Recomposing Conformance: Closing the Circle on Decomposed Alignment-Based Conformance Checking in Process Mining

Wai Lam Jonathan Lee[a], H.M.W. Verbeek[b], Jorge Munoz-Gama[a], Wil M.P. van der Aalst[b], Marcos Sepúlveda[a]

[a]*Department of Computer Science, School of Engineering, Pontificia Universidad Católica de Chile, (Chile)*
[b]*Architecture of Information Systems Group, Department of Mathematics and Computer Science, Eindhoven University of Technology, (The Netherlands)*

Abstract

In the area of process mining, efficient conformance checking is one of the main challenges. Several process mining vendors are in the process of implementing conformance checking in their tools to allow the user to check how well a model fits an event log. Current approaches for conformance checking are *monolithic* and compute exact fitness values but this may take excessive time. Alternatively, one can use a *decomposition* approach, which runs much faster but does not always compute an exact fitness value.

This paper introduces a *recomposition* approach that takes the best of both: it returns the exact fitness value by using the decomposition approach in an iterative manner. Results show that similar speedups can be obtained as by using the decomposition approach, but now the exact fitness value is guaranteed. Even better, this approach supports a configurable time-bound: "Give me the best fitness estimation you can find within 10 minutes." In such a case, the approach returns an interval that contains the exact fitness value. If such an interval is sufficiently narrow, there is no need to spend unnecessary time to compute the exact value.

Keywords: conformance checking, process mining, business process management

1. Introduction

Process mining is a relatively young research discipline that sits between Business Process Management (BPM) and Data Science [34]. Traditionally, BPM focuses on process models and the design, execution, control, measurement, and optimization of business processes rather than dealing with the generated event data [11]. Contrastingly, data-oriented analysis such as data mining and machine learning looks at particular decisions or patterns and normally does not consider end-to-end processes [31]. By adapting data oriented analysis techniques to improve processes, process mining is the missing link between the two disciplines. More and more event data are made available with the increased use of information systems [1]. Information systems such as ERP (Enterprise Resource Planning) systems (SAP, Oracle, etc) and BPM (Business Process Management) systems (Pegasystems, Bizagi, Appian, IBM BPM, etc) support processes in different organizations. Event data from the event logs of information systems can be ordered to describe instances of the underlying process such that each event can be related to an activity and belongs to a particular case of the process. In essence, these event logs can be seen as "footprints" left behind by the execution of the process. Process mining uses these event logs as a starting point to "discover, monitor and improve real processes" [31].

There are three main areas in process mining [31]. Firstly, *process discovery* takes an event log and produces a model through different discovery algorithms. Secondly, *conformance checking* compares an existing process model with an event log of the same process. Thirdly, *enhancement* extends and improves an existing process model using information from the recorded event logs. In this paper, we focus on **conformance checking**. There are many reasons for performing conformance checking. For example, it is related to the auditing and compliance of business processes [31]. In businesses, certain boundaries on how thing should be done are set by different stakeholders, e.g., managers, government, and others. Audits are carried out to ascertain that these boundaries are being enforced, whether they are laws set by the government or company procedures decided by top level management. By facilitating the comparison of the executed events observed in real-life and the established rules for process execution, it is clear that conformance checking in process mining can support and help to automate audits [15, 35, 36]. Moreover, the company may also want to know whether the set procedures are meeting the requirements of reality. By identifying deviations between the event log and the process model, conformance checking can help with process re-design and modification of the BPM lifecycle [11]. Finally, conformance checking is also a key actor at the other process mining areas. For example, conformance checking is used to compare existing discovery algorithms [10], to recommend discovery techniques [22], or to be incorporated as a part of the discovery method [9, 5, 40].

Conformance checking has become a topic of extreme interest, not only academic but also commercially, where several process mining commercial tools such as *Celonis* and *Lana*

*Corresponding Author: Jorge Munoz-Gama
Vicuña Mackenna 4860, 7820436 Macul (Chile)
Email addresses: walee@uc.cl (Wai Lam Jonathan Lee), h.m.w.verbeek@tue.nl (H.M.W. Verbeek), jmun@uc.cl (Jorge Munoz-Gama), w.m.p.v.d.aalst@tm.tue.nl (Wil M.P. van der Aalst), marcos@ing.puc.cl (Marcos Sepúlveda)

Labs have recently incorporated preliminary conformance analysis. There are several existing conformance checking techniques [19], but most of the state-of-the-art approaches use **alignment-based** approaches due to their robustness and detailed view on deviations [2, 33, 4, 19, 3]. Unlike previous approaches using token-based replay [26], alignment-based conformance checking provides a more robust conformance analysis. In essence, alignment-based conformance checking aligns the events of a particular case with the closest matching path permitted by the process model. These alignments give an evaluation of the overall conformance between the observed events (the reality) and the process model. Moreover, deviations between the observed behavior and the modeled behavior can be pinpointed at the level of events, e.g., the identification of a particular activity that is required by the process model but is missing from the actual execution of the process. This conformance information is later used to improve the process model to be a more faithful representation of the reality (e.g., to be used for simulation), or to take specific actions over the execution of the process (e.g., extra security measures to avoid protocol violations in a hospital [24]).

The volume of data continues to grow and events logs are becoming larger as signified by the term "Big Data" [1, 12]. This has raised the need to make alignment-based conformance checking more scalable, i.e., information systems are recording more event data and are supporting larger and more complex processes. For example, Boeing jet engines can produce ten terabytes of operational information every thirty minutes and Walmart is logging one million customer transactions per hour [23]. Case studies of process mining in organizations have raised various challenges with respect to the performance of existing alignment-based conformance checking techniques when they are applied to larger and more complex processes. Process mining commercial tools also suffer from this limitation, or directly apply shortcuts and best-effort strategies, losing the optimality guarantee of the alignments. One of the most promising research lines is to **decompose** alignment problems. Rather than aligning the overall process and the overall event log, the two are decomposed into subprocesses and sublogs, and alignment is performed on these smaller subcomponents. If a deviation is found at one of the subcomponents, then it is clear that there is a deviation in the overall component. Otherwise, the overall process and the overall log are perfectly fitting. As such, alignment problems can be decomposed and distributed over a network of computers. Experimental results based on large-scale applications of decomposition techniques have shown significant improvements in performance, especially in computation time [20].

Existing decomposition techniques have tackled the original problem in two different ways. One is the computation of conformance at the decomposed subcomponents level [20, 47, 30], where instead of solving the overall problem, it focuses on identifying local conformance issues at individual subcomponents. The other is the approximation of the overall conformance between the process and the log, such as pseudo-alignments [45] or approximate alignments [27]. This means that there is still a gap in the application of decomposition techniques for the exact overall conformance between a process and a log. This paper covers this gap by proposing an approach to turn optimal decomposed pseudo-alignments into optimal proper alignments. This approach creates a full circle approach for decomposed alignment ('there and back again'), and makes it possible to solve alignment-based conformance problems that current techniques cannot handle. Importantly, our approach can balance quality and computation time. For example, in the experimental section of the paper, we demonstrate our approach on a real-life dataset (BPIC2012) for which the existing state-of-the-art monolithic conformance checking approach is not feasible while our proposed approach can compute an almost perfect approximation of the overall conformance in reasonable time, obtaining information about the specific problems of the model. This paper focuses on the fitness dimension which aims to measure the fraction of log traces that can be replayed by the process model.

The methodology of this work can be seen as being under the paradigm of Design Science and it appropriately follows the seven accepted guidelines as presented in [14]. The resulting artifact of this work is a novel conformance checking approach that computes the overall fitness of a process and a log in a divide-and-conquer manner (Guideline 1). The motivating general problem relating to the rapid data growth and the limitations of the existing techniques have been aptly presented above (Guideline 2 and 6). The utility and efficacy of the artifact is demonstrated through mathematical proofs and empirical experiments (Guideline 3 and 5). The artifact and new insights drawn from it are produced as the contributions (Guideline 4). Lastly, we have taken care to present the ideas with a balanced level of detail so that the described artifact can be implemented (Guideline 7).

As such, we extend the applicability of decomposition techniques by computing the overall fitness of a process and a log. This means that performance gains from decomposition techniques can be obtained whilst retaining the quality of the result in reflecting the overall conformance. As the main contributions of the paper:

- We formalize new properties in decompositions of processes and logs. These properties will later extend the conditions under which decomposition techniques can be applied to compute the overall fitness (Section 3.1)
- We also present a new fitness metric to facilitate the aggregation of conformance results from subcomponents to an overall conformance result (Section 3.3).
- Applying the formalized properties and the new fitness metric, we show the conditions under which conformance results from subcomponents can be aggregated. These conditions relax the current perfect fitness requirement and permit aggregation in the presence of deviations between the process and the log (Section 3.4 and Section 3.5).
- With the new aggregation conditions, we present a novel decomposed conformance checking method – **recomposing conformance** – to compute the overall fitness between a process and a log (Section 4).

- Sometimes, exactness is not the top priority. We also modify the exact recomposing conformance method to compute an interval as an approximation of the overall fitness between a process and a log (Section 5).
- The two methods are implemented as one configurable conformance checking approach. The performance gains from the proposed methods are demonstrated through extensive experimental results using both synthetic and real-life datasets (Section 6).

2. Preliminaries

This section introduces basic concepts related to process models, event logs, and their conformance.

2.1. Basic notation

Definition 1 (Multisets)**.** *Let X be a set, a multiset of X is a mapping $M : X \to \mathbb{N}$. $\mathcal{B}(X)$ denotes the set of all multisets over X. Let M and M' be multisets over X. M contains M', denoted $M \geq M'$, if and only if $\forall_{x \in X} M(x) \geq M'(x)$. The union of M and M' is denoted $M + M'$, and is defined by $\forall_{x \in X}(M + M')(x) = M(x) + M'(x)$. The difference between M and M' is denoted $M - M'$ and is defined by $\forall_{x \in X}(M - M')(x) = (M(x) - M'(x))$ max 0.*

Note that $(M - M') + M' = M$ only holds if $M \geq M'$. For sets X and X' such that $X' \subseteq X$, we consider every set X' to be an element of $\mathcal{B}(X)$, where $\forall_{x \in X'} X'(x) = 1$ and $\forall_{x \in X \setminus X'} X'(x) = 0$.

Definition 2 (Projection on sequences and multisets)**.** *Let X be a set, let $X' \subseteq X$ be a subset of X, let $\sigma \in X^*$ be a sequence over X, and let $M \in \mathcal{B}(X)$ be a multiset over X. With $\sigma\lceil_{X'}$ we denote the projection of σ on X', e.g. $\langle x, x, y, y, y, z \rangle \lceil_{\{x,z\}} = \langle x, x, z \rangle$. with $M\lceil_{X'}$ we denote the projection of M on X', e.g. $[x^2, y^3, z]\lceil_{\{x,z\}} = [x^2, z]$.*

Definition 3 (Function domains and ranges)**.** *Let $f \in X \nrightarrow X'$ be a (partial) function. With $dom(f) \subseteq X$ we denote the set of elements from X that are mapped onto some value in X' by f. With $rng(f) \subseteq X'$ we denote the set of elements in X' that are mapped onto by some value in X, i.e., $rng(f) = \{f(x) \mid x \in dom(f)\}$.*

Definition 4 (Functions on sequences and multisets)**.** *Let $f \in X \nrightarrow X'$ be a (partial) function, let $\sigma \in X^*$ be a sequence of X, and let $M \in \mathcal{B}(X)$ be a multiset of X. With $f(\sigma)$ we denote the application of f on all elements in σ, e.g., if $dom(f) = \{x, z\}$, then $f(\langle x, x, y, y, y, z \rangle = \langle f(x), f(x), f(z) \rangle$. With $f(M)$ we denote the application of f on all elements in M, e.g., if $dom(f) = \{x, z\}$, then $f([x^2, y^3, z]) = [f(x)^2, f(z)]$.*

2.2. Petri nets

As previously mentioned, processes are depicted using process models. There are many different process modeling languages, e.g., the Business Process Modeling Notation (BPMN), Event-Driven Process Chains (EPCs), Unified Modeling Language (UML) Activity diagrams, Yet Another Workflow Language (YAWL) and others [11]. In this paper, we use Petri nets to present our ideas [21], as this is the most often-used process modeling notation in process mining. We stress that process models denoted using Petri nets can be translated to models using other process modeling languages.

Definition 5 (Petri net)**.** *A Petri net is a tuple $N = (P, T, F)$ with P the set of places, T the set of transitions, $P \cap T = \emptyset$ and $F = (P \times T) \cup (T \times P)$ the set of arcs, which is sometimes referred to as the flow relation.*

Places are typically visualized by circles, whereas transitions are typically visualized by squares (or rectangles). Consider the Petri net $N_1 = (P_1, T_1, F_1)$ in Figure 1. N_1 has the set of places $P_1 = \{p_1, p_2, \ldots, p_{19}\}$, the set of transitions $T_1 = \{t_1, t_2, \ldots, t_{18}\}$ and the set of arcs $F_1 = \{(p_1, t_1), (t_1, p_2), \ldots, (t_{18}, p_{19})\}$.

The state of a Petri net is called a *marking*, and corresponds to a multiset of places. A marking is typically visualized by putting as many so-called *tokens* (black dots) at a place as the place occurs in the marking. For example, a possible marking of the net N_1 is $[p_2, {p_3}^2]$ which is visualized by one token at place p_2 and two tokens at place p_3.

Definition 6 (Marking)**.** *Let $N = (P, T, F)$ be a Petri net. A marking M is a multiset of places, i.e. $M \in \mathcal{B}(P)$*

Let $N = (P, T, F)$ be a Petri net. For a node $n \in P \cup T$ (a place or a transition), $\bullet n = \{n' \mid (n', n) \in F\}$ denotes the set of *input nodes* and $n \bullet = \{n' \mid (n, n') \in F\}$ denotes the set of *output nodes*.

A transition $t \in T$ is *enabled* by a marking M if and only if each of its input places $\bullet t$ contains at least one token in M, that is, if and only if $M \geq \bullet t$. An enabled transition may *fire* by removing one token from each of the input places $\bullet t$ and producing one token at each of the output places $t \bullet$. The firing of an enabled transition t in marking M is denoted as $(N, M)[t\rangle(N, M')$, where $M' = (M - \bullet t) + t \bullet$ is the resulting new marking.

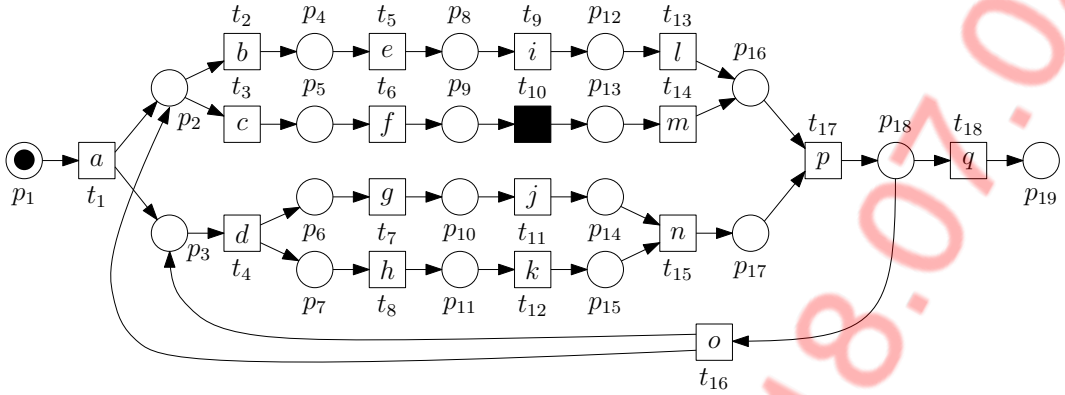
A marking M' is *reachable* from a marking M if and only if there is a sequence of transitions $\sigma = \langle t^1, t^2, \ldots, t^n \rangle \in T^*$ such that $\forall_{0 \leq i < n}(N, M^i)[t^{i+1}\rangle(N, M^{i+1})$ with $M^0 = M$ and $M^n = M'$. For example, $(N_1, [p_1])[\sigma\rangle(N_1, [p_2, p_{17}])$ with the sequence $\sigma = \langle t_1, t_4, t_7, t_8, t_{11}, t_{12}, t_{15} \rangle$ for the Petri net N_1 in Figure 1.

Definition 7 (Labeled Petri net)**.** *A* labeled *Petri net $N = (P, T, F, l)$ is a Petri net (P, T, F) with labeling function $l \in T \nrightarrow \mathcal{U}_A$ where $\mathcal{U}_A$ is some universe of activity labels. Let $\sigma_v = \langle a^1, a^2, \ldots, a^n \rangle \in \mathcal{U}_A^*$ be a sequence of activities. $(N, M)[\sigma_v \rhd (N, M')$ if and only if there is a sequence $\sigma \in T^*$ such that $(N, M)[\sigma\rangle(N, M')$ and $l(\sigma) = \sigma_v$.*

A transition t is called *invisible* if and only if it is not mapped to any activity label by the labeling function, that is, if and only if $t \notin dom(l)$. Otherwise, transition t is visible and corresponds to an observable activity $a = l(t)$.

Consider the labeled Petri net $N_1 = (P_1, T_1, F_1, l_1)$ in Figure 1. It has a labeling function l_1 that maps transition t_1 onto activity label a, t_2 onto b, etc. Note that $t_{10} \notin dom(l_1)$, hence this transition is an invisible transition.

$(N_1, [p_1])[\sigma_v \rhd (N_1, [p_{19}])$ for the sequence $\sigma_v = \langle a, c, f, m, d, g, h, j, k, n, p, q \rangle$ since $(N_1, [p_1])[\sigma\rangle(N_1, [p_{19}])$ with $\sigma = \langle t_1, t_3, t_6, t_{10}, t_{14}, t_4, t_7, t_8, t_{11}, t_{12}, t_{15}, t_{17}, t_{18} \rangle$ and $l_1(\sigma) = \sigma_v$. Note that



a = start bank transfer	g = enter sender account	m = update local bank form
b = open overseas bank form	h = enter receiver account	n = complete verification
c = open local bank form	i = foreign currency conversion	o = redo bank transfer
d = start verification	j = verify sender account	p = finish bank transfer
e = enter oversea bank code	k = verify receiver account	q = send bank transfer
f = enter local bank code	l = update oversea bank form	

Figure 1: Running example: The system net SN_1 that contains the (labeled) Petri net N_1

since t_{10} is not mapped to any activity label, it is not observable in σ_v.

In the context of process mining, the focus is mainly on processes with an initial state and a well-defined final state. For the net N_1, we are interested in *complete* firing sequences, starting from marking $[p_1]$ and ending at marking $[p_{19}]$. The notion of a system net is defined to include the initial and final marking.

Definition 8 (System net)**.** *A system net is a triplet $SN = (N, I, O)$ where $N = (P, T, F, l)$ is a labeled Petri net, $I \in \mathcal{B}(P)$ is the initial marking and $O \in \mathcal{B}(P)$ is the final marking. $\mathcal{U}_{SN}$ is the* universe of system nets.

The net $SN_1 = (N_1, I_1, O_1)$ in Figure 1 is a system net, with an initial marking $I_1 = [p_1]$ and a final marking $O_1 = [p_{19}]$. This system net models a bank transfer process in which a client makes a bank transfer from one account (sender account) to another account (receiver account). The receiver account can be of a local bank or an overseas bank.

Consider the visible sequence $\sigma_v = \langle a, b, e, i, l, d, g, h, j, k, n, p, q\rangle$. This sequence describes the activities that are executed for a bank transfer to an overseas bank account. It is initiated with activity a (*start bank transfer*) and is ended with activity q (*send bank transfer*). For an overseas bank transfer, the bank employee has to open a new overseas bank form, enter the overseas bank code, convert the transfer amount into the foreign currency and fill in the bank form with the converted amount. The bank employee also has to verify both the sender and receiver account before making the transfer. The completion of the bank form and the account verification can be done concurrently as shown in SN_1. In σ_v, the bank form is completed ($\langle \ldots, b, e, i, l, \ldots\rangle$) before the account verification ($\langle \ldots, d, g, h, j, k, \ldots\rangle$).

Definition 9 (System net notations)**.** *Let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$.*

- $T_v(SN) = dom(l)$ *is the set of* visible transitions *in SN.*
- $A_v(SN) = rng(l)$ *is the set of corresponding observable activities in SN.*
- $T_v^u(SN) = \{t \in T_v(SN) \mid \forall_{t' \in T_v(SN)} l(t) = l(t') \Rightarrow t = t'\}$ *is the set of unique visible transitions in SN (such that no other transition has the same visible label).*
- $A_v^u(SN) = \{l(t) \mid t \in T_v^u(SN)\}$ *is the set of corresponding unique observable activities in SN.*

For a given system net, the set of visible traces starting from marking I to marking O is projected onto observable activities yields set $\phi(SN)$.

Definition 10 (Traces)**.** *Let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net. $\phi(SN) = \{\sigma_v \mid (N, I)[\sigma_v \rhd (N, O)\}$ is the set of visible traces starting in marking I and ending in marking O. $\phi_f(SN) = \{\sigma \mid (N, I)[\sigma\rangle(N, O)\}$ is the corresponding set of complete firing sequences.*

For the system net SN_1 in Figure 1, $\phi(SN_1) = \{\langle a, b, e, i, l, d, g, j, h, k, n, p, q\rangle, \langle a, c, f, m, d, g, j, h, k, n, p, q\rangle, \ldots\}$ and $\phi_f(SN_1) = \{\langle t_1, t_2, t_5, t_9, t_{13}, t_4, t_7, t_{11}, t_8, t_{12}, t_{15}, t_{17}, t_{18}\rangle, \langle t_1, t_3, t_6, t_{10}, t_{14}, t_4, t_7, t_{11}, t_8, t_{12}, t_{15}, t_{17}, t_{18}\rangle, \ldots\}$. Due to the loop involving transition t_{16} there is an infinite number of visible traces and complete firing sequences.

