Improving merging conditions for recomposing conformance checking

Wai Lam Jonathan Lee[1], Jorge Munoz-Gama[1], H.M.W. Verbeek[2], Wil M.P. van der Aalst[3], and Marcos Sepúlveda[1]

[1] Pontificia Universidad Católica de Chile (Chile)
{walee, jmun}@uc.cl, marcos@ing.puc.cl,
[2] Eindhoven University of Technology, (The Netherlands)
h.m.w.verbeek@tue.nl
[3] RWTH Aachen University, (Germany)
wvdaalst@pads.rwth-aachen.de

Abstract. Efficient conformance checking is a hot topic in the field of process mining. Much of the recent work focused on improving the scalability of alignment-based approaches to support the larger and more complex processes. This is needed because process mining is increasingly applied in areas where models and logs are "big". Decomposition techniques are able to achieve significant performance gains by breaking down a conformance problem into smaller ones. Moreover, recent work showed that the alignment problem can be resolved in an iterative manner by alternating between aligning a set of decomposed subcomponents before merging the computed sub-alignments and recomposing subcomponents to fix merging issues. Despite experimental results showing the gain of applying recomposition in large scenarios, there is still a need for improving the merging step, where log traces can take numerous recomposition steps before reaching the required merging condition. This paper contributes by defining and structuring the recomposition step, and proposes strategies with significant performance improvement on synthetic and real-life datasets over both the state-of-the-art decomposed and monolithic approaches.

Keywords: Recomposition, Conformance Checking, Process Mining

1 Introduction

In today's organizations, it is important to ensure that process executions follow the protocols prescribed by process stakeholders so that compliance is maintained. Conformance checking in process mining compares event data with the corresponding process model to identify commonalities and discrepancies [2]. Detailed diagnostics provide novel insights into the magnitude and effect of deviations. The state-of-the-art in conformance checking are alignment-based techniques that provide detailed explanations of the observed behavior in terms of modeled behavior [4].

However, one of the limitations of alignment-based approaches is the explosion of state-space during the alignment computation. For example, the classic cost-based alignment approach [4] in the worst case is exponential with respect to the model size [5].

One research line focuses on decomposition techniques which break down a conformance problem into smaller sub-problems [1]. Experimental results have shown that decomposed approaches can be several times faster than their monolithic counterparts and can compute alignments for datasets that were previously infeasible. But until recently, decomposition techniques have been limited to resolving the decision problem of deciding if a log trace is perfectly fitting with the model. As a result, reliable diagnostics are missing. However, recent work has shown that overall alignment results can be computed under decomposed conformance checking by using the so-called *recomposition* approach. A framework that computes overall alignment results in a decomposed manner was presented in [10, 11].

A key result of the work is in defining and proving the border agreement condition which permits the merging of sub-alignment results as an overall result. If the condition is not met, the decomposed sub-components are "recomposed" to encourage the merging condition in the next alignment iteration. Experimental results have shown significant performance gains using recomposition, but they have also shown that the merging aspect of the framework can become a performance bottleneck where log traces may require numerous recompositions to reach the merging condition. Under this context, this paper is a step towards that direction by defining and structuring the recomposition step, proposing different recomposition strategies, and evaluating their impact to the overall computation time. The experimental results show that by applying the presented recomposition strategies, exact alignment results can be computed on synthetic and real-life datasets much faster.

The remainder of the paper is structured as follows: Section 2 introduces the required notations and concepts. In particular, Section 2.2 presents the recomposition approach as the focus of the paper. Section 3 defines and structures the recomposition step and sheds light on the limitations of the existing recomposition strategies. Section 4 presents four recomposition strategies that can be used in the recomposition step. Section 5 details the experimental setup for the evaluation of the proposed strategies, and Section 6 analyzes the experimental results. Section 7 presents the related work. Finally, Section 8 presents some conclusions and future work.

2 Preliminaries

This section introduces basic concepts related to process models, event logs, and alignment-based conformance checking techniques.

Let X be a set. $\mathcal{B}(X)$ denotes the set of all possible multisets over set X, and X^* denotes the set of all possible sequences over set X. $\langle\rangle$ denotes the empty sequence. Concatenation of sequences $\sigma_1 \in X^*$ and $\sigma_2 \in X^*$ is denoted as $\sigma_1 \cdot \sigma_2$. Given a tuple $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in X_1 \times X_2 \times \ldots \times X_n$, $\pi_i(\boldsymbol{x}) = x_i$ denotes the projection operator for all $i \in \{1, \ldots, n\}$. This operator is extended to sequences so that given a sequence $\boldsymbol{\sigma} \in (X_1 \times X_2 \times \ldots \times X_n)^*$ of length m with $\boldsymbol{\sigma} = \langle(x_{1_1}, x_{2_1}, \ldots, x_{n_1}), (x_{1_2}, x_{2_2}, \ldots, x_{n_2}), \ldots, (x_{1_m}, x_{2_m}, \ldots, x_{n_m})\rangle$, $\pi_i(\boldsymbol{\sigma}) = \langle x_{i_1}, x_{i_2}, \ldots, x_{i_m}\rangle$ for all $i \in 1, \ldots, n$. Projection is also defined over sets and functions recursively. Given $Y \subseteq X$ and a sequence $\sigma \in X^*$, $\langle\rangle{\restriction_Y} = \langle\rangle$, and $(\langle x\rangle \cdot \sigma){\restriction_Y} = \langle x\rangle \cdot \sigma{\restriction_Y}$ if $x \in Y$, and

