Efficient Alignment Between Event Logs and Process Models

Wei Song, Xiaoxu Xia, Hans-Arno Jacobsen, *Senior Member, IEEE*, Pengcheng Zhang, and Hao Hu

Abstract—The aligning of event logs with process models is of great significance for process mining to enable conformance checking, process enhancement, performance analysis, and trace repairing. Since process models are increasingly complex and event logs may deviate from process models by exhibiting redundant, missing, and dislocated events, it is challenging to determine the optimal alignment for each event sequence in the log, as this problem is NP-hard. Existing approaches utilize the cost-based A^* algorithm to address this problem. However, scalability is often not considered, which is especially important when dealing with industrial-sized problems. In this paper, by taking advantage of the structural and behavioral features of process models, we present an efficient approach which leverages effective heuristics and trace replaying to significantly reduce the overall search space for seeking the optimal alignment. We employ real-world business processes and their traces to evaluate the proposed approach. Experimental results demonstrate that our approach works well in most cases, and that it outperforms the state-of-the-art approach by up to 5 orders of magnitude in runtime efficiency.

Index Terms—Event logs, process models, alignment, process decomposition, trace segmentation, trace replaying

1 Introduction

Process mining is attracting growing interest due to ever increasing needs for automatically extracting actionable business intelligence from event logs [1], [2], [3], [4], [5]. There are many process mining techniques, such as, conformance checking, process enhancement, and performance evaluation, which need both event logs and process models for decision making [2], [3], [6], [7], [8], [9]. In these cases, it is the prerequisite to align event logs with process models, that is, to relate event sequences in the log to traces in the model and vice versa. The alignment requires that the events in the log are correlated with the activities in the process model [6], [10]. It is often assumed that the correlated event and activity share the same name [6], [11].

The problem of event log-model alignment is, however, not trivial because event sequences recorded in logs may deviate from process models due to missing, redundant, and dislocated activities. Theoretically, each event sequence corresponds to an exponential number of candidate traces due to *parallel* and *alternative routings* in the process model. If the process model involves *iterative routings* (e.g., loops), the number of candidate traces could even be infinite. Here, *routing* refers to the structure of a workflow. For conformance checking, process enhancement, and performance analysis, we need to determine the optimal alignment, that is, to determine a trace through the process model which is closest (i.e., exhibits the minimum distance) to the event sequence in the log. Unfortunately, it has been proven that the problem of finding the optimal alignment is NP-hard [11], [12]. This indicates that when the process is large, it is intractable to determine the optimal alignment [13].

From another perspective, today's business processes continuously produce large volumes of event data in real-time during execution [14], [15], [16]. The generated events trigger other business processes and lead to the execution of other business activities in return, e.g., event monitoring and processing [15], [16], online performance evaluation [7], [17], online conformance checking [7], [13], instance migration [18], etc. These business activities usually call for efficient optimal log-model alignment. However, due to NP-hardness of the problem, we have to sacrifice the accuracy of seeking the optimal alignment to improve the alignment speed.

Existing approaches provide cost-based techniques to measure the distance between two event sequences (traces) [6], [7], [8], [12]. The A^* algorithm is employed to find the optimal alignment between an event sequence in the log and a trace in the model. These techniques work well for small and medium-sized problem instances, but have difficulties coping with industrial-sized problem instances [13], because in the worst case the technique needs to enumerate all possible traces to determine the optimal alignment. The crux of resolving this problem depends on the effective pruning of the search space for seeking the optimal alignment. Our observation is that the structural and behavioral features of process models can be leveraged to reduce the search space. On the one hand, any process model can be horizontally decomposed into several independent sub-processes without alternative routings [17]. Each of the sub-processes corresponds to a class

- *W. Song and X. Xia are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. E-mail: wsong@njust.edu.cn, xiaoxuxia1@gmail.com.*
- *H.-A. Jacobsen is with the Middleware Systems Research Group. E-mail: arno.jacobsen@msrg.org.*
- *P. Zhang is with the College of Computer and Information, Hohai University, Nanjing 211110, China. E-mail: pchzhang@hhu.edu.cn.*
- *H. Hu is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: myou@nju.edu.cn.*

Manuscript received 30 Nov. 2015; revised 31 July 2016; accepted 12 Aug. 2016. Date of publication 17 Aug. 2016; date of current version 8 Feb. 2017. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSC.2016.2601094

of cases (process instances). Thus, each event sequence in the log follows only one of these sub-processes. According to this structural feature, we are in a position to convincingly determine the qualified sub-process which could yield the optimal alignment for the event sequence in the log, whereas the unqualified sub-processes could be safely pruned without the need for further investigation.

On the other hand, each sub-process is executable. According to this behavioral feature, for each event sequence in the log, we can "simulate" its execution in the qualified sub-process. In this way, a "reference" trace which is close to the input event sequence can be obtained. The merit of the technique of simulated execution (i.e., trace replaying) lies in that it avoids enumerating all possible traces (some of which are equivalent) in a sub-process, thus, significantly reducing the search space for seeking the optimal alignment.

Some approaches do exist that focus on repairing event logs by using process models [11], [19], [20]. Despite the fact that the goal of log repairing is different from that of log-model alignment, both types of approaches rely on finding in the process model the reference trace which corresponds to the input event sequence in the event log. However, existing studies on log repairing only focus on the recovery of missing events, while redundant events, and dislocated events are not considered. In this paper, we extend our log recovery approach [20] by handling missing, redundant, and dislocated events in a general framework.

In our experimental evaluation, we compare our approach against the cost-based alignment approach [6] and the log recovery approach [11] on 65 real-world business processes and their traces. The results demonstrate that our approach is able to find the optimal alignment in most cases and is more efficient than the other approaches in terms of processing latency.

This paper builds on our previous work [20] and makes the following new contributions:

1) We develop a general framework to seek the optimal alignment between process models and event logs with missing, redundant, and dislocated events and activities, respectively. This framework is not only used to align event logs with process models, but is also utilized for repairing event logs.
2) We present an efficient approach that is based on effective heuristics and trace relaying to significantly reduce the search space for the optimal alignment. More specifically, we utilize a divide-and-conquer strategy. For processes with no alternative routings or iterative routings, we present a linear-time algorithm that is based on trace replaying to find the optimal alignment for each event sequence in the logs. In general cases, we first decompose the process into several sub-processes, and then present effective heuristics to prune the search space, thus, accelerating the search.
3) We realize our approach in the tool Effa which acts as a plugin of ProM [21]. We employ this tool to conduct an experiment with 65 real-world business processes. Our experimental results demonstrate both the effectiveness and the efficiency of our approach vis-a-vis two state-of-the-art approaches [6], [11] against which we compare ourselves.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the preliminaries. Section 4 defines the research problem. Section 5 elaborates on our alignment approach. Section 6 reports on the experimental results. Section 7 concludes the paper.

2 Related Work

In this section, we review recent approaches relevant to finding the alignment between event logs and process models. Process discovery, conformance checking, and process enhancement are three main types of process mining techniques [21]. The latter two heavily rely on being able to correlate between activities in the model and events in the log, which is required to map an event sequence in the log to the reference trace in the model. In this way, discrepancies and conformance degrees between a log and a model can be determined [2], [3], [6], [8]. Based on this alignment, log repairing [11], [19], [20], model repairing and enhancement [9], [21], as well as performance analysis [7] become feasible. Thus, existing alignment approaches mainly originate from conformance checking techniques.

Rozinat et al. [2] introduce a trace-replaying-based conformance checking technique. However, during replaying, the approach focuses on missing and remaining tokens in Petri net models while the reference trace of the optimal alignment is not taken into account. Obtaining the optimal alignment is the primary step of cost-based conformance checking techniques [6], [7], [8]. The cost of an alignment depends on both skipped activities in process models and inserted events in event logs. Since this is an NP-hard problem [12], the A^* algorithm is applied to find the optimal alignment (with minimum cost). While ingenious, this approach is inefficient and sometimes fails to yield results, as the in worst case it enumerates and searches over all possible traces (i.e., activity interleavings) in the process model. Since existing alignment approaches [6], [7], [8] tend to encounter difficulties when dealing with industry-sized process models and event logs, Munoz-Gama et al. propose to partition the process model into several interconnected sub-models so that a divide-and-conquer method can be used to address the problem [13]. However, since their approach directly adopts existing alignment approaches (i.e., [6], [7], [8]) to seek the optimal alignment in each process partition, the essence of the alignment approach remains the same, and, thus, the approach only reduces the overall runtime overhead to some extent. In contrast, our approach utilizes heuristics and trace replaying to seek the optimal alignment, which significantly increases the processing speed, as we demonstrate experimentally.

The alignment problem is also relevant to log repairing and model repairing. The former concentrates on repairing event sequences according to a predefined process model [11], [19], [20], while the latter focuses on repairing the necessary parts of the process model such that it can replay event sequences in the log [9], [21]. In the following, we only focus on log repairing because it is more relevant to our work. Rogge-Solti et al. [19] utilize the cost-based alignment approach [6] to recover missing events in logs. Wang et al. [11] study how to find the minimum recovery sequence for an incomplete event sequence from a process model. They prove that it is still an NP-hard problem to

recover missing events in logs. Furthermore, they propose a branching framework with advanced indexing and pruning techniques to reduce the search space for finding the minimum recovery sequence. In our previous work [20], we also present a log recovery approach. Different from the approach in [11], in our approach, the pruning granularity is sub-processes, potentially resulting in a single sub-process remaining after pruning. This significantly reduces the search space and facilitates seeking the minimum recovery sequence based on the trace replaying technique. Nevertheless, all these log repairing approaches only focus on the recovery of missing events, whereas redundant and dislocated events are not considered.

Another related approach is *trace alignment* introduced by Bose et al. [22]. The aim of trace alignment is to cluster traces such that process discovery and process diagnosis are facilitated. It is worth mentioning that trace alignment is more tractable than log-model alignment because it is not susceptible to combinatorial explosion.

All above approaches assume that event logs and process models share the same labels for events and activities, respectively. Recently a few approaches have become available for seeking alignment between heterogeneous logs and models [23], [24]. They are complementary to existing alignment approaches.

3 Preliminaries

In this section, we introduce necessary preliminaries based on which we define our research problem and approach. We use Petri nets to represent process models. To ease the understanding of subsequent sections, we define a few basic concepts related to Petri nets [25].

