Fast Conformance Analysis based on Activity Log Abstraction

P.M. Dixit
Eindhoven University of Technology
Eindhoven, Netherlands
p.m.dixit@tue.nl

H.M.W. Verbeek
Eindhoven University of Technology
Eindhoven, Netherlands
h.m.w.verbeek@tue.nl

W.M.P. van der Aalst
RWTH, Aachen, Germany
Aachen, Germany
wvdaalst@pads.rwth-aachen.de

***Abstract*—Process mining techniques focus on bridging the gap between activity logs and business process management. Process discovery is a sub-field of process mining which uses activity logs in order to discover process models. Some process discovery techniques, such as interactive process discovery and genetic algorithms, rely on the so-called conformance analysis. In such techniques, process models are discovered in an incremental way, and the quality of the process models is quantified by the results of conformance analysis. State-of-the-art conformance analysis techniques are typically optimized and devised for one-time use. However, in process discovery settings which are incremental in nature, it is imperative to have fast conformance analysis. Moreover, the activity logs used for conformance analysis at each stage remain the same. In this paper, we propose an approach that exploits this fact in order to expedite conformance analysis by approximating the conformance results. We use an abstracted version of an activity log, which can be used to compare with the changing (or new) process models in an incremental process discovery setting. Our results show that the proposed technique is able to outperform traditional conformance techniques in terms of performance by approximating conformance scores.**

***Index Terms*—process mining, conformance analysis, incremental**

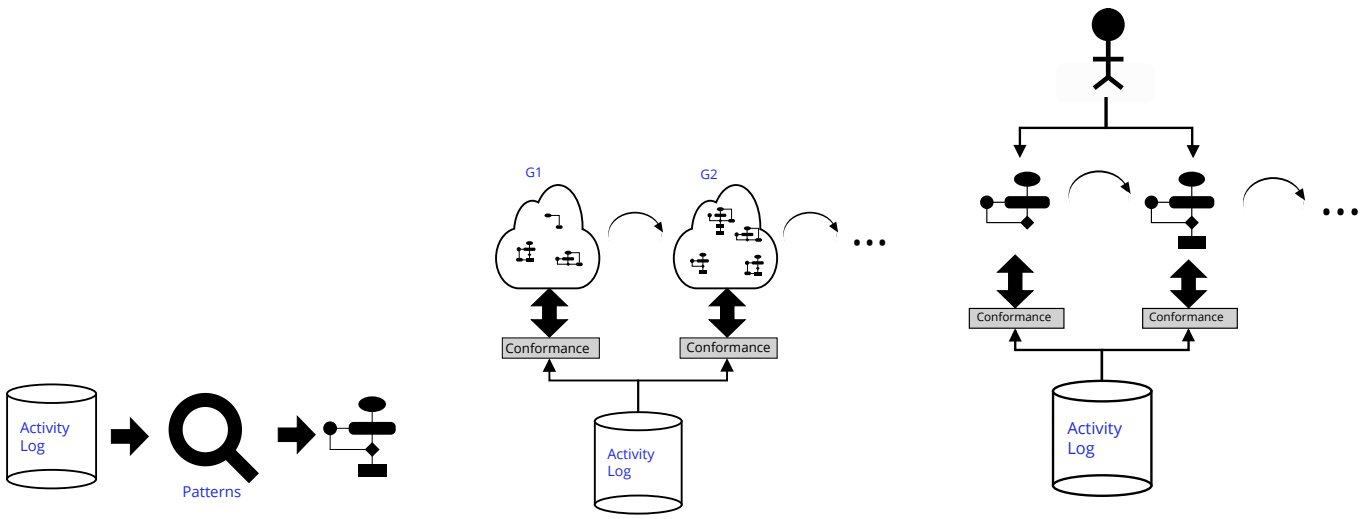
I. Introduction

Process mining has become a key enabler for the control, diagnosis and (re)design of processes in a BPM life cycle. The execution histories of processes, called the activity logs, can be extracted from the corresponding information systems. For example, the activity log of an administrative process can be extracted from an ERP system. These activity logs are the key-enablers for the application of process mining techniques. Broadly, process mining can be categorized into (1) process discovery, and (2) conformance checking [1]. The aim of process discovery is to learn process models using the information from the activity log. Conformance checking techniques on the other hand, focus on finding the goodness-of-fit of a pre-existing process model and the reality as depicted by the activity log.

In an ideal world, the activity logs extracted from the information systems would contain all the necessary information needed by a process discovery algorithm in order to discover the desired process model. However, in reality, this is seldom the case. The activity logs can contain noisy information, due to data quality issues during logging or processing of the activity log. Activity logs can also contain incomplete cases. Hence, the discovery techniques have to deal with such noisy and/or incomplete information from the activity logs extracted from the information systems. State-of-the-art process discovery techniques typically try to overcome such inaccuracies from the activity log, while using other relevant information from the activity logs, in order to discover process models.

Process discovery techniques can use the information from the activity log in three ways. First, the process discovery techniques extract some patterns, which are then translated into process models directly. Figure 1a shows the typical overview of such algorithms. Second, the discovery algorithms incrementally discover newer versions of process models, starting with an arbitrary process model. Genetic algorithms are a prime example of this technique, and work as shown in Figure 1b. Every process model discovered is compared with the activity log to validate the goodness of the process model. Third, the discovery approach could be interactive, and involve a human-in-the-loop (see Figure 1c). In such techniques, the user can build-up a process model, starting with an initially empty process model. Every action of the user during interactive process discovery leads to a new variant of a process model, which can be evaluated against the activity log. Since the process models are generated in an incremental way for both genetic and interactive process discovery techniques, we refer to them as incremental discovery techniques.

Typically, there are four quality dimensions of process mining (*fitness, precision, generalization and simplicity* [2]), that are used as the guiding principle by the incremental process discovery techniques. The *fitness* dimension determines how much behavior from the activity log is allowed by a process model. The *precision* dimension on the other hand determines how much extra behavior does a process model allow which is not present in the activity log. The *generalization* dimension evaluates the degree to which the discovered model is generic, i.e., the extent to which it is coherent with some *unseen* future behavior. Finally, the *simplicity* dimension addresses the comprehensibility of the discovered model. There is no real consensus on the best way to measure generalization and simplicity, and multiple viewpoints are possible. However, there is more argument on fitness and precision dimensions and these can be computed entirely based on the activity log. The incremental process discovery techniques rely on the



(a) Process discovery techniques which abstract knowledge from the activity log, and then use it for deducing process model.

(b) Genetic algorithms which construct process models first, and then compare it with the activity log. G1, G2,... are the 1^{st} and 2^{nd} generations resp., and each generation contain multiple process models.