The union of two system nets is defined for composing and decomposing of process models.

Definition 11 (Union of nets)**.** *Let $SN = (N, I, O) \in \mathcal{U}_{SN}$ with $N = (P, T, F, l)$ and $SN' = (N', I', O') \in \mathcal{U}_{SN}$ with $N' = (P', T', F', l')$ be two system nets.*

- *$l'' \in (T \cup T') \nrightarrow \mathcal{U}_A$ with $dom(l'') = dom(l) \cup dom(l')$, $l''(t) = l(t)$ if $t \in dom(l)$, and $l''(t) = l'(t)$ if $t \in dom(l') \setminus dom(l)$ is the union of l and l'.*
- *$N \cup N' = (P \cup P', T \cup T', F \cup F', l'')$ is the union of N and N'.*
- *$SN \cup SN' = (N \cup N', I + I', O + O')$ is the union of systems nets SN and SN'.*

Note that since the unioned labeling function $l''(t) = l(t)$ rather than $l''(t) = l'(t)$ if $t \in dom(l)$ and $t \in dom(l')$, the union of nets is not commutative, i.e., $SN \cup SN' \neq SN' \cup SN$.

2.3. Event logs

Event logs serve as the starting point for process mining. An event log is a multiset of *traces*. Each trace describes a particular *case*, i.e., a process instance, in terms of the activities executed.

Definition 12 (Trace, Event log)**.** *Let $A \subseteq \mathcal{U}_A$ be a set of activities. A trace $\sigma \in A^*$ is a sequence of activities. An event log $L \in \mathcal{B}(A^*)$ is a multiset of traces.*

In this simple definition of an event log, an event refers to just an activity. Often event logs store additional information about events such as resources, timestamps or additional data elements recorded with the event log. In this paper, we abstract from such information and limit conformance to solely the control flow aspect.

$$\begin{aligned} L_1 = [&\sigma_1 = \langle a, b, e, i, l, d, g, j, h, k, n, p, q\rangle^{20}, \\ &\sigma_2 = \langle a, c, f, m, d, g, j, k, h, n, p, q\rangle^{5}, \\ &\sigma_3 = \langle a, e, i, l, d, g, j, h, k, n, p, q\rangle^{5}] \end{aligned}$$

$$\begin{aligned} L_2 = [&\sigma_4 = \langle a, c, f, m, d, g, j, h, k, n, p, q\rangle^{20}, \\ &\sigma_5 = \langle a, b, e, i, l, d, j, g, h, k, n, p, q\rangle^{5}, \\ &\sigma_6 = \langle a, b, e, i, l, d, g, h, n, j, k, p, q\rangle^{5}] \end{aligned}$$

Figure 2: Running example: Event logs L_1 and L_2

Consider the event logs L_1 and L_2 in Figure 2. Both event logs have three distinct traces. L_1 contains 30 traces, with 20 σ_1 traces, 5 σ_2 traces and 5 σ_3 traces. L_2 contains 30 traces with 20 σ_4 traces, 5 σ_5 traces and 5 σ_6 traces. Note that an event log is a multiset of traces as a distinct trace (like σ_1) can occur multiple times.

The projection function $\restriction_X$ as introduced earlier also directly applies to event logs. For example, consider the event log L_1 in Figure 1, $L_1 \restriction_{\{a,b,e\}} = [\langle a, b, e\rangle^{20}, \langle a\rangle^5, \langle a, e\rangle^5]$ and $L_1 \restriction_{\{a,b\}} = [\langle a, b\rangle^{20}, \langle a\rangle^{10}]$. For a log $L_3 = [\langle\rangle]$ with an empty trace, the projection $L_3 \restriction_{\{a,b,e\}} = [\langle\rangle]$ returns a log with an empty trace. These projected event logs are referred to as sublogs and the traces in sublogs are referred to as subtraces.

2.4. Alignment-based conformance checking

The main idea of conformance checking is to compare the observed behavior of an event log $L \in \mathcal{B}(A^*)$ with the modeled behavior of the related model, i.e. the related system net $SN = (N, I, O)$.

There are four commonly accepted quality dimensions for comparing a model and a log: (1) fitness, (2) precision, (3) simplicity and (4) generalization [25]. A model has good fitness if it can mimic the behavior of the event log. Yet a fitting model is not necessarily a good model. For example, a "flower model" (a model that has only transitions and no places to constrain the behavior) is able to replay all the traces in the event log and allows sequences that are not seen in the event log. Such a model does not contain any knowledge of the process other than its activities and is unlikely to be of much use. A model is precise if it does not allow "too much" behavior. A model that lacks precision is underfitting. Moreover, a model that does not allow any behaviour other than the traces in the event log is unlikely to be a good model. An event log often does not contain all the possible runs of the process. Generalization means that a model should not be overfitting. Lastly, the simplicity dimension refers to Occam's Razor; a model should be as simple as possible. In the remainder, we will focus on fitness. However, the ideas presented are also applicable to the other quality dimensions, e.g. precision by counting escaping arcs [3] and generalization [33].

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & e & i & l & d & g & j & h & k & n & p & q \\ \hline a & b & e & i & l & d & g & j & h & k & n & p & q \\ t_1 & t_2 & t_5 & t_9 & t_{13} & t_4 & t_7 & t_{11} & t_8 & t_{12} & t_{15} & t_{17} & t_{18} \\ \hline \end{array}$$

$$\gamma_2 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline a & c & f & \gg & m & d & g & j & k & h & \gg & n & p & q \\ \hline a & c & f & \tau & m & d & g & j & \gg & h & k & n & p & q \\ t_1 & t_3 & t_6 & t_{10} & t_{14} & t_4 & t_7 & t_{11} & & t_8 & t_{12} & t_{15} & t_{17} & t_{18} \\ \hline \end{array}$$

$$\gamma_3 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline a & \gg & e & i & l & d & g & j & h & k & n & p & q \\ \hline a & b & e & i & l & d & g & j & h & k & n & p & q \\ t_1 & t_2 & t_5 & t_9 & t_{13} & t_4 & t_7 & t_{11} & t_8 & t_{12} & t_{15} & t_{17} & t_{18} \\ \hline \end{array}$$

Figure 3: Alignments for event log L_1

In alignment-based conformance checking, fitness is measured by alignments between the traces in the event log and the visible traces of the process model. Consider the alignments of the three traces in the event log L_1 in Figure 3. For each alignment, the top row corresponds to the trace in the event log,

the middle row corresponds to the visible trace in the model, whereas the bottom row corresponds to the corresponding firing sequence in the model.

If an activity in the model cannot be mimicked by an activity in the log, then a $\gg$ ("no step") appears in the top row. Similarly, if an activity in the log cannot be mimicked by an activity in the model, then a $\gg$ ("no step") appears in the bottom row. Note that we use the symbol τ as the surrogate activity label for invisible transitions.

For example, the fourth column of γ_2 indicates that the net SN_1 has fired transition t_{10}, which is an invisible transition, and that the log trace σ_2 could not mimic this firing. The ninth column indicates that the activity k was performed in the log trace σ_2, but that the net SN_1 could not mimic this. These columns containing $\gg$ point to deviations between model and log.

A *move* is a pair (a, m) where the first element a refers to the activity in the log and the second element m refers to the transition in the net.

Definition 13 (Legal moves)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log and let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$. $A_{LM} = \{(a,(a,t)) \mid a \in A \wedge t \in T \wedge l(t) = a\} \cup \{(\gg,(a,t)) \mid a \in A \wedge t \in T \wedge l(t) = a\} \cup \{(\gg,(\tau,t)) \mid t \in T \wedge t \notin dom(l)\} \cup \{(a,\gg) \mid a \in A\}$ is the set of* legal moves*. The function $\alpha \in A_{LM} \to A \cup \{\tau\}$ provides the activity (possibly τ) associated with a move: for all $t \in T$ and $a \in A$, $\alpha(a,(a,t)) = a$, $\alpha(\gg,(a,t)) = a$, $\alpha(\gg,(\tau,t)) = \tau$, and $\alpha(a,\gg) = a$.*

An alignment is a sequence of legal moves. This means that, after removing all $\gg$ symbols, the top row corresponds to a log trace and the bottom row corresponds to a firing sequence in the net from marking I to marking O. The middle row corresponds to a visible trace after also removing all τ symbols.

Definition 14 (Alignment)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log with $A \subseteq \mathcal{U}_A$, let $\sigma_L \in L$ be a log trace and $\sigma_M \in \phi_f(SN)$ a complete firing sequence of system net SN. An* alignment *of σ_L and σ_M is a sequence $\gamma \in A^*_{LM}$ such that the projection on the first element (ignoring any $\gg$) yields σ_L and the projection on the last element (ignoring any $\gg$) yields σ_M.*

$\gamma_2' =$

a	c	f	$\gg$	m	d	g	j	k	h	$\gg$	$\gg$	$\gg$	n	p	q
a	c	f	τ	m	d	$\gg$	$\gg$	$\gg$	h	k	g	j	n	p	q
t_1	t_3	t_6	t_{10}	t_{14}	t_4				t_8	t_{12}	t_7	t_{11}	t_{15}	t_{17}	t_{18}

Figure 4: Alternative alignment for trace σ_2

Given a log trace and a model, there could be multiple or even infinitely many alignments. Consider trace σ_2 in event log L_1 in Figure 2. One possible alignment is γ_2 at Figure 3 while another could be γ_2' at Figure 4. It is clear that γ_2 is a better alignment as it better matches the log trace with the model trace. In general, costs can be assigned to different types of moves so that an optimal alignment with the lowest costs can be computed.

Definition 15 (Cost of alignment)**.** *The cost function $\delta \in A_{LM} \to \mathbb{Q}$ assigns costs to legal moves. Moves where the log and the model agree have no costs, i.e. $\delta(a,(a,t)) = 0$ for all $a \in A$. A move in the model also has no costs if the transition is invisible, i.e. $\delta(\gg,(\tau,t)) = 0$ if $t \notin dom(l)$. A move in the model has a cost of $\delta(\gg,(a,t)) > 0$ if $l(t) = a$ and $a \in A$. Similarly, a move in the log has a cost of $\delta(a,\gg) > 0$. The* cost *of an alignment $\gamma \in A^*_{LM}$ is the sum of all costs: $\delta(\gamma) = \sum_{(a,m)\in\gamma} \delta(a,m)$.*

In this paper, we assume a standard cost function δ_1 that assigns unit costs: $\delta_1(a,(a,t)) = 0$, $\delta_1(\gg,(\tau,t)) = 0$ and $\delta_1(\gg,(a,t)) = \delta_1(a,(\gg,t)) = 1$ for all $a \in A$. For example, $\delta_1(\gamma_1) = \delta_1(\gamma_4) = 0$, $\delta_1(\gamma_2) = \delta_1(\gamma_5) = \delta_1(\gamma_6) = 2$ and $\delta_1(\gamma_3) = 1$.

Definition 16 (Optimal alignment)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log with $A \subseteq \mathcal{U}_A$ and let $SN \in \mathcal{U}_{SN}$ be a system net with $\phi(SN) \neq \emptyset$.*

- *For $\sigma_L \in L$, an alignment γ between σ_L and a complete firing sequence of the system net $\sigma_M \in \phi_f(SN)$ is optimal if the associated misalignment costs are lower or equal to the costs of any other possible alignment γ'.*
- *$\lambda(\sigma_L, SN) \in A^* \to A^*_{LM}$ is a deterministic mapping that assigns any log trace σ_L to an optimal alignment.*

$\gamma_4 =$

a	c	f	$\gg$	m	d	g	j	h	k	n	p	q
a	c	f	τ	m	d	g	j	h	k	n	p	q
t_1	t_3	t_6	t_{10}	t_{14}	t_4	t_7	t_{11}	t_8	t_{12}	t_{15}	t_{17}	t_{18}

$\gamma_5 =$

a	b	e	i	l	d	j	g	$\gg$	h	k	n	p	q
a	b	e	i	l	d	$\gg$	g	j	h	k	n	p	q
t_1	t_2	t_5	t_9	t_{13}	t_4		t_7	t_{11}	t_8	t_{12}	t_{15}	t_{17}	t_{18}

$\gamma_6 =$

a	b	e	i	l	d	g	h	n	j	k	$\gg$	p	q
a	b	e	i	l	d	g	h	$\gg$	j	k	n	p	q
t_1	t_2	t_5	t_9	t_{13}	t_4	t_7	t_8		t_{11}	t_{12}	t_{15}	t_{17}	t_{18}

Figure 5: Alignments for event log L_2

Consider the system net SN_1 in Figure 1 and the event logs L_1 and L_2 in Figure 2. The alignments γ_1, γ_2, and γ_3 in Figure 3 are possible optimal alignments for the traces in L_1 and their corresponding firing sequences in the system net SN_1. The alignments γ_4, γ_5, and γ_6 in Figure 5 are possible optimal alignments for the traces in L_2 and their corresponding firing sequences in the same system net. The mapping function would return one of the possible optimal alignments in a deterministic manner so that the same one is always returned.

With an optimal alignment γ between σ_L and SN, we can quantify fitness by comparing its associated misalignment costs with the costs of a default alignment.

Definition 17 (Fitness metric)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log and let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$. Let $l^{\tau} \in T \to A$ be such that $l^{\tau}(t) = l(t)$ if $t \in dom(l)$ and $l^{\tau}(t) = \tau$ if $t \notin dom(l)$. Let $\sigma_L \in L$ be a log trace. Let $move_M(SN) = min_{\sigma_M \in \phi_f(SN)} \sum_{t \in \sigma_M} \delta(\gg, (l^{\tau}(t), t))$ be the minimal costs of an alignment between an empty log trace and a complete firing sequence of the system net. Let $move_L(\sigma_L) = \sum_{a \in \sigma_L} \delta(a, \gg)$ be the costs of an alignment between σ_L and an empty model trace.*

Given the single-trace fitness function fit, the fitness of the trace σ_L is computed as follows:

$$fit(\sigma_L, SN, \delta) = 1 - \frac{\delta(\lambda(\sigma_L, SN))}{move_M(SN) + move_L(\sigma_L)}$$

The fitness function is also overloaded to compute the fitness of the event log L as follows:

$$fit(L, SN, \delta) = 1 - \frac{\sum_{\sigma_L \in L} \delta(\lambda(\sigma_L, SN))}{|L| \times move_M(SN) + \sum_{\sigma_L \in L} move_L(\sigma_L)}$$

This is a relative fitness metric presented in many conformance related papers [4, 2, 33]. The metric normalizes the misalignment costs associated to γ by the costs of the extreme case, a default alignment where all the steps in the log trace are aligned as log moves and all the steps of the minimum model trace are aligned as model moves. Using an optimal alignment $\gamma = \lambda(\sigma_L, SN)$, the fitness metric computes a value between 0 and 1. A trace that perfectly fits the system net would yield a fitness value of 1 and a trace that does not fit the system net at all would yield a fitness value of 0.

Consider the trace σ_3 in Figure 2 and the net SN_1 in Figure 1. Assuming that the optimal alignment $\gamma_3 = \lambda(\sigma_3, SN_1)$ in Figure 3 is used, the fitness of σ_3 and SN_1 is $fit(\sigma_3, SN_1, \delta_1) = 1 - \frac{1}{12+12} = \frac{23}{24} \approx 0.958$. The fitness of the event log L_1 and SN_1 is $fit(L_1, SN_1, \delta_1) = 1 - \frac{0\times20+2\times5+1\times5}{12\times30+13\times20+12\times5+12\times5} = 1 - \frac{15}{740} = \frac{145}{148} \approx 0.980$. Recall that the cost function δ as a function is defined for multisets. This means that the cost function δ is applied to a trace σ multiple times if there are multiple cases with the trace σ.

3. Total border agreement and exact decomposed fitness

Alignment-based conformance checking can be time consuming. This is because the time needed for computing conformance and optimal alignments is heavily influenced by the size of the net and the log, as well as by the complexity of the underlying process. One of the ways to tackle this limitation is through decomposition techniques. The alignment problem can be decomposed by splitting the overall net and the overall log into subcomponents (subnets and sublogs) and then solving this set of smaller problems. Under the assumption that the complexity of the alignment algorithm is significantly worse than linear, solving multiple small alignment problems is often faster than solving one large alignment problem. In addition, the set of decomposed alignment problems can be distributed over a network of computer nodes to further reduce computation time.

However, existing decomposition techniques have limited applicability in computing the overall conformance between the net and the log at the alignment level, i.e., computing conformance using optimal alignments between the net and log traces. Following the decomposition of the overall net and the overall log, existing decomposition techniques only guarantee that the aggregation of conformance results from subcomponents will reflect the exact overall conformance if there is perfect fitness between the net and the log [30]. This has led to the focus on using decomposition to identify problematic sections of the process.

As previously mentioned, there are many scenarios where the precise alignments or costs are required. This motivates our work on extending the conditions under which the conformance results from subcomponents can be aggregated to reflect the exact overall conformance. In this section, we present the core ideas that will extend the applicability of decomposition techniques in computing the overall fitness. These ideas will be later used in two novel alignment-based conformance checking methods to compute an exact or an interval overall fitness result.

3.1. Border activities

In [30] the author presented the concept of *valid decomposition* of a Petri net. A decomposition of a Petri net is valid if each place and invisible transition resides in just one subnet. Moreover, if there are multiple transitions with the same label, they should reside in the same subnet. Only unique visible transitions can be shared among different subnets.

Definition 18 (Valid decomposition [30])**.** *Let $SN \in \mathcal{U}_{SN}$ be a system net with labeling function l. $D = \{SN^1, SN^2, \ldots, SN^n\} \subseteq \mathcal{U}_{SN}$ is a valid decomposition if and only if the following properties are fulfilled:*

- *$SN^i = (N^i, I^i, O^i)$ is a system net with $N^i = (P^i, T^i, F^i, l^i)$ for all $1 \le i \le n$.*
- *$l^i = l \restriction_{T^i}$ for all $1 \le i \le n$.*
- *$P^i \cap P^j = \emptyset$ for all $1 \le i < j \le n$.*
- *$T^i \cap T^j \subseteq T^u_v(SN)$ for all $1 \le i < j \le n$.*
- *$SN = \bigcup_{1 \le i \le n} SN^i$.*

$\mathcal{D}(SN)$ is the set of all valid decompositions of SN.

For example, the decomposition D_1 in Figure 6 is a valid decomposition of the system net SN_1, that is, $D_1 \in \mathcal{D}(SN_1)$. The system nets in D_1 are referred to as the subnets of SN_1.

Given a valid decomposition, an activity may be shared by multiple subnets. We define these activities as the *border activities* of the decomposition. This overlapping property will be used later as a common ground between subcomponents in order to obtain an overall result from local results.

Definition 19 (Border activities)**.** *Let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$. Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of SN. For all $1 \le$*

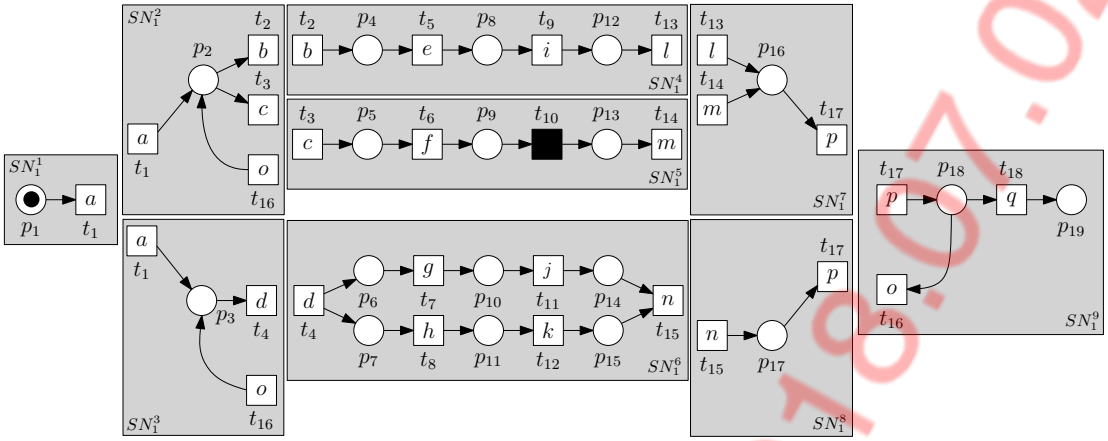


Figure 6: Components resulting from a possible valid decomposition D_1 of the system net SN_1

$i \leq n$, $SN^i = (N^i, I^i, O^i,)$ is a subnet with $N^i = (P^i, T^i, F^i, l^i)$. $A_b(D) = \{l(t) \mid \exists_{1 \leq i < j \leq n}\ t \in T^i \cap T^j\}$ is the set of border activities of the valid decomposition D.

For an activity $a \in rng(l)$, $SN_b(a, D) = \{SN^i \mid SN^i \in D \wedge a \in A_v(SN^i)\}$ is the set of subnets that contain a as an observable activity.

Due to the properties of a valid decomposition, a border activity can only be an activity that has a unique label, i.e., $A_b(D) \subseteq A_v^u(SN)$. For example, the valid decomposition D_1 in Figure 6 has the set of border activities of $\{a, b, c, d, l, m, n, o, p\}$.

In addition, non-unique activities will appear in precisely one subnet, i.e., for all $a \in A_v(SN) \setminus A_v^u(SN)$ it holds that $|SN_b(a, D)| = 1$. Contrastingly, unique activities may appear in multiple subnets, i.e., for all $a \in A_v^u(SN)$ it holds that $|SN_b(a, D)| \geq 1$. Border activities are unique activities that appear in multiple subnets, i.e., for all $a \in A_b(D)$ it holds that $|SN_b(a, D)| > 1$.