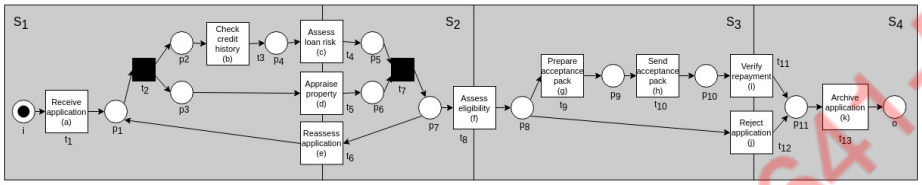

Fig. 1. System net S that models a loan application process (adapted from [6])

$(\langle x \rangle \cdot \sigma)\restriction_Y = \sigma\restriction_Y$ if $x \notin Y$. Similarly, given a function $f : X \to Y$ and a sequence $\sigma = \langle x_1, x_2, \ldots, x_n \rangle \in X^*$, $f(\sigma) = \langle f(x_1), f(x_2), \ldots, f(x_n) \rangle$.

2.1 Preliminaries on Petri net, event log, and net decomposition

In this paper, Petri nets are used to represent process models.

Definition 1 (Labeled Petri net). *Let P denote a set of places, T denote a set of transitions, and $F \subseteq (P \times T) \cup (T \times P)$ denote the flow relation. 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.*

In a process setting, there is typically a well-defined start and end to an instance of the process. This can be denoted with the initial and final marking of a system net.

Definition 2 (System net). *A system net is a triplet $S = (N, I, O)$ where $N = (P, T, F, l)$ is a labeled Petri net, $I \in \mathcal{B}(P)$ is the initial state and $O \in \mathcal{B}(P)$ is the final state. $\phi_f(S)$ is the set of transition sequences that reach the final state when started in the initial state. If σ is a transition sequence, then $l(\sigma\restriction_{dom(l)})$ is an activity sequence.*
$T_v(S) = dom(l)$ is the set of visible transitions *in S. $T_v^u(S) = \{t \in T_v(S) \mid \forall_{t' \in T_v(S)} l(t) = l(t') \Rightarrow t = t'\}$ is the set of unique visible transitions in S.*

Figure 1 presents a system net S that models a loan application process (ignore the grey boxes in the background for now). [i] is the initial marking and [o] is the final marking. An example activity sequence is $\langle \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{f}, \mathsf{g}, \mathsf{h}, \mathsf{i}, \mathsf{k} \rangle$ which corresponds to the occurred events of a successful loan application. The process executions in real-life are recorded as event data and can be expressed as an event log.

Definition 3 (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.*

Figure 2 presents an event log L corresponding to the system net in Figure 1. Log L has 20 cases in total with 5 cases following trace σ_1, 10 cases following trace σ_2, and 5 cases following trace σ_3. In cost-based alignment conformance checking, a trace is aligned with the corresponding system net to produce an alignment.

$$\mathsf{L} = [\overbrace{\langle \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{f}, \mathsf{g}, \mathsf{h}, \mathsf{i}, \mathsf{k} \rangle}^{\sigma_1}{}^{5}, \overbrace{\langle \mathsf{a}, \mathsf{c}, \mathsf{b}, \mathsf{d}, \mathsf{f}, \mathsf{g}, \mathsf{i}, \mathsf{h}, \mathsf{k} \rangle}^{\sigma_2}{}^{10}, \overbrace{\langle \mathsf{a}, \mathsf{c}, \mathsf{b}, \mathsf{d}, \mathsf{f}, \mathsf{g}, \mathsf{j}, \mathsf{k} \rangle}^{\sigma_3}{}^{5}]$$

Fig. 2. Running example: Event log L

Definition 4 (Alignment [4]). *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 $\sigma_M \in \phi_f(S)$ a complete transition sequence of system net S. An alignment of σ_L and σ_M is a sequence of pairs $\gamma \in ((A \cup \{\gg\}) \times (T \cup \{\gg\}))^*$ where $\pi_1(\gamma)\restriction_A = \sigma_L$, $\pi_2(\gamma)\restriction_T = \sigma_M$, $\forall_{(a,t)\in\gamma}\ a \neq \gg \vee t \neq \gg$, and $\forall_{(a,t)\in\gamma}\ a \neq \gg \wedge (t = \gg \vee a = l(t))$.*