(c) Interactive process discovery setting wherein the user interactively models/discovers a process model. Bigger process models are obtained starting with smaller process models. Every process model variant is compared to the activity log.

Fig. 1: Usage of activity logs by different process discovery techniques.

outcome of conformance techniques for these dimensions.

Traditionally, conformance techniques are used to analyze how well a pre-existing process model conforms to the reality as depicted by the activity logs. Hence, much emphasis is put on the accuracy of the results. Traditionally, conformance results can be used to perform analysis such as compliance issues in the process behavior. Furthermore, conformance results can also be used to analyze the performance of the process in reality, for e.g., to find out where the possible bottlenecks in the process are. However, in the case of incremental process discovery techniques, the in-depth analysis offered by conformance techniques are not very relevant. It can be argued that the accuracy of the conformance outcome can be compromised for speed to a certain extent. For e.g., in the case of genetic discovery techniques, a multitude of process models can be produced in each generation. Each of these process models need to be compared with the activity log, in order to judge the quality of the process model. Hence, it is important to calculate the conformance (i.e. fitness and precision) scores for each of the process models in an efficient way. A similar argument holds in an interactive process discovery setting. Typically, in such settings, the waiting times suffered by the user should be minimal. That is, it is undesirable for the user to wait for a long time to get the conformance scores upon making a change in the process model.

In order to address these issues pertaining to the usage of conformance techniques in the context of incremental process discovery, we propose a novel framework based on activity log abstraction for conformance checking. In the context of incremental process discovery, even though the process model(s) changes over time, the activity log remains the same. Hence, it is not necessary to go through the entire activity log again and again in order to perform conformance analysis. In this paper, we exploit this fact. The main contribution of this paper is to discuss fast and approximate conformance calculations that can be used to enable efficient incremental process discovery, without losing too much on the accuracy of the results.

The rest of the paper is structured as follows. In Section II we discuss the related work from the literature that is relevant to our approach. In Section III we provide the preliminaries, followed by the details of the proposed approach in Section IV. In Section V we evaluate our approach by comparing it with several state-of-the-art techniques, followed by concluding remarks and future directions in Section VI.

II. Related Work

The work presented in this paper centers around proposing faster conformance analysis in order to enable process discovery based on incremental approaches. Since the scope is conformance analysis rather than process discovery itself, in this section we discuss techniques from literature which propose conformance analysis in the context of process mining, that are comparable to our technique. Conformance checking techniques typically match the behavior of the activity logs with the behavior as depicted by process models. [3] was among the first one's to propose conformance checking in the context of process mining using the token based replay in Petri nets. Conformance checking techniques have been

developed for procedural models. For example, alignment-based conformance checking techniques such as [4]–[6] have been proposed that aim to optimally align the behavior of activity traces and the possible process model execution traces. It should be noted that the problem of conformance checking is not restricted by the so-called procedural process models, or traditional information systems based activity logs. Some techniques address conformance checking from other perspectives such as natural language processing, formal methods, real time setting etc. [7]–[11]. However, the main focus of most of these techniques is to provide accurate conformance results. And hence, there is not a lot of emphasis on the performance of the technique itself. Furthermore, these techniques require scanning of the complete activity log in order to perform conformance. This is not ideal in a process discovery setting which is incremental in nature and would require conformance checking in each incremental step, as the performance of the overall system would be drastically impacted if the size of the activity log is very large. There have been specialized strategies proposed to incrementally repair alignments in the context of Evolutionary Tree Miner (ETM) algorithm [12]. However, the class of process models supported by this approach is limited to block-structured process models (process trees).

There are also techniques proposed in order to improve the performance of calculating conformance of a process model and an activity log. Many of these techniques use the so-called divide-and-conquer strategy [13]–[17]. The idea behind these techniques is to decompose a process model into various sub-models based on a specific decomposition strategy, and then to compute alignments on the smaller model (and a corresponding smaller log) and aggregate the information across all the models. Another strategy to improve the computation time of conformance analysis is by using the Projected Conformance Checking technique [18], which projects process models and activity logs onto a subset of activities, and aggregates the results over all combinations of activity subsets. In our approach, we choose a similar direction. However, we first abstract information from the activity logs, and then use this abstracted information in order to calculate conformance on a set of projected activity combinations. We argue that our approach is faster by not having to scan through the entire activity log during every stage (step) of incremental process discovery. This distinguishes it from other approaches.

III. Preliminaries

In this section we discuss some preliminaries used throughout the paper. A bag over some set S is a function from S to the natural numbers that assigns only a finite number of elements from S a positive value. For a bag B over set S and $s \in S$, $B(s)$ denotes the number of occurrences of s in B, often called the cardinality of s in B. Note that a finite *set* of elements of S is also a bag over S, namely the function yielding 1 for every element in the set and 0 otherwise. We use brackets to explicitly enumerate a bag and superscripts to denote cardinalities. For example $[a^2, b^3, c]$ denotes a bag which contains the elements a, b and c 2, 3 and 1 times resp. Bag B is a subbag of bag B', denoted $B \leq B'$, iff, for all $s \in S$, $B(s) \leq B'(s)$. Having defined bags, we now discuss activity logs, workflow nets and fitness and precision in process mining.

TABLE I: An example activity log

Activity trace	Frequency
$\langle a, b, c, d \rangle$	1
$\langle a, b, c, c, d \rangle$	5
$\langle a, b, c, c, c, d \rangle$	1
$\langle a, e, c, c, d \rangle$	1
$\langle a, e, c, d \rangle$	6
$\langle a, c, b, d \rangle$	3
$\langle a, c, e, c, c, c, d \rangle$	1

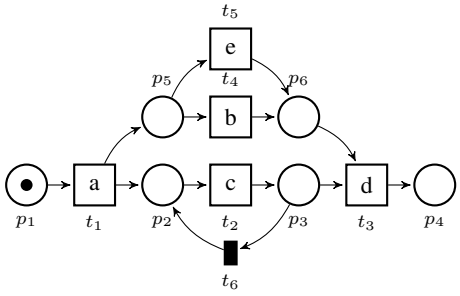


Fig. 2: An example workflow net.

A. Activity Logs

Information systems record the information related to the execution of processes, using the so-called *events*. Events are key-value pairs, which contain values for several attributes related to the event. For e.g., a recorded event can contain information related to the attribute resource, which records the resource responsible for the particular event. An event also contains key value pair for attributes which map the activity to which the event belongs, and the time when the activity was performed. We can use these attributes to extract the so-called activity logs. An activity log is thus a bag of sequences of activities, wherein the ordering of sequences is extracted from the time attribute of the events. Each sequence of activities is called a trace. Table I shows an example of an activity log which contains 7 different activity traces and 18 traces in total. Furthermore, the activity log contains 5 unique activities and 83 activities in total [1].

