can change based on the perspective of analysis. If we are interested in finding common execution patterns or the backbone sequence of a process,

[2] The entropy of a column is defined as $E = \sum_{a\in\Sigma\cup\{-\}} -p_a \log_2(p_a)$ where $p_a$ is the probability of occurrence of $a$ in the column.

the above definition of outliers may work fine. However, if we are interested in finding non-conforming traces or deviations in anomalous traces from normal traces, then the above definition might not always be appropriate.

**Realigning alignments.** Variation in the lengths of the traces (mostly due to recurring patterns and loops), the choice of scoring matrices used, the method and parameter choices used in the generation of guide tree, strategies used in resolving conflicts during traceback can all lead to disturbances in the alignment. Furthermore, disturbances in earlier stages of progressive alignment strategy percolate to later stages. Detecting such disturbances and realigning them might vastly improve the quality of the final alignment. Figure 5 depicts an example of an alignment before and after realignment. For the original alignment, the information score for the columns 4 and 6 is 0.66 whereas the information score for the same columns after realignment is 0.73. More than the improvement in score, what is important it that the conserved activity sequence `ahbd` is preserved after realignment.

$$\begin{array}{c}\begin{array}{ccccccccccccc} &&&\downarrow&&\downarrow&&&&&&& \end{array}\\ \left[\begin{array}{ccccccccccccc} a&h&b&-&f&d&k&a&-&h&b&d&i\\ a&h&b&d&f&-&k&a&-&h&b&d&-\\ a&h&b&d&f&-&k&a&-&h&b&d&i\\ a&h&b&-&-&d&k&a&f&h&b&d&i \end{array}\right]\end{array} \xrightarrow{\text{Realignment}} \left[\begin{array}{ccccccccccccc} a&h&b&-&f&d&k&a&-&h&b&d&i\\ a&h&b&d&f&-&k&a&-&h&b&d&-\\ a&h&b&d&f&-&k&a&-&h&b&d&i\\ a&h&b&d&-&-&k&a&f&h&b&d&i \end{array}\right]$$

**Fig. 5.** Example of Realignment

The alignment procedure described in Section 3 is also called as *global trace alignment*. Depending on the scoring functions, global trace alignment can sometimes penalize gaps at the beginning and/or end of the traces in the alignment. In order to allow gaps to be inserted at the beginning/end of any trace in an alignment, a variant of the global trace alignment called the *semi-global trace alignment* can be considered. Here the best score of the alignment is defined to be the one that is the maximum in the last row or last column of the $F$ matrix defined in Section 3. Traceback procedure starts from that cell and proceeds until it stops at the first position it reaches in the top row or left column. Gaps can then be inserted in the appropriate trace in the positions subsequent to the maximum value cell in the last row/column and prior to the position it reached in the top row or left column. Figure 6 depicts the difference between global trace alignment and semi-global trace alignment of two traces aligned using the same scoring functions. It is easy to see that the alignment obtained using semi-global alignment is preferable to the one obtained using global-alignment. We recommend to consider semi-global trace alignment (at any iteration of progressive alignment) in scenarios where the traces to be aligned differ in their lengths vastly (for example, due to the manifestation of loop constructs).

$$\begin{array}{cccccccccccccc} \overline{T}_1 & j&g&c&-&a&h&b&-&-&-&-&f&d\\ \overline{T}_2 & j&g&c&f&a&h&b&d&k&a&h&b&d \end{array} \qquad \begin{array}{cccccccccccccc} \overline{T}_1 & j&g&c&-&a&h&b&f&d&-&-&-&-&-\\ \overline{T}_2 & j&g&c&f&a&h&b&-&d&k&a&h&b&d \end{array}$$

Global Trace Alignment Semi−global Trace Alignment

**Fig. 6.** Example of global trace alignment and semi-global trace alignment

## 5 Experimental Results and Discussion

Based on the techniques and framework presented in sections 3 and 4, we built a trace alignment plug-in in ProM[3]. We present the results of applying trace alignment on two event logs in the subsequent sections.

