Measuring Fitness and Precision of Automatically Discovered Process Models: A Principled and Scalable Approach

Adriano Augusto, Abel Armas-Cervantes, Raffaele Conforti,
Marlon Dumas, Marcello La Rosa, Daniel Reissner

Abstract—Automated process discovery techniques allow us to generate a process model from an event log consisting of a collection of business process execution traces. The quality of process models generated by these techniques can be assessed with respect to several criteria, including *fitness*, which captures the degree to which the generated process model is able to recognize the traces in the event log, and *precision*, which captures the extent to which the behavior allowed by the process model is observed in the event log. A range of fitness and precision measures have been proposed in the literature. However, recent studies have shown that these existing measures do not fulfil a set of intuitive properties, including monotonicity properties. In addition, existing fitness and precision measures suffer from scalability issues when applied to models discovered from real-life event logs. This article presents a family of fitness and precision measures based on the idea of comparing the k-th order Markovian abstraction of a process model against that of an event log. The article shows that this family of measures fulfils the aforementioned properties for suitably chosen values of k. An empirical evaluation shows that representative exemplars of this family of measures yield intuitive results on a synthetic dataset of model-log pairs, while outperforming existing measures of fitness and precision in terms of execution times on real-life event logs.

Index Terms—Process mining, automated process discovery, conformance checking, fitness, precision.

1 Introduction

Contemporary information systems store detailed records of the execution of the business processes they support, including records of the creation of process instances (a.k.a. *cases*), the start and completion of tasks, and other events associated with a case. These records can be extracted as event logs consisting of a set of traces, each trace itself consisting of a sequence of events associated with a case. Automated process discovery techniques [1], [2] allow us to extract process models from such event logs. The quality of process models discovered in this way can be assessed with respect to several quality criteria related to simplicity and accuracy.

Two commonly used criteria for assessing accuracy are fitness and precision [3]. *Fitness* captures the extent to which the behavior observed in an event log is allowed by the discovered process model (i.e. can the process model generate every trace observed in the event log?). Reciprocally, *precision* captures the extent to which the behavior allowed by a discovered process model is observed in the event log. A low precision indicates that the model under-fits the log, i.e. it can generate traces that are unrelated or only partially related to traces observed in the log, while a high precision indicates that it over-fits (i.e. it can only generate traces in the log and nothing more).

Several measures of fitness and precision have been proposed and empirically evaluated in the literature [3]. Until recently however, the theoretical properties of these measures had not been investigated. A recent study [4] has shown that none of the existing measures of precision fulfils a set of five intuitive properties (namely *axioms of precision*). Another recent study [5] postulates a set of desirable properties of fitness measures (namely *fitness propositions*), which existing fitness measures do not fulfil as shown later in this article. In addition, existing precision measures (and to a lesser extent also fitness measures) suffer from scalability issues when applied to models discovered from real-life event logs.

In this setting, this article presents a family of fitness and precision measures based on the idea of comparing the k^{th}-order Markovian abstraction of a process model against that of an event log. We show that the proposed fitness and precision measures fulfil all aforementioned properties for a suitable k dependent on the log. In other words, when measuring fitness and precision, we do not need to explore the entire state space of a process model but only its state space up to a certain memory horizon.

The article empirically evaluates exemplars of the proposed family of measures using: (i) a synthetic collection of models and logs previously used to assess the suitability of precision measures (applicable also to fitness measures); and (ii) a set of models discovered from 20 real-life event logs using three automated process discovery techniques. The synthetic evaluation shows that the exemplar measures rank the model-log pairs in an intuitive manner. Moreover, the precision measures closely approximate the rankings of two existing precision measures that have been proposed as ground truths. Meanwhile, the evaluation based on real-life logs shows that for values of up to $k = 5$, the k^{th}-order Markovian precision measure is considerably more computationally efficient than existing precision measures, while the k^{th}-order Markovian fitness measure outperforms a commonly used fitness measure, except in cases where the process model is

- *A. Augusto, M. Dumas are with the University of Tartu, Estonia. E-mail: {adriano.augusto,marlon.dumas}@ut.ee,*
- *A. Augusto, A. Armas-Cervantes, R. Conforti, M. La Rosa, D. Reissner are with the University of Melbourne, Australia. E-mail: {raffaele.conforti,marcello.larosa}@unimelb.edu.au*

Manuscript received XX; revised XX.

highly permissive (i.e. when it allows arbitrary combinations with repetition of a set of tasks).