Each pair in an alignment is a *legal move*. There are four types of legal moves: a *synchronous move* (a, t) means that the activity matches the activity of the transition, i.e., $a = l(t)$, a *log move* $(a, \gg)$ means that there is a step in the log that is not matched by a corresponding step in the model, a *model move* $(\gg, t)$ where $t \in dom(l)$ means that there is a step in the model that is not matched by a corresponding step in the log, and an *invisible move* $(\gg, t)$ where $t \in T \setminus dom(l)$ means that the step in the model corresponds to an invisible transition that is not observable in the log.

Definition 5 (Valid decomposition [1] and Border activities [11]). *Let $S = (N, I, O)$ with $N = (P, T, F, l)$ be a system net. $D = \{S_1, S_2, \ldots, S_n\}$ is a valid decomposition if and only if the following properties are fulfilled:*

- *$S_i = (N_i, I_i, O_i)$ is a system net with $N_i = (P_i, T_i, F_i, l_i)$ for all $1 \leq i \leq n$.*
- *$l_i = l\restriction_{T_i}$ for all $1 \leq i \leq n$.*
- *$P_i \cap P_j = \emptyset$ and $T_i \cap T_j \subseteq T_v^u(S)$ for all $1 \leq i < j \leq n$.*
- *$P = \bigcup_{1\leq i\leq n} P_i$, $T = \bigcup_{1\leq i\leq n} T_i$, and $F = \bigcup_{1\leq i\leq n} F_i$.*

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

$A_b(D) = \{l(t) \mid \exists_{1\leq i<j\leq n}\ t \in T_i \cap T_j\}$ is the set of border activities of the valid decomposition D. To retrieve the sub-nets that share the same border activity, for an activity $a \in rng(l)$, $S_b(a, D) = \{S_i \in D \mid a \in rng(l_i)\}$ is the set of sub-nets that contain a as an observable activity.

Figure 1 presents a valid decomposition D of net S where sub-nets are marked by the grey boxes. For example, sub-net S_1 consists of the transitions t_1, t_2, t_3, t_4, t_5, and t_6. Border activities can be identified as the activities of the transitions that are shared between two sub-nets. They are t_4, t_5, t_6, t_8, t_{11}, and t_{12}. Under the recomposition approach framework, overall alignments can be computed in a decomposed manner.

2.2 Recomposing conformance checking

Figure 3 presents an overview of the recomposing conformance checking framework [11, 10] which consists of the following five steps: (1) The net and log are decomposed using a decomposition strategy, e.g., maximal decomposition [1]. (2) Alignment-based

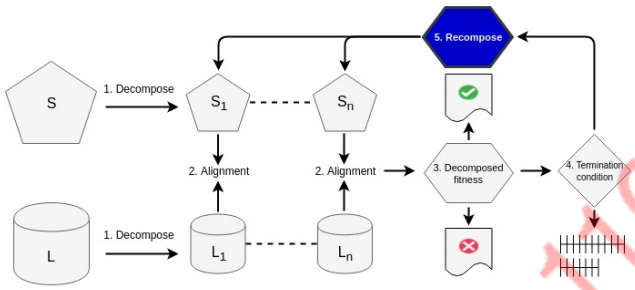

Fig. 3. Recomposing conformance checking framework with the recomposition step highlighted in dark blue

conformance checking is performed per sub-net and sub-log to produce a set of sub-alignments for each log trace. (3) Since sub-components overlap on border activities, the set of sub-alignments for each log trace also overlap on moves involving border activities. In [11], it was shown that if the sub-alignments synchronize on these moves, then they can be merged as an overall optimal alignment using the merging algorithm presented in [18]. This condition was formalized as the *total border agreement* condition. Log traces that do not meet the requirement are either rejected or left for the next iteration. As such, only border activities can cause merge conflicts. (4) User-configured termination conditions are checked at the end of each iteration. If the framework is terminated before computing the overall optimal alignments for all log traces, then an approximate overall result is given. The results of the framework consist of a fitness value and a set of alignments corresponding to the log traces. In the case of an approximate result, the fitness value would be an interval bounding the exact fitness value and the set of alignments would have pseudo alignments. (5) If there are remaining log traces to be aligned and the termination conditions are not reached, then a recomposition step is taken to produce a new net decomposition and a corresponding set of sub-logs. The next iteration of the framework then starts from Step (2).

While experimental results have shown significant performance gains from the recomposition approach over its monolithic counterpart, large scale experimentation has shown that recomposition is a potential bottleneck. In particular, the strategies used at the recomposition step can have a significant impact. The following section takes a more detailed look at the recomposition step and discusses the limitations of the current recomposition strategies.