Definition 1 (*Petri Net*). *A Petri net is a triple $PN = (P, T, F)$ where*

- *P is a finite set of places;*
- *T is a finite set of transitions such that $P \cap T = \emptyset$;*
- *$F \subseteq (P \times T) \cup (T \times P)$ is a set of flow relations.*

A Petri net is a directed graphs, where places, transitions, and flow relations are represented by circles, bars, and directed arcs, respectively. For any node $x \in P \cup T$, ${}^\bullet x = \{y \mid <y, x> \in F\}$ and $x^\bullet = \{y \mid <x, y> \in F\}$.

Definition 2 (*Free-Choice Petri Net*). *$PN = (P, T, F)$ is a free-choice Petri net iff $\forall t_1, t_2 \in T$, ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$, then ${}^\bullet t_1 = {}^\bullet t_2$.*

Tokens are introduced to express the state of a Petri net. The distribution of tokens over places is often called the marking (state) of the Petri net. A marking M and a transition t is enabled (denoted as $M[t>$) if each of its input places has at least one token. An enabled transition t can be fired, and the firing consumes a token from each of ${}^\bullet t$ and generates a new one in each of $t^\bullet$. Consequently, the firing of t transitions a Petri net from a marking M to a new marking M'. This step is denoted by $M[t>M'$. If T^* represents the set of all transition sequences over T, $M[\sigma>M'$ denotes that the firing sequence $\sigma \in T^*$ transitions the Petri net from marking M to marking M'.

Definition 3 (*Process Model*). *A process model is a free choice Petri net $PN = (P, T, F)$ which has a unique source place $p_{source} \in P$ (${}^\bullet p_{source} = \emptyset$), and a unique sink place $p_{sink} \in P$ ($p_{sink}^\bullet = \emptyset$) such that each place or transition is on a path from p_{source} to p_{sink}.*

Following [1], [17], we employ free-choice Petri nets to represent process models and use transitions to represent activities in the process models. Commonly-used workflow patterns such as sequential routings, alternative routings, parallel routings, and iterative routings can all be modeled with free-choice Petri nets.

Process instances can be initiated and executed according to the process models. If executed transitions are recorded as events, an event sequence is obtained after the instance terminates. An event (transition) sequence σ conforms to a process model PN, denoted as $\sigma \models PN$ if σ is a complete firing sequence of PN.

Definition 4 (*Complete Firing Sequence*). *Let $PN = (P, T, F)$ be a process model whose initial marking is M_0 and final marking is M_f. A complete firing sequence $\sigma \in T$ of PN is a transition sequence which leads PN from M_0 to M_f,* i.e., *$M_0[\sigma>M_f$.*

At the initial marking M_0, only p_{source} contains a token while at the final marking M_f, only p_{sink} contains a token. Ideally, event logs record conforming event sequences of processes. In practice, however, manual recording, system failures, and interleaved event logs usually lead to nonconforming event sequences [11], [19], [26]. On the other hand, nonconforming event sequences can also be encountered when comparing event logs with the processes extracted from them [2], [6]. A nonconforming event sequence σ of a process model PN is denoted as $\sigma \not\models PN$.

The notation used in this paper is summarized in Table 1.

TABLE 1
Notation Used in the Sequel

Notation	Description
$PN = (N, T, F)$	A process model P
$M[\sigma>M'$	σ is enabled at marking M and its firing results in a new marking M'
$\sigma \models PN$	σ is a complete firing sequence of PN
$S(\sigma)$	The set of events in σ
$\lvert\sigma\rvert$, $\lvert T\rvert$	The number of events in σ, T
$\sigma[i]$	The i^{th} activity in the event sequence σ
$S_1 \uparrow S_2$	Discarding S_1's elements not in S_2
$D(\sigma_1, \sigma_2)$	The edit distance between σ_1 and σ_2
$MS = (S, m)$	A multi-set MS whose underlying set is S and whose multiplicity function is m

4 Problem Statement

As compared with conforming event sequences, nonconforming ones may involve missing events, redundant events, and dislocated events. Our objective is to repair a nonconforming event sequence σ, i.e., transform it into a conforming one σ' according to a priori process model PN. More specifically, following the principle of minimum change to improve data quality [27], [28], our goal is to find in PN a conforming event sequence σ' that is most similar to σ. We refer to the qualified conforming event sequence as the *reference trace* for the nonconforming one.

To formalize the reference trace, let us first review the concept of an *edit distance* between two strings (event sequences) [29]. The edit distance is defined based on the following edit operations: Deleting one event from an event sequence, inserting one event into an event sequence, and interchanging the positions of two adjacent events in an event sequence. In an event sequence, changing one event e_1 into another event e_2 can be realized by first deleting e_1 and then inserting e_2 into the original position of e_1. Although different edit operations could be associated with different costs, in this paper, we assume that the cost of all three edit operations is 1.

Definition 5 (*Edit Distance*). *Given two event sequences σ_1 and σ_2, the edit distance (or distance for short) $D(\sigma_1, \sigma_2)$ between them is measured by the minimum number (cost) of edit operations required to change one event sequence into the other.*

It follows that the distance between two event sequences is no less than 0, that is, for any two event sequences, say, σ_1 and σ_2, $D(\sigma_1, \sigma_2) \geq 0$.

Definition 6 (*Reference Trace*). *Given a process model $PN = (P, T, F)$, the reference trace σ' of a nonconforming event sequence σ satisfies the following two conditions:*

1) $\sigma' \models PN$,
2) *$D(\sigma, \sigma')$ is minimal, i.e., for any other conforming event sequence, say, λ, $D(\sigma, \lambda) \geq D(\sigma, \sigma')$.*

For a nonconforming event sequence σ, there could be more than one reference trace. All such qualified traces are regarded to be equivalent (cf. Definition 7). A concrete example is shown in Example 1 in Section 5.1.

Definition 7 (*Equivalent Reference Traces*). *Given a process model $PN = (P, T, F)$, two event sequences σ' and σ'' are regarded as equivalent reference traces of a nonconforming event sequence σ if the following two conditions are met:*

1) $\sigma' \models PN$, *and* $\sigma'' \models PN$,
2) *$D(\sigma, \sigma') = D(\sigma, \sigma'')$ is minimal.*

When there are more than one reference trace, our goal is to find only one of them. So, the optimal alignment problem we address in this paper, can be stated as follows.

Research Problem 1 (*Optimal Alignment*). *Given a nonconforming event sequence σ with respect to a process model $PN = (P, T, F)$, the optimal alignment problem is to determine one reference trace σ' of σ in PN.*

It has been proven that the optimal alignment problem is NP-hard in general [11], [12]. As mentioned before, many applications relevant to this problem call for efficiency. It seems intractable to find the right reference trace when the process model is complex and large. However, as we demonstrated experimentally, our approach works well for most cases, and, also, it scales well for industry-size processes. We present our approach in the following section.

5 Efficient Alignment

In this section, we present our approach to aligning event sequences in the event log with the corresponding reference traces in the process model.

5.1 Alignment on Causal Nets

Initially, we consider a simple yet fundamental case where process models involve neither alternative routings nor iterative routings. Processes of that kind can be represented by *causal nets* [11], [30]. This fundamental case also serves to illustrate the intuition behind our approach, leaving the presentation of more complex cases to subsequent sections.

Definition 8 (*Causal Net*). *A process specification $PN = (P, T, F)$ is a causal net if $\forall\, p \in P$, $|{}^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$.*

A causal net PN can at most contain sequential and parallel routings. This is to say, during the lifecycle of a process instance, every transition in PN is executed only once. This property is helpful to us in detecting the reference trace of nonconforming event sequences with missing, redundant, and dislocated events.

The intuition behind our approach is that no matter to what degree an event sequence deviates from the process, it follows one reference trace in the process. To determine the reference trace of a nonconforming event sequence σ in a causal net of a process model $PN = (P, T, F)$, we have to align events in σ with the corresponding transitions in PN. More specifically, if an event $e \in S(\sigma) \setminus T$, then it is redundant and, thus, there is no need to replay it in PN. If an event $e \in S(\sigma) \cap T$ occurs more than once in σ, then only one occurrence of e is necessary and the others are redundant. If an event $e \in T \setminus S(\sigma)$, then it is a missing event of σ and it needs to be recovered. If two events e_1, $e_2 \in T \cap S(\sigma)$ but their order in σ is not consistent with that in PN, then they are dislocated events of σ and need to be rearranged according to the control flow of PN. To accomplish the entire task in one pass, we propose a trace replaying technique (cf. Algorithm 1) that handles all these nonconforming cases.

Algorithm 1. SequenceRepair(σ, PN)

Input: Event sequence σ, causal net $PN = (P, T, F)$ whose initial and final markings are M_0 and M_f
Output: Reference trace σ'

$\sigma' \leftarrow \Phi$, $M_c \leftarrow M_0$, $\sigma \leftarrow \sigma \uparrow T$, $i \leftarrow 1$
for each $e \in S(\sigma)$ **do**
  $replayed(e) \leftarrow$ false
while $i \leq |\sigma|$ **do**
  if $replayed(\sigma[i])$ **then**
    $i \leftarrow i + 1$
    continue
  if $M_c[\sigma[i] > M_c'$ **then**
    $\sigma' \leftarrow Append(\sigma', \sigma[i])$
    $replayed(\sigma[i]) \leftarrow$ true
    $i \leftarrow i + 1$
  else if $\exists t \in T \setminus S(\sigma)$, $M_c[t > M_c'$ **then**
    $\sigma' \leftarrow Append(\sigma', t)$
  else if $\exists \sigma[j]$, $M_c[\sigma[j] > M_c'$ & $\nexists k, i < k < j$, $M_c[\sigma[k] >$ **then**
    $\sigma' \leftarrow Append(\sigma', \sigma[j])$
    $replayed(\sigma[j]) \leftarrow$ true
  $M_c \leftarrow M_c'$
if $M_c \neq M_f$ **then**
  Find $\rho \in T^*$: $M_c[\rho > M_f$
  $\sigma' \leftarrow Append(\sigma', \rho)$
return σ'

Algorithm 1 works as follows. Line 1 is the variable initialization, where $\sigma \leftarrow \sigma \uparrow T$ is to remove redundant events not in T. Lines 2–3 initialize each event in σ to be not replayed. Lines 4–17 are in charge of replaying σ in PN. Among them, Lines 5–7 check whether an event has been replayed already; if yes, this event is a redundant occurrence. Thus, it is not necessary to replay it. Instead, we check whether $\sigma[i]$ can be replayed (cf. Line 8) at the current marking M_c. If yes, $\sigma[i]$ is added to the end of σ' and we go on to replay $\sigma[i+1]$ (cf. Lines 9–11). Otherwise, Lines 12–13 select a missing event to be fired; Lines 14–16 are responsible for dislocated events, which fire the first event behind $\sigma[i]$ in σ that is enabled and mark it as fired. In either case, the fired event is appended to σ' and the current marking of PN is updated to go to the next iteration (cf. Line 17). If all events in σ have been handled but M_f is not reached, we have to find an event sequence ρ that transitions PN from M_c to M_f (cf. Lines 18–20). After ρ is appended to σ' (cf. Line 20), σ' constitutes the reference trace of σ (cf. Line 21).