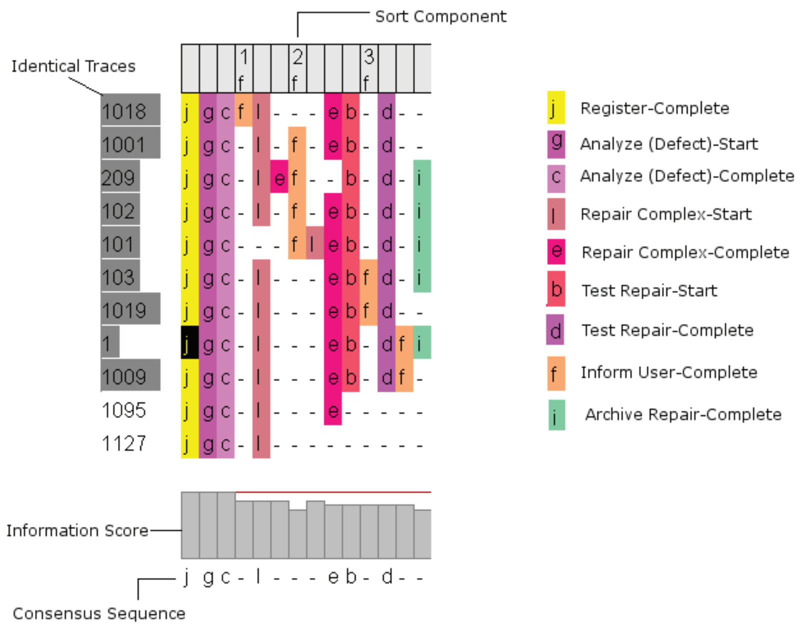
### 5.1 Telephone Repair Log

The telephone repair event log [8] is defined over 12 event classes and consists of 1104 traces, of which only 77 traces are distinct when represented as activity sequences. Since duplicate traces add to the complexity of alignment without yielding any additional benefits, we applied the trace alignment on these 77 traces (but at the same time maintain the fact that there exists identical traces in the log). The log consists of cases where the repair can be classified as a simple or complex one. For our discussion here, we further distinguish two types of cases based on the difficulty level of repair viz., cases where the telephone repair was easy and cases where it was difficult (in both simple and complex types). Difficult cases required multiple tries of the repair diagnosis for failing the quality assessment test.

As mentioned earlier, the guide tree inherently captures the notion of clustering. We have split the event log into four clusters for the example log and Figure 7 depicts the trace alignment for one of the four clusters. This cluster corresponds to traces where the repair type was easy and a complex repair procedure was done to fix the problem. The length of the alignment is 14 for this cluster. The left panel depicts the process instance identifier (as in the log) and identifiers with a grey background indicate traces that have identical duplicates. For example there are traces identical to process instance 1018 (corresponding to activity sequence `jgcflebd` in the event log) while there are no identical traces for the process instance 1127. The top panel depicts a sorting component where the traces involved in the alignment can be sorted based on the activities in a column and the number in the column indicates the priority of sorting. For example in Figure 7, the traces are sorted based on activity `f` (which indicates the `inform user` activity) with traces having `f` in column 4 having first priority and then with those having `f` in column 7 and finally with those having `f` in column 11. The bottom panel depicts the information score metric for each column as well as a consensus sequence for the alignment. *The consensus sequence captures the major activity in each column and can be considered as a back-bone sequence for the process.* Columns with an information score of 1.0 indicate well conserved patterns. For example in this alignment, the columns $1-3$ depicting the encoded activity sequence `jgc` (corresponding to activities `Register-complete`, `Analyze (Defect)-start` and `Analyze (Defect)-complete`) is well conserved and appears in all the traces as the beginning subsequence. It is obvious to see that the encoded activity `f` corresponding to `Inform User - complete`

[3] ProM is an extensible framework that provides a comprehensive set of tools/plugins for the discovery and analysis of process models from event logs. See `http://www.processmining.org` for more information and to download ProM.

is a concurrent activity. *Concurrent activity manifests in mutually exclusive traces across different columns in the alignment.* The encoded activities `l`, `e`, `b`, `d` and `i` correspond to `Repair Complex-start`, `Repair Complex-complete`, `Test Repair-start`, `Test Repair-complete` and `Archive Repair-complete` respectively. Annotating the traces with additional information such as performance metrics, customer feedback etc over the alignment might give further insights. For example, let us assume that the customer was not happy for the cases 1 and 1009, it is obvious to see that these traces differ from the rest in that the activity `f` appears quite late in these traces. It could be inferred that these customers were not timely informed about the status of their complaint and thus were not satisfied. The rest of the 3 clusters for this event log corresponded to the following difficulty level and repair type categories: *easy* and *simple*, *difficult* and *complex* and *difficult* and *simple/complex* where the last cluster pertained to cases where a simple repair procedure was first tried and finally a complex repair procedure was done.