3 Recomposition step

The recomposition step refers to Step (5) of the framework overview presented in Figure 3 and is highlighted in dark blue. We formalize the step in two parts: the production of a new net decomposition and a corresponding set of sub-logs.

Definition 6 (Recomposition step). *Let $D \in \mathcal{D}(S)$ be a valid decomposition of system net S and let $L = \mathcal{B}(A^*)$ be an event log. For $1 \leq i \leq n$, where $n = |D|$, let*

$M_i = (A_i \cup \{\gg\}) \times (T_i \cup \{\gg\})$ *be the possible alignment moves for a sub-component so that* $\Gamma_D = [(\gamma_{i_1}, \ldots, \gamma_{i_n}) \in M_1^* \times \ldots \times M_n^* \mid \exists_{\sigma_i \in L} \forall_{j \in \{1,\ldots,n\}} \pi_1(\gamma_{i_j}) \restriction_{A_j} = \sigma_i \restriction_{A_j}]$ *contains the latest sub-alignments for all log traces. Given the valid decomposition, and the latest sub-alignments,* $R_S : \mathcal{D}(S) \times \mathcal{B}(M_1^* \times \ldots \times M_n^*) \rightarrow \mathcal{D}(S)$ *creates a new valid decomposition* $D' \in \mathcal{D}(S)$ *where* $m = |D'| < |D|$*. Then, given the new and current net decompositions, the event log, and the latest sub-alignments,* $R_L : \mathcal{D}(S) \times \mathcal{D}(S) \times \mathcal{B}(A^*) \times \mathcal{B}(M_1^* \times \ldots \times M_n^*) \nrightarrow \mathcal{B}({A'_1}^*) \times \ldots \times \mathcal{B}({A'_m}^*)$ *creates a set of sub-logs to align in the following iteration of the recomposition approach. Overall, the recomposition step* R *creates a new net decomposition and a corresponding set of sub-logs,* $R : \mathcal{D}(S) \times \mathcal{B}(A^*) \times \mathcal{B}(M_1^* \times \ldots \times M_n^*) \nrightarrow \mathcal{D}(S) \times \mathcal{B}({A'_1}^*) \times \ldots \times \mathcal{B}({A'_m}^*)$.

The current recomposition strategy involves recomposing on the most frequent conflicting activities (MFC) and constructing sub-logs that contains to-be-aligned traces which carry conflicting activities that have been recomposed upon (IC).

Most frequent conflict (MFC) recomposes the current net decomposition on the activity set $A_r = \{a \in A_b(D) \mid a \in \arg\max_{a' \in A_b(D)} \sum_{\boldsymbol{\gamma_i} \in \text{Supp}(\Gamma_D)} C(\boldsymbol{\gamma_i})(a')\}$ where $\Gamma_D \in \mathcal{B}(M_1^* \times \ldots \times M_n^*)$ are the latest sub-alignments and $C : M_1^* \times \ldots \times M_n^* \rightarrow \mathcal{B}(A)$ is a function that gives the multiset of conflicting activities of sub-alignments. Hence, A_r contains the border activities with the most conflicts.

Inclusion by conflict (IC) then creates a log $L_r = [\sigma_i \in L \mid \exists_{a \in A_b(D)} C(\boldsymbol{\gamma_i})(a) > 0 \wedge a \in A_r]$ where $\boldsymbol{\gamma_i} \in \Gamma_D$ are the sub-alignments of trace $\sigma_i \in L$ and net decomposition $D \in \mathcal{D}(S)$. As such, log L_r includes to-be-aligned log traces which have conflicts on at least one of the border activities that have been recomposed upon. Later, log L_r is then projected onto the new net decomposition to create the corresponding sub-logs.

3.1 Limitations to the current recomposition strategies

To explain the limitations, we refer to the set of optimal sub-alignments in Figure 4 from aligning net decomposition D in Figure 1 and log L in Figure 2. We first note that for the conflicting activities which are highlighted in grey: $\sum_{\boldsymbol{\gamma} \in \Gamma_\mathsf{D}} C(\boldsymbol{\gamma})(\mathsf{c}) = 2$, $\sum_{\boldsymbol{\gamma} \in \Gamma_\mathsf{D}} C(\boldsymbol{\gamma})(\mathsf{i}) = 1$, and $\sum_{\boldsymbol{\gamma} \in \Gamma_\mathsf{D}} C(\boldsymbol{\gamma})(\mathsf{j}) = 1$, where $\Gamma_\mathsf{D} = \{\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\gamma}_3\}$. With activity c being the most frequent conflicting activity, MFC recomposes the current net decomposition on $A_r = \{\mathsf{c}\}$ and IC creates the corresponding sub-logs containing $L_r = \{\sigma_2, \sigma_3\}$ since both traces have activity c as a conflicting activity. The new net decomposition will contain three sub-nets rather than four where sub-net S_1 and sub-net S_2 are recomposed upon activity c as a single sub-net. The corresponding sub-log set is created by projecting log L_r onto the new net decomposition.