Based on the above, the running time complexity of Algorithm 1 is $O(|\sigma|+|\sigma'|)$ or $O(|\sigma|+|T|)$. This is promising when the process model involves a considerable number of concurrently executing transitions, because our approach avoids enumerating all possible event sequences in seeking the optimal alignment.

Next, we show the soundness of Algorithm 1. In the following, events in different branches of the same parallel routing are referred to as *concurrent events*.

Lemma 1. *σ' returned by Algorithm 1 is a conforming event sequence of PN, i.e., $\sigma' \models PN$.*

Proof. From the specification of Algorithm 1, it can be seen that $M_0[\sigma' > M_f$. Therefore, $\sigma' \models PN$. □

Lemma 2. *The partial orders between concurrent events in σ are preserved in the returned σ'.*

Proof. In Algorithm 1, when $\sigma[i]$ cannot be replayed at the current marking M_c of PN, we first check whether there is a missing event that can be fired at M_c. If no, but more than one event in σ can be replayed at M_c, we choose the one with the minimal index (i.e., position) to be fired (cf. Line 14–16). Hence, in this way, all partial orders between concurrent events in σ are preserved in σ'. □

Lemma 3. *For concurrent events that are missing in σ, different partial orders between them in σ' result in conforming event sequences with the same distance to σ.*

Proof. In Algorithm 1, when $\sigma[i]$ cannot be replayed at the current marking M_c of PN, we first check whether there exists a missing event that can be fired (cf. Lines 12–13). In this situation, if there are at least two missing concurrent events that can be fired, i.e., e_1 and e_2, we randomly select one of them to be replayed. Thereby, if Algorithm 1 runs several times, we can obtain two different conforming event sequences σ_1 and σ_2 with opposite partial orders between e_1 and e_2 as their sole difference. That is, $\sigma_1 \uparrow(T \setminus \{e_1, e_2\}) = \sigma_2 \uparrow(T \setminus \{e_1, e_2\})$. Thus, $D(\sigma, \sigma_1) = D(\sigma, \sigma_1 \uparrow(T \setminus \{e_1, e_2\}))+2 = D(\sigma, \sigma_2 \uparrow(T \setminus \{e_1, e_2\}))+2 = D(\sigma, \sigma_2)$. Therefore, Lemma 3 is proven. □

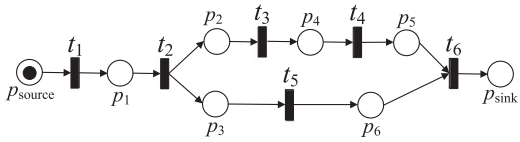

Fig. 1. A process model of a causal net PN.

Theorem 1. *Algorithm 1 returns the reference trace of the nonconforming event sequence σ.*

Proof. We prove this by contradiction. Assume that the returned event sequence σ' is not the reference trace. Let σ'' be the reference trace of σ with respect to $PN = (P, T, F)$. According to Lemma 1, $\sigma' \models PN$. Since σ'' is the reference trace, $\sigma'' \models PN$. Hence, σ' and σ'' share the same set of events (i.e., $S(\sigma') = S(\sigma'') = T$), but there are events with different partial order relations in σ' and σ''. It can be proven that these events must be concurrent events. Assuming these concurrent events are summarized in T_c, we consider three cases in the proof.

1) All events of T_c are in σ. This case is further divided into two sub-cases:
   a) The partial orders between events of T_c in σ are the same as those in σ'' but opposite to those in σ'. Since events in T_c are concurrent events, this situation contradicts Lemma 2.
   b) The partial orders between events of T_c in σ are the same as those in σ' but opposite to those in σ''. In this situation, we can change in σ'' the positions of events in T_c, which results in a new event sequence σ'''. It follows that $D(\sigma, \sigma''') < D(\sigma, \sigma'')$. This contradicts the assumption that σ'' is the reference trace.
2) All events of T_c are missing in σ. Since the missing events in T_c are concurrent events, according to Lemma 3, $D(\sigma, \sigma') = D(\sigma, \sigma'')$. Since $\sigma' \models PN$ and σ'' is the reference trace, σ' is also a reference trace equivalent to σ''. This result contradicts the assumption.
3) Partial events of T_c are in σ. Let $T_{c1} = T_c \cap S(\sigma)$ and $T_{c2} = T_c \setminus S(\sigma)$. It follows that $D(\sigma \uparrow T_c, \sigma' \uparrow T_c) = D(\sigma \uparrow T_{c1}, \sigma' \uparrow T_{c1}) + |T_{c2}|$. Likewise, $D(\sigma \uparrow T_c, \sigma'' \uparrow T_c) = D(\sigma \uparrow T_{c1}, \sigma'' \uparrow T_{c1}) + |T_{c2}|$. According to Lemma 2, $D(\sigma \uparrow T_{c1}, \sigma' \uparrow T_{c1}) = 0$. Since σ'' is the reference trace, $D(\sigma \uparrow T_c, \sigma'' \uparrow T_c) < D(\sigma \uparrow T_c, \sigma' \uparrow T_c)$. From the above four formulae, we can derive that $D(\sigma \uparrow T_{c1}, \sigma'' \uparrow T_{c1}) < 0$, which contradicts the fact that the distance between two event sequences should be no less than 0.

Since all cases lead to contradictions, Theorem 1 holds. □

We use the casual net depicted in Fig. 1 to illustrate Algorithm 1.

Example 1. Consider the process model of a casual net $PN = (P, T, F)$ in Fig. 1 and three event sequences $\sigma_1 = t_1t_2t_3t_5t_4t_6$, $\sigma_2 = t_1t_2t_4t_3t_5t_6$, and $\sigma_3 = t_1t_4t_3t_5t_7t_6$. By using Algorithm 1, we see that σ_1 can be fully replayed in PN, and, thus, $\sigma_1 \models PN$. If we try to replay σ_2, t_1 and t_2 can be replayed, but t_4 cannot. In this case, we first check whether there is a missing event that can be fired. Thus,

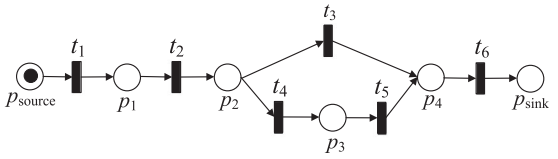

Fig. 2. A process model PN with an alternative routing.

t_5 is found. After the firing of t_5, t_4 can still not be replayed. Since there is no missing event that can be fired, we choose t_3 for firing. After this, t_4 is replayed. The second t_3 is skipped because it has been replayed before. After t_6 is replayed, the final marking of PN is reached, and we get the reference trace $\sigma_2' = t_1t_2t_5t_3t_4t_6$ of σ_2. Note that $\sigma_2'' = t_1t_2t_3t_4t_5t_6$ is also a reference trace of σ_2, because $\sigma_2'' \models PN$ and $D(\sigma_2, \sigma_2') = D(\sigma_2, \sigma_2'') = 3$. Since our goal is to find only one reference trace, other equivalent reference traces such as σ_2'' are not generated. Before the replaying of σ_3, t_7 is removed by the projection operation $\sigma_3 \uparrow T$. After t_1 is replayed, the missing event t_2 is recovered. t_4 cannot be replayed in the next step, but both t_3 and t_5 can. According to Algorithm 1, t_3 is selected because it precedes t_5 in σ_3. After t_3 is replayed, t_4, t_5, t_6 are replayed in order. The reference trace $\sigma_3' = t_1t_2t_3t_4t_5t_6$ of σ_3 is obtained.

5.2 Alignment on Processes with Alternative Routings

In this section, we extend our approach to process models involving alternative routings but without iterative routings. A process model with alternative routings can be regarded as the composition of a set of causal nets with common process fragments [30]. Each of the causal nets (referred to as sub-processes) begins with p_{source} and ends with p_{sink} such that it works as the template for a kind of process instances [17], [30]. In other words, a specific process instance only follows one causal net as it executes. With this in mind, if we decompose the process model into a set of independent causal nets (sub-processes), the detection of the reference traces is facilitated.

Since such a process decomposition algorithm has been presented in [17], [30], we can just leverage it to decompose the process. Here, our focus is on how to determine the causal net that can generate the reference trace for the nonconforming event sequence. According to the convention in process modeling [11], it is not permitted that different transitions (events) in a process model share the same name. Based thereon, the following criterion (cf. Heuristic 1) is presented to identify which causal net can contribute to the reference trace of the nonconforming event sequence.

Definition 9 (*Reference Sub-Process*). *Given a process model $PN = (P, T, F)$ without iterative routings, a causal net $PN_s = (P_s, T_s, F_s)$ decomposed from PN is referred to as the reference sub-process of a nonconforming event sequence σ if PN_s can generate the reference trace of σ.*

Heuristics 1. Given a process model $PN = (P, T, F)$ without iterative routings, a causal net $PN_s = (P_s, T_s, F_s)$ decomposed from PN is regarded as the reference sub-process of a nonconforming event sequence σ, if $\frac{|S(\sigma)\cap T_s|}{|S(\sigma)\cup T_s|}$ is maximal, where $S(\sigma)$ represents the set of events in σ.

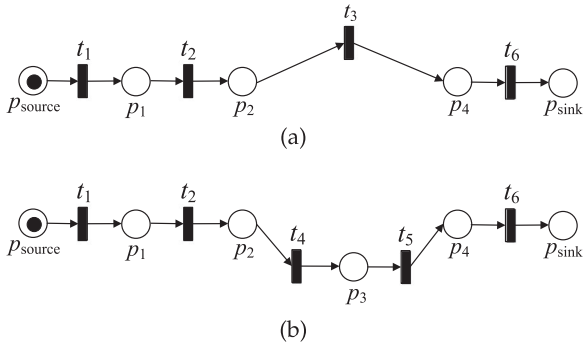

Fig. 3. Causal nets decomposed from the process model PN in Fig. 2: (a) Sub-process PN_1, (b) Sub-process PN_2.