For example with the valid decomposition D_1 in Figure 6, $SN_b(a, D_1) = \{SN_1^1, SN_1^2, SN_1^3\}$ as the border activity a is shared by the subnets SN_1^1, SN_1^2 and SN_1^3.

3.2. Alignment for subnets with border activities

Following a valid decomposition, border activities would either have no input places or output places in the subnets which share the border activities but do not contain the corresponding places. This means that these subnets would have an empty initial marking and/or an empty final marking.

For example, in the valid decomposition D_1 in Figure 6, the activity set of SN_1^8 consists of the border activities n and p. For subnet SN_1^8, border activity n has no input places and border activity p has no output places. This means subnet SN_1^8 has an empty initial and final marking. We note that the computation of an optimal alignment remains the same as for system nets with non-empty initial and final markings.

For the sake of simplicity, say we are aligning the subtrace $\sigma_3^8 = \sigma_3 \restriction_{A_1^8} = \langle n, p \rangle$ with subnet SN_1^8, i.e., subtrace σ_3^8 is obtained by projecting trace σ_3 onto the activity set of subnet SN_1^8. At the start of the alignment procedure, we would yield an empty alignment since no legal moves have been added yet. Since the initial marking equals the final marking (both are empty), by Definition 10, an empty sequence is a valid complete firing sequence. However, the empty alignment is not a valid alignment in this case since Definition 14 requires the projection on the first element (ignoring any $\gg$) to yield the log trace, i.e., σ_3^8, and the projection on the last element (ignoring any $\gg$) to yield a complete firing sequence of the net. While the projection on the last element of an empty alignment gives a complete firing sequence, the projection on the first element does not yield σ_3^8. As such, we try to fire the corresponding transition of activity n from subtrace σ_3^8. Transition t_{15} is always enabled in subnet SN_3^8 since it has no input places. Firing transition t_{15} produces a token in place p_{17} and enables transition t_{17}. This means that the log and model execution of activity n corresponds to a synchronous move. Similarly, the corresponding transition of activity p, which is the next activity in subtrace σ_3^8, can be fired in the model as there is a token in its input place p_{17}. This means that the log and model execution of activity p corresponds to a synchronous move. Since transition t_{17} has no output places, the marking reached by its firing is an empty multiset. This corresponds to the final marking and is therefore a complete firing sequence. The produced alignment corresponds to subalignment γ_3^8 in Figure 7. This is a valid alignment as the projection on the first element (ignoring any $\gg$) yields σ_3^8 and the projection on the last element (ignoring any $\gg$) yields a complete firing sequence.

We note that there are alternative decomposed replay approaches that do not involve removing all irrelevant transitions, places, and arcs. The paper [42] presents the *Hide* and *Hide and Reduce* decomposed replay approaches which create subnets by making irrelevant transitions invisible. Lastly, we refer interested readers to the paper [30] for further details on the replay of subnets created by valid decompositions.

3.3. Decomposed Fitness

Let us consider trace $\sigma_3 = \langle a, e, i, l, d, g, j, h, k, n, p, q \rangle$ and the system net SN_1 in Figure 1. Under a decomposition, the optimal subalignments $\gamma_3^1, \ldots, \gamma_3^9$ can be obtained by first pro-

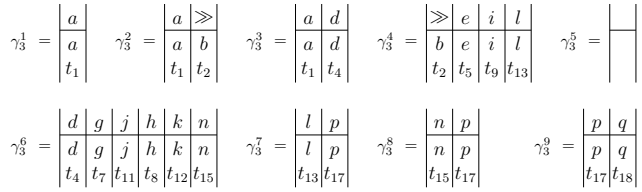


Figure 7: Subalignments between the trace $\sigma_3 = \langle a, e, i, l, d, g, j, h, k, n, p, q\rangle$ in event log L_1 and the valid decomposition D_1 in Figure 6

jecting σ_3 onto the subnets $SN_1^1, \ldots, SN_1^9$ in Figure 6 and later aligning each subtrace with the corresponding subnet as shown in Figure 7.

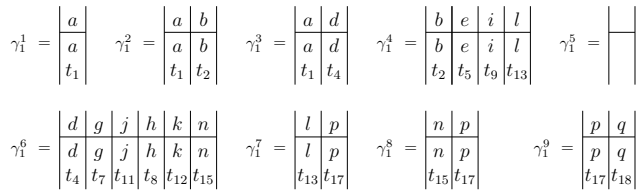


Figure 8: Subalignments between the trace $\sigma_1 = \langle a, b, e, i, l, d, g, j, h, k, n, p, q\rangle$ in event log L_1 and valid decomposition D_1 in Figure 6

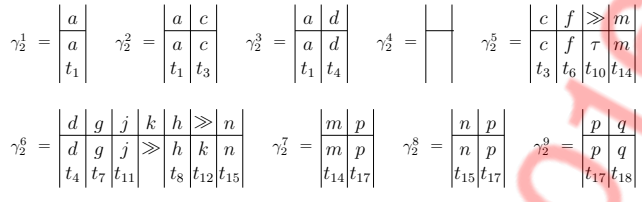


Figure 9: Subalignments between the trace $\sigma_2 = \langle a, c, f, m, d, g, j, k, h, n, p, q\rangle$ in event log L_1 and valid decomposition D_1 in Figure 6

All moves in alignments γ_3^1, γ_3^3, γ_3^6, γ_3^7, γ_3^8 and γ_3^9 are synchronized. Alignment γ_3^5 is an empty alignment. Alignments γ_3^2 and γ_3^4 both have a model move involving transition t_2 with the label b. Similarly, optimal subalignments for the traces σ_1 and σ_2 are shown in Figures 8 and 9 respectively.

A naive approach to aggregate the results per subcomponent, would be to sum up all the misalignment costs of the subalignments under the standard cost function. For $\gamma_3^1, \gamma_3^2, \ldots, \gamma_3^9$, we would get a total of 2. However the misalignment costs associated to the optimal overall alignment γ_3 is 1 as shown in Figure 3. The wrong result is produced because border activities appear in multiple subnets and therefore moves involving these transitions will be counted multiple times when their associated costs are simply aggregated. We would like the result computed using $\gamma_3^1, \ldots, \gamma_3^9$ to equal the optimal cost computed using a overall alignment such as γ_3. Hence, we use the adapted cost function presented in [30], to avoid counting moves that involve border activities multiple times.

Definition 20 (Adapted cost function [30])**.** *Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of some system net SN and $\delta \in A_{LM} \to \mathbb{Q}$ a cost function. The adapted cost function $\delta_D \in A_{LM} \to \mathbb{Q}$ for decomposition D is defined as follows:*

$$\delta_D(a,m) = \begin{cases} \frac{\delta(a,m)}{|SN_b(\alpha(a,m),D)|} & \text{if } \alpha(a,m) \neq \tau; \\ \delta(a,m) & \text{otherwise.} \end{cases}$$

The cost of each legal move is divided by the number of subnets in which the corresponding activity may appear, for example, $\delta_D(a, \gg) = \frac{\delta(a,\gg)}{|SN_b(a,D)|}$. This avoids counting misalignment costs of the same legal move multiple times. For example, consider the set of subalignments $\gamma_3^1, \ldots, \gamma_3^9$ in Figure 7, $|SN_b(b, D_1)| = |\{SN_1^2, SN_1^4\}|$ and $|SN_b(d, D_1)| = |\{SN_1^3, SN_1^6\}|$. For the adapted standard cost function δ_{D_1}, $\delta_{D_1}(\gg, (b, t_2)) = \frac{1}{2}$ and $\delta_{D_1}(d, (d, t_4)) = 0$. The aggregated cost of $\gamma_3^1, \ldots, \gamma_3^9$ is 1, i.e. identical to the costs of the overall optimal alignment γ_3 as shown in Figure 3.

Having defined the adapted cost function, the fitness values associated with the optimal subalignments per sublog and subnet can then be aggregated. This gives a decomposed fitness metric.

Definition 21 (Decomposed fitness metric)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log and let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$.*

Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of SN. For all $1 \leq i \leq n$, $SN^i = (N^i, I^i, O^i)$ is a subnet with an observable activity set $A_v^i = A_v(SN^i)$.

For a log trace $\sigma_L \in L$, $\sigma_L^i = \sigma_L{\restriction_{A_v^i}}$ is the projection of σ_L on the activity set of subnet SN^i.

$$fit_D(\sigma_L, SN, \delta) = 1 - \frac{\sum_{i \in \{1,\ldots,n\}} \delta_D(\lambda(\sigma_L^i, SN^i))}{move_M(SN) + move_L(\sigma_L)}$$

For an event log L, its decomposed fitness metric is computed as follows:

$$fit_D(L, SN, \delta) = 1 - \frac{\sum_{\sigma_L \in L} \sum_{i \in \{1,\ldots,n\}} \delta_D(\lambda(\sigma_L^i, SN^i))}{|L| \times move_M(SN) + \sum_{\sigma_L \in L} move_L(\sigma_L)}$$

In the decomposed fitness metric, the misalignment costs of each subalignment are first aggregated using the adapted cost function. Afterwards, the total is normalized using the same value as the undecomposed relative fitness metric so that both metric values are normalized in the same manner. In the paper [30], it is shown that the decomposed fitness metric provides an upper bound to the fitness computed using the full alignment between the overall log and model.

Let us consider again the trace $\sigma_3 = \langle a, e, i, l, d, g, j, h, k, n, p, q\rangle$ and the valid decomposition D_1 in Figure 6. Assuming that for $SN_1^1, \ldots, SN_1^9$, λ gives the subalignments $\gamma_3^1, \ldots, \gamma_3^9$ as shown in Figure 7. The decomposed fitness metric between the trace and the subnets is computed as $fit_{D_1}(\sigma_3, SN_1, \delta_1) = \frac{23}{24} \approx 0.958$. This is identical to the fitness value for the overall trace and the overall net.

Similarly, the formula can be applied to the log. Let the three subalignments shown in Figure 8, Figure 9 and Figure 7 be the optimal subalignments that correspond to the traces σ_1, σ_2 and σ_3. The decomposed fitness value of the event log L_1 and SN_1 is $fit_{D_1}(L_1, SN_1, \delta_1) = 1 - \frac{0\times20+2\times5+(\frac{1}{2}+\frac{1}{2})\times5}{12\times30+13\times20+12\times5+12\times5} = \frac{145}{148} \approx 0.980$. This is again identical to the fitness value for the overall log and the overall net. As such, the approach has decomposed the conformance checking problem whilst providing a conformance value for the overall log and net as output. However, this is not always the case, and cannot be generalized for the general case but only for cases satisfying specific properties. These properties relate to alignment moves corresponding to border activities.

3.4. Total border agreement

$$\gamma_6^1 = \begin{vmatrix} a \\ a \\ t_1 \end{vmatrix} \quad \gamma_6^2 = \begin{vmatrix} a & b \\ a & b \\ t_1 & t_2 \end{vmatrix} \quad \gamma_6^3 = \begin{vmatrix} a & d \\ a & d \\ t_1 & t_4 \end{vmatrix} \quad \gamma_6^4 = \begin{vmatrix} b & e & i & l \\ b & e & i & l \\ t_2 & t_5 & t_9 & t_{13} \end{vmatrix} \quad \gamma_6^5 = \begin{vmatrix} \\ \\ \end{vmatrix}$$

$$\gamma_6^6 = \begin{vmatrix} d & g & h & n & j & k & \gg \\ d & g & h & \gg & j & k & n \\ t_4 & t_7 & t_8 & & t_{11} & t_{12} & t_{15} \end{vmatrix} \quad \gamma_6^7 = \begin{vmatrix} l & p \\ l & p \\ t_{13} & t_{17} \end{vmatrix} \quad \gamma_6^8 = \begin{vmatrix} n & p \\ n & p \\ t_{15} & t_{17} \end{vmatrix} \quad \gamma_6^9 = \begin{vmatrix} p & q \\ p & q \\ t_{17} & t_{18} \end{vmatrix}$$

Figure 10: Subalignments between the trace $\sigma_6 = \langle a, b, e, i, l, d, g, h, n, j, k, p, q\rangle$ in event log L_2 and valid decomposition D_1 in Figure 6

As mentioned earlier, the decomposed fitness does not always match the overall fitness metric in the general case. That is because the legal moves involving a particular activity may differ from one subnet to another. Let us consider the trace $\sigma_6 = \langle a, b, e, i, l, d, g, h, n, j, k, p, q\rangle$ and the valid decomposition D_1 in Figure 6. A set of optimal subalignments between the trace and the subnets is shown in Figure 10. According to the system net SN_1 in Figure 1, transition t_{15} with label n is to be executed after transitions t_{11} with label j and t_{12} with label k; in the trace σ_6, t_{15} is executed before t_{11} and t_{12}. This results in a log move of activity n at the fourth position and a model move of transition t_{15} with label n at the seventh position of alignment γ_6^6.

As activities j and k are not present in the subnet SN_1^8, the move in the log and the move in the model are synchronized for transition n at alignment γ_6^8. Therefore, the moves involving border activity n are not identical between subalignments γ_6^6 and γ_6^8; the moves involving border activity n in the two subalignments are not in agreement. In this case, the decomposed fitness metric would not result in a value that is equal to the fitness value of the overall log and the overall net.

To compute the exact fitness value, a specific property must be satisfied: sequences of moves involving the same border activity have to be in agreement across all subalignments. We define that property as *border agreement* of subalignments and we formalize it as follows:

Definition 22 (Border agreement)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log and let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$.*

Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of SN. For all $1 \leq i \leq n$, $SN^i = (N^i, I^i, O^i)$ is a subnet with an observable activity set $A_v^i = A_v(SN^i)$.

For a border activity $a \in A_b(D)$, let $a_{LM} = \{(a, (a, t)), (\gg, (a, t)), (a, \gg)\}$ be the set of legal moves for activity a, where $t \in T$ such that $l(t) = a$.

Let $SN^i \in SN_b(a, D)$ be a subnet that has the border activity a. For a log trace $\sigma_L \in L$, $\sigma_L^i = \sigma_L{\restriction}_{A_v^i}$ is the projection of σ_L on the activity set of SN^i. $\gamma^i \in A_{LM}^*$ denotes an optimal alignment between the sublog trace σ_L^i and some complete firing sequence of a subnet $\sigma_M^i \in \phi_f(SN^i)$.

The set of subalignments $\gamma^1, \ldots, \gamma^n$ are under border agreement on a border activity $a \in A_b(D)$ if, and only if, $\gamma^i{\restriction}_{a_{LM}} = \gamma^j{\restriction}_{a_{LM}}$, for all $SN^i, SN^j \in SN_b(a, D)$.

The set of subalignments $\gamma^1, \ldots, \gamma^n$ are under total border agreement (t.b.a.) if, and only if, border agreement is achieved one by one on all the border activities in $\gamma^1, \ldots, \gamma^n$ following the order of their occurrences across $\gamma^1, \ldots, \gamma^n$, starting with the first occurring border activity in subnet $SN^i \in D$ where $I^i \geq I$.

Given the properties of a sequence, there is border agreement if the following three conditions are satisfied:

1. $\gamma^i{\restriction}_{a_{LM}}$ has an equal number of moves as $\gamma^j{\restriction}_{a_{LM}}$.
2. $\gamma^i{\restriction}_{a_{LM}}$ has the same move types as $\gamma^j{\restriction}_{a_{LM}}$, i.e. if $\gamma^i{\restriction}_{a_{LM}}$ has one log move, then $\gamma^j{\restriction}_{a_{LM}}$ must also have one log move.
3. The order of moves in $\gamma^i{\restriction}_{a_{LM}}$ and $\gamma^j{\restriction}_{a_{LM}}$ are the same.

Note that if the subalignment γ^i is empty then the projection of the subalignment will also be an empty sequence.

For example, there is total border agreement between the valid decomposition D_1 in Figure 6 and the log trace $\sigma_3 = \langle a, e, i, l, d, g, j, h, k, n, p, q\rangle$. As shown by the corresponding subalignments in Figure 7, all the moves corresponding to each border activity are under border agreement.

Contrastingly, the subalignments of the trace $\sigma_6 = \langle a, b, e, i, l, d, g, h, n, j, k, p, q\rangle$ are not under total border agreement. The moves involving border activity n in subalignment γ_6^6 is a log move followed by a model move which does not "agree" with the synchronous move in subalignment γ_6^8.

3.5. Properties of decomposed fitness

We now formalize the properties of the decomposed fitness metric with consideration to the border agreements of subalignments, i.e., if the total border agreement is not satisfied, the decomposed fitness metric corresponds only with an upper bound of the overall metric; when the total border agreement is satisfied, the decomposed fitness matches exactly the overall fitness. This results will be used in Sections 4 and 5 as part of the proposed conformance methods.

First, we note that the metric is an upper bound to the fitness metric computed using the full alignment between the overall log and model. This has been shown as Theorem 3 in the earlier paper [30].

However, under total border agreement, we can prove that the decomposed fitness value from the set of subalignments corresponds to the exact fitness value computed with the overall

alignment. This applies to the decomposed log fitness value as well. We extend the properties of the decomposed fitness metric to include the capability of computing a conformance result that corresponds to an exact overall fitness value regardless of the conformance level.

Theorem 1 (Exact value for decomposed fitness metric under total border agreement)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log and let $\sigma_L \in L$ be a log trace. Let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net and let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of SN. For all $1 \leq i \leq n$, $SN^i = (N^i, I^i, O^i)$ is a subnet with an observable activity set $A^i_v = A_v(SN^i)$. $\sigma^1_L, \ldots, \sigma^n_L$ are the subtraces from the projection of σ_L onto the activity sets of $SN^1, \ldots, SN^n$ such that $\sigma^i_L = \sigma_L\restriction_{A^i_v}$. γ^i is an optimal subalignment between σ^i_L and some complete firing sequence of the corresponding subnet $\sigma^i_M \in \phi_f(SN^i)$.*

Let $\gamma^1, \ldots, \gamma^n$ be the set of subalignments and let them be under total border agreement. The decomposed fitness metric computed using this set of subalignments equals the relative fitness metric computed with the overall alignment between σ_L and SN:

$$fit(\sigma_L, SN, \delta) = fit_D(\sigma_L, SN, \delta)$$

For the log L, if for all the log traces in L, their corresponding set of subalignments is under total border agreement,

$$fit(L, SN, \delta) = fit_D(L, SN, \delta)$$

Proof. The paper [45] presents a stitching function that merges a set of subalignments into an optimal alignment if all the legal moves from the subalignments can be stitched together without conflict and a trace pseudo-alignment otherwise. We prove by contradiction that under total border agreement, the set γ^1, $\ldots, \gamma^n$ can always be stitched together as an optimal alignment without conflict.

Suppose that $\gamma^1, \ldots, \gamma^n$ is a set of subalignments and are under total border agreement but cannot be stitched together without conflict. The set of subalignments are comprised of only legal moves and therefore it must be that there is a stitching conflict for particular moves between the subalignments.

Let $(a_l, m) \in A_{LM}$ be a move involved in the first conflict as $\gamma^1, \ldots, \gamma^n$ are being stitched together. As for a conflict we need another move, and hence another subnet, we know that $|SN_b(\alpha(a_l, m), D)| > 1$. Therefore, the activity $\alpha(a_l, m)$ must be a border activity. Under total border agreement, all subalignments for this activity are the same across all subalignments. Therefore, there cannot be a conflict.

Furthermore, since border agreement is achieved according to the occurrence order of the border activities across the subalignments, $\gamma^1, \ldots, \gamma^n$ can be stitched together without occurring any conflicts.

As a result, $\gamma^1, \ldots, \gamma^n$ can always be merged into an optimal alignment between σ and a complete firing sequence $\sigma_M \in \phi_f(SN)$. With the merged optimal alignment γ, the sum of misalignment costs associated with $\gamma^1, \ldots, \gamma^n$ under the adapted cost function equals the misalignment cost of γ. Therefore, the decomposed fitness value with $\gamma^1, \ldots, \gamma^n$ equals the fitness value with γ.

The decomposed log fitness metric compares the sum of the misalignment costs from the sets of subalignments with the sum of the worst-case scenario costs for all the log traces. Since the set of subalignments corresponding to each log trace is under total border agreement, all sets of subalignments can be merged into optimal alignments. This means that the sum of the misalignment costs under the adapted cost function equals the sum of optimal misalignment costs associated with all the log traces. The decomposed log fitness value equals the log fitness value. □

As previously shown, the event log L_1 has the exact same value under both the decomposed fitness metric and the undecomposed fitness metric at about 0.980.

We now propose two novel alignment-based conformance checking methods that use the results from Theorem 1 and the upper bound property of the decomposed fitness metric.

4. Recomposing method for exact decomposed fitness

As previously mentioned, the main contribution of this paper is to make decomposition techniques more applicable in computing the overall conformance between a net and log. We now use the border properties formalized in the last section in a novel method to compute the overall conformance between a net and a log: *recomposing conformance*.

Similar to many existing decomposition techniques, the net is decomposed into subnets by activities and the log is projected onto the subnets to create sublogs. For each pair of subnet and sublog, alignments are created for the subnet and each subtrace in the sublog. Each of the sublogs can be analyzed in parallel, and together with the reduced size and complexity of the net, the approach obtains a significant performance gain in time [20, 47, 39, 44, 8]. Following the per subcomponent computation, the results are aggregated if there was agreement on the border. Otherwise, some disagreeing subnets are "recomposed" to get a more coarse-grained decomposition of the net, in which the disagreeing subnets have become a single subnet. As a result, they cannot disagree anymore. Traces whose conformance results disagreed on the border previously are recomputed under the new decomposition. This iterative process is repeated until completion. Figure 11 shows an overview of the summarized method. In this paper, we apply this approach to decompose the relative fitness metric.