B. Workflow nets

A workflow net is a Petri net containing a source place and a sink place. Figure 2 shows an example of a workflow net [19], with the source place p_1 and the sink place p_4. We use the regular Petri net semantics for workflow nets [20]. Figure 2 also shows the start state of the workflow net, which corresponds to one token (the black dot) in place p_1. This state *enables* the transition t_1, as all its input places contain a token. As a result transition t_1 may *fire*, which would result in removing a token from every input place and adding a token

to every output place. As a result in the resulting state, the places p_2 and p_5 would contain a token each. This new state would enable transitions t_2, t_4 and t_5, etc.

The visible transitions in this net are labeled with an activity (like t_1 is labeled a); the invisible transitions are not labeled (like t_6). If we consider only the accepting firing sequences of the workflow net, that is the firing sequences that start with a token in $[p_1]$ and end with a token in $[p_4]$, then it is straightforward to check that every activity trace from the example log in Table I can be replayed in the example workflow net of Figure 2. For example, the activity sequence $\langle a, e, c, c, d \rangle$ can be replayed by firing the transitions t_1, t_5, t_2, t_6, t_2, and t_3 in the given order. Note that as invisible transitions are not related to any activity, they do not contribute to the resulting activity sequence, only the visible activities do. Furthermore, in our approach, we only deal with the class of safe workflow nets. That is, at any given state, each place in the workflow net can carry at-most 1 token.

C. Fitness and Precision

An indication of fitness and precision metrics can be obtained using the activity log and the process model. Both the activity log and the process model describe a language, i.e. all the observed behavior in the activity log, and all the allowable behavior by a process model. Fitness metric indicates how much behavior from the activity log is allowed by a process model. On the other hand, precision metric indicates how much extra behavior does a process model allow that is not seen in the activity log. It should be noted that, for constructs such as loops, a process model can allow for infinite behavior. Hence, the precision metric should inherently also account for such behavior.

In this paper, we use the so-called (unary and binary) footprint patterns to estimate the fitness and precision metrics. In the case of fitness, we also consider the frequencies of patterns, as will become evident in the sections to follow.

IV. Approach

We begin by first introducing a framework to enable fast conformance analysis, based on the abstraction of the activity log. This is followed by instantiation of the proposed framework.

A. Framework

The central idea behind our approach is motivated by the various divide and conquer strategies which were introduced to speed-up conformance calculations. Figure 3 shows a high-level overview of how these techniques typically work. To speed-up the conformance analysis approach, these techniques split up one big conformance problem into several smaller problems. That is, instead of checking the conformance of the complete process model in one go, the process model is *split up* into several smaller process fragments. Next, the activity log is also split up corresponding to each of these process fragments. Then, the process fragments are compared with the smaller activity logs in order to calculate the conformance per process fragment. Finally, the overall conformance score is calculated by aggregating the conformance score of each individual process fragment.

The traditional divide-and-conquer-based conformance techniques can offer sufficient speed-up when conformance needs to be calculated only once. However, in our case, the conformance scores have to be calculated several times when the process model changes (or is newly created), e.g. in the process discovery settings that are incremental in nature. It should be noted that in such scenarios, the activity log remains unchanged. *Hence, in our approach, instead of splitting the activity log into several smaller logs based on each process fragment, we pre-compute a meta-structure which abstracts the activity log*. This meta-structure is then used to calculate conformance score. This structure is computed only *once*, and whilst calculating the conformance, the time spent on processing of the activity log is saved. This is especially handy when the activity logs are very big, and the conformance algorithms spend a lot of time to process the activity log, for e.g. during each step (or for each model) of the incremental process discovery techniques. The overview of our approach is shown in Figure 4.

The proposed framework could be instantiated in multiple ways. For example, declare-like constructs can be extracted from activity logs, and then used for computing conformance with a declarative process model. Alternatively, declare-like constructs extracted from the activity logs can also be compared with declare-like constructs extracted from a procedural process model. Another solution could be to use the so-called

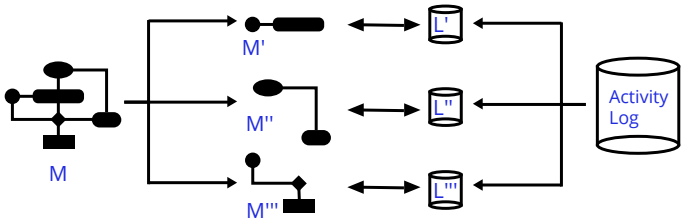


Fig. 3: The traditional divide-and-conquer approach. Each time conformance analysis needs to be performed, the process model as well as the activity log needs to be decomposed and compared.

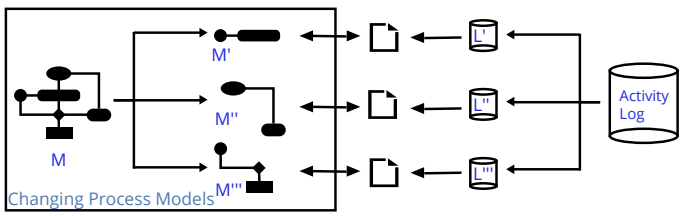


Fig. 4: The divide-and-conquer strategy based on activity log abstraction proposed in this paper. Activity logs are abstracted *once*, and then compared with process models that are changing, or with newer process models.

footprint matrix from an activity log, in order to compute the directly-follows and precedes relations between activities. This footprint matrix can then be compared with the process model to calculate conformance. It should be noted that in scenarios such as interactive process discovery, the process model is built-up interactively, starting with an initial empty process model. Since all the process activities would not be present until the complete process model is constructed, the directly follows/precedes relations would be inadequate. An alternative could be the usage of eventually follows/precedes relation. However, it should be noted that the eventually follows/precedes relation may still be unable to cope very well with looping instances in the traces. We now discuss a solution of the framework which builds upon the footprint relations.

B. Application of the framework

In this section we discuss a solution of the approach. We begin by first discussing the abstraction of activity logs based on the so-called *footprint patterns*, which is central to our approach. This is followed by extraction of footprint patterns from a process model. Finally, we discuss the comparison of footprint patterns of process models and the footprint patterns of activity logs (i.e. abstracted activity logs) to calculate conformance.

1) Footprint patterns from activity log: Information from the activity logs can be abstracted in multiple ways. In our approach, we calculate the so-called footprint patterns from the activity log. The footprint pattern of an activity log is composed of two types: (i) unary footprint patterns and (ii) binary footprint patterns.