That is, for any other decomposed causal net $PN_s' = (P_s', T_s', F_s')$, $\frac{|S(\sigma)\cap T_s'|}{|S(\sigma)\cup T_s'|} \leq \frac{|S(\sigma)\cap T_s|}{|S(\sigma)\cup T_s|}$.

Despite the fact that Heuristic 1 does not always guarantee to find the reference sub-process (also the reference trace), it performs well in most cases. For the other cases, the heuristic returns an acceptable solution approximating the optimal one. In practice, if there are some sub-processes with comparable value to $\frac{|S(\sigma)\cap T_s|}{|S(\sigma)\cup T_s|}$, they can all be regarded as reference sub-processes, and the right reference trace could probably be generated from them. On the other hand, the formula $\frac{|S(\sigma)\cap T_s|}{|S(\sigma)\cup T_s|}$ in Heuristic 1 measures the conformance degree of the event sequence σ to the sub-process. Thus, the formula itself is useful in practice. For example, if for each sub-process $PN_s = (P_s, T_s, F_s)$, the value of $\frac{|S(\sigma)\cap T_s|}{|S(\sigma)\cup T_s|}$ is too low, we draw the conclusion that σ is of bad quality, e.g., poorly recorded.

Once the qualified causal net is found, we are able to use Algorithm 1 to find the reference trace. The time cost of our approach in this case is as follows. Assuming that a process model contains n alternative routings, each of which consists of m branches, the number of decomposed causal nets is m^n. Since we decompose the process model offline, the overhead of this step is negligible. Since there are m^n candidate causal nets, the runtime cost of checking Heuristic 1 for these causal nets is $O(|S(\sigma)| \times |T_c| \times m^n)$, where $|T_c|$ represents the average number of events in the m^n decomposed causal nets. We prune those decomposed causal nets that fail to meet Heuristic 1, thus, significantly reducing the search space. As analyzed in Section 5.1, the runtime cost of finding the reference trace in the reference sub-process PN_s is $O(|\sigma|+|T_s|)$. So, the overall time complexity is $O(|S(\sigma)| \times |T_c| \times m^n+|\sigma|+|T_s|)$. In practice, neither n nor m is big, and, hence, our approach scales well.

The process model in Fig. 2 is used to illustrate how our approach operates in the presence of alternative routings.

Example 2. Consider the process model $PN = (P, T, F)$ in Fig. 2 and two event sequences $\sigma_1 = t_1t_4t_6t_5t_3$ and $\sigma_2 = t_6t_1t_2t_4t_5$. PN includes an alternative routing, where places P_2 and P_4 represent the beginning (OR-split) and the ending (OR-join) of the alternative routing. The model is first decomposed into two causal nets $PN_1 = (P_1, T_1, F_1)$ and $PN_2 = (P_2, T_2, F_2)$, depicted in Figs. 3a and 3b, respectively. We follow Heuristic 1 to identify which causal net

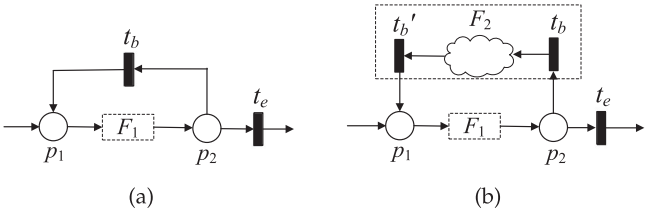

Fig. 4. Structure of an iterative routing in process models: (a) Standard form, (b) Extended form.

is the reference sub-process of σ_1 (σ_2). We see that $\frac{|S(\sigma_1)\cap T_1|}{|S(\sigma_1)\cup T_1|} = \frac{3}{6} = 0.5$, $\frac{|S(\sigma_1)\cap T_2|}{|S(\sigma_1)\cup T_2|} = \frac{4}{6} = 0.67$. Thereby, PN_2 is deemed to be the reference sub-process of σ. Next, we invoke Algorithm 1 to obtain the reference trace of σ in PN_2, while PN_1 is pruned without further investigation. Algorithm 1 first deletes the redundant event t_3, then recovers the missing event t_2 between the replaying of t_1 and t_4, and, finally, swaps the dislocated events t_6 and t_5. The returned reference trace of σ_1 is $\sigma_1' = t_1t_2t_4t_5t_6$. It follows that $D(\sigma_1, \sigma_1') = 3$. If we replay σ_1 in PN_1, the obtained "reference trace" is $\sigma_1'' = t_1t_2t_3t_6$. It can be seen that $D(\sigma_1, \sigma_1'') = 4$. Therefore, Heuristic 1 does not miss the right reference trace σ_1'. To seek the optimal alignment of σ_2, $\frac{|S(\sigma_2)\cap T_1|}{|S(\sigma_2)\cup T_1|} = \frac{3}{6} = 0.5$ and $\frac{|S(\sigma_2)\cap T_2|}{|S(\sigma_2)\cup T_2|} = \frac{5}{5} = 1$. According to Heuristic 1, the reference trace of σ_2 is in PN_2. By invoking Algorithm 1, we obtain the reference trace $\sigma_2' = t_1t_2t_4t_5t_6$ such that $D(\sigma_2, \sigma_2') = 4$. If we replay σ_2 in PN_1, we obtain the "reference trace" $\sigma_2'' = t_1t_2t_3t_6$ such that $D(\sigma_2, \sigma_2'') = 5$. Thus, even in this extreme case, Heuristic 1 does not collapse.

5.3 Alignment on Processes with Iterative Routings

Finally, we extend our approach to the general case that process models contain sequential, parallel, alternative, and iterative routings, as well as their nested forms. To ease understanding, we first explain how iterative routings are modeled with Petri nets. Fig. 4a illustrates an iterative routing in which the loop semantic is realized by the "back" transition t_b such that transitions in fragment F_1 can be executed more than once [31]. The "exit" transition t_e simulates the quit of the iterative routing. When t_e is fired, the corresponding instance cannot re-enter the iterative routing any more. In practice, the back transition t_b can be refined into a *Single-Exit-Single-Entry* (SESE [13]) fragment F_2 with two special transitions t_b and t_b' such that ${}^\bullet t_b = p_2$ and $t_b'^\bullet = p_1$ (cf. Fig. 4b). Based on the structural feature of the iterative routing, we can identify all iterative routings involved in a process model.

For a process model with iterative routings, our approach to seeking the optimal alignment consists of the following four steps.

1) *Decompose the process model into sub-processes.* When the process model involves alternative routings, we need to decompose it into several sub-processes which involve only sequential, parallel, and iterative routings.
2) *Seek the reference sub-process.* For each nonconforming event sequence σ, we employ Heuristic 1 to identify the reference sub-process that can generate the reference trace of σ, while unqualified sub-processes are pruned.
3) *Determine the right number of occurrences for each event in the nonconforming sequence.* Due to iterative routings, redundant occurrences may obscure real ones. We leverage the structure of the reference sub-process and event occurrences in σ to address this problem.
4) *Seek the reference trace in the reference sub-process.* According to the number of occurrences of each event in σ, we replay σ in the reference sub-process to seek the reference trace (i.e., the optimal alignment) of σ.

Note that the first two steps are similar to the case where process models involve no iterative routings. The difference is that iterative routings need to be identified in the first step. For each iterative routing in the sub-process, its back transition and exit transition, the fragments F_1 and F_2 (cf. Fig. 4b) are identified as well. The first step is done offline.

In the third step, based on event occurrences in the event sequence σ and the control-flow of the reference sub-process, we follow the principle of the *minority subordinates to the majority* in order to determine the right number of occurrences for each event in σ. When the iterative routings are not nested, the following criterion (Heuristic 2) is used directly for this problem.

Definition 10 (*Multi-Set*). *A multi-set is a two-tuple $M = (S, m)$, where S is the underlying set of elements and $m: S \rightarrow N_{\geq 1}$ is a function from S to the set $N_{\geq 1}$ of positive natural numbers.*

Definition 11 (*Multiplicity*). *Given a multi-set $M = (S, m)$, for $\forall\ e \in S$, $m(e)$ is referred to as the multiplicity* (i.e., *the number of occurrences) of e in M.*

Definition 12 (*Mode*). *Given a finite multi-set $M = (S, m)$, $e \in S$ is the mode of M, denoted as $mode(M)$, if for any other $e' \in S$, $m(e) \geq m(e')$.*

Heuristics 2. Given a nonconforming event sequence σ and its reference sub-process PN_s, let $m(e)$ and $quot(e)$ represent the actual and right number of occurrences of an event e in σ, respectively, then

- If e is not in an iterative routing of PN_s, $quot(e) = 1$.
- If e is in F_1 of an iterative routing (cf. Fig. 4) of PN_s, $quot(e) = mode(M)$, where the multi-set $M = \{m(e') |\ e' \in S(\sigma) \cap F_1\} \cup \{m(e') + 1 | e' \in S(\sigma) \cap F_2\}$.
- If e is in F_2 of an iterative routing (cf. Fig. 4) of PN_s, $quot(e) = mode(M) - 1$.

Theoretically, there are an infinite number of candidate alignments when processes involve iterative routings. Heuristic 2 is promising in practice, because it significantly reduces the search space.