Sort Component
Identical Traces
1 f
2 f
3 f
1018 j g c f l - - - e b - d - -
1001 j g c - l - f - e b - d - -
209 j g c - l e f - - b - d - i
102 j g c - l - f - e b - d - i
101 j g c - - - f l e b - d - i
103 j g c - l - - - e b f d - i
1019 j g c - l - - - e b f d - -
1 j g c - l - - - e b - d f i
1009 j g c - l - - - e b - d f -
1095 j g c - l - - - e - - - - -
1127 j g c - l - - - - - - - - -
Information Score
j g c - l - - - e b - d - -
Consensus Sequence
j Register-Complete
g Analyze (Defect)-Start
c Analyze (Defect)-Complete
l Repair Complex-Start
e Repair Complex-Complete
b Test Repair-Start
d Test Repair-Complete
f Inform User-Complete
i Archive Repair-Complete

**Fig. 7.** Trace alignment of traces in telephone repair log for one of the clusters

## 5.2 Rental Agency Log

We applied trace alignment on a real life log of a rental agency where the cases corresponded to cancellation of a current rental agreement and subsequent registration of a new rental agreement. This log was provided by a large Dutch agency that rents houses and apartments (the organization has approximately

1000 employees and handles 80,000 houses). There were 74 event classes, one event type, 210 traces and 6100 events in this log. As we can see, this log is sufficiently complex in terms of the size of the alphabet and the number of cases. The traces are first encoded into activity sequences where each activity is encoded as a two character sequence. Figure 8 depicts the alignment for one of the four clusters of this log. Since the whole alignment is not legible[4], we highlight the interesting patterns/activities (that we refer to for our further discussion) at the top and the bottom of the figure. The length of the alignment is 88. At the outset, we can see certain patterns in the form of well conserved regions (columns) in the alignment. *Deviations and exceptional behavior are captured in regions that are sparsely filled i.e., regions with lot of gap symbols (−).* We will present the results of analysis of some of these deviations. It could be seen that only one of the traces (third trace in the alignment) has the activity subsequence `b4a8b0` in columns 9 − 11. Activity `b0` in column 8 corresponds to the check, `is first inspection done?` and the activity subsequence `b4a8b0` corresponds to the scenario where the result of the check was negative due to the fact that the tenant was not at home. `b4` corresponds to the activity of sending a letter to the tenant and `a8` corresponds to the activity of rescheduling the first inspection.