4.1. Decomposed fitness metric

As illustrated in Figure 11, the first step of the approach is to decompose the system net. This enables the performance gain in the conformance checking process. The decomposition of the system net has to fulfill the properties of a valid decomposition, first defined in [30].

Following the initial decomposition, the log is projected onto the subnets of the decomposition to get the sublogs for alignment.

The alignment of subnets and sublogs and the computation of the decomposed fitness metric are marked as step two and

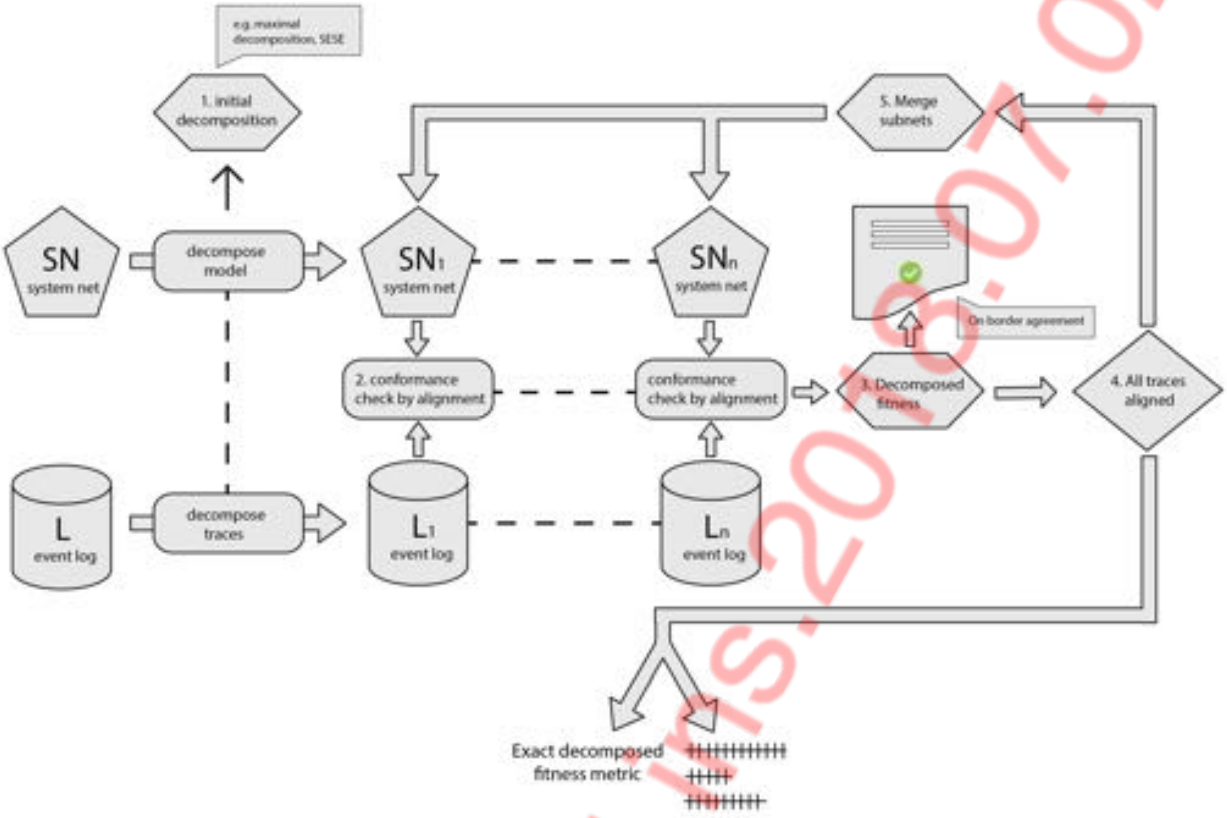


Figure 11: Overview of the exact decomposed conformance metric

three of the algorithm in Figure 11. Decomposed fitness values of subalignments computed under total border agreement are recorded. Afterwards, their associated traces are taken out of the process and are marked as completed. As for the remaining traces, we resolve their border agreement problems by selectively "recomposing" subnets by their matching border activities on which they disagree.

4.2. Subnet recomposition

As illustrated in step five in Figure 11, the existing set of subnets are recomposed as a new set of subnets. Recomposing subnets by their matching border activities resolves any border agreement problems associated with the recomposed border activities as it ceases to be shared between multiple subnets. "Recomposition" can be done on single or multiple border activities and different selection criteria can be used to select the border activities. For example, recomposing the subnets by multiple border activities is likely to resolve more border agreement problems than if the subnets were to be recomposed on only one border activity. However, there would be less performance gain under the multiple border activities selection approach as the resulting subnets would be larger and more complex.

For the sake of simplicity, in this paper we consider selecting the single activity that has the highest number of border agreement problems. This selection criterion resolves the most problematic border activity at each recomposition. Following the valid decomposition in Figure 6, for event log L_2, there is a

t_1	t_2	t_3	t_4	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}
0	0	0	0	0	0	5	0	0

Figure 12: Vector showing the number of border agreement problems at each border activity for event log L_2

border agreement issue with trace $\sigma_6 = \langle a, b, e, i, l, d, g, h, n, j, k, p, q \rangle$ on activity n. At the recomposition, activity n is identified and selected since it is the activity with the highest number of border agreement issues as shown in Figure 12. Retrieval of all the subnets that have the selected border activity is done by the shared function defined in Definition 19. These subnets are then merged, after which the set of recomposed subnets is a new valid decomposition of the system net.

Theorem 2 (Recomposition results in valid decomposition)**.** *Let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$. Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of SN.*

Let $a \in A_b(D)$ be a border activity that is shared between $|SN_b(a, D)|$ subnets in D. Recomposing $SN^1, SN^2, \ldots, SN^n$ on border activity a joins all the subnets in $SN_b(a, D)$ on the activity a. Following the recomposition, which leads to a new decomposition D', $|SN_b(a, D')| = 1$ and $a \notin A_b(D')$ i.e., a ceases to be a border activity.

Let $A' \subseteq A_b(D)$ be a subset of the border activities of D. Let D' be the recomposition of D on A', i.e., decomposition D' is recomposed on all activities $a \in A'$.

$D' \in \mathcal{U}_{SN}$ i.e., D' is a valid decomposition of SN.

Proof. Recomposing $SN^1, SN^2, \ldots, SN^n$ on a particular border activity $a \in A'$ joins the set of subnets $SN_b(a, D)$ on activity a into one single subnet $SN^+ = \bigcup SN_b(a, D)$. SN^+ has the same set of edges as $SN_b(a, D)$ has. Therefore, $D' = (D \setminus SN_b(a, D)) \cup \{SN^+\}$ is a partitioning of the edges in SN. Given that there was no creation of a new transition or a new place in the recomposition to SN^+, it follows trivially that D' is a valid decomposition. The recomposition on any remaining border activities $a' \in A' \setminus \{a\}$ can be done in the same manner, and hence also yields a valid decomposition. □

4.3. New border agreement problems following recomposition

While recomposition can solve existing border agreement problems at merged border activities, new border agreement problems may arise at locations where there were no issues previously. Here we showcase such a case using the running example net SN_1 in Figure 1 to explain the underlying intuition. This example is slightly more involved than the previous log examples in that it involves the loop construct of the net.

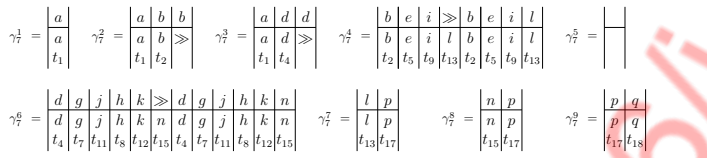


Figure 13: Subalignments between trace $\sigma_7 = \langle a, b, e, i, d, g, j, h, k, b, e, i, l, d, g, j, h, k, n, p, q \rangle$ and valid decomposition D_1 in Figure 6

Consider trace $\sigma_7 = \langle a, b, e, i, d, g, j, h, k, b, e, i, l, d, g, j, h, k, n, p, q \rangle$. Replaying trace σ_7 on decomposition D_1 in Figure 6 would produce the set of subalignments as shown in Figure 13. There are border agreement problems with border activities b, d, l, and n. This is caused by the fact that SN_1^2, SN_1^3, SN_1^7, and SN_1^8 are not aware that most of the subsequent activities of the branches associated to border activity b and d have been executed in trace σ_7. This indicates that border activities b, d, l, and n should not be marked as log moves. This shows that the information asymmetry or disparity between subnets on conformance is the cause of the border agreement problems. To resolve the border agreement problems, the subnets SN_1^2, SN_1^3. SN_1^4. SN_1^5. SN_1^6. SN_1^7. and SN_1^8 are recomposed so that activity b, d, l, and n are no longer border activities. This produces the valid decomposition D_2 as shown in Figure 14. Replaying trace σ_7 on decomposition D_2 would produce the set of subalignments as shown in Figure 15. There are border agreement problems with border activities p. Similar to the previous iteration, this is caused by the fact that subnet SN_1^{13} is not aware of the need for a model move on border activity p. One further recomposition on border activity p is required until the set of subalignments can be merged as the overall alignment as shown in Figure 16.

4.4. Iterative conformance checking

As such, at each iteration, a new valid decomposition is created from the recomposed subnets. Then, traces which had border agreement problems in the previous iteration are projected onto the new valid decomposition to be rechecked. Recomposition and conformance checking can be repeated until the decomposed fitness metric of all traces are computed under total border agreement.

At completion, the set of alignments and the decomposed fitness value of the log are returned as output. This fitness value corresponds to the exact fitness value of the overall log and overall net.

While it can be crucial to have an exact fitness value of the overall log and net, in some scenarios an interval value may suffice. This is addressed in the next section.

5. Recomposing method for interval decomposed fitness

In some conformance checking scenarios, it may be sufficient to have an interval fitness value of the overall log and the overall net. For example, in the selection of candidates of genetic algorithms for the creation of a new generation of models, an interval value of each candidate model might already be sufficient to decide whether or not it should be kept. In addition, not requiring an exact fitness value can increase performance gain as there can potentially be less iterations of the recomposition and checking steps. Moreover, we note that while recomposition guarantees that any border agreement problems at the merged border activities will be gone in the next conformance checking iteration, new border agreement problems may arise at other border activities which had no problems previously. Lastly, it is possible that there are a few traces whose decomposed fitness value cannot be computed under total border agreement for many iterations or unless the overall trace and the overall net are used. As shown in Figure 17, the exact decomposed conformance algorithm is modified to compute an interval decomposed fitness value.

5.1. Interval decomposed fitness conformance

As previously mentioned, the decomposed fitness metric has been shown to be an upper bound to the fitness metric of the overall log and the overall net. Using this property, an interval can be computed such that the fitness value of the overall log and the overall net is within the interval.

Definition 23 (Decomposed fitness interval)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log and let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$. Let $\sigma_L \in L$ be a log trace.*

Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of SN.

$$fit_D^{int}(\sigma_L, SN, \delta) = [fit_D^{low}(\sigma_L, SN, \delta), fit_D(\sigma_L, SN, \delta)]$$

$fit_D^{low}(\sigma_L, SN, \delta)$ defines the lower bound of the decomposed fitness interval such that it equals the decomposed fitness metric

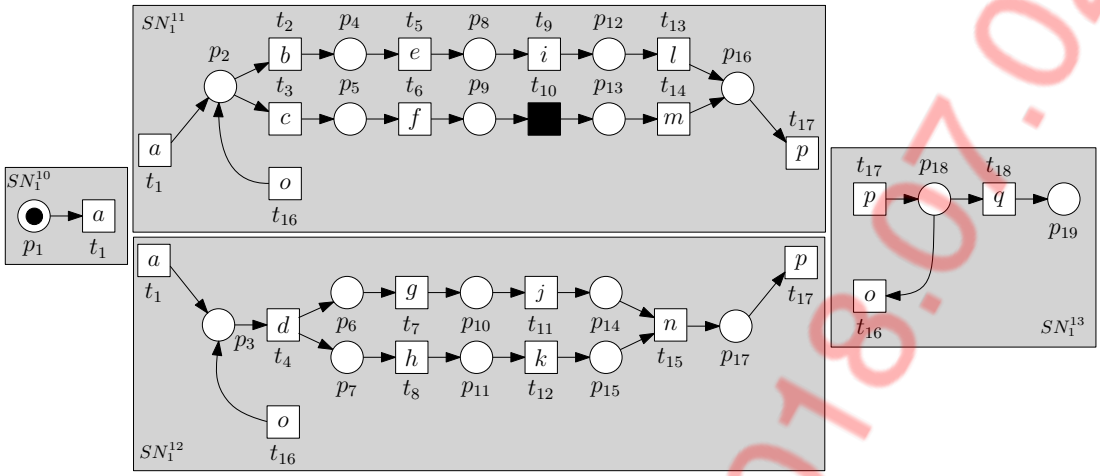


Figure 14: Valid decomposition D_2 of system net SN_1 following the recomposition of subnets SN_1^2, SN_1^3, SN_1^4, SN_1^5, SN_1^6, SN_1^7 of decomposition D_1 in Figure 6

$$\gamma_7^{10} = \begin{array}{|c|}\hline a \\ a \\ t_1 \\ \hline \end{array} \qquad \gamma_7^{12} = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline a & d & g & j & h & k & \gg & \gg & \gg & d & g & j & h & k & n & p \\ a & d & g & j & h & k & n & p & o & d & g & j & h & k & n & p \\ t_1 & t_4 & t_7 & t_{11} & t_8 & t_{12} & t_{15} & t_{17} & t_{16} & t_4 & t_7 & t_{11} & t_8 & t_{12} & t_{15} & t_{17} \\ \hline \end{array}$$

$$\gamma_7^{11} = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline a & b & e & i & \gg & \gg & \gg & b & e & i & l & p \\ a & b & e & i & l & p & o & b & e & i & l & p \\ t_1 & t_2 & t_5 & t_9 & t_{13} & t_{17} & t_{16} & t_2 & t_5 & t_9 & t_{13} & t_{17} \\ \hline \end{array} \qquad \gamma_7^{13} = \begin{array}{|c|c|}\hline p & q \\ p & q \\ t_{17} & t_{18} \\ \hline \end{array}$$

Figure 15: Subalignments between trace $\sigma_7 = \langle a, b, e, i, d, g, j, h, k, b, e, i, l, d, g, j, h, k, n, p, q\rangle$ and valid decomposition D_2 in Figure 14

$$\gamma_7 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline a & b & e & i & \gg & d & g & j & h & k & \gg & \gg & \gg & b & e & i & l & d & g & j & h & k & n & p & q \\ a & b & e & i & l & d & g & j & h & k & n & p & o & b & e & i & l & d & g & j & h & k & n & p & q \\ t_1 & t_2 & t_5 & t_9 & t_{13} & t_4 & t_7 & t_{11} & t_8 & t_{12} & t_{15} & t_{17} & t_{16} & t_2 & t_5 & t_9 & t_{13} & t_4 & t_7 & t_{11} & t_8 & t_{12} & t_{15} & t_{17} & t_{18} \\ \hline \end{array}$$

Figure 16: Alignment between trace $\sigma_7 = \langle a, b, e, i, d, g, j, h, k, b, e, i, l, d, g, j, h, k, n, p, q\rangle$ and system net SN_1 in Figure 1

if the optimal subalignments $\gamma^1, \ldots, \gamma^n$ (where $\gamma^i = \lambda(\sigma_L^i, SN^i)$) are under total border agreement (t.b.a.):

$$cost_D(\sigma_L, SN, \delta) = \begin{cases} \sum_{i\in\{1,\ldots,n\}} \delta_D(\lambda(\sigma_L^i, SN^i)) & \text{if under t.b.a.;} \\ move_M(SN) + move_L(\sigma_L) & \text{otherwise.} \end{cases}$$

$$fit_D^{low}(\sigma_L, SN, \delta) = 1 - \frac{cost_D(\sigma_L, SN, \delta)}{move_M(SN) + move_L(\sigma_L)}$$

The decomposed fitness interval of the event log L is computed as follows:

$$fit_D^{int}(L, SN, \delta) = [fit_D^{low}(L, SN, \delta), fit_D(L, SN, \delta)],$$

$$fit_D^{low}(L, SN, \delta) = 1 - \frac{\sum_{\sigma_L\in L} cost_D(\sigma_L, SN, \delta)}{|L| \times move_M(SN) + \sum_{\sigma_L\in L} move_L(\sigma_L)}$$

Consider the subalignments for the trace $\sigma_6 = \langle a, b, e, i, l, d, g, h, n, j, k, p, q\rangle$ in Figure 10. Due to the border agreement problem with activity n, the decomposed fitness interval for the trace is $fit_{D_1}^{int}(\sigma_6, SN_1, \delta_1) \approx [0, 0.962]$. The decomposed fitness interval for event log L_2 is computed as $fit_{D_1}^{int}(L_2, SN_1, \delta_1) \approx [0.815, 0.979]$. We note that for both the trace σ_6 and the log L_2, their fitness values are within the respective intervals: $fit(\sigma_6, SN_1, \delta_1) = 0.923 \in [0, 0.962]$ and $fit(L_2, SN_1, \delta_1) = 0.973 \in [0.815, 0.979]$.

Theorem 3 (Overall fitness is within interval fitness)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log and let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$.*

Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of SN. For all $1 \le i \le n$, $SN^i = (N^i, I^i, O^i)$ is a subnet with an observable activity set $A_v^i = A_v(SN^i)$.

For a log trace $\sigma_L \in L$, $\sigma_L^i = \sigma_L{\restriction_{A^i}}$ is the projection of σ_L on the activity set of subnet SN^i.

$$fit(\sigma_L, SN, \delta) \in fit_D^{int}(\sigma_L, SN, \delta)$$

For log L,

$$fit(L, SN, \delta) \in fit_D^{int}(L, SN, \delta)$$

Proof. We prove the theorem by cases.

For $SN^1, \ldots, SN^n$, let $\gamma^1, \ldots, \gamma^n$ be the optimal subalignments where $\gamma^i = \lambda(\sigma_L^i, SN^i)$ is the optimal subalignment of σ_L^i and SN^i. There are two cases.

Case 1: $\gamma^1, \ldots, \gamma^n$ are under total border agreement

$fit_D^{int}(\sigma_L, SN, \delta) = [fit_D(\sigma_L, SN, \delta), fit_D(\sigma_L, SN, \delta)]$. By Theorem 1, the fitness of the overall trace and net equals the decomposed fitness metric. Hence the overall fitness value is within the interval.

Case 2: $\gamma^1, \ldots, \gamma^n$ are not under total border agreement

$fit_D^{int}(\sigma_L, SN, \delta) = [0, fit_D(\sigma_L, SN, \delta)]$. The overall fitness is within the interval since the decomposed fitness metric computes an upper bound of the fitness value for the overall trace and net.

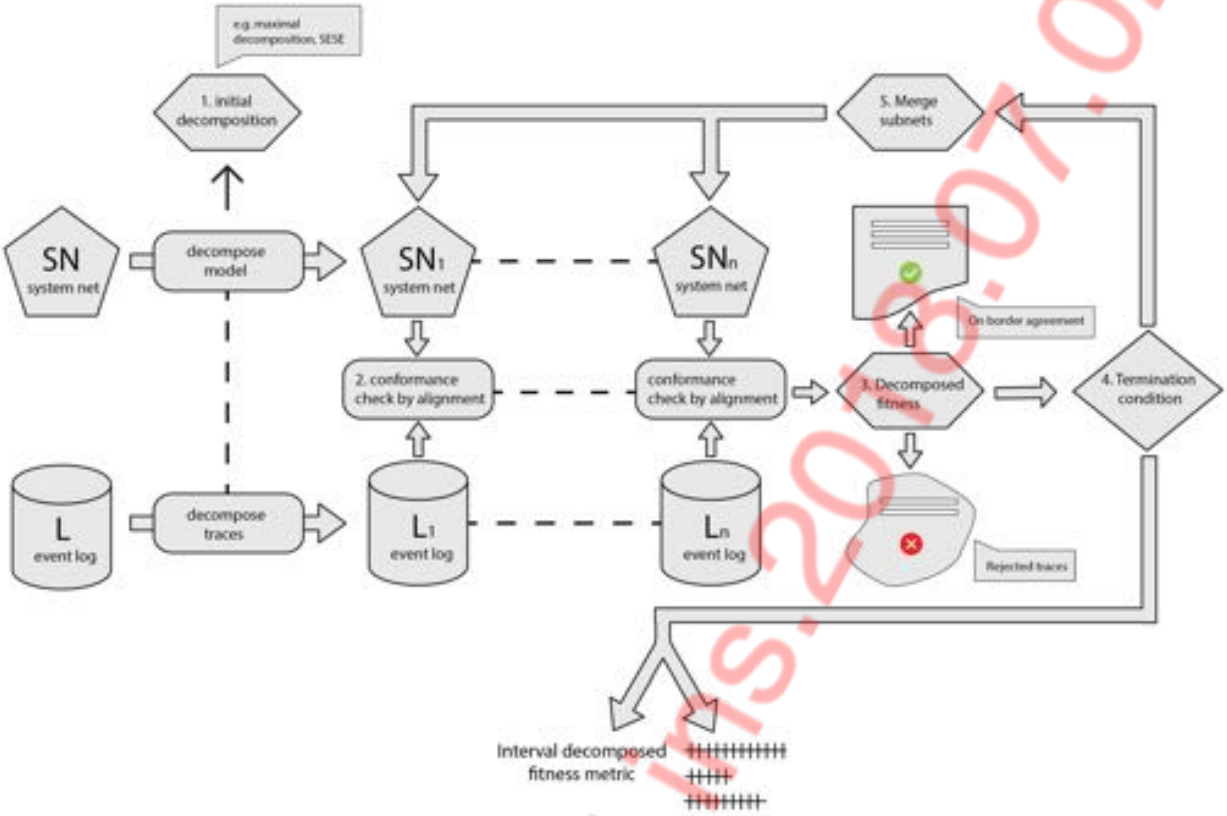


Figure 17: Overview of the interval decomposed conformance metric

For the decomposed log fitness interval, the upper bound of the interval is the decomposed log fitness which has been shown to be an upper bound to the exact overall log fitness. The lower bound of the interval uses the total misalignment costs of the subalignments under the adapted cost function if there is total border agreement. Otherwise, it defaults to the worst-case scenario cost. In Theorem 1, it was proven that the exact misalignment cost is obtained if the set of subalignments is under total border agreement. This means that the sum of misalignment cost is an upper bound to the overall misalignment costs and therefore the lower bound of the decomposed log fitness interval is a lower bound to the exact overall log fitness value. Hence, the exact overall log fitness is within the decomposed log fitness interval. □

As previously shown, the fitness value for the log L_2, $fit(L_2, SN_1, \delta_1) \approx 0.973$ is within its decomposed fitness interval $fit^{int}_{D_1}(L_2, SN_1, \delta_1) \approx [0.815, 0.979]$. To compute the interval decomposed metric, traces are exempted from the computation of an exact decomposed fitness value through trace reject and termination conditions.