Iterative routings in process models can be nested. To still apply Heuristic 2 in this case, we propose a solution based on trace segmentation. Let IR_1 be an iterative routing in which there is another iterative routing IR_2. Let $F_1 \cup F_2$ and $F_1' \cup F_2'$ be the event sets of IR_1 and IR_2, respectively. To determine the right number of executed iterations $quot(IR_1)$ of IR_1 in an event sequence σ, we only need to consider events in $F_1 \cup F_2 \backslash (F_1' \cup F_2')$ when applying Heuristic 2. Specifically, $quot(IR_1) = mode(M)$, where $M = \{m(e) | e \in S(\sigma) \cap F_1 \setminus (F_1' \cup F_2')\} \cup \{m(e) + 1 |\ e \in S(\sigma) \cap F_2 \setminus (F_1' \cup F_2')\}$. To get the right number of iterations of IR_2, we can choose one event $e \in F_1$ or $e' \in F_2$ satisfying $m(e) =$

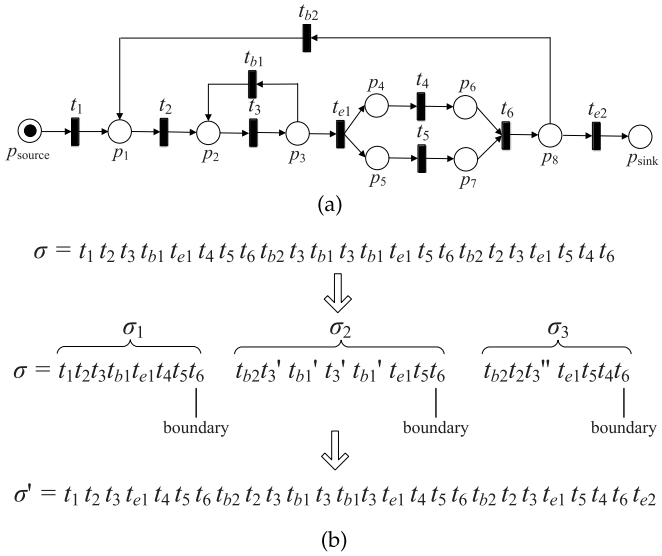

Fig. 5. An example illustrating sequence alignment when the process model contains nested iterative routings: (a) Process model PN with nested iterative routings, (b) Trace σ, its segments $\sigma_1 \sim \sigma_3$, and its reference trace σ'.

$m(e')$+1 = $quot(IR_1)$. Different occurrences of e (or e') in σ are used as boundaries for trace segmentation. In each of the $quot(IR_1)$ segments, we obtain $quot(IR_2)$ in the same way as determining $quot(IR_1)$. This procedure can be recursively invoked, if IR_2 contains other iterative routings. We employ a process model PN with a two-layer iterative routing (cf. Fig. 5a) to illustrate our solution.

Example 3. In Fig. 5a, the outer iterative routing and the inner one are denoted as IR_1 and IR_2, respectively. Consider the nonconforming event sequence $\sigma = t_1t_2t_3t_{b1}t_{e1}t_4t_5t_6t_{b2}t_3t_{b1}t_3t_{b1}t_{e1}t_5t_6t_{b2}t_2t_3t_{e1}t_5t_4t_6$ depicted in Fig. 5b. Since t_2, t_{e1}, t_4, t_5, t_6, and t_{b2} are events in IR_1 but not in IR_2, the multi-set $M = \{m(t_2), m(t_{e1}), m(t_4), m(t_5), m(t_6)\} \cup \{m(t_{b2}) + 1\} = \{2, 3, 2, 3, 3, 3\}$ is obtained, and the right number of iterations of the IR_1 is $quot(IR_1) = mode(M) = 3$. At the same time, $quot(t_2) = quot(t_{e1}) = quot(t_4) = quot(t_5) = quot(t_6) = mode(M) = 3$, and $= quot(t_{b2}) = mode(M) - 1 = 2$. Since t_1 and t_{e2} are not in IR_1, $quot(t_1) = quot(t_{e2}) = 1$. We choose different occurrences of t_6 as the boundaries to segment σ, and, thus, we obtain three sequence segments σ_1, σ_2, σ_3 (cf. Fig. 5b). We can adopt the same manner to determine the number of iterations of IR_2 in each of the three sequence segments. For instance, to get $quot(IR_2)$ in σ_1, we first obtain the events of IR_2 in σ_1, that is, t_3 and t_{b1}. The multi-set $M_1 = \{m(t_3)\} \cup \{m(t_{b1}) + 1\} = \{1, 2\}$ is obtained. The right number of iterations of IR_2 in σ_1 is $mode(M_1) = 1$ or $mode(M_1) = 2$. Either one is acceptable for the subsequent trace replaying. Following the principle of minimum change to improve data quality [27], [28], here, we choose $mode(M_1) = 1$. Accordingly, $quot(t_3) = mode(M_1) = 1$ and $quot(t_{b1}) = mode(M_1) - 1 = 0$. In the same manner, we can obtain the number of iterations of IR_2 in σ_2 and σ_3.

For realizing the fourth step, we specify Algorithm 2. In Algorithm 2, $e.ft$ captures the number of times e fired in T_s; $isBoundary(e)$ reflects whether e is a boundary event determined in the third step; T_B and T_E denote the sets of back transitions and exit transitions in PN_s, respectively. There are two major difference between Algorithms 1 and 2. First, every event in a causal net is executed only once, while in a process model with iterative routings, the number of executions of each event e can be more than one, which is captured in $quot(e)$ in the third step. Second, when an exit transition t_e is enabled, it is not allowed to fire if the number of firings of the corresponding back transition t_b is less than $quot(t_b)$.

Algorithm 2. SequenceRepair+(σ, PN_s)

Input: the event sequence σ; the sub-process $PN_s = (P_s, T_s, F_s)$ whose initial marking and final marking are M_0 and M_f
Output: the reference trace σ'

1: $\sigma' \leftarrow \Phi$, $M_c \leftarrow M_0$, $\sigma \leftarrow \sigma \uparrow T_s$, $i \leftarrow 1$
2: **for** each $e \in S(\sigma)$ **do**
3: $e.ft \leftarrow 0$
4: **while** $i \leq |\sigma|$ **do**
5: **if** $M_c[\sigma[i] > M_c'$, $\sigma[i].ft < quot(\sigma[i])$ **then**
6: **if** $\sigma[i] \in T_E$, $\exists t_b \in T_B$, $M_c[t_b > M_c''$, $t_b.ft < quot(t_b)$ **then**
7: $\sigma' \leftarrow Append(\sigma', t_b)$, $t_b.ft \leftarrow t_b.ft + 1$, $M_c' \leftarrow M_c''$
8: **else**
9: $\sigma' \leftarrow Append(\sigma', \sigma[i])$, $\sigma[i].ft \leftarrow \sigma[i].ft + 1$, $i \leftarrow i + 1$
10: **else if** $t \in T_s \setminus S(\sigma[i, k])$, $M_c[t > M_c'$, $t.ft < quot(t)$, where $k > i$, $isBoundary(\sigma[k])$ = true & $\forall k'$, $i < k' < k$, $isBoundary$ ($\sigma[k']$) = false **then**
11: **if** $t \in T_E$, $\exists t_b \in T_B$, $M_c[t_b > M_c''$, $t_b.ft < quot(t_b)$ **then**
12: $\sigma' \leftarrow Append(\sigma', t_b)$, $t_b.ft \leftarrow t_b.ft + 1$, $M_c' \leftarrow M_c''$
13: **else**
14: $\sigma' \leftarrow Append(\sigma', t)$, $t.ft \leftarrow t.ft + 1$
15: **else if** $\exists \sigma[j]$, $M_c[\sigma[j] > M_c'$, $\sigma[j].ft < quot(\sigma[j])$ & $\nexists k$, $i < k < j$, $M_c[\sigma[k] >$ or $isBoundary(\sigma[k])$ = true **then**
16: **if** $\sigma[j] \in T_E$, $\exists t_b \in T_B$, $M_c[t_b > M_c''$, $t_b.ft < quot(t_b)$ **then**
17: $\sigma' \leftarrow Append(\sigma', t_b)$, $t_b.ft \leftarrow t_b.ft + 1$, $M_c' \leftarrow M_c''$
18: **else**
19: $\sigma' \leftarrow Append(\sigma', \sigma[j])$, $\sigma[j].ft \leftarrow \sigma[j].ft + 1$
20: $M_c \leftarrow M_c'$
21: **if** $M_c \neq M_f$ **then**
22: Find $\rho \in T_s^*$: $M_c[\rho > M_f$ & $S(\rho) \cap T_B = \Phi$
23: $\sigma' \leftarrow Append(\sigma', \rho)$
24: **return** σ'

We analyze the running time complexity of our approach for the general case. The runtime overhead of the first step is neglected, for it happens offline. As discussed in Section 5.2, the time complexity of the second step is $O(|S(\sigma)| \times |T_c| \times m^n)$, where $|T_c|$ represents the average number of events in the m^n decomposed causal net. The third step needs to traverse the input sequence σ, and, thus, the runtime cost is $O(|\sigma|)$. The time cost of the fourth step (i.e., Algorithm 2) is the same as for Algorithm 1, i.e., $O(|\sigma|+|T_s|)$, where T_s is the event set of the reference sub-process. Thus, the time complexity of our approach is $O(|S(\sigma)| \times |T_c| \times m^n + |\sigma| + |T_s|)$.

Example 4 (Continuation of Example 3). Since events t_3 and t_{b1} in different sequence segments σ_1, σ_2, σ_3 are not adjacent and the corresponding $quot(t_3)$, $quot(t_{b1})$ are independent in different segments, t_3 and t_{b1} are differentiated via renaming in σ_1, σ_2 and σ_3 (cf. Fig. 5b), but their mapping to events in PN must be preserved for replaying. Here, we show this differentiation and the

TABLE 2
Approaches Compared in the Experiment

Ref.	Approaches
Alignment	The cost-based alignment approach [6]
Branching	The sequence recovery approach [11]
Effa	Our trace-relaying based approach

boundaries of trace segmentation to facilitate the fourth step. When the first occurrence of t_3 in σ is replayed, although there are other occurrences of t_3 left in σ, Algorithm 2 jumps out of the iterative routing IR_2, because the right number of occurrences of t_3 in σ_1 is $quot(t_3) = 1$. When the segment σ_2 is replayed, after the second t'_{b1} is replayed, t_3 (t'_3) is regarded as a missing event although in the segment σ_3 there is another t_3 (t''_3). After σ is replayed, we obtain its reference trace $\sigma' = t_1t_2t_3t_{e1}t_4t_5t_6t_{b2}t_2t_3t_{b1}t_3t_{b1}t_3t_{e1}t_4t_5t_6t_{b2}t_2t_3t_{e1}t_5t_4t_6t_{e2}$ (cf. Fig. 5b).

6 Evaluation

In this section, we report on the experimental comparison results of our approach with two state-of-the-art approaches (i.e., the alignment approach [6]) and the approach to missing event recovery presented in [11] (cf. Table 2). Our experiment was performed on a computer with Inter(R) 2.5GHz CPU and 6 GB memory running Window 7 and JDK 1.7. With our experiment, we aim to answer the following two research questions.