This article is an extended and revised version of a conference paper [6]. The main extension with respect to the conference version is the definition of the fitness measures, the proofs that these fitness measures fulfil the properties spelled out in [5], and their empirical evaluation. The article also provides a more detailed description of the approach, including algorithms to compute the k^{th}-order Markovian abstraction of a log and a process model.

The article is structured as follows. Section 2 introduces existing fitness and precision measures and examines them with respect to the theoretical properties postulated in [4] and [5]. Section 3 describes the Markovian abstractions and how to extract them from a process model and an event log. Section 4 defines fitness and precision measures in terms of these Markovian abstractions, and shows that these measures fulfil the theoretical properties. Finally, Section 5 presents the empirical evaluation, while Section 6 draws conclusions and outlines directions for future work.

2 Background and Related Work

As stated above, the quality of a process model discovered from an event log is usually assessed in terms of *fitness* and *precision* [1]. Fitness (a.k.a. *recall*) refers to the amount of behavior observed in the event log that is recognized by the process model. A perfectly fitting process model is one that can recognize every trace in the log. Precision refers to the amount of behavior captured in the process model that is observed in the log. A perfectly precise model is one that can only recognize traces that are observed in the log. Two other quality criteria used in this context are *generalization* and *complexity* (or equivalently, *simplicity*) [1]. Generalization refers to the extent to which the process model captures behavior that, while not observed in the log, is implied by it. Meanwhile, simplicity refers to the understandability of a process model, and is typically measured via size and structural complexity measures. This paper focuses on fitness and precision.

Below, we provide an overview of existing measures of fitness and precision. We then introduce a collection of theoretical properties fitness and precision measures introduced in previous work, and we discuss the limitations of existing fitness and precision measures with respect to these properties.

2.1 Fitness Measures

One of the earliest and simplest measures of fitness is *token-based fitness* [7]. Given an event log and a process model, represented as a Workflow net[1], token-based fitness is equal to one minus a normalized count of the number of tokens that need to be added or deleted when replaying every trace of the event log against the Workflow net. If while replaying a trace, we reach a state (a.k.a. *marking*) where the next event in the trace does not match any enabled transition, then tokens are added to the Workflow net in order to enable (and fire) the transition corresponding to the event. When we finish replaying a trace, if any tokens are left behind in any place other than the end place of the Workflow net, these tokens are deleted. This fitness measure was successively extended by De Medeiros, who proposed two variants: the *continuous parsing* and the *improved continuous semantics fitness*, which optimize the computational efficiency of token-based fitness at the cost of exactness. Another variant of token-based fitness was proposed by vanden Broucke et al. [8]. The idea of this latter approach is to decompose the Workflow net into single-entry single-exit regions and to analyse each region independently, so as to increase scalability and to provide a more localised feedback. While computationally efficient, especially with the aforementioned optimizations, token-based fitness measures have been criticized as being arbitrary, because they make "local decisions" when deciding how to fix a replay error.

Adriansyah et al. [9] argue that a more robust approach to measuring fitness is to compute, for each trace in the log, the closest trace recognized by the process model, and to measure fitness based on the minimal number of error-corrections required to align these two traces (a.k.a. *minimal alignment cost*). This observation underpins the *alignment-based fitness* measure [9], which is equal to one minus a normalized sum of the minimal alignment cost between each trace in the log and the closest corresponding trace recognized by the model. While conceptually sound, *alignment-based fitness* measures sometimes suffer from scalability issues when applied to real-life event logs, due to the need to compute the optimal alignment between the process model and each trace in the log.

In a recent study, Leemans et al. [10] proposed the *projected conformance checking (PCC) fitness* measure. The idea of *PCC fitness* is to construct an automaton from the process model (namely A_m) and from the event log (A_l). To control the size of these automata, the *PCC fitness* focuses on a subset of the activities in the event log, by ignoring the less frequent activities. The two automata A_m and A_l are then used to generate a third automaton, namely $A_{l,m}$, capturing their common behavior. The fitness value is then computed as the ratio between the number of outgoing edges of each state in $A_{l,m}$ and the number of outgoing edges of the corresponding state(s) in A_l. As we will observe in Section 5 , *PCC fitness* suffers from similar (and sometimes more pronounced) scalability issues as alignment-based fitness.