5.2. Trace reject and termination conditions

As previously mentioned, it is possible that there are a few problematic traces which cannot be checked within a short time. As such, trace reject and termination conditions can be used to configure the balance between result quality and computation time.

Let $L \in \mathcal{B}(A^*)$ be an event log and let $SN = (N, I, O) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$. Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of SN. For all $1 \leq i \leq n$, $SN^i = (N^i, I^i, O^i)$ is a subnet with an observable activity set $A^i_v = A_v(SN^i)$.

For a log trace $\sigma_L \in L$, $\sigma^i_L = \sigma_L \restriction_{A^i}$ is the projection of σ_L on the activity set of subnet SN^i. For $1 \leq i \leq n$, γ^i is a subalignment between subnet SN^i and subtrace σ^i_L. $\gamma^1, \ldots, \gamma^n$ is the set of subalignments corresponding to σ_L.

Following the computation of the decomposed fitness metric in step three of Figure 17, the traces in log L are partitioned over C, R, and B, i.e., $L = C + R + B$. Traces that are computed under total border agreement are added to the accepted traces multiset $C \in \mathcal{B}(A^*)$. Due to the number of border agreement issues or alignment time, traces that are not computed under total border agreement may be added to the rejected traces multiset $R \in \mathcal{B}(A^*)$. Otherwise, traces are added to the to-be-aligned multiset $B \in \mathcal{B}(A^*)$.Let the recomposition algorithm be at its kth iteration.

For trace σ_L, if its corresponding set of subalignments $\gamma^1, \ldots, \gamma^n$ is not under total border agreement, the trace is added to multiset R if,

- the number of border agreement issues of subalignments $\gamma^1, \ldots, \gamma^n$ is greater than the given threshold $x \in \mathbb{N}$, i.e., $|\{a \in A_b(D) \mid \exists_{1 \leq i < j \leq n} \gamma^i \restriction_{\{a\}} \neq \gamma^j \restriction_{\{a\}}\}| > x$.

- The time spent on aligning a subtrace σ_L^i with a subnet SN^i is higher than the given threshold $y \in \mathbb{Q}$, e.g. a threshold of $y = 1$ millisecond per subalignment.

A rejected trace $\sigma'_L \in \{\sigma_L \in L \mid R(\sigma_L) > 0\}$ is not checked in future iterations. The criteria for the trace reject conditions can be adjusted to reflect the trade-off between result quality and computation time. A decomposed fitness interval is computed if $\exists_{\sigma_L \in L} R(\sigma_L) > 0$, i.e., at least one trace is rejected.

At the end of each iteration, termination conditions are examined to decide whether to proceed to the next iteration of the algorithm. If termination conditions are met, the decomposed fitness interval $fit_D^{int}(L, SN, \delta)$ is returned. These termination conditions are defined on the log level. The termination conditions are as follows:

- All log traces have been either aligned under total border agreement or have been rejected, i.e., $C + R = L$.
- Surpassing the overall time threshold $z \in \mathbb{Q}$ for conformance checking, e.g., if more than $z = 1$ minute is needed in the conformance checking process.
- Having aligned a target percentage of traces $0 \leq v \leq 1$ in the log under total border agreement, i.e., $\frac{\sum_{\sigma_L \in L} C(\sigma_L)}{\sum_{\sigma_L \in L} L(\sigma_L)} \geq v$. For example, if more than $v = 0.9 = 90\%$ of the traces in the event log have been aligned under total border agreement.
- The overall fitness interval value $0 \leq w \leq 1$ is narrow enough, i.e., $fit_D(L, SN, \delta) - fit_{SN}^{low}(D, L, \delta) \leq w$. For example, if the interval range is less than $w = 0.1$.
- The maximum number of iterations $m \in \mathbb{N}$ is reached, i.e., $k \geq m$. For example, if $k = 100$ iterations have been done.

As illustrated in Figure 17, if the termination conditions are met in step four, the algorithm terminates and returns a decomposed fitness interval value $fit_D^{int}(L, SN, \delta)$ and the alignments of traces whose decomposed fitness value had been computed under total border agreement.

6. Implementation and Evaluation

We have implemented the proposed conformance checking framework, and evaluated it on both artificial and real-life datasets. Our evaluations demonstrate the following main contributions:

1. Recomposing conformance checking enables replays of model-log pairs that were previously not feasible under the monolithic approach. This increases the applicability of alignment-based conformance checking.
2. Logs associated to large models can take a tremendous amount of time to replay under the current monolithic approach. For these logs, the proposed recomposition strategy can lead to significant performance gains, speeding up the conformance checking process. The significant performance gains are present both in noiseless and noisy scenarios.
3. Recomposing conformance checking allows configurable approximations of conformance through interval fitness values. The computation time of interval fitness values is often shorter than the needed time for exact fitness values under both the monolithic and recomposition approach. This means that end users can get an idea of the conformance level within shorter times, or whenever there is a hard time constraint.

The section is structured as follows: First, we discuss the implementation details, the characteristics of the datasets used in the experiments, and the general conditions applied on the experimentation. Second, the exact fitness computation and the resulting speed-ups and improvement in the feasability are evaluated in detail, in both noiseless and noisy scenarios. Third, we illustrate the improvement in the feasibility of the alignment-based approach for different time-constrained scenarios using the recomposition strategy, and the effect of the time limits in the interval narrowing. Finally, the applicability of the approach for real-life scenarios is shown.

6.1. Implementation, datasets, and evaluations

The presented recomposing conformance checking framework has been implemented as the *Replay with Recomposition* plugin in the *DecomposedReplayer* package of ProM6.7[1] [43]. *ProM* is an extensible framework that supports a wide variety of Process Mining techniques in the form of plug-ins. It is platform independent as it is implemented in Java, and can be downloaded free of charge.[2] Figure 19 shows a dialog box of the plugin at which the user can configure different values for the parameters introduced in the previous section, and Figure 18 shows the conformance analysis overview for one of the datasets tested. The implementation allows for the setting of all the parameters defined in the paper, and to manually select an initial decomposition for the approach (including the maximal decomposition if desired).

Figure 18: Resulting alignments for deviation diagnosis

[1] At this moment, ProM6.7 has not been officially released, consequently, ProM6.7beta was used instead. However, no significant differences are to be expected.

[2] http://www.promtools.org/

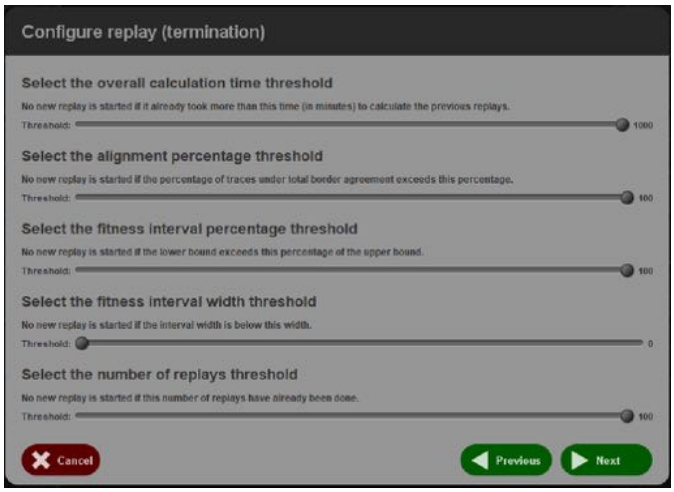


Figure 19: Dialog for *Replay using Recomposition*

Name	Source	|A|	|AND|	|XOR|	|Loop|	|Initial decomposition|
P241	synth	117	13	12	3	12
P246	synth	137	16	12	2	15
P272	synth	201	21	24	4	32
P275	synth	101	13	9	1	13
P284	synth	170	19	18	1	26
P285	synth	140	13	19	1	25
P291	synth	170	19	21	2	25
P297	synth	125	7	18	2	15
P307	synth	193	15	18	2	22
P313	synth	185	9	18	1	14
P347	synth	212	15	23	3	20
P381	synth	111	15	15	3	12
P383	synth	150	29	20	1	16
P430	synth	160	14	19	6	25
P436	synth	230	12	38	9	30

Table 1: Characteristics of the synthetic nets

The experiments were conducted including a wide-range of different datasets to represent the variability of possible scenarios. The datasets used include both synthetically generated datasets (represented as PX), and two real-life datasets (represented as BPIC201X). The process models of the synthetic datasets were randomly generated using the PLG2 tool [6], and they include large processes containing combinations of all the most common workflow patterns, such as XOR, AND, loops, or invisible transitions. The model characteristics are shown in Table 1. Logs were generated from the models using simulation, and different operations were applied to emulate different plausible noise scenarios: no noise, noise by removing events and noise by swapping. This is explained in detail in Section 6.3. The names of the datasets refer to their time of creation and have no relation to any property of the processes such as the number of activities or arcs. Each log has 1000 cases. Following the Open Data Science principles, the datasets are open and publicly available.[3] The experimental evaluation also include two real-life datasets – based on the real cases BPIC2012 [37] and BPIC2017 [38] – to illustrate the applicability and benefits of the proposed approach for real cases. Details on the dataset are presented in Section 6.6, and the models and logs used are also publicly available.

The evaluations were performed on a desktop with two processors Intel E5-2470, 8 Cores, 2.3 Ghz, 32 GB RAM, running Linux 2.6.32 and using a 64-bit version of Java 7 where 6GB of RAM was allocated to the Java VM. Note that the approach can be distributed over a network of computers, but for the reported experiments, we only used one computing node. Although decomposed parts could be analyzed in parallel on a distributed system, this has not been implemented yet. Each experiment was conducted 3 times.

Parameters	MaxRecomposing
GlobalDurationThreshold	3600 seconds
LocalDurationThreshold	80 seconds
RelativeIntervalThreshold	100%
AbsoluteIntervalThreshold	0
MaxConflictThreshold	100
AlignmentPercentageThreshold	100%
MaxIterationThreshold	200

Table 2: MaxRecomposing configuration

In the evaluations, we experimented with different configurations adjusting the termination thresholds. The configuration with all the parameter values adjusted to the maximum will be identified as the *MaxRecomposing* configuration as shown in Table 2. For the sake of generality, this is the configuration shown in the results unless it is stated otherwise. Moreover, a *Hide and Reduce* projection strategy was used from the three net projection strategies proposed in [41]. Net projection is used to decompose the overall net into subnets.

As previously explained, the recomposition approach is instantiated with an initial decomposition. Different strategies can be used to decompose a model into subcomponents, e.g., maximal decomposition [30], SESE-based decomposition [20], passage-based decomposition [29], or cluster-based decomposition [46]. For the sake of simplicity, the experiments use a manual decomposition where the decompositions are manually decided through visual inspection of the model. As shown in Figure 20, the partitions follow closely to a SESE-based decomposition [20]. As such, the initial decomposition is done without previous knowledge of the locations where there are conformance issues. This is the default decomposition method unless stated otherwise. The number of subnets at the initial decomposition of each model is shown in Table 1.

6.2. Exact fitness in noiseless scenarios

We first experimented with the simplest scenario where the model and log are perfectly fitting. In this section we present conformance checking results for logs without noise ("no-noise" logs). The goals of this section are:

- To compare the feasibility of the recomposition approach with the existing monolithic approach.

[3] DOI will be requested after the acceptance of the manuscript. Meanwhile, the datasets can be temporarily accessed in `http://www.jorgemunozgama.com/data/uploads/ds/isci17dataset.zip`.

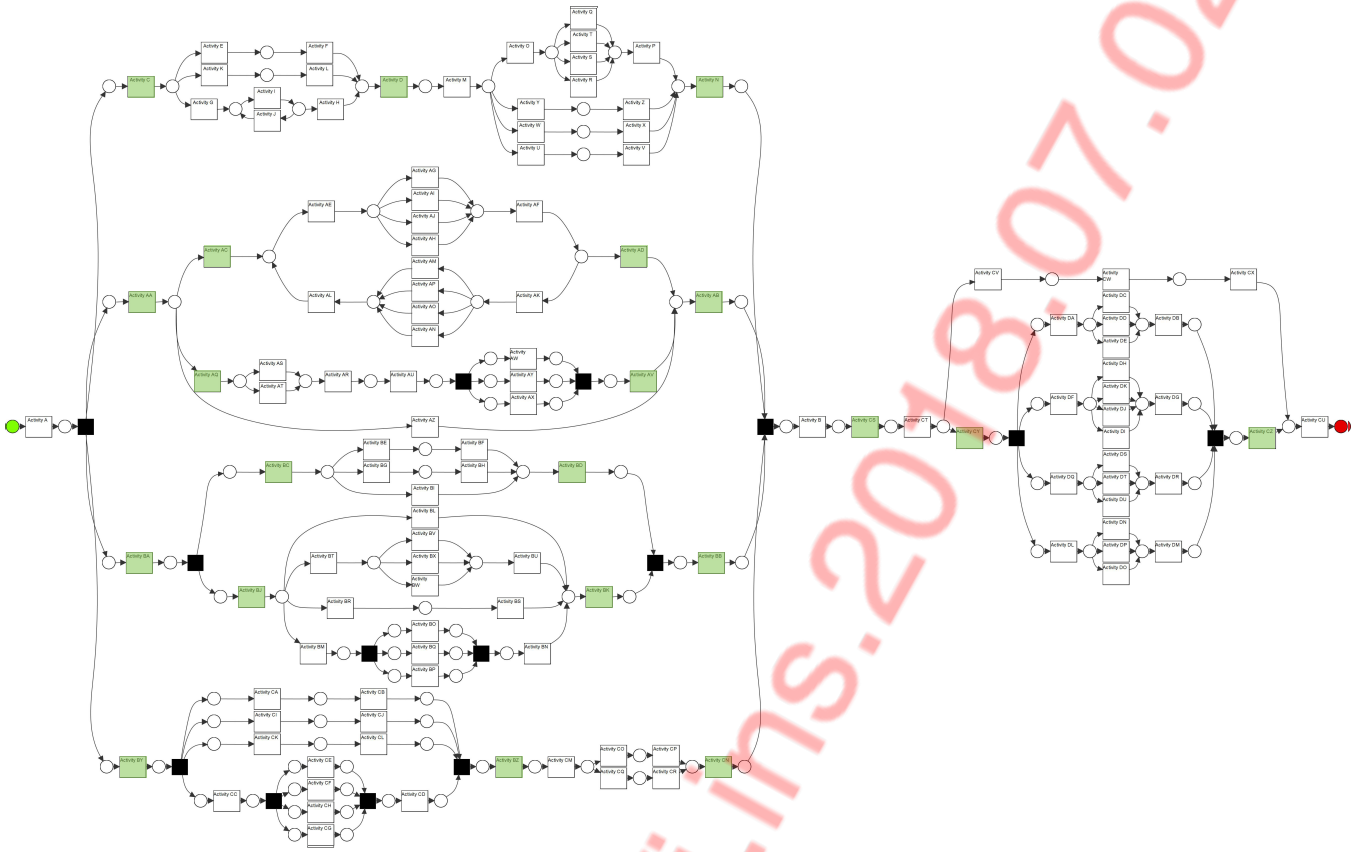


Figure 20: Initial manual decomposition of model P297 where border transitions are colored in green

- To compare and analyze computation times of replays under both approaches.

The synthetic process models and "no-noise" logs generated by the PLG2 tool are used in the experiments. "No-noise" logs are perfectly fitting with the corresponding models. This means that all the behaviour observed in the log can be matched to the behaviour modeled by the model so that the log can be replayed perfectly on the model and the fitness value equals to 1. We note that this is the least computationally intensive of all possible scenarios since the alignment algorithm can always match a log trace with a corresponding model trace as an alignment of solely synchronous moves. This means that the iterative process of the recomposition approach does not occur for the replays of these model-log pairs because moves associated with border transitions are always synchronous moves and in agreement.

There are 15 model-log pairs and for each dataset we conduct two experiments using the recomposition approach and the monolithic approach respectively. For each experiment, a 1 hour time limit is set such that replays which are still running after the time limit are stopped and deemed infeasible. Other than the replay feasibility, fitness, time, average speedup, and median speedup, we also report the number of activities ($|A|$) in the models, the number of log traces ($|L|$), the number of trace variants ($|\{\sigma \in L\}|$), and the average trace lengths ($\overline{\sigma}$) of the logs.

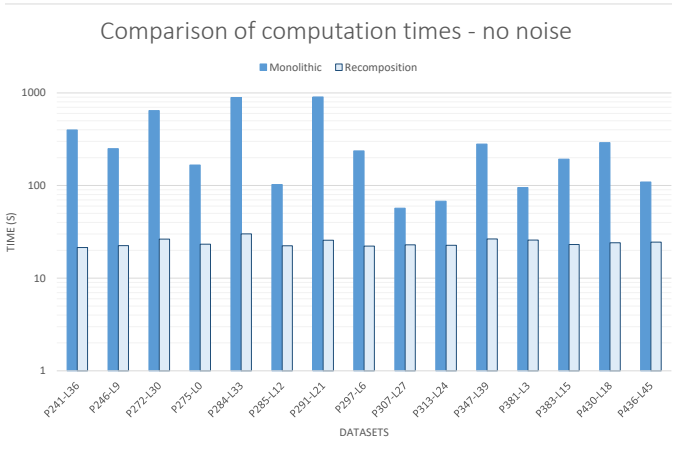


Figure 21: Feasible computation times for synthetic logs without noise

Dataset							Monolithic			Recomposition				
Name	Source	Noise	$\|A\|$	$\overline{\|\sigma\|}$	$\|L\|$	$\|\{\sigma \in L\}\|$	Feasible	Fitness	$\overline{\text{Time (s)}}$	Feasible	Fitness	$\overline{\text{Time (s)}}$	$\overline{\text{Speedup}}$	$\widetilde{\text{Speedup}}$
P241-L36	synth	no	117	95	1000	1000	✓	1	398	✓	1	21	18.6	14.6
P246-L9	synth	no	137	77	1000	1000	✓	1	250	✓	1	23	11.1	8.8
P272-L30	synth	no	201	101	1000	1000	✓	1	645	✓	1	26	24.4	19.3
P275-L0	synth	no	101	77	1000	967	✓	1	167	✓	1	23	7.2	6.0
P284-L33	synth	no	170	102	1000	570	✓	1	891	✓	1	30	29.6	24.3
P285-L12	synth	no	140	46	1000	967	✓	1	103	✓	1	22	4.6	5.1
P291-L21	synth	no	170	90	1000	1000	✓	1	904	✓	1	26	35.1	41.5
P297-L6	synth	no	125	59	1000	1000	✓	1	236	✓	1	22	10.6	12.3
P307-L27	synth	no	193	22	1000	572	✓	1	57	✓	1	23	2.5	2.7
P313-L24	synth	no	185	20	1000	480	✓	1	68	✓	1	23	3.0	3.4
P347-L39	synth	no	212	60	1000	1000	✓	1	280	✓	1	26	10.6	12.1
P381-L3	synth	no	111	73	1000	978	✓	1	95	✓	1	26	3.7	5.2
P383-L15	synth	no	150	108	1000	631	✓	1	193	✓	1	23	8.3	9.3
P430-L18	synth	no	160	104	1000	1000	✓	1	291	✓	1	24	12.1	13.7
P436-L45	synth	no	230	35	1000	621	✓	1	109	✓	1	24	4.5	4.9

Table 3: Replay feasibility and computation times for synthetic logs without noise

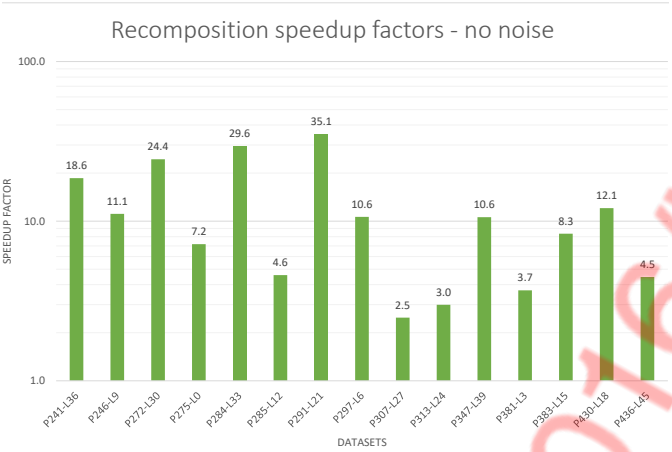


Figure 22: Speedup factors from recomposition approach over monolithic approach for synthetic logs without noise

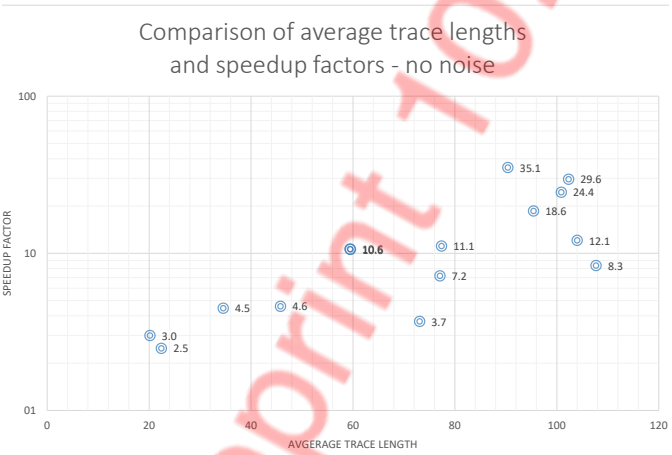


Figure 23: Speedup from recomposition approach in relation to average trace length for synthetic logs without noise

The results are presented in Table 3. We first note that the results from feasible replays are, as expected, all equal to 1, i.e., both approaches reported the conformance of the datasets correctly. Replay is feasible for all datasets under both approaches. Figure 21 compares the computation times of the monolithic replays with the computation times of the recomposing replays on a **logarithmic scale**. The figure shows that there is a clear performance gain in time across all datasets under the recomposition approach. Furthermore, the total computation times under the recomposition approach have much lower variability than the monolithic approach. Computation times range from 21s to 30s under the recomposition approach while they range from 57s to 904s under the monolithic approach. The stability in computation time under the recomposition approach allows its users to be more certain about the required execution time. The speedup factor per dataset is presented in Figure 22 where we see that it ranges from 2.5× to 35.1×. A 2.0× speedup factor means that the recomposition approach is twice as fast as the monolithic approach and a 10.0× speedup factor means that the recomposition approach is ten times faster. Furthermore, speedup factors with respect to the average trace length of the log are shown in Figure 23. We can observe that datasets with longer average trace lengths generally have greater speedup factors. While there are differences between the average speedups and the median speedups, they do not significantly affect the presented conclusions.