While one merge conflict is resolved by recomposing on activity c, the merge conflicts at activity i and j will remain in the following iteration. In fact, under the current recomposition strategy, trace σ_2 and σ_3 have to be aligned three times each to reach the required merging condition to yield overall alignments. This shows the limitation of MFC in only partially resolving merge conflicts on the trace level and IC in leniently including to-be-aligned log traces whose subsequent sub-alignments are unlikely to reach the necessary merging condition.

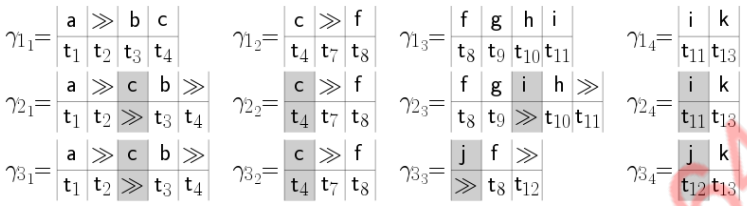

Fig. 4. Sub-alignments $\boldsymbol{\gamma_1} = (\gamma_{1_1}, \gamma_{1_2}, \gamma_{1_3}, \gamma_{1_4})$, $\boldsymbol{\gamma_2} = (\gamma_{2_1}, \gamma_{2_2}, \gamma_{2_3}, \gamma_{2_4})$, and $\boldsymbol{\gamma_3} = (\gamma_{3_1}, \gamma_{3_2}, \gamma_{3_3}, \gamma_{3_4})$ of log L_1 and net decomposition D_1 with merge conflicts highlighted in grey

As such, the key to improving the existing recomposition strategies is in lifting conflict resolution from the individual activity level to the trace level so that the net recomposition strategy resolves merge conflicts of traces rather than activities and the log recomposition strategy selects log traces whose merge conflicts are likely to be fully resolved with the latest net recomposition. In the following section, three net recomposition strategies and one log recomposition strategy are presented. These strategies improve on the existing ones by looking at merge conflict sets, identifying co-occurring conflicting activities, and minimizing the average size of the resulting recomposed subnets. The later experimental results show that the strategies lead to significant performance improvements in both synthetic and real-life datasets.

4 Recomposition strategies

In this section, three net recomposition strategies and one log recomposition strategy are presented.

4.1 Net recomposition strategies

As previously shown, resolving individual conflicting activities may only partially resolve the merge conflicts of traces. This key observation motivates the following net recomposition strategies which target conflicts at the trace level.

Top k most frequent conflict set (MFCS-k) constructs a multiset of conflict sets $A_{cs} = [\text{Supp}(C(\boldsymbol{\gamma})) \subseteq A_b(D) \mid \boldsymbol{\gamma} \in \Gamma_D \wedge |C(\boldsymbol{\gamma})| > 0]$. Then the top k most frequent conflict set $A_{cs,k} \subseteq \{a_{cs} \subseteq A_b(D) | A_{cs}(a_{cs}) > 0\}$ is selected. If $|A_{cs}| < k$, then all conflict sets are taken. Afterwards, the recomposing activity set $A_r = \cup(A_{cs,k}) \subseteq A_b(D)$ is created. We note that in the case where two conflict sets have the same occurrence frequency, a random one is chosen. This secondary criterion avoids bias, and gives better performances empirically than any other straightforward criteria.
Merge conflict graph (MCG) recomposes on conflicting activities that co-occur on the trace level by constructing a weighted undirected graph $G = (V, E)$ where $E = \{\{a_1, a_2\} \mid \exists_{\boldsymbol{\gamma} \in \Gamma_D} a_1 \in C(\boldsymbol{\gamma}) \wedge a_2 \in C(\boldsymbol{\gamma}) \wedge a_1 \neq a_2\}$ with a weight function $w : E \to \mathbb{N}^+$ such that $w((a_1, a_2)) = |\{\boldsymbol{\gamma} \in \Gamma_D \mid C(\boldsymbol{\gamma})(a_1) > 0 \wedge C(\boldsymbol{\gamma})(a_2) > 0\}|$ and $V = \{a \in A_b(D) \mid \exists_{(a_1,a_2) \in E}\ a = a_1 \vee a = a_2\}$. Then, with a threshold $t \in [0, 1]$, edges

are filtered so that $E_f = \{e \in E \mid w(e) \geq t \times w_{\max}\}$ where $w_{\max}$ is the maximum edge weight in E. The corresponding vertex set and filtered graph can be created as $V_f = \{a \in A_b(D) \mid \exists_{(a_1,a_2)\in E_f} a = a_1 \vee a = a_2\}$ and $G_f = (V_f, E_f)$. Finally, the current net decomposition is recomposed on activity set $A_r = V_f$.