2.2 Precision Measures

Greco et al. [11] define the precision of a (model, log) pair as the *set difference* (*SD*) between the set of traces recognized by the process model and the set of traces in the event log. This measure is a direct operationalization of the concept of precision, but it is not applicable to models with cycles since the latter have an infinite set of traces.

Rozinat and van der Aalst [12] proposed the *advanced behavioral appropriateness* (*ABA*) precision measure. *ABA* precision is based on the comparison between the sets of activity pairs that sometimes but not always follow each other, and the set of activity pairs that sometimes but not always precede each other. The comparison is performed on the sets of activity pairs extracted both from the model and the log behaviors. The *ABA* precision measure does not scale to large models and it is undefined for process models without concurrency or conflict relations [4].

De Weerdt et al. [13] proposed the *negative events* precision measure (*NE*). This method works by inserting inexistent (so-called negative) events to enhance the traces in the log. A negative event is inserted after a given prefix of a trace if this event is never observed preceded by that prefix anywhere in the log. The traces extended with negative events are then replayed on the model. If the model can parse some of the negative events, it means that the

1. A Workflow net is a Petri net with one single start place, one single end place, and such that every transition is on a path from the start to the end place.

model has additional behavior. This approach is however heuristic: it does not guarantee that all additional behavior is identified.

Muñoz-Gama and Carmona [14] proposed the *escaping edges* (*ETC*) precision measure. This approach starts by building a *prefix automaton* from the event log. It then replays the process model behavior on top of this prefix automation and counts the number of times that the model can perform a move that the prefix automaton cannot. Each of these mismatches is called an *escaping edge*. The original *ETC* precision measure assumes that the process model perfectly fits the log. An extension of *ETC* precision applicable to logs containing non-fitting traces, namely *alignments-based ETC* precision (herein ETC_a), in proposed in [15]. *ETC* and ETC_a are not very accurate, since they only count the escaping edges without taking into account how much the process model behavior diverges from the log behavior after each escaping edge.

More recently, van Dongen et al. [16] proposed the *anti-alignment* precision (*AA*). This measure analyses the anti-alignments of the process model behavior to assess the model's precision. An anti-alignment of length n is a trace in the process model behavior of length at most equal to n, which maximizes the Levenshtein distance from all traces in the log. The major drawback of *AA* precision is that it cannot be applied in a real-life context due to its scalability issues and the difficulties in tuning its input parameters [16].

Alongside the *PCC fitness* measure outlined above, Leemans et al. [10] propose a dual measure of precision, namely *PCC precision*. This latter measure is computed in a similar way as *PCC fitness*, with the difference that *PCC precision* is the ratio between the number of outgoing edges of each state in $A_{l,m}$ and the number of outgoing edges of the corresponding states occurring in A_m. We note that *PCC* is the only one of the above approaches where fitness and precision measures are computed based on the same abstraction of the model and the log behavior, specifically an automata-based abstraction. This paper follows up on this idea by proposing fitness and precision measures based on a unified behavioral abstraction, but designed to be more scalable than *PCC* and to fulfill the theoretical limitations of *PCC* exposed below.

2.3 Fitness Propositions

Van der Aalst [5] proposed seven propositions to capture intuitive properties behind the concept of fitness in automated process discovery. Before introducing these propositions, we formally define the notions of process model behavior and event log behavior (and the auxiliary concepts of trace and sub-trace) upon which these propositions rely.

Definition 1. **[Trace]** Given a set of activity labels Ω, we define a trace on Ω as a sequence $\tau_\Omega = \langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle$, such that $\forall 1 \leq i \leq n, t_i \in \Omega$. [2] Furthermore, we denote with τ_i the activity label in position i, and we use the symbol Γ_Ω to refer to the universe of traces on Ω. With abuse of notation, hereinafter we refer to any $t \in \Omega$ as an activity instead of an activity label.