Unary footprint patterns are obtained by projecting a log on a single activity. A unary footprint pattern is a bag of projected traces on an activity. We project a log on an activity by removing all the other activities from the activity log. For e.g., the bag of unary footprint patterns of an activity c in the activity log of Table I is $[\langle c\rangle^{10}, \langle c, c\rangle^{6}, \langle c, c, c\rangle, \langle c, c, c, c\rangle]$. The unary footprint pattern thus considers how often an activity is repeated a particular number of times.

Binary footprint patterns are used to calculate relations between activities in a pair-wise manner, for a given trace. We introduce this using an example. Consider a pair of activities (x, y) and a trace $\sigma = \langle \sigma_1, \sigma_2, ..., \sigma_n\rangle$, such that $\forall_{1\leq i\leq n}(\sigma_i = x \vee \sigma_i = y)$. Then, for any i, the binary footprint pattern of σ is calculated as shown in Figure 5. An arc from the binary footprint pattern is removed, if the condition mentioned on the arc is not satisfied. The 1^{st} activity of trace σ can either be x or y. Similarly, the last activity can either be x or y. Hence, for any given trace, there could only be two dashed arcs, one (incoming arc) related to the first activity, and one (outgoing arc) related to the last activity of the trace. Figure 6 shows all the possible footprint patterns with some example traces, when the first activity of the trace is fixed to x, i.e. $\sigma_1 = x$. Note that for a given activity sequence we use the footprint pattern with lowest number of arcs. For e.g., consider a trace $\langle x, y, y\rangle$. Even though the footprint patterns shown in Figure 6k or Figure 6l are valid for such a trace sequence, Figure 6e is chosen as the appropriate footprint pattern as it contains the lowest number of arcs. Unlike the unary footprint patterns, which record the number of times an activity is repeated, the binary footprint patterns do not record the number of times an activity is repeated. Also, when only one activity occurs in a trace sequence (Figure 6a), then the repetition of activity is considered irrelevant for a binary footprint pattern. We can thus obtain binary footprint patterns for all the pairs of activities in the activity log, by *projecting* the activity log on pairs of activities. In order to project an activity log on a pair of activities, we simply remove the activities from the activity log, which are not a part of the pair. Using such a projected activity log, we can then compute binary footprint patterns for all the activity pairs of the entire activity log. Binary footprint patterns for the entire activity log are constructed as a bag of binary footprint patterns. For example, consider a pair (c, e) from the activity log of Table I. The projected activity log for the pair (c, e) is $[\langle c\rangle^{4}, \langle c, c\rangle^{5}, \langle c, c, c\rangle^{1}, \langle e, c, c, c\rangle,$

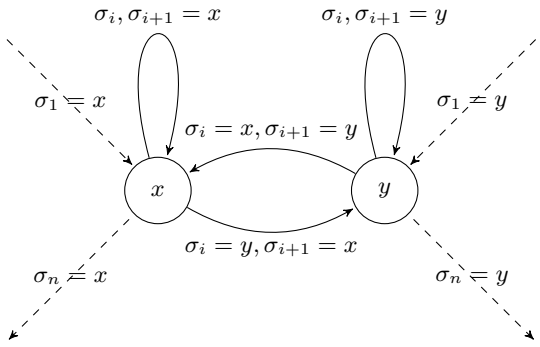
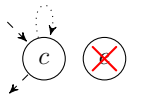
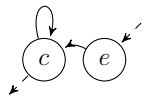
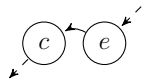
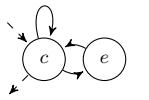
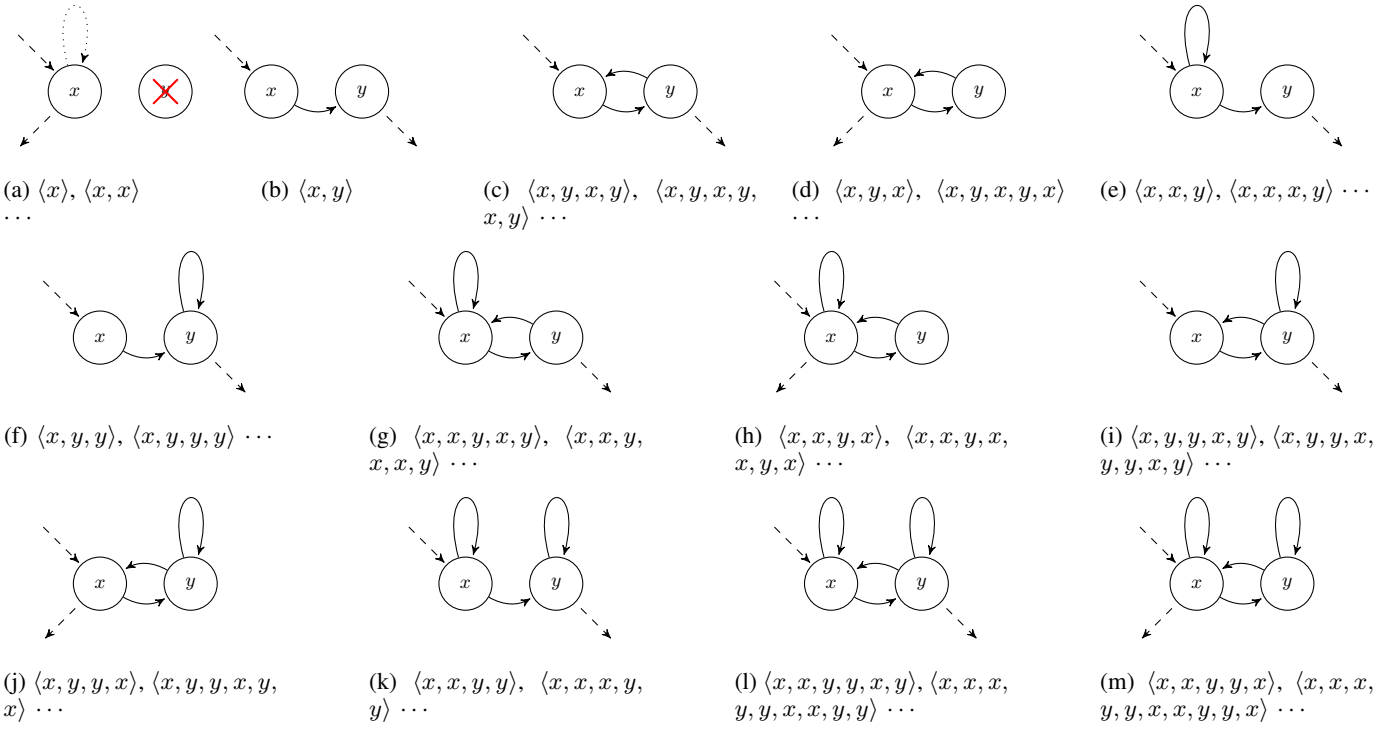


Fig. 5: Binary footprint calculation mechanism for a trace σ.