Balanced. This recomposition strategy extends the MFCS-k strategy but also tries to minimize the average size of the sub-nets resulting from the recomposition. For a border activity $a \in A_b(D)$, $|(a, D)| = |\cup_{S_i \in S_b(a,D)} A_v(S_i)|$ approximates the size of the recomposed sub-net on activity a. The average size of the recomposed sub-nets for a particular conflict set can then be approximated by $|(A_c, D)| = \frac{\sum_{a\in A_c} |(a,D)|}{|A_c|}$. The score of the conflict set can be computed as a weighted combination $\beta(A_c, D) = w_0 \times \frac{m(A_c)}{\max_{A'_c \in A_{cs}} m(A'_c)} + w_1 \times (1 - \frac{|(A_c,D)|}{\max_{A'_c \in A_{cs}} |(A'_c,D)|})$ where higher scores are assigned to frequent conflict sets that do not recompose to create large sub-nets. The activities of the conflict sets with the highest score, $A_r = \{a \in A_c \mid A_c \in \arg\max_{A'_c \in A_{cs}} \beta(A'_c, D)\}$, are then recomposed upon to create a net decomposition.

4.2 Log recomposition strategy

Similar to the net recomposition strategies, the existing IC strategy can be too lenient in including log traces which have conflicting activities that are unlikely to be resolved in the following decomposed replay iteration.

Strict include by conflict (SIC) increases the requirement for a to-be-aligned log trace to be selected for the next iteration. This addresses the limitation of IC which can include log traces whose merge conflicts are only partially covered by the net recomposition. Given the recomposed activity set A_r, SIC includes log traces as $L_r = [\sigma_i \in L \mid \forall_{a\in C(\gamma_i)}\ a \in A_r]$ with merge conflict if the corresponding conflict set is a subset of set A_r. However, this log strategy only works in conjunction with the net strategies that are based on conflict sets, i.e., MFCS-k and Balanced, so that at least one to-be-aligned log trace is included.

5 Experiment setup

Both synthetic and real-life datasets are used to evaluate the proposed recomposition strategies. Dataset generation is performed using the PTandLogGenerator [8] and information from the empirical study [9]; it is reproducible as a RapidProM workflow [3]. The BPIC 2018 dataset is used [16] as the real-life dataset. Moreover, two baseline net recomposition strategies are used: **All** recomposes on all conflicting activities, and **Random** recomposes on a random number of conflicting activities. Similarly, a baseline log recomposition **All** which includes all to-be-aligned log traces is used. For the sake of space, the full experimental setup and datasets are available at the GitHub repository[4] so that the experimental results can be reproduced.

[4] See `https://github.com/wailamjonathanlee/Characterizing-recomposing-replay`

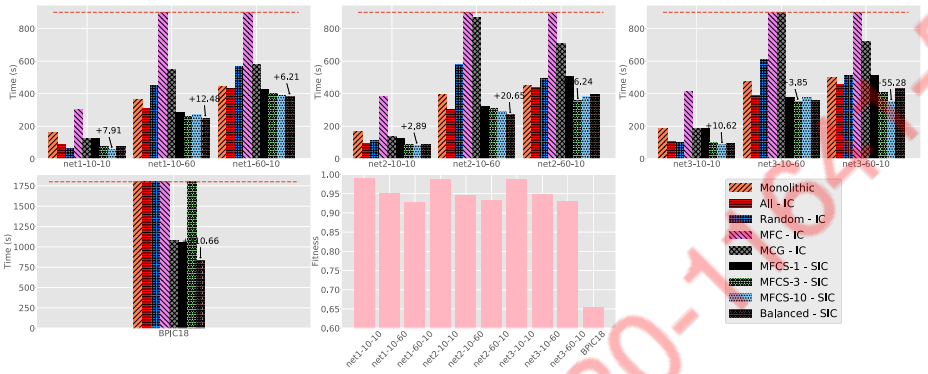

Fig. 5. Bar chart showing fitness and overall time per net recomposition strategy (including the monolithic approach). The time limit is shown as a dashed red line and indicates infeasible replays. Best performing approaches and their time gains from the second fastest times are specified by black arrows.

6 Results

The results shed light on two key insights: First, the selection of the recomposition approach may lead to very different performances. Second, good performance requires both selecting appropriate conflicting activities and well-grouped to-be-aligned log traces.

Figure 5 presents the experimental results for both synthetic and real-life datasets. For each of the synthetic models, there are three event logs of different noise profiles described as net*X*-*noise probability-dispersion over trace* where $X \in \{1, 2, 3\}$. For the sake of readability, we only show the results of three out of five synthetic datasets, but the results are consistent across all five synthetic datasets). Readers interested in more details are referred to the GitHub link for a detailed explanation on noise generation and the rest of the experimental results. For the MFCS-k and Balanced strategies, only configurations using the SIC log strategy are shown; results showed that the SIC log strategy provides a better performance. For the others where SIC is not applicable, only configurations using the IC log strategy are shown as results indicated better performances. Overall, the results show that for both the monolithic and recomposition approach, it is more difficult to compute alignment results for less fitting datasets.