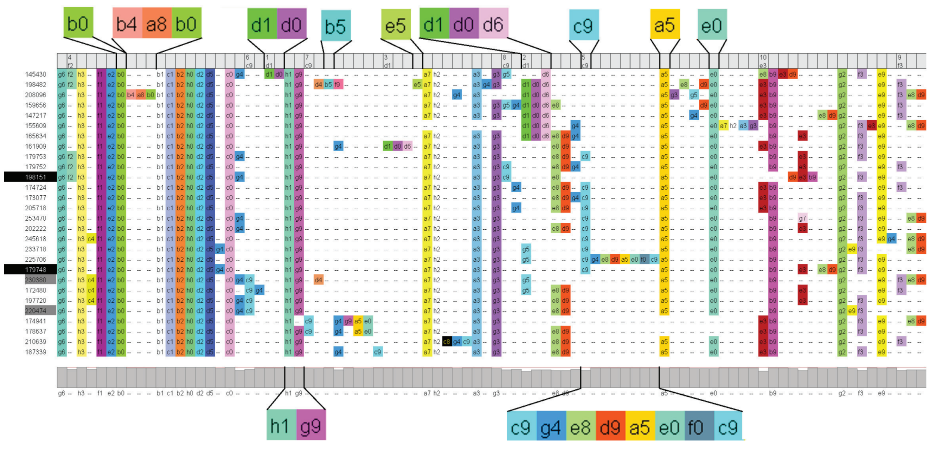


**Fig. 8.** Trace alignment for one of the clusters of rental agency log

The activity sequence `h1g9` corresponding to the checks `is final inspection done?` (`h1`), and `are there new/repaired defects?` (`g9`) is well conserved across all but one of the traces . We see an exceptional activity sequence `d1d0` corresponding to the offering of a flat in one of the traces (first trace) before the activity sequence `h1g9`. It is strange that a flat was offered before the final inspection was done as in all the other traces where the flat was offered, it happened subsequent to the final inspection. Upon further inspection, we observed

[4] The actual alignment can be inspected at `www.win.tue.nl/~jcbose/AlignmentAnaysis3a.png`

that though the flat was offered, the actual registration/check of the candidate corresponding to activity `d6` happened subsequent to the final inspection. Furthermore, in all the cases where the flat was offered and the candidate registered, the activity sequence `d1d0d6` was well conserved except for the first trace. Trace alignment helps us uncover such anomalies and deviations. Similarly, we notice that in only one of traces (second trace) there was a need for second inspection (activity `b5` corresponds to the planning of second inspection and `e5` corresponds to the check, `is second inspection done?`).

The activity `c9` corresponds to the determination of a candidate tenant and the activities `a5` and `e0` correspond to registration of lease and signing of contract respectively. It could also be observed from the alignment that there is an exceptional behavior in one of the cases where we see a manifestation of the activity subsequence `c9g4e8d9a5e0f0c9` (the activity `c9` appears twice). This indicates the fact that for this case, there was a need for determining the candidate tenant twice. The determination of the second candidate tenant followed the activity `f0` which corresponds to the termination of provisional lease. In this fashion trace alignment assists the analyst in getting diagnostic insights by uncovering interesting patterns and deviations.

Finding good quality alignments is intriguingly challenging. Efficient preprocessing techniques for transforming the log with abstractions might help in finding better alignments. One can try to find alignments in a multi-phase approach, with abstractions defined over conserved patterns in each iteration of the alignment. Alternatively, one can also adopt our abstraction techniques proposed in [9].

## 6 Related Work

Song and Aalst [10] have proposed the dotted chart analysis to analyze process performance by depicting process events in a graphical way. The dotted chart analysis (analogous to Gantt charts) primarily focuses on the time dimension of events and presents a "helicopter view" of the event log along with some metrics for performance such as the minimum, maximum and average interval between events. The business analyst need to *manually* investigate the dotted chart to identify any potential performance issues. For logs with medium to large number of activities (of the order of a few tens to hundreds), the manual inspection and comprehension of the dotted chart becomes cumbersome and often infeasible to identify interesting patterns. Trace alignment alleviates this problem, by finding those patterns automatically and depicting it to the user. In the parlance of dotted chart analysis, trace alignment considers the *logical relative* time perspective of the event log. Furthermore, it would be simple and a natural extension to project the performance metrics proposed in [10] onto the aligned traces.

Conformance checking compares an *apriori* model with the observed behavior as recorded in the log and aims at detecting inconsistencies/deviations between a process model and its corresponding execution log [11]. Conformance checking has inherent limitations in its applicability especially for diagnostic purposes. Firstly, it assumes the existence of a process model (the current realization of the conformance checker plugin in ProM requires the process to be modeled as Petrinet). However, in reality, process models are either not present or if present are incorrect or outdated (their quality typically leaves much to be desired). One can argue that process models can be discovered from the event logs and conformance checking be applied on the discovered models. However, this approach is not suitable for the analysis of highly complex and/or flexible processes, the class of models which most of the real-life logs fall into and where the discovered models are "spaghetti-like". Even in cases where the process models are available as Petrinets, it is difficult to look inside of the processes to identify and locate problems especially with models that are large. Trace alignment analyzes the raw event traces and highlights the deviations.