TABLE II: Bag of footprint patterns on the complete log for the pair (c, e) from Table I.

Binary footprint pattern	Frequency
	10
	1
	6
	1



(a) $\langle x \rangle$, $\langle x, x \rangle$ $\cdots$ (b) $\langle x, y \rangle$ (c) $\langle x, y, x, y \rangle$, $\langle x, y, x, y, x, y \rangle$ $\cdots$ (d) $\langle x, y, x \rangle$, $\langle x, y, x, y, x \rangle$ $\cdots$ (e) $\langle x, x, y \rangle$, $\langle x, x, x, y \rangle$ $\cdots$

(f) $\langle x, y, y \rangle$, $\langle x, y, y, y \rangle$ $\cdots$ (g) $\langle x, x, y, x, y \rangle$, $\langle x, x, y, x, x, y \rangle$ $\cdots$ (h) $\langle x, x, y, x \rangle$, $\langle x, x, y, x, x, y, x \rangle$ $\cdots$ (i) $\langle x, y, y, x, y \rangle$, $\langle x, y, y, x, y, y, x, y \rangle$ $\cdots$

(j) $\langle x, y, y, x \rangle$, $\langle x, y, y, x, y, x \rangle$ $\cdots$ (k) $\langle x, x, y, y \rangle$, $\langle x, x, x, y, y \rangle$ $\cdots$ (l) $\langle x, x, y, y, x, y \rangle$, $\langle x, x, x, y, y, x, x, y, y \rangle$ $\cdots$ (m) $\langle x, x, y, y, x \rangle$, $\langle x, x, x, y, y, x, x, y, y, x \rangle$ $\cdots$

Fig. 6: Binary footprint patterns when the activity traces start with x.

$\langle e, c \rangle^6, \langle c, e, c, c, c \rangle]$. The bag of binary footprint patterns of the pair (c, e) over the projected activity log is shown in Table II.

The footprint patterns of activity log are calculated only once in our approach.

C. Footprint patterns of a process model

We now discuss the computation of footprint patterns from the process models.

Unary footprint patterns In a process model, the unary footprint pattern corresponding to an activity x, is a pair of values (M_x^{min}, M_x^{max}) calculated as follows:

1) Project a process model on x. This is done by making all the activities other than x invisible.
2) We use a minimal trace-based replay to calculate the X_{min} and X_{max} values. Lets call a trace $\sigma^m = \langle x, \cdots, x \rangle$, where $|\sigma^m| = m$.
3) Check whether it is possible to replay an empty trace $\sigma^0 = \langle\rangle$ on the process model. If it is possible, then set $M_x^{min} = 0$. Otherwise, check if the replay of $\sigma^1 = \langle x \rangle$ is possible on the projected model. If yes, then set $M_x^{min} = 1$, and so-on.
4) Next, count the number of visible transitions labeled with the activity x. Let this number be n. Check if a trace σ^{n+1} can be replayed on the projected model. If it is possible, then set $M_x^{max} = \infty$. Otherwise, check if replay of σ^n is possible on the projected model. If it is possible, then set $M_x^{max} = n$, and so-on.

The unary footprint pair for the process model of Figure 7 corresponding to activity c is $(1, \infty)$, whereas the unary footprint pair of the process model of Figure 7 corresponding to the activity e is $(0, 1)$.

Binary footprint patterns are computed based on pairs of activity combinations, for all the activities present in the process model. For any pair of activities, we calculate a set of binary footprint patterns which is *allowed* by a process model. In order to calculate this set, we use the following steps:

1) Project the pair of selected activities on the process model. This is done by making all the transitions which are not part of the activity pair invisible. For example, Figure 7 shows the projection for activity pair (c, e) on the process model as shown by Figure 2. We can further use language preserving reduction rules, to reduce a process model in order to remove unnecessary nodes [21].
2) Again, we use some minimal traces corresponding to each binary footprint pattern, and try to replay them on the projected process model. For example, in order to verify if the binary footprint pattern of Figure 6a holds for a process model, a trace $\langle x \rangle$ is run over the process model. If the process completes successfully, i.e. the end state is reached, then the binary footprint pattern as shown in Figure 6a is added to the set of binary footprint patterns for a pair. These minimal traces for each binary footprint pattern are the first traces of the corresponding binary footprint pattern from Figure 6.

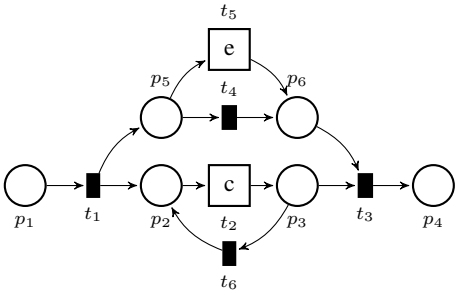


Fig. 7: Projecting activities (c, e) on Figure 2.

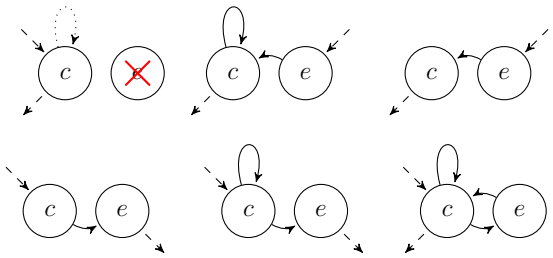


Fig. 8: Set of footprint patterns of the process model as shown in Figure 7.

The set of binary footprint patterns for the activity pair (c ,e) and the workflow net as shown in Figure 7, is shown in Figure 8 .

D. Comparing Process Model and Activity Logs

We compare the footprint patterns of activity logs and process models in order to calculate conformance of an activity log with a process model, based on two metrics: *fitness* and *precision*. It should be noted that, in the case of activity logs, the footprints patterns are computed in terms of bags, whereas in the case of process models, the footprint patterns are computed in terms of sets (for binary footprints) or pairs (for unary footprints).

1) Fitness: Fitness score reflects how much behavior from the activity log is represented by the process model. In our case, we first calculate the individual fitness of each activity based on the unary footprint patterns, as well as the fitness of activity pairs using the binary footprint patterns for all the activity combinations, and then aggregate the results to calculate the overall fitness.

Fitness based on unary footprint patterns Let x be an activity from an activity log. Let L_x^1 be the bag of unary footprint patterns from the activity log for the activity x. Let $M_x^1 = (M_x^{min}, M_x^{max})$ be the unary footprint pattern from the process model for the activity x. Then the fitness f_x^1 of the activity x based on unary footprint patterns is calculated as follows:

$$f_x^1 = \frac{\Sigma_{P \in L_x^1 \wedge M_x^{min} \leq |P| \leq M_x^{max}} L_x^1(P)}{\Sigma_{P \in L_x^1} L_x^1(P)}$$

Using this, the fitness of activity c for the activity log as shown in Table I and process model as shown in Figure 2 is $f_{(c)}^1 = 1$.