- RQ1: Effectiveness - How well can our approach seek the optimal alignment between the event log and the process model?
- RQ2: Efficiency - Compared with existing approaches, how efficient is our approach in seeking the optimal alignment? And, does the approach scale well in practice?

6.1 Experimental Setup

Tool Support. The alignment approach has been implemented as a plugin of ProM [21]. We use the plugin for our experiment. Note that the original plugin in ProM only returns the conformance values while the optimal alignment (i.e., the reference trace) is not provided. Since the plugin is open-source, we added several lines of code to derive the reference trace. In order to provide for a fair comparison, we also used Java to realize our approach as a ProM plugin, called Effa, which is publicly accessible at: http://bit.ly/MyEffa. In terms of the Branching approach [11], since the authors do not provide a publicly available tool, we faithfully implemented the approach in Java. The input to the three tools are a process model in PNML format and an event log in XES format [21]. The output of the three tools are reference traces found for each nonconforming event sequence in the event log.

Data Set. We employ a total of 65 real-world processes for our experiment, where 35 stem from Dongfang Boiler, a Chinese manufacturing company, and the remaining processes are from SAP. These processes are Petri net models in PNML format, but not all of them meet the syntax requirements of process models (cf. Definition 3). We manually inserted some dummy places and transitions into the model to guarantee that each one of the processes has one source place and one sink place. Among the 65 processes, 15 (i.e., 23.1 percent) are process models of causal nets, 36 (i.e., 55.4 percent) contain alternative routings but no iterative routings, and 14 (i.e., 21.5 percent) involve iterative routings. The number of activities in these processes range from 4 to 53. Besides process models, we also need corresponding nonconforming event sequences in order to conduct our experiment. Therefore, we randomly generate nonconforming event sequences according to the process models, introduced in more detail below. Nonconforming event sequences are recorded in log files in XES format.

Experimental Procedure. First, for each process model, we obtain its conforming event sequences by simulating the execution of process instances. The simulated executions were repeated 10 times, resulting in a total of 10 conforming event sequences for each of the 65 process models. Second, in order to obtain nonconforming event sequences, we modified each conforming event sequence with specific proportions of missing, redundant, and dislocated events. In this way, for each conforming event sequence σ, we obtain four collections of nonconforming event sequences: The first collection involves nonconforming event sequences which are obtained by deleting different proportions of events (ranging from 10 to 90 percent) from σ; the second collection contains nonconforming event sequences that are obtained by adding different proportions of redundant events to σ; the third collection includes nonconforming event sequences with different proportions of dislocated events in σ; the final collection of nonconforming event sequences contains different proportions of random mutations which involve at least one of the three mutations, i.e., missing, redundant, and dislocated events. Third, for each nonconforming event sequence, we utilize different approaches to seek the reference trace in the corresponding process model. Finally, we compare the performance (in terms of alignment accuracy and runtime overhead) of different approaches based on the evaluation criteria.

Evaluation Criteria. Let σ represent the original conforming event sequence which is used to derive the nonconforming one σ'. Let σ'' represent the reference trace of σ' returned by either of the three approaches. We utilize σ as the ground truth to determine whether σ'' is the right reference trace. That is, if σ'' and σ are equivalent, σ'' is regarded as the right reference trace. With this in mind, we use the following formula to calculate the alignment accuracy of different approaches: $accuracy = \frac{|S(\sigma''=\sigma)|}{|S(\sigma')|}$, where $S(\sigma')$ and $S(\sigma'' = \sigma)$ denote the set of nonconforming event sequences and the set of right reference traces, respectively. Additionally, as mentioned before, the efficiency of seeking the reference trace is of great importance for the optimal alignment problem. Therefore, we also recorded the runtime overheads of different approaches for comparison. An approach is better than others if it is more accurate (with a higher alignment accuracy) and more efficient.

6.2 RQ1: Effectiveness

Experimental Results on Causal Nets. For processes without alternative or iterative routings, our approach only uses Algorithm 1 to obtain the reference trace without employing heuristics. Figs. 6a, 6b, and 6c show the average accuracies

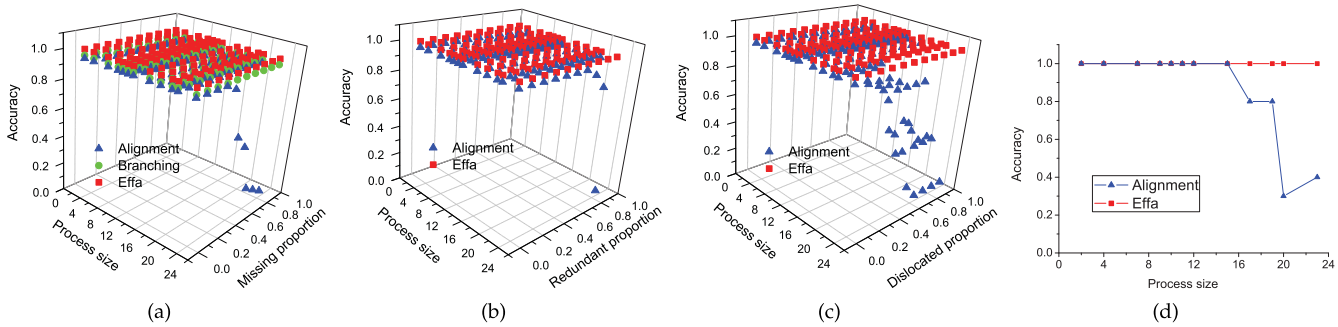

Fig. 6. Accuracy results on causal nets.

of different approaches on 15 causal nets across various proportions of missing, redundant, and dislocated events in event sequences, respectively (ranging from 0 to 90 percent). Fig. 6d illustrates the average accuracies of different approaches on 15 causal nets with random proportions of missing, redundant, and dislocated events in the event sequences. From Fig. 6, we observe that:

- Our approach realized in Effa performs well across different cases, that is, the accuracy of seeking the reference trace remains 100 percent regardless of changes to causal nets and proportions of missing, redundant, dislocated events. This confirms that our approach guarantees obtaining the right reference trace for process models of causal nets.
- The Branching approach also works well, i.e., its accuracy is 100 percent, when event sequences involve only missing events (cf. Fig. 6a).
- The Alignment approach performs well when the process size is small and missing event proportions are low. However, it does not perform well when the number of activities in the process model is larger than 15. Take Fig. 6a for example. For processes with 23 activities, the accuracy of Alignment decreases with the increase in the proportion of missing events, because there are growing nonconforming sequences whose reference traces cannot be obtained within a reasonable time span, say, five minutes. Specifically, when the missing event proportions are 40, 50, 60, 70, 80, 90 percent, accuracies are 90, 55, 45, 10, 5, 0 percent, respectively. Similar results can be found in Figs. 6b and 6c. In addition, the impact of dislocated events on Alignment is largest, while the impact of redundant events on Alignment is smallest. Fig. 6d implies that if input sequences involve random proportions of missing, redundant, and dislocated events, and when the process size is larger than 15 activities, the accuracy of Alignment decreases significantly. For example, for process models which contains 20 activities, the average accuracy of Alignment reduces to 30 percent.

Experimental Results on Process Models with Alternative Routings. For processes with alternative routings but no iterative routings, our approach employs both Heuristic 1 and Algorithm 1 to seek the reference trace. Figs. 7a, 7b and 7c illustrate the average accuracies of different approaches on 36 process models with alternative routings and various proportions of missing, redundant, and dislocated events ranging from 0 to 90 percent in event sequences, respectively. Fig. 7d shows the average accuracies of different approaches applied to 36 process models with random proportions of missing, redundant, and dislocated events in event sequences. From Fig. 7, it follows that:

- For nonconforming sequences with only missing events, the accuracies of the Branching approach and of our approach are generally comparable; both are better than the Alignment approach (cf. Fig. 7a). More specifically, when the proportion of missing events is lower than 50 percent, no matter how many activities are in the process models, the accuracies of all three approaches remains 100 percent. When the proportion of missing events is higher than 50 percent, neither of the three approaches is able to return the right reference trace for all cases, because the optimal alignment is generated from another sub-process, instead of the original sub-process that generates the nonconforming input sequence. For example, when the process size is 10 and the proportion of missing events is 60 percent, accuracies of all three

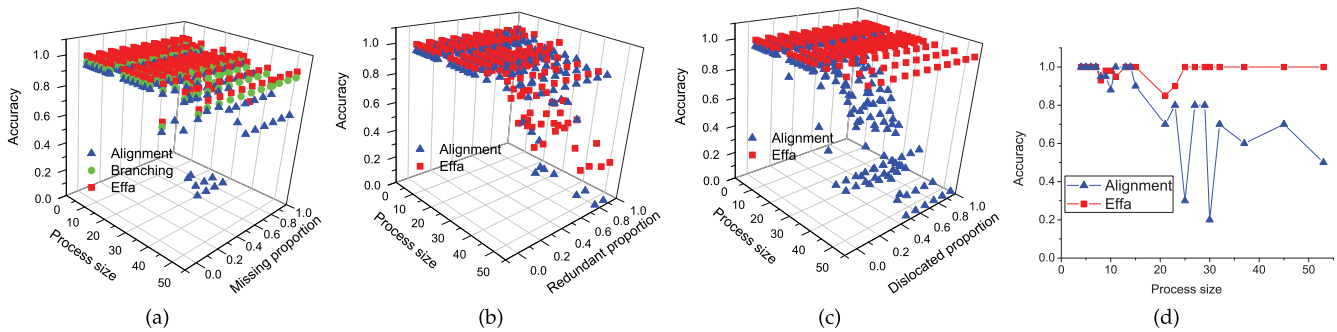

Fig. 7. Accuracy results on process models with alternative routings.