Definition 2. **[Subtrace]** Given a trace $\tau = \langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle$, with the notation $\tau^{i \to j}$, we refer to the subtrace $\langle t_i, t_{i+1}, \ldots, t_{j-1}, t_j \rangle$, where $0 < i < j \leq n$. We extend the subset operator to traces, i.e., given two traces τ and $\widehat{\tau}$, $\widehat{\tau}$ is contained in τ, shorthanded as $\widehat{\tau} \subset \tau$, if and only if (iff) $\exists i, j \in \mathbb{N} \mid \tau^{i \to j} = \widehat{\tau}$.

Definition 3. **[Process Model Behavior]** Given a process model P (regardless of its representation) and being Ω the set of its activities. We refer to the model behavior as $\mathscr{B}_P \subseteq \Gamma_\Omega$, where $\forall \langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle \in \mathscr{B}_P$ there exists an execution of P that allows to execute the sequence of activities $\langle t_1, t_2, \ldots, t_{n-1}, t_n \rangle$, where t_1 is the first activity executed, and t_n the last. [3]

Definition 4. **[Event Log and Event Log Behavior]** Given a set of activities Ω, an event log L is a finite multiset of traces defined over Ω. The event log behavior of L is defined as $\mathscr{B}_L = support(L)$. [4]

Given the above definitions, the seven fitness propositions are spelled out below.

Definition 5. **[Fitness Propositions]**

- **Proposition-1.** A fitness measure is a deterministic function $fit : \mathscr{L} \times \mathscr{P} \to [0,1] \cap \mathbb{R}$, where $\mathscr{L}$ is the universe of event logs, and $\mathscr{P}$ is the universe of processes.
- **Proposition-2.** Given two process models P_1, P_2 and a log L, if the behavior of P_1 is equal to the behavior of P_2, the fitness value of P_1 must be equal to the fitness value of P_2. Formally, if $\mathscr{B}_{P_1} = \mathscr{B}_{P_2} \Longrightarrow fit(L, P_1) = fit(L, P_2)$.
- **Proposition-3.** Given two process models P_1, P_2 and a log L, if the behavior of P_1 is contained in the behavior of P_2, the fitness value of P_2 must be equal to or greater than the fitness value of P_1. Formally, if $\mathscr{B}_{P_1} \subseteq \mathscr{B}_{P_2} \Longrightarrow fit(L, P_2) \geq fit(L, P_1)$.
- **Proposition-4.** Given a process model P and two event logs L_1, L_2, if the behavior of L_2 is contained in the behavior of P, the fitness value of the model measured over the union of L_1 and L_2 must be equal to or greater than the fitness value measured over L_1. Formally, if $\mathscr{B}_{L_2} \subseteq \mathscr{B}_P \Longrightarrow fit(L_1 \cup L_2, P) \geq fit(L_1, P)$.
- **Proposition-5.** Given a process model P and two event logs L_1, L_2, if none of the behavior of L_2 is contained in the behavior of P, the fitness value of the model measured over L_1 must be equal to or greater than the fitness value measured over the union of L_1 and L_2. Formally, if $\mathscr{B}_{L_2} \cap \mathscr{B}_P = \varnothing \Longrightarrow fit(L_1, P) \geq fit(L_1 \cup L_2, P)$.
- **Proposition-6.** Given a process model P and an event log L, altering the frequencies of the traces recorded in L without altering their distribution should not alter the fitness value. Formally, let $n \in \mathbb{N}$ and $L^n = \bigcup_n L \Longrightarrow fit(L, P) = fit(L^n, P)$.
- **Proposition-7.** Given a process model P and a log L, if the behavior of L is contained in the behavior of P, the fitness value of P must be equal to 1. Formally, if $\mathscr{B}_L \subseteq \mathscr{B}_P \Longrightarrow fit(L, P) = 1$.

Van der Aalst [5] does not assess the existing fitness measures against these propositions. Below, we show via counter-examples that none of the aforementioned fitness measures fulfils all the propositions. *Token-based fitness* does not fulfil Proposition-7, indeed, given the log in Table 1 and the model in Figure 6 (Section 5) its score is 0.960 even if the log behavior is fully contained in the model behavior. *Alignment-based fitness* does not fulfil Proposition-3, let us consider two process models P_1 and P_2, respectively Fig. 1 and 2, we can see that the behavior of P_1 is contained in the behavior of P_2, i.e. $\mathscr{B}_{P_1} \subseteq \mathscr{B}_{P_2}$. Nevertheless, given the log in Table 1 Alignment-based fitness scores 0.821 for P_1 and 0.672 for P_2, breaking Proposition-3. Finally, *PCC fitness* does not fulfil Proposition-5, let the process model in Fig. 6 be P, the log L_1

2. For the sake of simplicity, we refer to τ_Ω as τ where there is no ambiguity.

3. When $\mathscr{B}_P = \Gamma_\Omega$, we say that P is a *flower model*. Intuitively, a flower model is a process model that recognizes any trace consisting of any number of occurrences of a given set of tasks, in any order.