In terms of feasibility, both the monolithic and recomposition approach have the same performance as they were able to complete all replays. However, the results show a significant improvement in computation time using the recomposition approach. As shown in [20], the speedup results from the decomposition of the alignment problem itself and the trace groupings formed by the decomposition. Smaller components usually take significantly less time to replay than large components such that replaying several subcomponents of a decomposition is faster than replaying the original component. Furthermore, distinct traces can share identical subsections. Trace groupings from decomposition can be these subsections such that the pre-

viously distinct traces become trace groupings where some are identical. Since alignments are not recomputed for the same trace, the number of alignment operations can be reduced.

In real-life cases, it is likely that the observed and modeled behaviour are not perfectly fitting. Therefore, further experiments are conducted using logs with noise, i.e., the log is not perfectly fitting with the model.

6.3. Exact fitness in noisy scenarios

In this section, we present conformance checking results for logs with noise. This means that there are discrepancies between the modeled and observed behaviour such that the fitness value is less than 1. Similar to the previous section, the goals of this section are:

- To compare the feasibility of the proposed recomposition approach with the existing monolithic approach.
- To compare and analyze computation times of replays under both approaches.

For the sake of comparison, the processes and models are the same as the previous section, but noise is included in the logs at their generation. To test the performance of the two approaches in detecting different types of deviations, we experimented with two noise types by including them into two separate logs. This means that each model is associated with two "noisy" logs. Each noise type mimics a specific scenario where there is a mismatch between modeled and observed behaviour. Alignments between log traces and net are computed on synchronous product nets which combine the two; different noise characteristics in the log produce different log traces which can have different impacts on the performance of alignment computation. The characteristics of the noise types and the process of generating these "noisy" logs can be summarized as follows:

- *MissingTrace* noise is where a trace has a probability of missing a part of its head, tail and/or episode. The noise is included into the log by log generation using the PLG2 tool. The specific configuration used for the log generation is shown in Table 4 (we refer to [6] for a detailed explanation of each parameter).
- *Swapped* noise is where two events in a trace can be inverted. This noise type mimics the scenario where the process has a specific location where deviation always occurs. We generate a log with *Swapped* noise by swapping two activities at a specific location in the model before generating a noiseless log using the modified model. The point of deviation is randomly chosen but it is ensured that cases always pass through this location.

There are 30 model-log pairs and for each dataset we conduct two experiments using the recomposition approach and the monolithic approach. At each experiment, a 1 hour time limit is set such that replays which are still running after the time limit are stopped and deemed infeasible. Other than the replay feasibility, fitness, time, average speedup, median speedup, and the number of recompositions, we also report the number of transitions in the models, the number of unique log traces, and the average trace lengths of the logs.

Parameters	MissingTrace
Number of traces	1000
Integer data object error probability	0%
Integer data object error delta	0
String data object error probability	0%
Change activity name probability	0%
Trace missing head probability	0.1%
Head max size	2
Trace missing tail probability	0.1%
Tail max size	2
Trace missing episode probability	0.1%
Episode max size	2
Perturbed event order probability	0%
Doubled event probability	0%
Alien event probability	0%

Table 4: PLG2 log generation configurations for *MissingTrace* dataset

Table 5 presents the results. On the whole, monolithic replay was not feasible for 1 model-log pair (P284-L48) due to the 1 hour time limit. At the time column, the infeasible replay is marked with "> 3600". Figure 24 compares the computation times of monolithic replays and recomposing replays for logs with *MissingTrace* noise and *Swapped* noise. Computation times are shown on a **logarithmic scale** in both figures. There are clear performance gains from adopting the recomposition approach. Figure 25 shows the speedup factors for logs with *MissingTrace* noise and the speedups for logs with *Swapped* noise. As previously explained, a 2.0× speedup means that the recomposition approach is twice as fast as the monolithic approach and a 10.0× speedup means that it is 10× faster. We observe that speedups from the recomposition approach range from 1.3× to at least 85.5×. For the particular model-log pair P275-L52, computation time was reduced by a factor 26.6 from 17.5 minutes (1052 seconds) under the monolithic approach to 39 seconds under the recomposition approach. Speedup factors in relation to the average trace length of logs with *MissingTrace* noise and *Swapped* noise are shown in Figure 26. We can observe that in general logs with longer average trace lengths have greater speedup factors.

In terms of feasibility, the recomposition approach outperforms the monolithic approach in replaying logs with *MissingTrace* or *Swapped* noise as the monolithic approach was infeasible for 1 model-log pair while the recomposition approach was feasible for all datasets. More importantly, for these model-log pairs, the recomposition approach was able to compute their exact fitness values in very little time (less than 2 minutes).

The positive relationship between average trace lengths and speedups reflects the advantage of the recomposition approach. Logs with long average trace lengths tend to be associated with models that are more complex and therefore more challenging when computing alignments. As previously explained, the

Dataset							Monolithic			Recomposition					
Name	Source	Noise	$\|A\|$	$\overline{\|\sigma\|}$	$\|L\|$	$\|\{\sigma \in L\}\|$	Feasible	Fitness	$\overline{\text{Time (s)}}$	Feasible	Fitness	$\overline{\text{Time (s)}}$	$\overline{\text{Recompose}}$	$\overline{\text{Speedup}}$	$\widetilde{\text{Speedup}}$
P241-L37	synth	miss	117	94	1000	1000	✓	0.999	395	✓	0.999	34	5	11.5	9.8
P241-L50	synth	swap	117	95	1000	1000	✓	0.989	374	✓	0.989	26	1	14.4	11.7
P246-L10	synth	miss	137	77	1000	1000	✓	0.998	265	✓	0.998	31	2	8.6	6.9
P246-L49	synth	swap	137	77	1000	1000	✓	0.987	257	✓	0.987	28	1	9.1	7.6
P272-L31	synth	miss	201	101	1000	1000	✓	0.999	685	✓	0.999	81	15	8.4	8.5
P272-L62	synth	swap	201	101	1000	1000	✓	0.989	626	✓	0.989	40	1	15.6	13.8
P275-L1	synth	miss	101	73	1000	984	✓	0.998	168	✓	0.998	36	9	4.6	3.9
P275-L52	synth	swap	101	74	1000	965	✓	0.984	1052	✓	0.984	39	1	26.6	21.6
P284-L34	synth	miss	170	106	1000	708	✓	0.998	1147	✓	0.998	61	10	18.7	23.1
P284-L48	synth	swap	170	106	1000	589	✓		> 3600	✓	0.997	42	1	> 85.5	> 85.9
P285-L13	synth	miss	140	46	1000	978	✓	0.997	112	✓	0.997	32	6	3.5	3.9
P285-L56	synth	swap	140	48	1000	973	✓	0.947	202	✓	0.947	29	0	7.0	8.0
P291-L22	synth	miss	170	90	1000	1000	✓	0.999	935	✓	0.999	46	6	20.3	21.5
P291-L51	synth	swap	170	91	1000	1000	✓	0.989	913	✓	0.989	35	1	25.9	30.1
P297-L7	synth	miss	125	59	1000	1000	✓	0.998	228	✓	0.998	31	4	7.5	7.7
P297-L55	synth	swap	125	60	1000	1000	✓	0.982	241	✓	0.982	30	1	8.0	9.2
P307-L28	synth	miss	193	22	1000	660	✓	0.994	65	✓	0.994	52	11	1.3	1.4
P307-L53	synth	swap	193	23	1000	581	✓	0.996	66	✓	0.996	27	0	2.4	2.3
P313-L25	synth	miss	185	20	1000	534	✓	0.993	71	✓	0.993	29	1	2.4	2.6
P313-L59	synth	swap	185	20	1000	465	✓	0.986	67	✓	0.986	28	0	2.4	2.5
P347-L40	synth	miss	212	60	1000	1000	✓	0.998	298	✓	0.998	54	9	5.5	6.2
P347-L58	synth	swap	212	59	1000	1000	✓	0.996	292	✓	0.996	40	1	7.3	8.5
P381-L4	synth	miss	111	73	1000	983	✓	0.998	112	✓	0.998	28	1	3.9	4.6
P381-L63	synth	swap	111	74	1000	973	✓	0.985	90	✓	0.985	28	1	3.2	3.4
P383-L16	synth	miss	150	106	1000	737	✓	0.999	224	✓	0.999	31	2	7.1	8.4
P383-L61	synth	swap	150	111	1000	632	✓	0.979	795	✓	0.979	43	1	18.7	19.8
P430-L19	synth	miss	160	104	1000	1000	✓	0.999	293	✓	0.999	46	10	6.4	7.2
P430-L57	synth	swap	160	104	1000	1000	✓	0.997	306	✓	0.997	30	0	10.0	11.2
P436-L46	synth	miss	230	35	1000	735	✓	0.996	120	✓	0.996	68	12	1.8	2.0
P436-L54	synth	swap	230	35	1000	633	✓	0.982	107	✓	0.982	35	1	3.0	3.3

Table 5: Replay feasibility and computation times for synthetic logs with *MissingTrace* and *Swapped* noise

decomposition of a complex component into several decomposed subcomponents reduces replay time significantly. However, current techniques only guarantee that conformance results from decomposed subcomponents can be aggregated to an overall result if the model and log are perfectly fitting. By formalizing border activities and border agreement, we relax the strict requirement for perfect fitness and allow the aggregation of results from decomposed subcomponents even when there are mismatches between the model and log. This means that the recomposition approach gets the performance gains from decomposition and the resulting exact fitness result would correspond to the fitness value that one would get using the monolithic approach. The increasing speedups as average trace length increases demonstrate the said performance gains from decomposition despite the presence of noise in the log. This is further supported by observing the minimal effect that the average trace length has on the computation time of the recomposing replays. Analyzing the number of recompositions, we note that the larger and more complex datasets require more recompositions. For example, P272-L31 required 15 recompositions while P241-L37 required 5 recompositions. Also, the noise type has a significant influence on the number of recompositions regardless of the model characteristics. Experiments on dataset with missing noise never require more than one recomposition while experiments on dataset with missing noise can require a maximum of 15 recompositions. These two points are further examined and explained in Section 6.4. While there are differences between the average speedups and the median speedups, they do not significantly affect the presented conclusions.

In conclusion, these experiments empirically show the potential performance gains of the recomposition approach. As the results illustrate, this new approach not only increases replay feasibility, but also results in significant performance gains for datasets that are feasible under both recomposition and monolithic approaches.

6.4. *Bottlenecks for the monolithic and recomposition approach*

One principal motivation of investigating decomposition techniques for alignment-based conformance checking is to improve the alignment computation time for large and complex processes. It has been previously shown that by decomposing the conformance problem into parts, this bottleneck can be alleviated significantly [20]. This section evaluates the bottlenecks of both approaches by analyzing the computation times of all the previous experiments. The computation time refers to the time required for fitness computation and corresponds to the reported times in Table 3 and Table 5. Specifically, there are three main goals in this section:

- Understand the bottleneck of the monolithic approach by analyzing the percentage of total computation time spent in alignment computation.

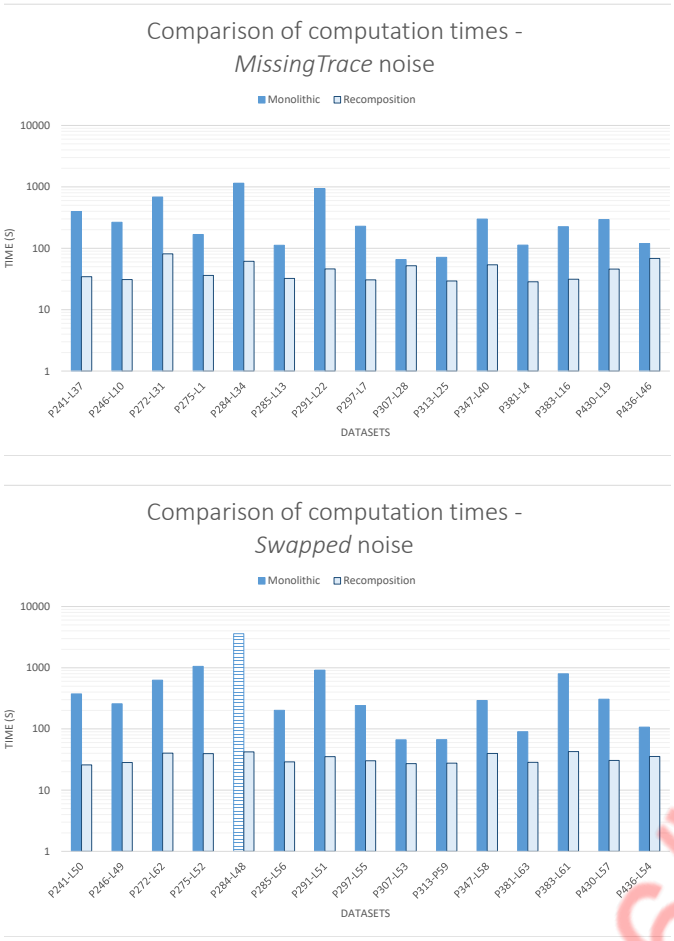


Figure 24: Feasible computation times for synthetic logs with *MissingTrace* noise (top) and *Swapped* noise (bottom) (infeasible replays are shown using a dashed pattern instead of a solid fill)

Figure 25: Speedup factors from recomposition approach over monolithic approach for synthetic logs with *MissingTrace* noise (top) and *Swapped* noise (bottom) (infeasible replays are shown using a dashed pattern instead of a solid fill)

- Understand the bottleneck of the recomposition approach by analyzing the percentage of total computation time spent in alignment computation and additional details, e.g., the number of recompositions.
- Compare the bottleneck analysis of both approaches.

The fitness computation between a particular log trace and net requires several other steps such as building a synchronous product net, and constructing the alignments [2]. However, since the majority of the computation time is in the computation of the alignment via the replay of the synchronous product net, previous works often do not always distinguish between the different sources of the total computation time.

Figure 27 compares the total computation time with the percentage of time spent in replay for all the conducted experiments. It shows the minimum replay percentage as 81% for P313-L59 and the maximum replay percentage of feasible monolithic replays as 99% for P284-L34. The shape of the scatter plot illustrates that the percentage of time spent in replay quickly approximates to 100% as total computation time grows. This confirms that the bottleneck for the monolithic approach is at the replay. Also, this bottleneck exists regardless of the noise type in the data. Next, a similar analysis is conducted for the recomposition approach with several extensions to consider the additional components in the framework.

As illustrated by the overview of the recomposition approach in Figure 11, if there is no total border agreement between two subalignments, the recomposition approach undertakes a recomposition to merge the corresponding subcomponents before recomputing the alignment in the next iteration. Figure 28 compares the number of recompositions with the total computation time for the "exact experiments" under the recomposition approach. In spite of the variances in total computation time at particular recomposition values, it is quite clear that the total computation time increases with the number of recompositions. It is also evident that results of different noise types show different characteristics. We first note that none of the "no-noise" experiments required any recompositions. This is because the subalignments only have synchronous moves so that total border agreement is guaranteed. Contrastingly, this is not the case for the experiments on noisy datasets. For the "swapped" experiments, a subset of the experiments is able to compute the exact fitness value without any recompositions and the rest required

Figure 26: Speedup from recomposition approach in relation to average trace length for synthethic logs with *MissingTrace* noise (top) and *Swapped* noise (bottom)

Figure 27: Percentage of time spent in alignment computation in relation to total computation time

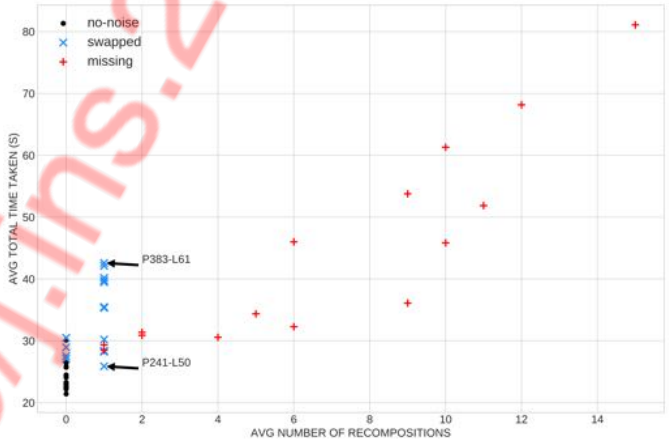


Figure 28: Total computation time in relation to number of recompositions for synthetic logs at exact experiments under the recomposition approach

exactly one recomposition. The maximum of one recomposition can be attributed to the fact that only one pair of activities is swapped to create noise during the data generation. This means that if the net is decomposed on the swapped activities then border agreement issue is bound to occur. This is because the subcomponents which have only one of the activities would not be able to detect the activity swap. Since the nets are not maximally decomposed in the experiments, all subcomponents are comprised of more than two unique visible transitions. In the worst case, decomposition would occur on one of the two swapped activities. This means that border agreement issues are immediately resolved when the subcomponents are merged on the decomposed activity. The variances in total computation times for particular recomposition values can be attributed to the underlying data complexity. For example, Figure 28 shows that there is a notable variance for the experiments which underwent one recomposition. The minimum time is from P241-L50 and the maximum time is from P383-L61. Referring to Table 1, we find that P241 has one of the smallest and simplest structure with 117 activities, 13 ANDs, and 12 XORs. In contrast, P383 has one of the largest and most complex structure with 150 activities, 29 ANDs, and 20 XORs. For the "missing" experiments, the number of recompositions ranges from 1 to 15. Furthermore, they particularly illustrate the direct positive correlation between the two factors.

While it is clear that the total computation time should increase with the number of recompositions, we would like to know whether the proportion of total time spent in replay remains the same as the number of recompositions increases. Figure 29 compares the total computation time with the percentage of time spent in replay at the exact experiments under the recomposition approach. Firstly, the figure suggests that the percentage of time spent in replay decreases as total computation time increases. This is a surprising result as it contrasts the previous analysis on the monolithic approach where the percentage of time spent in replay increases with total computation time. Furthermore, the percentage of time spent in replay never exceeds 60% under the recomposition approach whereas the percentage of time spent in replay never falls below 80% under the monolithic approach. This suggests that under the recomposition approach, as computation complexity increases, replay becomes less of a contributing factor to computation time.

One possible culprit would be the recomposition component

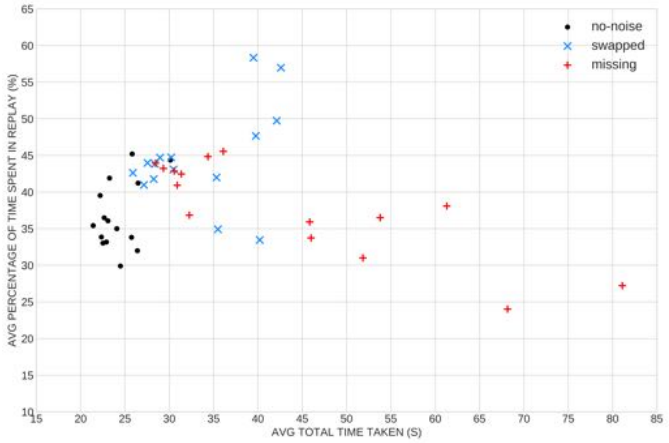


Figure 29: Total computation time in relation with to the percentage of time spent in replay for synthetic logs at exact experiments under the recomposition approach

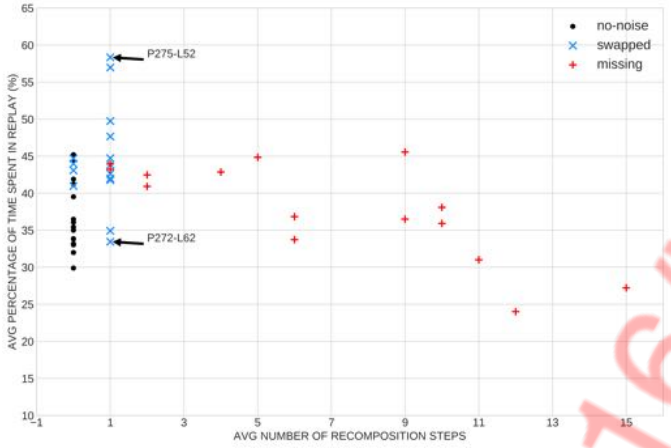


Figure 30: Percentage of time spent in replay in relation to number of recompositions for synthetic logs at exact experiments under the recomposition approach

of the framework which can be associated to the number of recompositions. Figure 30 shows the percentage of time spent in replay in relation to the number of recompositions. Indeed, the figure resembles Figure 29 to show that replay has less of an impact on performance as complexity increases. Recall that Figure 28 suggested that the number of recompositions and total computation time are positively correlated. Similar to before, there can be variances in the percentage of time spent in replay at particular recomposition values. This can be explained by the number of subnets for each dataset. For example, for experiments that underwent one recomposition, the maximum time percentage is of P275-L52 and the minimum time percentage is of P272-L62. The P275 net has 13 subnets in its initial decomposition and P272 has 32 subnets in its initial decomposition. Having more subnets means more overhead before and after the alignment computation. However, this can likely be alleviated with the use of more computer nodes.