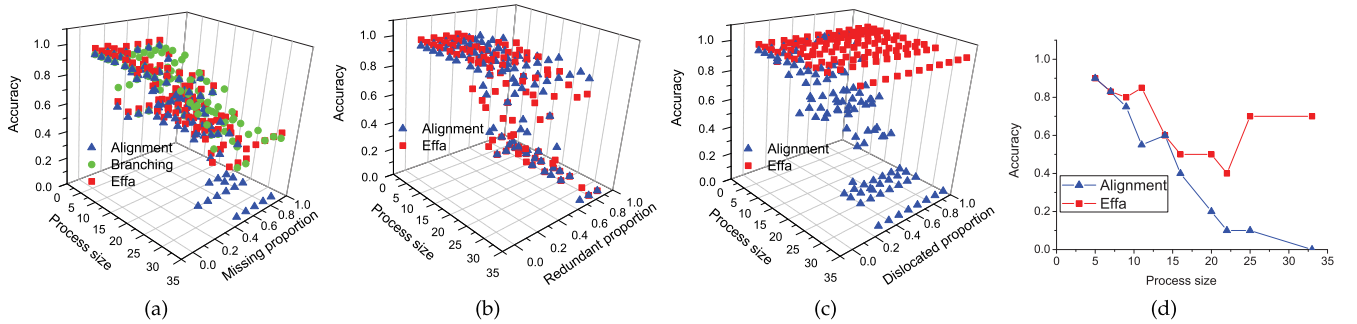

Fig. 8. Accuracy results on process models with iterative routings.

approaches reduces from 100 to 95 percent. When the process size is 15 and the proportion of missing events is 90 percent, accuracies of both Branching and Effa reduce to their low-point of 50 percent; the accuracy of Alignment is also 50 percent, which is not its lowest accuracy. When the process size is 25 and the proportion of missing events is 70 percent, the accuracy of Alignment reduces to 0, as it does not yield any result after a reasonable amount of time.

- For nonconforming sequences with only redundant events, both accuracies of Alignment and Effa generally decrease with the increase of process size and the proportion of redundant events, while Effa performs better than Alignment (cf. Fig. 7b). For example, when the process size is 30 and the proportion of redundant events is 80 percent, accuracies of both Alignment and Effa reduce to 60 percent; when the process size is 30 and the proportion of redundant events is 90 percent, the accuracy of Alignment becomes 0, while the accuracy of Effa remains at 60 percent. When the process size is 53 and the proportion of redundant events is 90 percent, both accuracies of Alignment and Effa become 0 and 30 percent, respectively. The main reason for the low accuracy of Alignment lies in that it cannot generate the reference trace when the process size is large and the proportion of redundant events is high. The main reason for the low accuracy of Effa is that Heuristic 1 could become invalid when the proportion of redundant events is high. Fortunately, the chances of this happening are small for the following reasons. First, the accuracy of Effa remains 100 percent when the proportion of redundant events is lower than 50 percent. Second, in practice, the possibility of a sequence with 90 percent redundant events is slim.
- For nonconforming sequences with only dislocated events, Effa performs better than Alignment, because the accuracy of Alignment decreases as both the process size and the proportion of dislocated events increase, while that of Effa remains 100 percent (cf. Fig. 7c). For instance, the accuracy of Alignment becomes 0 when the process size is 30 and the proportion of dislocated events is 50 percent. The reason for the low accuracy of Alignment is that it cannot produce the reference trace in a reasonable time. The 100 percent accuracy of Effa indicates the advantage of Heuristic 1 when event logs only involve dislocated events.
- For nonconforming sequences with random proportions of missing, redundant, and dislocated events, overall, Effa performs better than Alignment (cf. Fig. 7d). The lowest accuracies of both approaches occur when the process size is between 20 and 30. The reason is that accuracies of both approaches are heavily influenced by the number of branches (activities) in parallel and alternative routings, but not simply by the process size alone.

Experimental Results on Process Models with Iterative Routings. For processes with iterative routings, our approach employs Heuristic 1, Heuristic 2 and Algorithm 2 to seek the reference trace. Figs. 8a, 8b, and 8c demonstrate the average accuracies of different approaches on 14 process models with iterative routings and various proportions of missing, redundant, and dislocated events (ranging from 0 to 90 percent) in event sequences, respectively. Fig. 8d shows the average accuracies of the different approaches on those 14 process models with random proportions of missing, redundant, and dislocated events in event sequences. From Fig. 8 it follows that:

- For nonconforming sequences with only missing events, the accuracies of the three approaches generally decrease as the process size and proportion of missing events increase (cf. Fig. 8a). In some cases, the accuracy of the Branching approach is higher than those of the other two approaches. For instance, when the process size is 16 and the proportion of missing events is 70 percent, accuracies of the Alignment approach and our approach are both 60 percent, while the accuracy of Branching is 75 percent. For input sequences with 40 percent of events missing, we find that both Alignment and Effa detect reference traces with minimum distances to the input sequences. However, these reference traces are not equivalent to the original conforming sequences. This also applies to the 25 percent sequences whose right reference traces that Branching is not able to obtain. For the 15 percent sequences, Branching can find the reference traces equivalent to the original sequences, because it does not take redundant and dislocated events into account, whereas the other two approaches can find the optimal alignment because all deviating situations are considered. In fact, this implies that Branching does not always guarantee to obtain the optimal alignments.

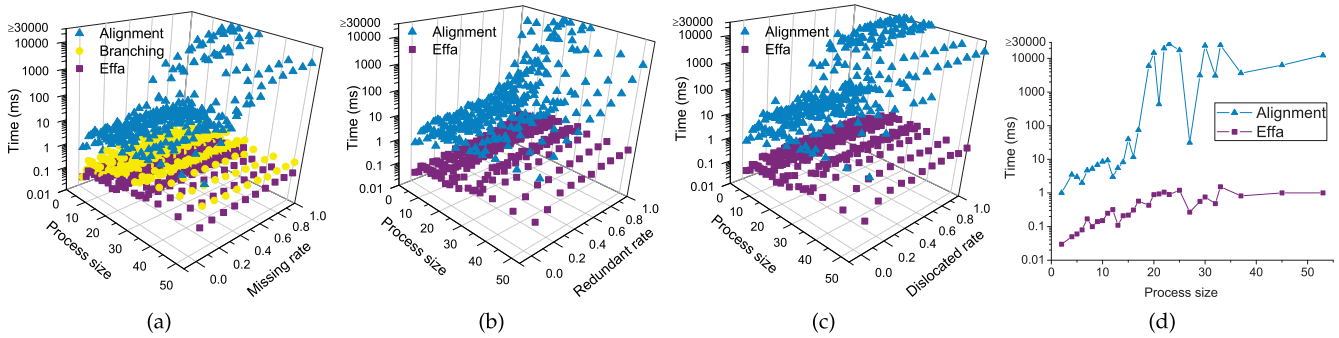

Fig. 9. Runtime overhead per approach.

- For nonconforming sequences with only redundant events, both accuracies of Alignment and Effa generally decrease with the increase of process size and the proportion of redundant events (cf. Fig. 8b). In addition, the performance of both approaches is similar. In some cases the accuracy of Alignment is higher than that of Effa, while in other cases the accuracy of Effa is higher than that of Alignment. For example, when the process size is 20 and the proportion of redundant events is 60 percent, accuracies of Alignment and Effa are 80 percent and 60 percent, respectively. When the process size is 33 and the proportion of redundant events is 70 percent, accuracies of Alignment and Effa are 0 and 10 percent, respectively. The reason for the low accuracy of Effa in this case is that some redundant events in iterative routings are regarded as normal occurrences, especially when the proportion of redundant events is higher than 50 percent.
- For nonconforming sequences with only dislocated events, Effa performs better than Alignment, because the accuracy of Alignment decreases with increasing process size and proportion of dislocated events while that of Effa remains close to 100 percent (cf. Fig. 8c). Moreover, Alignment is significantly impacted by dislocated events. For example, for the process with 33 activities, when the proportion of dislocated events is 30 percent, the accuracy of Alignment reduces to 0 while the accuracy of Effa remains 100 percent. Only for a few specific cases are the accuracies of Effa below 100 percent. For instance, for the process with only seven activities and a nested loop, when the proportion of dislocated events is 40 percent, the accuracy of Effa is 89 percent. This is because our trace segmentation technique is greatly impacted by dislocated events. Yet, the accuracy of Alignment is even worse (i.e., around 50 percent).
- For nonconforming sequences with random proportions of missing, redundant, and dislocated events, accuracies of Alignment and Effa decrease with the increase of the proportion of nonconforming events. Both approaches are comparable when the process size is small; Effa performs better than Alignment when the process size becomes larger (cf. Fig. 8d). For example, when the process sizes are 5, 7, 9, accuracies of Alignmentand Effa are 90, 83, 75, and 90, 83, 80 percent, respectively. When the process sizes are 22, 25, 33, accuracies of Alignment and Effa are 10, 10, 0, and 40, 70, 70 percent, respectively.

Through our experimental analysis, we conclude that the reason why Alignment is unable to detect the right reference trace is two-fold. First, when the process is complex (i.e., involves parallel, alternative, and iterative routings), Alignment may not provide the result in a reasonable time. Second, when the nonconforming event proportion is large, the principle of relying on minimum change to improve data quality becomes invalid. The second is also the reason why Effa does not always produce the right reference trace.

6.3 RQ2: Efficiency

To compare the efficiency of different approaches, we do not differentiate among the types of process models. In the experiment, we find that when the process is complex and the nonconforming event proportions are high, Alignment may not produce any results. Moreover, for cases where Alignment does produce output, results appear within the limit of 30,000 ms. When Alignment produces no results and in order to ease comparability to our approach, we set its runtime overhead to 30,000 ms instead of ∞ in our result graphs.

Figs. 9a, 9b, and 9c illustrate the average runtime overheads of different approaches on all 65 process models and various event proportions (ranging from 0 to 90 percent) of missing, redundant, and dislocated events in event sequences, respectively. Fig. 9d reports the average runtime overheads of different approaches on all 65 process models with random proportions of missing, redundant, and dislocated events. Fig. 9 shows that:

- For nonconforming sequences with only missing events, Branching scales well with increase in process size and missing event proportions. Although, Branching is much more efficient than Alignment, it generally takes twice as long as Effa. For example, when the process size is 22 and the missing event proportions are 90 percent, the runtime overheads of Alignment, Branching, and Effa are 30,000, 0.73, 0.31 ms, respectively.
- In other situations, the runtime overheads of Effa and Alignment increase in process size and nonconforming event proportions. However, Alignment does not scale well, because there is a surge of running time overheads when the process size is large and the nonconforming event proportion is high. Different from Alignment, Effa scales well. More specifically, when the process size is 23 (not too large)

and the redundant event proportion is 90 percent, then the average running time overhead of Alignment is 30,000 ms, whereas Effa's overhead is only 1.51 ms. When the process size is 23 and the dislocated event proportion is 90 percent, the average running time overhead of Alignment is 30,000 ms, whereas that of Effa is only 1.32 ms. When the process size is 53 and the nonconforming event proportion is random, the average running time overhead of Alignment is 12,327 ms whereas that of Effa is only 1.00 ms. To sum up, Effa outperforms the state-of-the-art approach Alignment by up to 5 orders of magnitude in runtime efficiency.