Multiple sequence alignment (MSA) is an active area of research in bioinformatics. Heuristic methods such as progressive alignment [12,13] and iterative alignment [14] have been proposed for MSA. However there are challenges in adapting these techniques for trace alignment. Alignment of biological sequences typically happens over sequences with less variation in length. However, traces in an event log in process mining can be of different lengths. Variation in lengths can occur due to variation in execution paths of the instances and due to manifestation of process model constructs such as choice/loop constructs. In biological sequence alignment, there are standard scoring matrices for substitution that are derived based on physio-chemical properties of the amino acids. Insertion/deletion operations are primarily considered either with a constant gap-score (or penalty) or as an affine function. Scoring matrices for trace alignment need to be derived automatically from the event log or provided by the domain experts. Biological sequences deal with an alphabet size of either 4 (for four nucleic acids) or 20 (for amino acids). However, the number of distinct activities (event classes) in a typical process mining log can be of the order of a few hundreds. This adds to the complexity of deriving good scoring matrices and aligning traces. We took inspiration from MSA techniques [12,13,15] and adapted them for trace alignment.

## 7 Conclusions

In this paper, we proposed a novel approach of aligning traces and showed that this approach uncovers interesting patterns and assists in getting better insights on process executions. We have listed some of the interesting questions in process diagnostics and showed how trace alignment can help in diagnostic efforts. Due to the computational complexity of multiple trace alignment, automatic generation of high-quality alignments is still challenging. Traces that are outliers (noise) in the log might mislead the alignment procedure and thereby result in a low quality alignment. Better techniques to identify and discard outliers during alignment are required. Metrics and realignment strategies in the perspective of process modeling constructs and their manifestation in traces is highly desirable and is an open area of research.

**Acknowledgments.** The authors are grateful to Philips Healthcare for funding the research in process mining.

## References

1. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
2. Waterman, M.S.: Introduction to Computational Biology: Maps, sequences and genomes. Chapman & Hall/CRC (2000)
3. Needelman, S., Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequences of two proteins. Journal of Molecular Biology 48, 443–453 (1970)
4. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological Sequence Analysis: Probabilistic models of proteins and nuclei acids. Cambridge University Press, Cambridge (2002)
5. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Journal of Computational Biology 1(4), 337–348 (1994)
6. Bose, R.P.J.C., van der Aalst, W.M.P.: Context aware trace clustering: Towards improving process mining results. In: Proceedings of the SIAM International Conference on Data Mining, pp. 401–412. SDM, Philadelphia (2009)
7. Simonsen, M., Mailund, T., Pedersen, C.N.S.: Rapid neighbor-joining. In: Algorithms in Bioinformatics, pp. 113–122 (2008)
8. de Medeiros, A.K.A., van der Aalst, W.M.P.: Process mining towards semantics. In: Advances in Web Semantics-I, pp. 35–80 (2008)
9. Bose, R.P.J.C., van der Aalst, W.M.P.: Abstractions in process mining: A taxonomy of patterns. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM. LNCS, vol. 5701, pp. 159–175. Springer, Heidelberg (2009)
10. Song, M., van der Aalst, W.M.P.: Supporting process mining by showing events at a glance. In: Proceedings of the 17th Annual Workshop on Information Technologies and Systems (WITS), pp. 139–145 (2007)
11. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)
12. Feng, D., Doolittle, R.: Progressive sequence alignment as a prerequisite to correct phylogenetic trees. Journal of Molecular Evoluation 25, 351–360 (1987)
13. Feng, D., Doolittle, R.: Progressive alignment of amino acid sequences and construction of phylogenetic trees from them. Methods in Enzymology 266, 368–382 (1996)
14. Barton, G., Sternberg, M.: A strategy for rapid multiple alignment of protein sequences, confidence levels from tertiary structure comparisons. Journal of Molecular Biology 198(2), 327–337 (1987)
15. Daniel, C., Paul, D., Vidhya, M., Marco, O., Eun-Jong, H., Yaoyu, W., Shyamal, S., Brian, C., Shobha, P., Enoch, H.: PFAAT version 2.0: A tool for editing, annotating, and analyzing multiple sequence alignments. BMC Bioinformatics 8(1), 381 (2007)