Fitness based on binary footprint patterns Let (x, y) be an activity pair. Let $L_{(x,y)}^2$ be the bag of binary footprint patterns from the activity log for the activity pair (x, y). Let $M_{(x,y)}^2$ be the set of binary footprint patterns from the process model for the pair (x, y). Then the fitness $f_{(x,y)}^2$ of the activity pair (x, y) based on binary footprint patterns is calculated as follows:

$$f_{(x,y)}^2 = \frac{\Sigma_{P \in L_{(x,y)}^2 \wedge P \in M_{(x,y)}^2} L_{(x,y)}^2(P)}{\Sigma_{P \in L_{(x,y)}^2} L_{(x,y)}^2(P)}$$

Using this, the fitness score for the activity pair (c, e) based on the binary footprint patterns of Table II and Figure 8 is $f_{(c,e)}^2 = 1$.

The fitness of the entire log is calculated as the average of all the individual finesses for all the pairs of activity combinations (in case of binary footprint patterns) as well as all the activities (in case of unary footprint patterns). The fitness of the complete activity log as shown in Table I and the process model as shown in Figure 2 using binary footprint patterns and unary footprint patterns is 1.

2) Precision: Precision score reflects how much extra behavior a process model allows, when compared with the activity log. Similar to the fitness scores, we first compute the precision value of every activity and every activity pair by comparing the unary footprint patterns and the binary footprint patterns as follows:

Precision based on unary footprint patterns Let x be an activity from an activity log. Let L_x^1 be the bag of unary footprint patterns from the activity log corresponding to the activity x. Let $M_x^1 = (M_x^{min}, M_x^{max})$ be the unary footprint pattern from the process model for the activity x. Then the precision p_x^1 of activity x is calculated as follows:

$$p_x^1 = \frac{L_m}{M_m}$$

where, L_m is the length of longest trace in L_x^1, i.e., $L_m = |P|$ s.t. $P \in L_x^1 \wedge \forall_{P' \in L_x^1} |P'| \leq |P|$ and

$$M_m = \begin{cases} L_m & \text{if } L_m > M_x^{min} \\ L_m + 1 & \text{if } M_x^{max} = \infty \\ M_x^{max} & \text{otherwise} \end{cases}$$

In order to avoid division by ∞, we use an upper bound for M_m, as $L_m + 1$, when $M_x^{max} = \infty$. This also penalizes shorter loops observations from the activity log, when compared to longer the loop observations. Using this, the precision scores of activity c and activity e for the activity log as shown in Table I and the process model as shown in Figure 2 are $p_c^1 = 0.8$ and $p_e^1 = 1$ respectively.

Precision based on binary footprint patterns Let (x, y) be an activity pair. Let $L_{(x,y)}^2$ be the bag of footprint patterns from the activity log for the activity pair (x, y). Let $M_{(x,y)}^2$ be the set of footprint patterns from the process

model for pair (x, y). Then the precision $p^2_{(x,y)}$ of the activity pair (x, y) is calculated as follows:

$$p^2_{(x,y)} = \frac{\sum_{P \in M^2_{(x,y)} \wedge P \in L^2_{(x,y)}} 1}{|M^2_{(x,y)}|}$$

The precision of the pair (c, e) calculated using Table II and Figure 8 is $p^2_{(c,e)} = 0.67$.

Precision of the overall process model with a given activity log is then taken as an average of all the individual precision scores (like with fitness). The precision of the process model as shown in Figure 2 and the activity log as shown in Table I is 0.94.

It should be noted that in the binary patterns for process models, our approach does not consider duplicate occurrences of activities. That is, by considering minimal traces in order to extract binary footprint patterns, the approach ignores the possible duplication of activities in the process model. However, to a certain extent, we address this by using unary patterns for single activities, which considers frequencies of the activity log, as well as the minimum and maximum possible occurrences from the process model.

V. Evaluation

We evaluate our approach based on the performance aspect and the accuracy aspect, by comparing it with state-of-the-art approaches which are relevant to our work. The state-of-the-art techniques used for comparing fitness scores are: the decomposed replay technique [14], the recomposed replay technique [15], the projected conformance checking [18] framework (with $k = 2$), and the alignment-based conformance technique [5]. The state-of-the-art techniques used for comparing precision scores are: the projected conformance checking [18] framework (with $k = 2$), and the escaping edges based ETC 1-align precision [22]. We performed the evaluation using 5 publicly available real-life activity logs: (i) the Sepsis activity log[1] containing the workflow data for roughly 1000 patients suffering from Sepsis in a hospital, (ii) BPIC 2012-A[2] and (iii) BPIC 2012-O[3] activity logs which are the application and the offer sub-logs resp. of a loan application process in Dutch financial institute, (iv) BPIC 2015[4] filtered activity log, which contains top 25 activities of a building permit application for one of the municipalities in the Netherlands, and (v) a hospital billing activity log[5], containing data from the financial modules of the ERP system of a regional hospital. We use the inductive miner-infrequent algorithm [23], in order to discover process models from activity logs. By default, discovery techniques such as the inductive miner and the ILP miner [24] aim at discovering fully fitting process models, i.e. process models which have a fitness value of 1. As we are interested in comparing the fitness and precision values across different conformance algorithms, we configure the inductive miner infrequent algorithm to be used in three settings: (i) *ind0:* a setting assuming 0 noise, which guarantees a fitness of 1, (ii) *ind5:* a setting with noise threshold set at 50% noise, and (iii) *ind10:* a setting with noise threshold set at 100%. We begin by discussing the performance evaluation.