Continuing with this line of reasoning, the low percentage of time spent in replay also prompts investigation into the number of remaining problematic traces that are taken to the next iteration by a recomposition. Figure 31 shows the number of remaining trace variants in relation to the number of recompositions for the experiments on the datasets with "missing" noise. We only focus on these experiments because the other experiments were able to align all traces in less than two recompositions. The figure illustrates that for all the experiments, less than 100 trace variants remains to be aligned after the first recomposition. This explains the previous findings which suggested that the percentage of time spent in replay decreases with the number of recompositions. Moreover, it implies that the bottleneck to computing the exact fitness value under the recomposition approach is in having the appropriate recomposition that will permit total border agreement for the remaining traces.

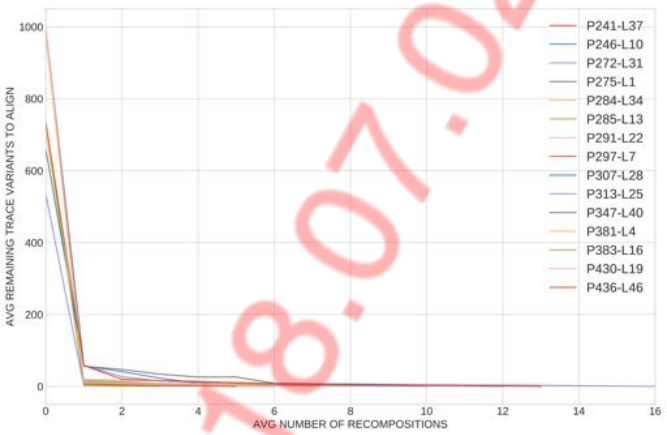


Figure 31: Number of remaining trace variants in relation to number of recompositions for the experiments on datasets with "missing" noise

Overall, the analysis on the different aspects of performance leads to several insights. First, we confirm that the bottleneck for the monolithic approach is at the alignment computation. Next, we find that the recomposition approach is able to partially shift the responsibility to the recomposition component. By decomposing the overall component, replay is only performed on small subcomponents even for large and complex processes. This prevents state space explosions during the alignment computation and keeps replay time in check for each subcomponent. It also means that the necessary number of recompositions have a large impact on performance. Moreover, we find that a small number of log traces may require several recompositions before yielding a set of subalignments that fulfills total border agreement. This means that the bottleneck to computing the exact fitness value is in finding the appropriate recomposition that will permit total border agreement. Since the experiments have been performed on only one computer node, distributing the computation across more nodes is likely to lead to further performance gains. The possibility of shifting the bottleneck from replay to recomposition motivates future work to investigate decomposition and recomposition strategies as well as different decomposed replay approaches. However, we once again emphasize that this shift is partial because the capacity to decompose a system net into small subnets is restricted by

whether if the net structure can be decomposed. Further experiments on different decompositions and decomposed replay approaches would give more insights but are outside the scope of this paper.

6.5. Feasibility and interval narrowing time constrained scenarios

One important contribution of the recomposition approach is to enable configurable approximations of the overall conformance between the model and log. This means that it would not be necessary to align all the traces in the log with the model. As previously mentioned, fitness approximations by fitness interval values can further reduce computation times and alleviate replay feasibility problems with specific log traces or sections of the model. As time is often the principal concern for alignment-based conformance checking, this section focuses on the effects of time as a hard constraint on the monolithic and recomposition approaches. There are two main goals in this section:

- To show how to compare the feasibility of the recomposition approach using the monolithic approach. We identify cases where the recomposition approach can provide either an exact or an interval fitness value while monolithic replay is infeasible.
- To analyze the effects on time upon the narrowing.

For the sake of comparison, the experiments are conducted using the same synthetic models and generated logs from the experiments of the previous section. As explained, each model has two associated "noisy" logs; the *no-noise* logs are excluded from these experiments. For each model-log pair, we conduct three experiments, each constrained by a different time limit on the overall alignment time. This is done by varying the value of the GlobalDurationThreshold parameter. Experiments are conducted with the GlobalDurationThreshold values of 1, 5 and 10 minutes.

For the results, in the case of the monolithic approach, an exact fitness value is provided if replay is feasible and it is marked with a cross otherwise. For the recomposition approach, since it defaults to approximation if an exact value cannot be computed, either an exact fitness value or a fitness interval is given.

Table 6 shows the results of the experiments. For the experiments with a 1 minute time constraint, monolithic replay is not feasible for any dataset. The recomposition approach can compute an exact fitness value for 27 model-log pairs and fitness interval values for the remaining 3 model-log pairs. For the fitness interval results, the interval range varies from 0.008 (P284-L34) to 0.027 (P272-L31). For the experiments with a 5 minute time constraint, monolithic replay is feasible for 19 model-log pairs. The recomposition approach can compute an exact fitness value for all 30 model-log pairs. For the experiments with a 10 minute time constraint, monolithic replay is feasible for 22 model-log pairs. The recomposition approach can compute an exact fitness value for all 30 model-log pairs. Previous experiments (cf. Section 6.3) have shown that the replay of 1 model-log pair cannot be finished within one hour under the monolithic approach.

In terms of feasibility, the results show that the recomposition approach can give an exact fitness value for more model-log pairs than the monolithic approach under all three time constraints. For the shortest 1 minute constraint, the recomposition approach can give an exact fitness value for almost all of the datasets while monolithic replay was not feasible for any dataset. Other than feasibility, the fitness interval results computed by the recomposition approach show that the exact fitness values (as shown in Table 5) are always within the interval.

In conclusion, the recomposition approach outperforms the monolithic approach in feasibility when there is a constraint on alignment time. More importantly, the recomposition approach can always give a fitness result to reflect the conformance level between the model and log whereas the monolithic approach can only give a result if replay can be fully completed.

Having compared the performance of the recomposition approach with the existing monolithic approach for different process characteristics and noise scenario, we show that the recomposition approach can be applied to real-life datasets.

6.6. Recomposed fitness in real-life cases

In this section we apply the monolithic approach and the recomposition approach on two real-life datasets. This is to demonstrate that the recomposition approach can be applied to real-life datasets and to compare the performances of both approaches. Notice that the final goal of any conformance approach is to gain insight on the conformance problems. Therefore, we perform a simple analysis using the conformance results from replay to showcase the applicability of alignment-based conformance checking for the analysis of deviations.

We used the BPIC2012 real-life dataset [37] adopted in 2012 for the BPI (Busines Process Intelligence) Challenge. This dataset is taken from a Dutch Financial Institute and contains 13,087 cases and 36 event classes. Apart from some anonymization, the log contains all data as it came from the financial institute.

The process represented in the event log is an application process for a personal loan or overdraft within a global financing organization. The event log is a merger of three intertwined sub processes such that the first letter of each task can be used to identify the subprocess that the task originated from.

Since no explicit process model is provided with the dataset, a model is constructed with consideration to the task semantics and the three sub processes as illustrated in Figure 32 (please ignore the color bars at the bottom of transitions and the color of transitions and places for now).

For the experiments, a 1 hour time limit is set such that replays which are still running after the time limit are stopped and deemed infeasible. The recomposition approach is initiated with a maximal decomposition of the model and log. A maximal decomposition is a valid decomposition where the subcomponents are as small as possible. We report the time and fitness of replay under both approaches.

As shown in Table 7, we find that the monolithic approach is not feasible for BPIC2012 under the 1 hour time limit. In contrast, the recomposition approach was able to provide a fitness

Dataset					Time constraint on overall alignment time (min)					
					1'		5'		10'	
Name	$\|A\|$	$\overline{\|\sigma\|}$	$\|L\|$	$\|\{\sigma \in L\}\|$	Monolithic	Recomposition	Monolithic	Recomposition	Monolithic	Recomposition
P241-L37	117	94	1000	1000	✗	0.999	✗	0.999	0.999	0.999
P241-L50	117	95	1000	1000	✗	0.989	✗	0.989	0.989	0.989
P246-L10	137	77	1000	1000	✗	0.998	0.998	0.998	0.998	0.998
P246-L49	137	77	1000	1000	✗	0.987	0.987	0.987	0.987	0.987
P272-L31	201	101	1000	1000	✗	0.972-0.999	✗	0.999	✗	0.999
P272-L62	201	101	1000	1000	✗	0.989	✗	0.989	✗	0.989
P275-L1	101	73	1000	984	✗	0.998	0.998	0.998	0.998	0.998
P275-L52	101	74	1000	965	✗	0.984	✗	0.984	✗	0.984
P284-L34	170	106	1000	708	✗	0.990-0.998	✗	0.998	✗	0.998
P284-L48	170	106	1000	589	✗	0.997	✗	0.997	✗	0.997
P285-L13	140	46	1000	978	✗	0.997	0.997	0.997	0.997	0.997
P285-L56	140	48	1000	973	✗	0.947	0.947	0.947	0.947	0.947
P291-L22	170	90	1000	1000	✗	0.999	✗	0.999	✗	0.999
P291-L51	170	91	1000	1000	✗	0.989	✗	0.989	✗	0.989
P297-L7	125	59	1000	1000	✗	0.998	0.998	0.998	0.998	0.998
P297-L55	125	60	1000	1000	✗	0.982	0.982	0.982	0.982	0.982
P307-L28	193	22	1000	660	✗	0.994	0.994	0.994	0.994	0.994
P307-L53	193	23	1000	581	✗	0.996	0.996	0.996	0.996	0.996
P313-L25	185	20	1000	534	✗	0.993	0.993	0.993	0.993	0.993
P313-L59	185	20	1000	465	✗	0.986	0.986	0.986	0.986	0.986
P347-L40	212	60	1000	1000	✗	0.998	0.998	0.998	0.998	0.998
P347-L58	212	59	1000	1000	✗	0.996	0.996	0.996	0.996	0.996
P381-L4	111	73	1000	983	✗	0.998	0.998	0.998	0.998	0.998
P381-L63	111	74	1000	973	✗	0.985	0.985	0.985	0.985	0.985
P383-L16	150	106	1000	737	✗	0.999	0.999	0.999	0.999	0.999
P383-L61	150	111	1000	632	✗	0.979	✗	0.979	✗	0.979
P430-L19	160	104	1000	1000	✗	0.999	0.999	0.999	0.999	0.999
P430-L57	160	104	1000	1000	✗	0.997	✗	0.997	0.997	0.997
P436-L46	230	35	1000	735	✗	0.983-0.996	0.996	0.996	0.995	0.996
P436-L54	230	35	1000	633	✗	0.982	0.982	0.982	0.982	0.982

Table 6: Time-constrained conformance analysis on synthetic logs with noise of dataset using manual initial decomposition

interval value with a range of less than 0.001 under 30 minutes (1800 seconds). The results also show that an interval value was given rather than an exact value because 1 trace was rejected during the replay process given the exclusion parameters defined (cf. Section 5). The fitness interval of $[0.647 - 0.648]$ indicates there are considerable deviations between the modeled behavior and observed behavior. Further analysis can be done by visualizing the conformance results.

Conformance results can be visualized through alignments as shown in Figure 18 or by projecting the deviation issues onto the corresponding places and transitions as shown in Figure 32. Transitions are augmented with two additional information. Firstly, the color of the visible transitions indicates the execution frequency of a particular transition in the log. A lighter color means that the transition is rarely executed and a darker color means that the transition is often executed. The color at the bottom of transitions indicates the distribution between synchronous moves (light green) and model moves (dark pink). A color bar with a high green portion means that there is little conformance issue with the corresponding transition and a low green portion means there is severe conformance issue with the transition. Transitions without color bars do not have any conformance issues. Log moves are shown by marking the places where log moves occurred. The occurrence frequency is indicated by the size of the places.

We first observe that out of the three subprocesses, subprocess *A* which relates to the application itself has minimal conformance issues. Since the application is submitted through the webpage, it makes sense that the observed behavior can be easily and accurately described by a corresponding model. Contrastingly, both subprocess *O* which relates to the offer of the application and subprocess *W* which relates to the work items belonging to the application have major conformance issues. For simplicity, we only focus on subprocess *O*. There are log moves relating to all parts of the subprocess and there is a high frequency of model moves relating to the transitions: *O_SELECTED*, *O_SENT*, *O_CREATED* and *O_ACCEPTED*. For example, figure 33 shows the ratio between synchronous moves (at 2243 cases) and model moves (at 10039 cases) for transition *O_ACCEPTED*. This means that 10039 cases did not execute transition *O_ACCEPTED* even though they were supposed to according to the model. This can be confirmed by inspecting the alignments of particular cases in the alignment visualization. Figure 34 shows the alignment for case 173733

Dataset								Monolithic			Recomposition					
Name	$\|A\|$	$\overline{\|\sigma\|}$	$\|L\|$	$\|\{\sigma \in L\}\|$	|AND|	|XOR|	|Loop|	Feasible	Fitness	Time (s)	Feasible	Fitness	Time (s)	$\overline{\text{Steps}}$	$\overline{\text{Speedup}}$	$\widetilde{\text{Speedup}}$
BPIC2012	36	20	13087	4366	4	19	7	✗		> 3600	✓	0.647-0.648	1761	11	> 2.0	> 2.0
BPIC2017	40	24	31509	7478	1	13	3	✓	0.992	71	✓	0.992	78	9	(1.1)	(1.1)

Table 7: Replay feasibility and computation times for BPIC2012

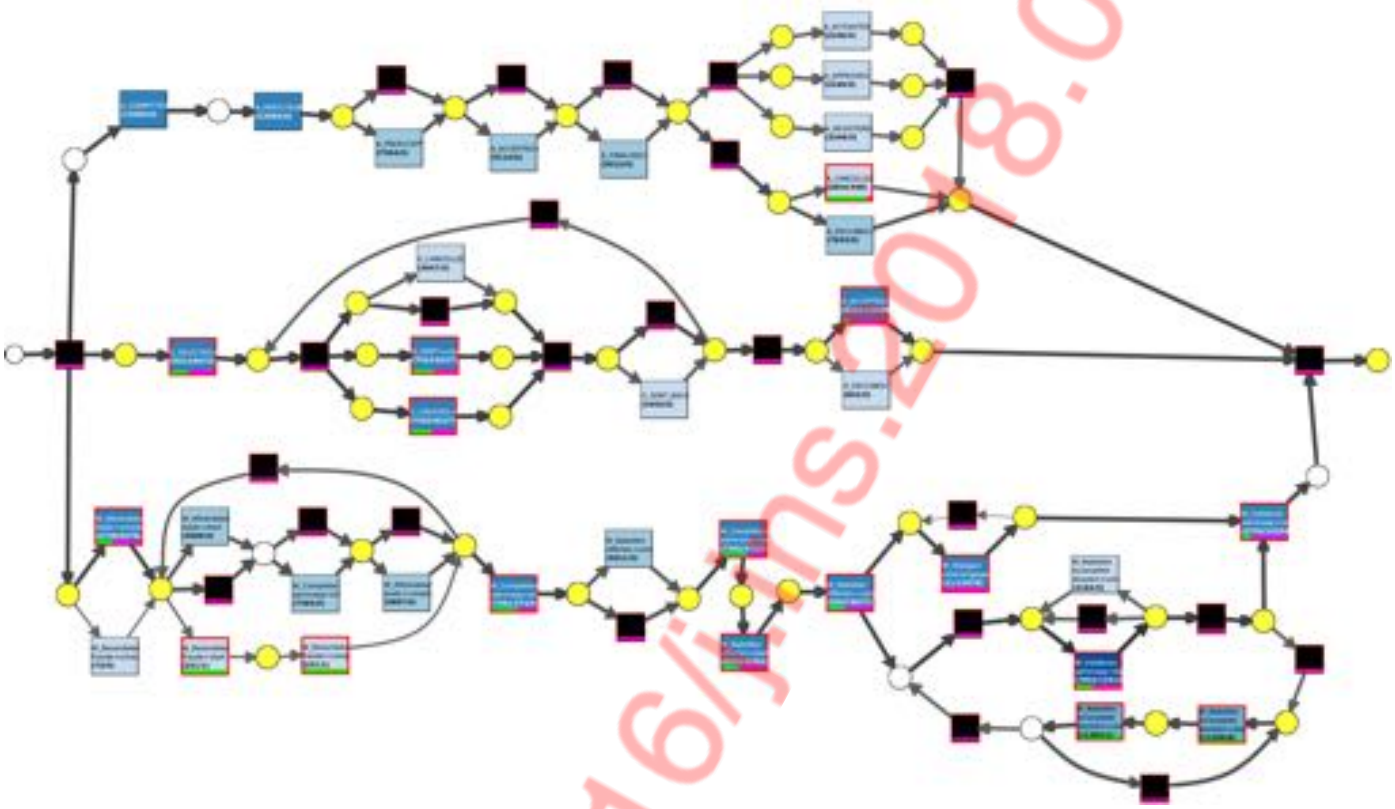


Figure 32: Handmade model for the BPIC2012 real-life dataset projected with the deviation issues between the model and log

Figure 33: Conformance diagnosis on transition *O_ACCEPTED*

which indicates a model move for transition *O_ACCEPTED*. As such, there is a discrepancy between the reality and the model suggested by the task semantics regarding to how an offer is processed. The identification of this conformance issue may prompt further investigations with related stakeholders such as

Figure 34: Alignment for case 173733 with model move on transition *O_ACCEPTED* (highlighted in white)

the employees who handled the deviating cases.

In the previous example, replay is not feasible under the monolithic approach. Here, we further showcase an example using a real-life dataset where replay is feasible under both approaches.

We used the BPIC2017 real-life dataset [38] adopted in 2017 for the BPI (Business Process Intelligence) Challenge. This dataset is taken from the same Dutch Financial Institute and contains 31,509 cases and 66 event classes. The data contains all the applications filed through the online system in 2016 and their subsequent events until the 1st of February 2017, 15:11. Note that the data is provided provided by the same company of the BPIC2012 dataset but has underwent several changes and improvements to the system. One important difference is that now the system can support multiple offers for a single applica-

tion.

In spite of the changes, the process is still a merger of three intertwined subprocesses such that the first letter of each task can be used to identify the subprocess that the task originated from. To ensure the replay feasibility of both approaches, the event log is filtered to keep only events with a schedule, start, and complete lifecycle attribute value. The filtered log has 40 event classes.

Since no explicit process model is provided with the dataset, a model is discovered using the inductive miner as illustrated in Figure 35 (please ignore the color bars at the bottom of transitions and the color of transitions and places for now).

The setup for the experiments is the same as the previous real-life dataset example with the same initial decomposition strategy for the recomposition approach and time limit for both approaches.

As shown in Table 7, replay was feasible under both approaches. Both approaches finished computing the model fitness in less than 1.5 minutes with the monolithic approach being 7 seconds faster on average so that there is an average (and median) speedup of 1.1 under the monolithic approach. This shows that even for simpler models, the recomposition approach is only slightly slower despite the additional recomposition procedures.

Both approaches yielded a fitness value of 0.992 which shows that the discovered model can describe nearly all the observed behavior in the log. Other than having the same fitness value, the computed alignments under both approaches are composed of the same alignment moves. This means that both approaches lead to the same enriched model in Figure 35 when the alignments are projected onto the model. Furthermore, examining most of the individual trace alignments finds that the alignment for the same case to be the same under both approaches. For example, Figure 36 shows the alignment for case Application_931736025. The alignment computed under both approaches indicates a model move on activity *A_Cancelled+complete*. As previously mentioned, given a cost function, there can be multiple optimal alignments for a particular log trace. Further investigations into the circumstances under which the monolithic and recomposition approaches compute different optimal alignments can be interesting, especially if there are reasons to prefer a particular one.

Lastly, this example demonstrates the need to consider the other quality dimensions for the evaluation of model quality. While the high fitness suggests that this is a "good" model, a closer look at the model by a user would conclude otherwise. For example, there are multiple loop constructs so that each loop construct approximates a flower model that can generate any sequences involving the transitions in the loop. Figure 37 illustrates one of these loop constructs. The construct initiates with an XOR split of 12 branches. Then, at the output place where all the branches terminate, there is an invisible transition that loops back to the previous XOR split. This means that if a case reaches a marking that includes the place of the XOR split, the model would then be able to replay any sequences that involve one or more of the branches. As such, this part of the model gives little information on the control flow of the observed behavior.

In conclusion, the results clearly demonstrate that the applicability of the recomposition approach make alignment-based conformance checking feasible for real-life datasets. In addition, for the example where replay is feasible under both approaches, an analysis of the computed alignments confirm that not only fitness values match but also alignments show the same behavioral analysis results. Lastly, the utility of the conformance results is shown through a simple analysis.