Different approaches give different performances. Comparing the monolithic and recomposition approach, it is clear that the recomposition approach provides a better performance than the monolithic counterpart under at least one recomposition strategy configuration. Furthermore, performance can vary significantly across different recomposition approaches. For example, the existing MFC strategy is the worst performing strategy where it is not able to give exact results for the real-life dataset and both the net*X*-10-60 and net*X*-60-10 noise scenarios of the synthetic datasets. The MFCS-k and Balanced strategies are shown to be the best performing strategies. While for high fitness scenarios, i.e., net*X*-10-10, MFCS-k give better performances with a high $k = 10$. This is because when there is little noise, it becomes simply a "race" to aligning traces with similar merge conflicts. Conversely, for low fitness scenarios, because merge con-

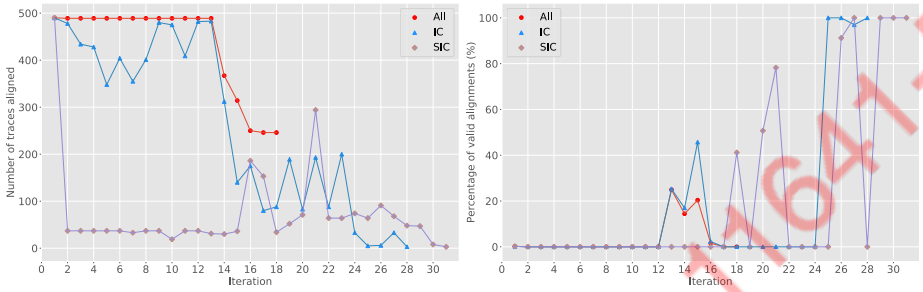

Fig. 6. Comparing log strategies by showcasing the number of aligned traces (left) and percentage of valid alignments (right) per iteration on the real-life dataset BPIC18.

flicts are potentially much more difficult to resolve, the Balanced strategy avoids quickly creating large sub-components that take longer to replay. In these cases, the time differences between the different feasible strategies can go up to three minutes. For all the experiments, the proposed recomposition strategies outperform the baseline strategies. Lastly, for the real-life dataset BPIC18, only the MFCS-1, Balanced, and MCG recomposition strategies are able to compute exact alignment results and the Balanced strategy outperforms MFCS-1 by more than three minutes.

Both net and log recomposition strategies matter. Figure 6 presents the number of aligned traces and percentage of valid alignments per iterations under All, IC, and SIC log strategies with net strategy fixed as Balanced on BPIC18. We first note that only the SIC log strategy resulted with exact alignment results. While all strategies start with aligning all traces in the first iteration, there are significant differences in the number of aligned traces across iterations. Similar to the All strategy, the existing IC strategy includes a high number of traces to align throughout all iterations; the number of aligned traces only tapered off in the later iterations as half of the traces have resulted as valid alignments. This confirms the hypothesis that the existing IC strategy can be too lenient with the inclusion of traces to align. Furthermore, up until iteration 13, none of the aligned traces reaches the necessary merging condition to result as a valid alignment; this means that both the All and IC strategies are "wasting" resources aligning many traces. Conversely, the SIC strategy keeps the number of traces to align per iteration comparatively lower. Moreover, at the peak of the number of traces to align at iteration 21, almost 80% of the $\sim$300 aligned traces resulted as valid alignments. These are likely to explain why only the SIC log strategy is able to compute an exact result.

7 Related work

Performance problems related to alignment-based conformance checking form a well-known problem. A large number of conformance checking techniques have been proposed to tackle this issue. Approximate alignments have been proposed to reduce the problem complexity by abstracting sequential information from segments of log traces [14]. The notion of indication relations has been used to reduce the model and log

prior to conformance checking [15]. Several approaches have been proposed along the research line of decomposition techniques. This include different decomposition strategies, e.g., maximal [1], and SESE-based [12]. Moreover, different decomposed replay approaches such as the hide-and-reduce replay [17] and the recomposition approach [11] have also been investigated. Compared to the existing work, this paper investigates different strategies for the recomposition approach in order to improve the overall performance in computation time.

Other than the alignment-based approach, there are also other conformance checking approaches. This includes the classical token replay [13], behavioral profile approaches [19] and more recently approaches based on event structures [7].