[1]https://data.4tu.nl/repository/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460

[2]https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

[3]https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

[4]https://doi.org/10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17

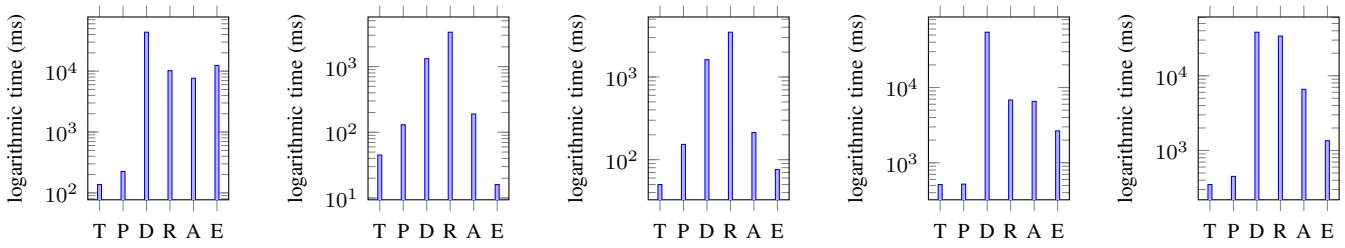
[5]https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741

A. Performance

The primary objective of the approach proposed in this paper is to enable *faster* conformance analysis which would be useful in scenarios such as incremental or interactive process discovery. The extraction of footprint patterns (unary and binary) from the activity logs would serve as an input to the conformance checking during incremental process discovery, and is required to be done only once. Hence, we did not use this processing step during comparison. This initial step took in between $10ms$ to $2500ms$, depending on the type of activity log. Contrary to this, other conformance techniques require the usage of the complete activity log during each step/stage of incremental process discovery. Figure 9 shows the performance of different approaches, in milli-seconds. Note that the scale is logarithmic. The computation time for the PCC framework [18] and our approach includes the calculation time for both the fitness and precision scores. The decomposed replay, recomposed replay and alignment-based replay are used to calculate only the fitness value. The ETC 1-align technique is used to calculate the precision value. It should be noted that, the ETC 1-align technique requires an alignment as an input for calculating the precision value. However, the plots of Figure 9 do not show the time taken for computing alignments in the case of ETC 1-align. It is easy to see that our approach is able to outperform most of the techniques for calculating faster conformance, for both fitness and precision scores. The PCC framework is the closest in terms of compute time compared to our approach. In settings such as interactive process discovery, the waiting times for the end user should be as small as possible. Waiting times of most state-of-the-art techniques can be greater than 10 seconds, and hence become unacceptable.

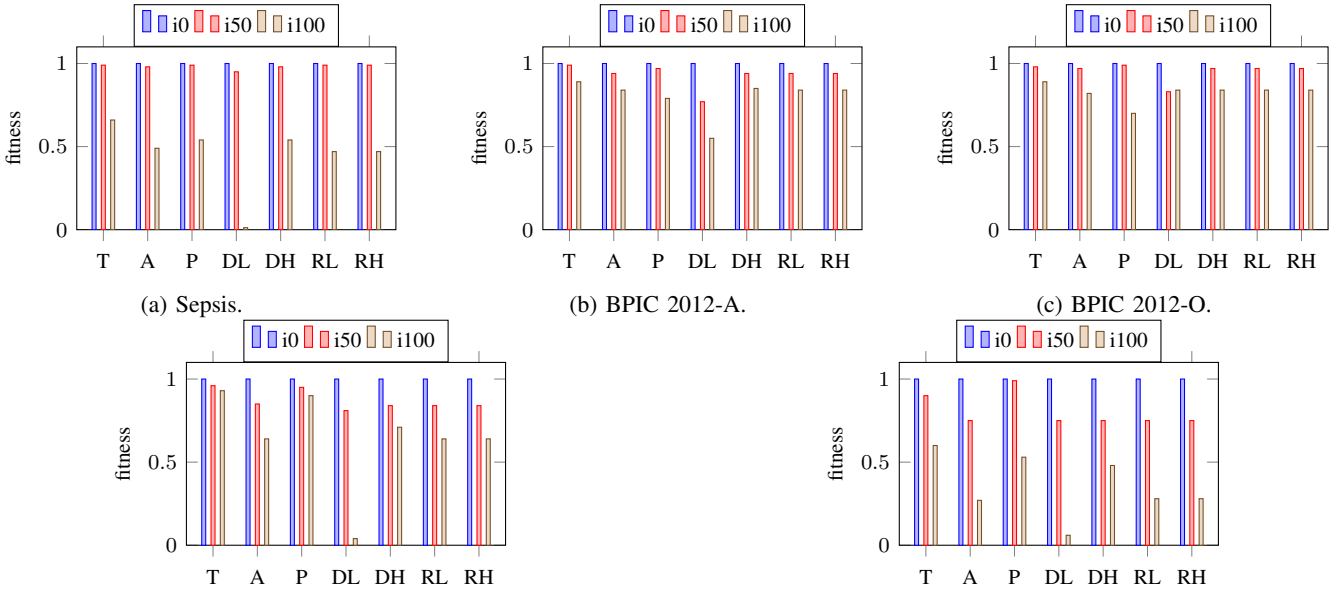
B. Fitness and Precision Comparison

In this sub-section, we compare the outcome of our result, in terms of fitness and precision scores, by comparing it with other approaches. All the techniques considered in this evaluation, calculate the fitness and precision scores in the (inclusive) range of 0-to-1. Higher scores indicate better the fitness/precision. Hence, a fitness score of 1 would indicate a perfect fitness/precision score. Moreover, the decomposed and recomposed replay techniques do not give a singular value for fitness scores, but instead give a lower bound and higher bound for fitness. The comparison outcome of fitness scores of all the techniques is shown in Figure 10. Similarly, the precision outcomes of various techniques is shown in Figure 11. As discussed earlier, we considered three process models for each type of activity log discovered using the inductive miner infrequent at various settings. The inductive miner infrequent



(a) Sepsis. (b) BPIC 2012-A. (c) BPIC 2012-O. (d) BPIC 2015. (e) Hospital billing.

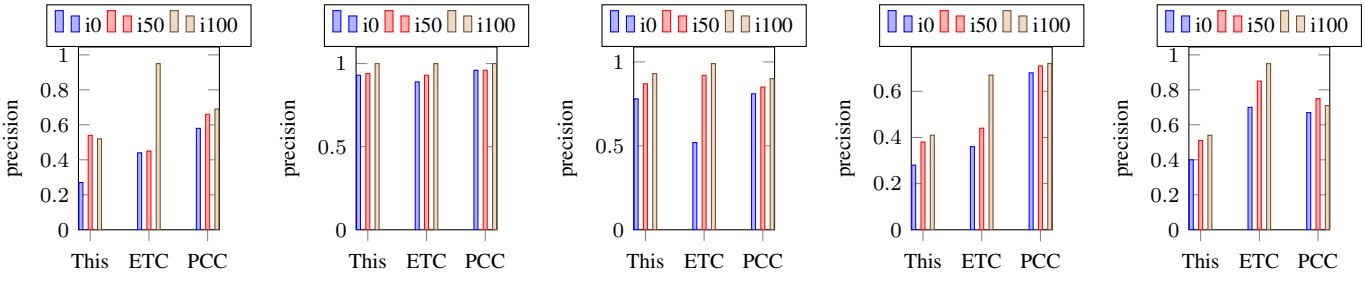
Fig. 9: Time (logarithmic) comparison for various approaches: T - This paper, P - PCC framework ($k = 2$), D - Decomposed Replay, R - Recomposed Replay, A - Alignment-based replay, E - ETC 1-align precision.