7. Related work

For an introduction to process mining, we refer to [31]. For an overview of best practices and challenges, we refer to the Process Mining Manifesto [32].

As previously mentioned, there are four competing quality criteria in conformance checking: fitness, precision, simplicity and generalization [26]. In this paper, we focused on fitness, however, the border properties of subnets can also be applied to quantify precision and generalization. There are many conformance checking techniques in the literature [26, 4, 3, 33, 31] but in the recent years, the use of alignment-based conformance checking has become the de facto standard [4, 18, 17]. Alignment techniques provide a robust analysis of fitness, especially in the presence of non-determinism in the model or noise in the logs.

Yet one limitation of the existing alignment algorithm is the explosion of state space when handling industrial sized problems. With the increasing availability of event data, this calls for solutions so that the alignment technique can be widely applicable in the evolving context.

Several approaches have been proposed to decompose conformance checking in the literature [46, 30, 47, 20]. Their experimental results showed an immense reduction in computation time over the existing monolithic approach. However, conformance results from most of these approaches only remain at the level of sublogs and subnets, the conformance between the overall process model and event log is not computed. For example, a recent paper [47] proposed a workflow decomposition technique that decomposes a workflow net into a set of smaller workflow nets by places after which conformance checking is performed on the subnets by alignment. This treats subcomponents independently and does not compute alignment at a global level. An exception is the implementation of an earlier proposed generic approach to decompose process discovery and conformance checking [46]. In the implementation, conformance checking is again performed by aligning sublogs and subnets but an extra step is taken afterwards to merge the subalignments into an overall pseudo alignment. The pseudo alignment gives a lower bound for the mismatch costs between the overall process model and log. This guarantee can provide an idea about the overall conformance. In relation to these works, the proposed recomposition approach advances existing decomposition solutions by extending decomposition techniques to compute an exact or interval value that reflects the overall conformance between a process model and log.

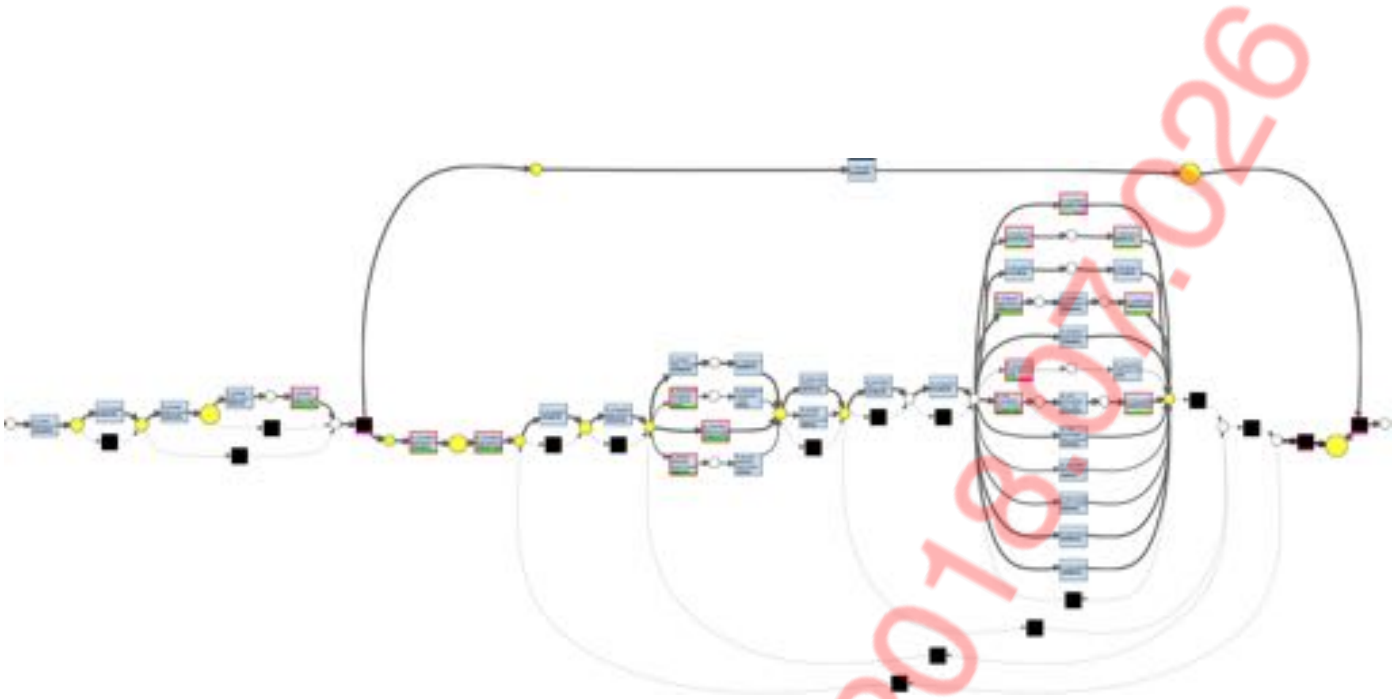


Figure 35: Discovered model for the BPIC2017 real-life dataset projected with the deviation issues between the model and log

Figure 36: Alignment for case Application_931736025 with a model move on transition *A_Cancelled+complete* (highlighted in white)

Other than decomposing conformance checking, other ideas have also been proposed. In [27], approximate alignments are proposed to view sections of step-moves in an alignment under a coarser granularity as multisets. This abstract view of alignment reduces the complexity of the alignment problem so that computation time is reduced. Another way to reduce the complexity of the problem is through reduction of the model and log. In [28], the notion of indication is used to reduce process models. The occurrence of an indicator transition requires the presence of a specified set of transitions (its indicated set) due to their causal relations. As such, these indication relations between transitions can be used to reduce models and logs. By using a reduced model and log, a macro-alignment can be computed in significantly shorter time. The missing alignment details are later expanded using an efficient alignment algorithm from bio-informatics. Other than abstracting alignment details to alleviate complexity, new interpretations of the alignment concept have also been proposed such as anti-alignments [7]. Compared to these alternatives, the recomposition approach presented in this paper preserves the alignment concept and does not need to abstract away details to achieve performance gain.

Another related work is conformance checking of proclets [13]. Proclets can be used to define so-called artifact centric processes, i.e., processes that are not monolithic but are composed of smaller interacting processes (called proclets). In [13], it is shown that conformance checking can be done per proclet by projecting the event log onto a single proclet while considering interface transitions in the surrounding proclets.

8. Conclusions and future work

This paper presents a novel approach to provide alignment-based conformance checking results for large and complex models where monolithic approaches are not suitable. This approach applies divide and conquer strategies to break down the overall component into subcomponents before iteratively recomposing these small pieces to get the final result on the overall fitness. As such, the approach significantly improves upon the performance issue of the existing monolithic approach for large and or more complex process models while retaining a result quality that is absent from existing decomposition techniques. In summary, the approach applies decomposition techniques to give a complete solution to alignment-based conformance checking problems.

To enable the aggregation of results from subcomponents, we formalized the border transitions of a valid decomposition. We showed that aggregation of conformance results from subcomponents is permissible if and only if there is total border agreement between all the subcomponents. In addition, we adapted the existing relative fitness metric as the decomposed fitness metric to compute an exact or interval value to reflect the fitness between a model and log.

The conformance checking approach has been implemented in ProM. Extensive experimental results from synthetic and real-life datasets demonstrate the potential performance gains of the recomposition approach over the existing state of the art monolithic approach. The use of decomposition increased replay feasibility and reduced computation time for datasets feasible under both approaches. The speedup from adopting the recomposition approach has shown to be attainable under different noise scenarios and increases with the average trace lengths of the event logs.

Our future work aims at improving and extending the presented approach as well as investigating new metrics for conformance checking. In the experiments, we found that the initial decomposition for instantiating recomposing replay can have an

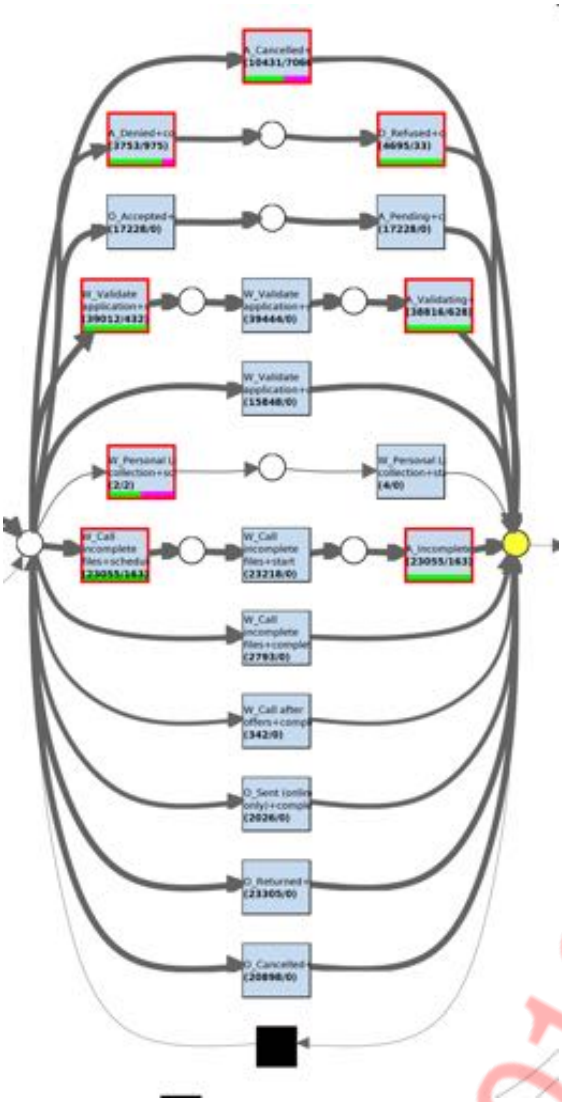


Figure 37: Loop construct in the discovered model of the BPIC2017 real-life dataset projected with deviation issues

impact on its performance. For the sake of simplicity, manual decompositions were used in the experiments on the synthetic datasets. We concede that using a manual decomposition is a possible weakness of the approach, but believe that we can generate such decompositions in an automated way by using SESE techniques. Also, a study of existing decomposition strategies as the initial decomposition may lead to new insights. On a similar note, better merging heuristics and strategies to encourage border agreement would also further improve the performance of the recomposition approach. Due to the process data generator, the synthetic datasets used in the empirical analysis are block structured. While this does not limit the validity and correctness of the theoretical results, extending empirical studies to consider non-block structured models by using more recent process data generators, e.g., the PTandLogGenerator [16], would provide further understanding on the performance of the proposed framework under a wider context. Furthermore, more extensive empirical studies to include a larger variety of noise scenarios may provide insights into their impacts on alignment computation. In this paper, we have focused on the fitness quality dimension. New metrics are needed to extend the recomposition approach to other quality dimensions such as precision. Lastly, we have based the approach upon the Petri net modeling notation and have restricted to the control flow aspect of event logs. It would be desirable to concretely extend the presented ideas to other modeling notations, e.g., BPMN, and to take into account the additional information contained in an event log.

Acknowledgements

This work is partially supported by *CONICYT-PCHA / Doctorado Nacional / 2017-21170612*, FONDECYT Iniciación 11170092, *CONICYT Apoyo a la Formación de Redes Internacionales Para Investigadores en Etapa Inicial REDI170136*, the *Vicerrectoría de Investigación de la Pontificia Universidad Católica de Chile / Concurso Estadías y Pasantías Breves 2016*, and the *Departamento de Ciencias de la Computación UC / Fond-DCC-2017-0001*. The authors would like to thank Felix Mannhardt and Boudewijn F. van Dongen for their comments on the implementation details.

References

[1] A. Abbasi, S. Sarker, R. H. L. Chiang, Big Data Research in Information Systems: Toward an Inclusive Research Agenda, J. AIS 17 (2).

[2] A. Adriansyah, Aligning Observed and Modeled Behavior, Ph.D. thesis, Technische Universiteit Eindhoven (2014).

[3] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, W. M. P. van der Aalst, Alignment Based Precision Checking, in: Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers, 2012, pp. 137–149.

[4] A. Adriansyah, B. F. van Dongen, W. M. P. van der Aalst, Conformance Checking Using Cost-Based Fitness Analysis, in: Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2011, Helsinki, Finland, August 29 - September 2, 2011, 2011, pp. 55–64.

[5] C. Bratosin, N. Sidorova, W. M. P. van der Aalst, Distributed genetic process mining, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010, IEEE, 2010, pp. 1–8.

[6] A. Burattin, PLG2: Multiperspective Processes Randomization and Simulation for Online and Offline Settings, CoRR abs/1506.08415.

[7] T. Chatain, J. Carmona, Anti-alignments in Conformance Checking - The Dark Side of Process Models, in: Application and Theory of Petri Nets and Concurrency - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings, 2016, pp. 240–258.

[8] M. de Leoni, J. Munoz-Gama, J. Carmona, W. M. P. van der Aalst, Decomposing Alignment-Based Conformance Checking of Data-Aware Process Models, in: R. Meersman, H. Panetto, T. S. Dillon, M. Missikoff, L. Liu, O. Pastor, A. Cuzzocrea, T. K. Sellis (eds.), On the Move to Meaningful Internet Systems: OTM 2014 Conferences - Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings, vol. 8841 of Lecture Notes in Computer Science, Springer, 2014, pp. 3–20.

[9] A. K. A. de Medeiros, A. J. M. M. Weijters, W. M. P. van der Aalst, Genetic process mining: an experimental evaluation, Data Min. Knowl. Discov. 14 (2) (2007) 245–304.

[10] J. De Weerdt, M. De Backer, J. Vanthienen, B. Baesens, A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs, Inf. Syst. 37 (7) (2012) 654–676.

[11] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2013.

[12] D. W. Embley, S. W. Liddle, Big data—conceptual modeling to the rescue, in: International Conference on Conceptual Modeling, Springer, 2013, pp. 1–8.

[13] D. Fahland, M. de Leoni, B. F. van Dongen, W. M. P. van der Aalst, Conformance Checking of Interacting Processes with Overlapping Instances, in: S. Rinderle-Ma, F. Toumani, K. Wolf (eds.), Business Process Management - 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011. Proceedings, vol. 6896 of Lecture Notes in Computer Science, Springer, 2011, pp. 345–361.

[14] A. R. Hevner, S. T. March, J. Park, S. Ram, Design science in information systems research, MIS Quarterly 28 (1) (2004) 75–105.

[15] M. J. Jans, M. G. Alles, M. A. Vasarhelyi, The case for process mining in auditing: Sources of value added and areas of application, International Journal of Accounting Information Systems 14 (1) (2013) 1–20.

[16] T. Jouck, B. Depaire, Ptandloggenerator: A generator for artificial event data, in: BPM (Demos), vol. 1789 of CEUR Workshop Proceedings, CEUR-WS.org, 2016, pp. 23–27.

[17] X. Lu, R. Mans, D. Fahland, W. M. P. van der Aalst, Conformance checking in healthcare based on partially ordered event data, in: Proceedings of the 2014 IEEE Emerging Technology and Factory Automation, ETFA 2014, Barcelona, Spain, September 16-19, 2014, 2014, pp. 1–8.

[18] P. Mukala, J. C. A. M. Buijs, M. Leemans, W. M. P. van der Aalst, Learning Analytics on Coursera Event Data: A Process Mining Approach, in: Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015), Vienna, Austria, December 9-11, 2015., 2015, pp. 18–32.

[19] J. Munoz-Gama, Conformance Checking and Diagnosis in Process Mining - Comparing Observed and Modeled Processes, Springer, 2016.

[20] J. Munoz-Gama, J. Carmona, W. M. P. van der Aalst, Single-Entry Single-Exit decomposed conformance checking, Inf. Syst. 46 (2014) 102–122.

[21] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (4) (1989) 541–580.

[22] J. Ribeiro, J. Carmona, M. Misir, M. Sebag, A Recommender System for Process Discovery, in: Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings, 2014, pp. 67–83.

[23] S. Rogers, Big data is scaling BI and analytics–data growth is about to accelearate exponentially, Information Management 21 (5) (2011) 14.

[24] E. Rojas, J. Munoz-Gama, M. Sepúlveda, D. Capurro, Process mining in healthcare: A literature review, Journal of Biomedical Informatics 61 (2016) 224–236.

[25] A. Rozinat, Process Mining: Conformance and Extension, Ph.D. thesis, Technische Universiteit Eindhoven (2010).

[26] A. Rozinat, W. M. P. van der Aalst, Conformance checking of processes based on monitoring real behavior, Inf. Syst. 33 (1) (2008) 64–95.

[27] F. Taymouri, J. Carmona, A Recursive Paradigm for Aligning Observed Behavior of Large Structured Process Models, in: Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings, 2016, pp. 197–214.

[28] F. Taymouri, J. Carmona, Model and Event Log Reductions to Boost the Computation of Alignments, in: Proceedings of the 6th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2016), Graz, Austria, December 15-16, 2016., 2016, pp. 50–62.

[29] W. M. P. van der Aalst, Decomposing Process Mining Problems Using Passages, in: S. Haddad, L. Pomello (eds.), Application and Theory of Petri Nets - 33rd International Conference, PETRI NETS 2012, Hamburg, Germany, June 25-29, 2012. Proceedings, vol. 7347 of Lecture Notes in Computer Science, Springer, 2012, pp. 72–91.

[30] W. M. P. van der Aalst, Decomposing Petri nets for process mining: A generic approach, Distributed and Parallel Databases 31 (4) (2013) 471–507.

[31] W. M. P. van der Aalst, Process Mining - Data Science in Action, Springer, 2016.

[32] W. M. P. van der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, R. P. J. C. Bose, P. van den Brand, R. Brandtjen, J. C. A. M. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, P. Delias, B. F. van Dongen, M. Dumas, S. Dustdar, D. Fahland, D. R. Ferreira, W. Gaaloul, F. van Geffen, S. Goel, C. W. Günther, A. Guzzo, P. Harmon, A. H. M. ter Hofstede, J. Hoogland, J. E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. L. Rosa, F. M. Maggi, D. Malerba, R. S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H. R. M. Nezhad, M. zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. S. Pérez, R. S. Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. D. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, M. T. Wynn, Process Mining Manifesto, in: F. Daniel, K. Barkaoui, S. Dustdar (eds.), Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I, vol. 99 of Lecture Notes in Business Information Processing, Springer, 2011, pp. 169–194.

[33] W. M. P. van der Aalst, A. Adriansyah, B. F. van Dongen, Replaying history on process models for conformance checking and performance analysis, Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery 2 (2) (2012) 182–192.

[34] W. M. P. van der Aalst, E. Damiani, Processes Meet Big Data: Connecting Data Science with Process Science, IEEE Trans. Services Computing 8 (6) (2015) 810–819.
URL `http://dx.doi.org/10.1109/TSC.2015.2493732`

[35] W. M. P. van der Aalst, K. M. van Hee, J. M. E. M. van der Werf, A. Kumar, M. Verdonk, Conceptual model for online auditing, Decision Support Systems 50 (3) (2011) 636–647.

[36] W. M. P. van der Aalst, K. M. van Hee, J. M. E. M. van der Werf, M. Verdonk, Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor, IEEE Computer 43 (3) (2010) 90–93.

[37] B. F. van Dongen, BPI Challenge 2012 (2012).
URL `https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`

[38] B. F. van Dongen, BPI Challenge 2017 (2017).

[39] S. K. L. M. vanden Broucke, J. Munoz-Gama, J. Carmona, B. Baesens, J. Vanthienen, Event-Based Real-Time Decomposed Conformance Analysis, in: R. Meersman, H. Panetto, T. S. Dillon, M. Missikoff, L. Liu, O. Pastor, A. Cuzzocrea, T. K. Sellis (eds.), On the Move to Meaningful Internet Systems: OTM 2014 Conferences - Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings, vol. 8841 of Lecture Notes in Computer Science, Springer, 2014, pp. 345–363.

[40] B. Vázquez-Barreiros, M. Mucientes, M. Lama, ProDiGen: Mining complete, precise and minimal structure process models with a genetic algorithm, Inf. Sci. 294 (2015) 315–333.

[41] H. M. W. Verbeek, Decomposed Replay using Hiding and Reduction, in: L. Cabac, L. Kristensen, H. Rölke (eds.), PNSE 2016 Workshop Proceedings, Torun, Poland, 2016, pp. 233–252.

[42] H. M. W. Verbeek, Decomposed replay using hiding and reduction as abstraction, LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) XII (2017) 166–186.

[43] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. v. Dongen, W. M. P. v. d. Aalst, ProM 6: The Process Mining Toolkit, in: M. La Rosa (ed.), Proc. of BPM Demonstration Track 2010, vol. 615 of CEUR Workshop Proceedings, CEUR-WS.org, Hoboken, USA, 2010, pp. 34–39.

[44] H. M. W. Verbeek, W. M. P. van der Aalst, Decomposed Process Mining: The ILP Case, in: F. Fournier, J. Mendling (eds.), Business Process Management Workshops - BPM 2014 International Workshops, Eindhoven, The Netherlands, September 7-8, 2014, Revised Papers, vol. 202 of Lecture Notes in Business Information Processing, Springer, 2014, pp. 264–276.

[45] H. M. W. Verbeek, W. M. P. van der Aalst, Merging Alignments for Decomposed Replay, in: F. Kordon, D. Moldt (eds.), Application and Theory of Petri Nets and Concurrency - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings, vol. 9698 of Lecture Notes in Computer Science, Springer, 2016, pp. 219–239.

[46] H. M. W. Verbeek, W. M. P. van der Aalst, J. Munoz-Gama, Divide and Conquer: A Tool Framework for Supporting Decomposed Discovery in Process Mining, The Computer Journal (2017) 1–26.

[47] L. Wang, Y. Du, W. Liu, Aligning observed and modelled behaviour based on workflow decomposition, Enterprise Information Systems 0 (0) (2016) 1–21.