8 Conclusions and Future work

This paper investigated the recomposition aspect of the recomposing conformance checking approach which can become a bottleneck to the overall performance. By defining the recomposition problem, the paper identifies limitations of the current recomposition strategy in not fully resolving merge conflicts on the trace level and also being too lenient in the inclusion of log traces for the subsequent decomposed replay iteration. Based on the observations, three net recomposition strategies and one log recomposition strategy have been presented. The strategies were then evaluated on both synthetic and real-life datasets with two baseline approaches. The results show that different recomposition strategies can significantly impact the overall performance in computing alignments. Moreover, they show that the presented approaches provide a better performance than baseline approaches, and both the existing recomposition and monolithic approaches. While simpler strategies tend to provide a better performance for synthetic datasets, a more sophisticated strategy can perform better for a real-life dataset. However, the results show that both the selection of activities to recompose on and log traces to include are important to achieve superior performances.

Future work. The results have shown that the recomposition strategy has a significant impact on performance. We plan to extend the evaluation of the presented approaches to a larger variety of models, noise scenarios, initial decomposition strategies, and other real-life datasets. For the current and presented approaches, new net decompositions are created by recomposing the initial decomposition on selected activities. Entirely different net decompositions can be created using the merge conflict information from the previous iteration; however, our preliminary results showed that this may be difficult. Lastly, in the current framework, the same strategies (both decomposition and recomposition) are used in all iterations; higher level meta-strategies might be useful. For example, it might be good to switch to the monolithic approach for a small number of log traces that cannot be aligned following many iterations.

Acknowledgments. 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 Com-*

putación UC / Fond-DCC-2017-0001. The authors would like to thank Alfredo Bolt for his comments on the data generation details.

References

1. van der Aalst, W.M.P.: Decomposing Petri nets for process mining: A generic approach. Distributed and Parallel Databases 31(4), 471–507 (2013)
2. van der Aalst, W.M.P.: Process Mining - Data Science in Action. Springer (2016)
3. van der Aalst, W.M.P., Bolt, A., van Zelst, S.J.: Rapidprom: Mine your processes and not just your data. CoRR abs/1703.03740 (2017)
4. Adriansyah, A.: Aligning Observed and Modeled Behavior. Ph.D. thesis, Technische Universiteit Eindhoven (2014)
5. van Dongen, B.F., Carmona, J., Chatain, T., Taymouri, F.: Aligning modeled and observed behavior: A compromise between computation complexity and quality. In: CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings. pp. 94–109 (2017)
6. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013)
7. García-Bañuelos, L., van Beest, N., Dumas, M., Rosa, M.L., Mertens, W.: Complete and interpretable conformance checking of business processes. IEEE Trans. Software Eng. 44(3), 262–290 (2018), https://doi.org/10.1109/TSE.2017.2668418
8. Jouck, T., Depaire, B.: Ptandloggenerator: A generator for artificial event data. In: BPM (Demos). CEUR Workshop Proceedings, vol. 1789, pp. 23–27. CEUR-WS.org (2016)
9. Kunze, M., Luebbe, A., Weidlich, M., Weske, M.: Towards Understanding Process Modeling — the Case of the BPM Academic Initiative. In: International Workshop on Business Process Modeling Notation. pp. 44–58. Springer (2011)
10. Lee, W.L.J., Verbeek, H.M.W., Munoz-Gama, J., van der Aalst, W.M.P., Sepúlveda, M.: Replay using recomposition: Alignment-based conformance checking in the large. In: Proceedings of the BPM Demo Track and BPM Dissertation Award, Barcelona, Spain, September 13, 2017. CEUR Workshop Proceedings, vol. 1920. CEUR-WS.org (2017)
11. Lee, W.L.J., Verbeek, H., Munoz-Gama, J., van der Aalst, W.M.P., Sepúlveda, M.: Recomposing Conformance: Closing the Circle on Decomposed Alignment-Based Conformance Checking in Process Mining. (under review) (2017), processmininguc.com/publications
12. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Single-Entry Single-Exit decomposed conformance checking. Inf. Syst. 46, 102–122 (2014)
13. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)
14. Taymouri, F., Carmona, J.: A Recursive Paradigm for Aligning Observed Behavior of Large Structured Process Models. In: BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings. pp. 197–214 (2016)
15. Taymouri, F., Carmona, J.: Model and Event Log Reductions to Boost the Computation of Alignments. In: SIMPDA 2016, Graz, Austria, December 15-16, 2016. pp. 50–62 (2016)
16. van Dongen, B.F., Borchert, F.: BPI Challenge 2018 (2018)
17. Verbeek, H.M.W.: Decomposed replay using hiding and reduction as abstraction. LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) XII, 166–186 (2017)
18. Verbeek, H.M.W., van der Aalst, W.M.P.: Merging Alignments for Decomposed Replay. In: Kordon, F., Moldt, D. (eds.) PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings. Lecture Notes in Computer Science, vol. 9698, pp. 219–239. Springer (2016)
19. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. Inf. Syst. 36(7), 1009–1025 (2011)