(a) Sepsis. (b) BPIC 2012-A. (c) BPIC 2012-O.

(d) BPIC 2015. (e) Hospital billing.

Fig. 10: Fitness values comparison for various approaches: T - This paper, A - Alignment-based replay, P - PCC framework ($k = 2$), DL - Decomposed Replay Lower Bound, DH - Decomposed Replay Higher Bound, RL - Recomposed Replay Lower Bound, RH - Recomposed Replay Higher Bound. The legends indicate the process models considered for calculating fitness, discovered using the inductive miner infrequent technique, such that: (i) *i0* - noise threshold set to 0, (ii) *i50* - noise threshold set to 50% (iii) *i100* - noise threshold set to 100%.



(a) Sepsis. (b) BPIC 2012-A. (c) BPIC 2012-O. (d) BPIC 2015. (e) Hospital billing.

Fig. 11: Precision values comparison for various approaches. The legends indicate the process models considered for calculating precision, discovered using the inductive miner infrequent technique, such that: (i) *i0* - noise threshold set to 0, (ii) *i50* - noise threshold set to 50% (iii) *i100* - noise threshold set to 100%.

guarantees that a process model discovered with a setting of no noise (i.e., noise threshold set to 0), are perfectly fitting

with the activity log. This is also evident in all the fitness scores of all the process models. As the noise threshold is increased, the inductive miner restricts uncommon behavior. Hence while increasing the noise threshold the fitness should drop. This is indeed the trend that is observed across the fitness measures for all the techniques (including ours). Moreover, it can be seen that for most of the scenarios, the fitness scores of our technique are extremely close to the alignment-based replay technique, which can be considered as a baseline for fitness scores. In a similar way, it is quite intuitive to note that as the process models become more restricted (with higher noise threshold), the precision of the process model increases. The PCC framework and our approach typically have a similar precision value, as shown in Figure 11, owing to some overlap of the two approaches in terms of calculating the precision scores. Even though the results are approximated in our approach, the fitness and precision scores computed using our approach still provide a good indication in comparison to the fitness and precision scores as computed by other approaches. Moreover, by abstracting the activity log using footprint patterns, our approach is able to compute these results in a much faster way.

VI. Conclusion and Future work

In this paper, we presented an approach to enable faster conformance analysis by abstracting information from the activity logs using footprint patterns. Furthermore, we presented a way to extract similar footprint patterns from the process models, and also presented a way to compare the footprint patterns from the activity logs with the footprint patterns from the process models, in order to deduce the fitness and precision scores. As shown in the evaluation, by re-using the pre-calculated footprint patterns of the activity log, we improve the performance times of calculating the conformance between a process model and the activity log. Furthermore, the approximated conformance results calculated using our technique are comparable to many state-of-the-art techniques. Hence, we argue that our technique is much more suited for computing conformance in the context of process discovery techniques which are incremental in nature, such as the genetic algorithms, and human-in-the-loop process discovery, and wherein the activity log remains unchanged in different steps/stages of the process discovery. In the future, we would like to extend our approach in order to also consider frequencies of self-loops in binary footprint patterns directly. Furthermore, we would like to allow the possibility for a user to influence the conformance, by adding or removing footprint patterns, thereby incorporating user's domain knowledge.

References

[1] W. M. P. van der Aalst, *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[2] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," *Int. J. Cooperative Inf. Syst.*, vol. 23, no. 1, 2014.

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

[4] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*. IEEE, 2011, pp. 55–64.

[5] ——, "Towards robust conformance checking," in *Business Process Management Workshops*, ser. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2011, vol. 66, pp. 122–133.

[6] T. Chatain and J. Carmona, "Anti-alignments in conformance checking – the dark side of process models," in *Application and Theory of Petri Nets and Concurrency*, F. Kordon and D. Moldt, Eds. Cham: Springer International Publishing, 2016, pp. 240–258.

[7] J. Carmona, *The Alignment of Formal, Structured and Unstructured Process Descriptions*. Cham: Springer International Publishing, 2017, pp. 3–11.

[8] M. De Leoni, F. M. Maggi, and W. M. P. van der Aalst, "Aligning event logs and declarative process models for conformance checking," in *Business Process Management*. Springer, 2012, pp. 82–97.

[9] F. Taymouri and J. Carmona, *A Recursive Paradigm for Aligning Observed Behavior of Large Structured Process Models*. Cham: Springer International Publishing, 2016, pp. 197–214.

[10] S. K. L. M. vanden Broucke, J. Munoz-Gama, J. Carmona, B. Baesens, and J. Vanthienen, *Event-Based Real-Time Decomposed Conformance Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 345–363.

[11] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni, "Verifiable agent interaction in abductive logic programming: The sciff framework," *ACM Trans. Comput. Logic*, vol. 9, no. 4, pp. 29:1–29:43, Aug. 2008.

[12] B. Vázquez-Barreiros, S. J. van Zelst, J. C. A. M. Buijs, M. Lama, and M. Mucientes, *Repairing Alignments: Striking the Right Nerve*. Cham: Springer International Publishing, 2016, pp. 266–281.

[13] W. M. P. van der Aalst, "Decomposing petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507, Dec 2013.

[14] H. M. W. Verbeek, W. M. P. van der Aalst, and J. Munoz-Gama, "Divide and conquer: A tool framework for supporting decomposed discovery in process mining," *The Computer Journal*, vol. 60, no. 11, pp. 1649–1674, 2017.

[15] H. M. W. Verbeek, *Decomposed Replay Using Hiding and Reduction as Abstraction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 166–186.

[16] J. Munoz-Gama, J. Carmona, and W. M. P. van der Aalst, "Single-Entry Single-Exit decomposed conformance checking," *Information Systems*, vol. 46, pp. 102–122, 2014.

[17] F. Taymouri and J. Carmona, "Model and event log reductions to boost the computation of alignments," 2016.

[18] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Scalable process discovery and conformance checking," *Software & Systems Modeling*, Jul 2016.

[19] W. M. P. van der Aalst, "The application of Petri nets to workflow management," *Journal of circuits, systems, and computers*, vol. 8, no. 01, pp. 21–66, 1998.

[20] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, Apr 1989.

[21] J. Desel and J. Esparza, *Free choice Petri nets*. Cambridge university press, 2005, vol. 40.

[22] J. Muñoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *Business Process Management*, R. Hull, J. Mendling, and S. Tai, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 211–226.

[23] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Business Process Management Workshops*. Springer, 2014, pp. 66–78.

[24] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," in *International Conference on Applications and Theory of Petri Nets*. Springer, 2008, pp. 368–387.