6.4 Threats to Validity

In this section, we analyze the *construct* and the *external validity* of our experiment.

Construct Validity. In our experimental evaluation, we construct the nonconforming event sequences via artificial manipulations of the conforming ones derived from real-world process models. The artificially constructed inputs may be different from real-world scenarios, which could introduce a threat to construct validity. However, in our opinion, the threat is not substantial for the following two reasons. First, we randomly generate the nonconforming event sequences with different proportions of deviations (including missing, redundant, and dislocated events). The generated inputs (nonconforming event sequences) include all possible situations in practice. Second, we generate a total of 26,000 event sequences for our experiment, which further reduces the likelihood of the threat.

The reason why we derive nonconforming event sequences via deviations in conforming ones is as follows: We need an oracle (i.e., the ground truth) to evaluate alignment accuracies of different approaches. In our experiment, the conforming event sequences of real-world processes act as the oracle, while the random manipulations of these event sequences correspond to the inputs (i.e., nonconforming ones). If the output (i.e., the reference trace) returned by an approach is equivalent (cf. Definition 7) to the oracle (i.e., the original event sequence), the effectiveness of the approach is validated. Otherwise, it would be difficult to obtain an oracle for the evaluation purpose.

External Validity. We employ 65 real-world process models and the corresponding nonconforming event sequences for our experiment. One may argue that threats to external validity can be introduced because neither the number of process models nor the number of activities in each process model is sufficiently large. Nevertheless, a recent survey reports that the number of activities in real-world process models tends to be no more than 60 [32]. Also, according to the process modeling guidelines proposed by Mendling et al. [33], process models should be decomposed if they contain more than 50 activities. Thus, the processes we employed and the event logs derived from them are representative. Besides, even for the process with only 23 activities, when the nonconforming event proportion is not very high, the state-of-the-art approach (i.e., Alignment) does not produce results due to the large search space. Altogether, we believe that our experimental results can be generalized.

7 Conclusion

This paper presents a novel and efficient approach to aligning event logs with process models. For process models of causal nets, we propose a linear algorithm which guarantees obtaining the right reference traces. This algorithm utilizes a trace replaying technique to explore the reference trace, and it avoids enumerating all possible event sequences of parallel routings, thus, significantly reducing the search space. For process models with alternative and iterative routings, we provide heuristics based on process decomposition and trace segmentation to prune the search space, further speeding up the search. We implement the proposed approach in our tool called Effa. We conduct extensive experiment over 65 real-world processes and their traces. The experimental results demonstrate that our approach is able to determine the right reference trace for most cases, and that it outperforms state-of-the-art approaches by up to 5 orders of magnitude in runtime efficiency.

Our heuristics-based approach to seeking the optimal alignment is not polynomial. The future work could be focused on investigating whether there are polynomial-time approximation algorithms to address this issue.

Acknowledgments

This work was supported in part by the National Basic Research Program of China under Grant No. 2015CB352202, the National Natural Science Foundation of China under Grant No. 61202003 and Grant No. 61572171, China Scholarship Council under Grant No. 201606845006, NSERC, and the Alexander von Humboldt Foundation. The work in this paper was done in parts at the University of Toronto, Canada and at the Technische Universität München, Germany. W. Song is the corresponding author of the article.

References

[1] W. M. P. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.

[2] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, 2008.

[3] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske, "Process compliance analysis based on behavioural profiles," *Inf. Syst.*, vol. 36, no. 7, pp. 1009–1025, 2011.

[4] P. Weber, B. Bordbar, and P. Tiňo, "A framework for the analysis of process mining algorithms," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 43, no. 2, pp. 303–317, Mar. 2013.

[5] W. Song, H.-A. Jacobsen, C. Ye, and X. Ma, "Process discovery from dependence-complete event logs," *IEEE Trans. Services Comput.*, Doi: 10.1109/TSC.2015.2426181.

[6] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *Proc. 15th IEEE Int. Enterprise Distrib. Object Comput. Conf.*, 2011, pp. 55–64.

[7] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdisciplinary Rev.: Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[8] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "Aligning event logs and declarative process models for conformance checking," in *Proc. 10th Int. Conf. Business Process Manage.*, 2012, pp. 82–97.

[9] D. Fahland and W. M. P. van der Aalst, "Model repair-aligning process models to reality," *Inf. Syst.*, vol. 47, pp. 220–243, 2015.

[10] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data," *Inf. Syst.*, vol. 47, pp. 258–277, 2015.
[11] J. Wang, S. Song, X. Zhu, and X. Lin, "Efficient recovery of missing events," *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 841–852, 2013.
[12] M. de Leoni and W. M. P. van der Aalst, "Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming," in *Proc. 11th Int. Conf. Business Process Manage.*, 2013, pp. 113–129.
[13] J. Munoz-Gama, J. Carmona, and W. M. P. van der Aalst, "Conformance checking in the large: Partitioning and topology," in *Proc. 11th Int. Conf. Business Process Manage.*, 2013, pp. 130–145.
[14] A. V. Baquero, R. C. Palacios, and O. Molloy, "Business process analytics using a big data approach," *IEEE IT Prof.*, vol. 15, no. 6, pp. 29–35, Nov./Dec. 2013.
[15] C. Doblander, T. Rabl, and H.-A. Jacobsen, "Processing big events with showers and streams," in *Proc. 1st Workshop Specifying Big Data Benchmarks*, 2014, pp. 60–71.
[16] L. Baresi and S. Guinea, "Event-based multi-level service monitoring," in *Proc. IEEE 20th Int. Conf. Web Services*, 2013, pp. 83–90.
[17] J. Li, Y. Fan, and M. Zhou, "Performance modeling and analysis of workflow," *IEEE Trans. Syst. Man Cybern.*, vol. 34, no. 2, pp. 229–242, Mar. 2004.
[18] W. Song and H.-A. Jacobsen, "Static and dynamic process change," *IEEE Trans. Services Comput.*, Doi: 10.1109/TSC.2016.2536025.
[19] A. Rogge-Solti, R. Mans, W. M. P. van der Aalst, and M. Weske, "Improving documentation by repairing event logs," in *Proc. 6th IFIP WG 8.1 Work. Conf. Practice Enterprise Modeling*, 2013, pp. 129–144.
[20] W. Song, X. Xia, H.-A. Jacobsen, P. Zhang, and H. Hu, "Heuristic recovery of missing events in process logs," in *Proc. IEEE Int. Conf. Web Services*, 2015, pp. 105–112.
[21] W. M. P. van der Aalst, *Process Mining-Discovery, Conformance and Enhancement of Business Processes*. Berlin, Germany: Springer, 2011.
[22] R. P. J. C. Bose and W. M. P. van der Aalst, "Trace alignment in process mining: Opportunities for process diagnostics," in *Proc. 8th Int. Conf. Business Process Manage.*, 2010, pp. 227–242.
[23] T. Baier, J. Mendling, and M. Weske, "Bridging abstraction layers in process mining," *Inf. Syst.*, vol. 46, pp. 123–139, 2014.
[24] X. Zhu, S. Song, J. Wang, P. S. Yu, and J. Sun, "Matching heterogeneous events with patterns," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 376–387.
[25] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[26] X. Liu, "Unraveling and learning workflow models from interleaved event logs," in *Proc. IEEE Int. Conf. Web Services*, 2014, pp. 193–200.
[27] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi, "A cost-based model and effective heuristic for repairing constraints by value modification," in *Proc. ACM SIGMOD Int. Conf. Manage.*, 2005, pp. 143–154.
[28] S. Kolahi and L. V. S. Lakshmanan, "On approximating optimum repairs for functional dependency violations," in *Proc. 12th Int. Conf. Database Theory*, 2009, pp. 53–62.
[29] R. Lowrance and R. A. Wagner, "An extension of the string-to-string correction problem," *J. ACM*, vol. 22, no. 2, pp. 177–183, 1975.
[30] J. Li, Y. Fan, and M. Zhou, "Timing constraint workflow nets for workflow analysis," *IEEE Trans. Syst. Man Cybern.*, vol. 33, no. 2, pp. 179–193, Mar. 2003.
[31] W. Song, X. Ma, C. Ye, W. Dou, and J. Lü, "Timed modeling and verification of BPEL processes using time Petri Nets," in *Proc. 9th Int. Conf. Quality Softw.*, 2009, pp. 92–97.
[32] J. Wang, T. Jin, R. K. Wong, and L. Wen, "Querying business process model repositories-A survey of current approaches and issues," *World Wide Web*, vol. 17, no. 3, pp. 427–454, 2014.
[33] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, "Seven process modeling guidelines (7PMG)," *Inf. Softw. Technol.*, vol. 52, no. 2, pp. 127–136, 2010.

Wei Song received the PhD degree from Nanjing University, China, in 2010. He is currently an associate professor in the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. At present, he is a visiting scholar at Technische Universität München, Germany. His research interests include software engineering and methodology, services computing, program analysis, business process management, and process mining. He is a member of the ACM.

Xiaoxu Xia received the BS degree from Suzhou University of Science and Technology, China, in 2014. He is currently working toward the master's degree in the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His research interests include services computing, process mining, and log repairing.

Hans-Arno Jacobsen received the PhD degree from Humboldt Universität, Germany. He engaged in postdoctoral research with INRIA near Paris, France, before moving to the University of Toronto, in 2001. He is a professor of computer engineering and computer science and directs the activities of the Middleware Systems Research Group. He conducts research at the intersection of distributed systems and data management, with particular focus on middleware systems, event processing, and cyber-physical systems. In 2011, he received the Alexander von Humboldt-Professorship to engage in research with the Technische Universität München, Germany. He is a senior member of the IEEE.

Pengcheng Zhang received the PhD degree from Southeast University, China, in 2010. He is currently an associate professor in the College of Computer and Information, Hohai University, China. His research interests include modeling, analysis, testing and verification of component based systems, software architectures, real-time and probabilistic systems, and service-oriented systems.

Hao Hu received the PhD degree from Nanjing University, China, in 2009. He is currently an associate professor in the State Key Laboratory for Novel Software Technology, Nanjing University, China. His research interests include software engineering and methodology, workflow management, business process management, and mobile services.