4. The support of a multiset is the set of distinct elements of the multiset.

be a single trace set $\{\langle a,b,c,d,e,f,g,h,i\rangle\}$, and the log L_2 be the set $\{\langle t_1,t_2,\cdots,t_8,t_9\rangle \mid t_i \in \{a,b,c,d,e,f,g,h,i\} \wedge t_i = t_j \iff i = j\}$. By construction, $\mathscr{B}_{L_2} \cap \mathscr{B}_P = \varnothing$, though $PCC(L_1 \cup L_2, P) = 1$ and $PCC(L_1, P) = 0$ breaking Proposition-5.

Traces	#
$\langle A,B,D,E,I\rangle$	1207
$\langle A,C,D,G,H,F,I\rangle$	145
$\langle A,C,G,D,H,F,I\rangle$	56
$\langle A,C,H,D,F,I\rangle$	23
$\langle A,C,D,H,F,I\rangle$	28

TABLE 1: Test log [16].

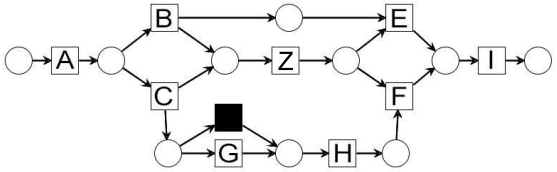


Fig. 1: Process 1.

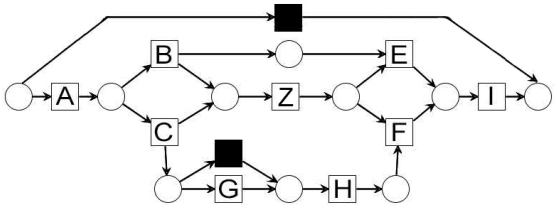


Fig. 2: Process 2.

2.4 Precision Axioms

Tax et al. [4] proposed five axioms to capture intuitive properties behind the concept of precision.[5] The axioms are formally defined below.

Definition 6. **[Precision Axioms]**

- **Axiom-1.** A precision measure is a deterministic function $prec : \mathscr{L} \times \mathscr{P} \to \mathbb{R}$, where $\mathscr{L}$ is the universe of event logs, and $\mathscr{P}$ is the universe of processes.
- **Axiom-2.** Given two process models P_1, P_2 and a log L, if the behavior of L is contained in the behavior of P_1, and this latter is contained in the behavior of P_2, the precision value of P_1 must be equal to or greater than the precision value of P_2. Formally, if $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subseteq \mathscr{B}_{P_2} \implies prec(L,P_1) \geq prec(L,P_2)$.
- **Axiom-3.** Given two process models P_1, P_2 and a log L, if the behavior of L is contained in the behavior of P_1, and P_2 is the flower model, the precision value of P_1 must be greater than the precision value of P_2. Formally, if $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subset \mathscr{B}_{P_2} = \Gamma_\Omega \implies prec(L,P_1) > prec(L,P_2)$.
- **Axiom-4.** Given two process models P_1, P_2 and a log L, if the behavior of P_1 is equal to the behavior of P_2, the precision values of P_1 and P_2 must be equal. Formally, if $\mathscr{B}_{P_1} = \mathscr{B}_{P_2} \implies prec(L,P_1) = prec(L,P_2)$.
- **Axiom-5.** Given a process model P and two event logs L_1, L_2, if the behavior of L_1 is contained in the behavior of L_2, and the behavior of L_2 is contained in the behavior of P, the precision value of the model measured over L_2 must be equal to or greater than the precision value measured over L_1. Formally, if $\mathscr{B}_{L_1} \subseteq \mathscr{B}_{L_2} \subseteq \mathscr{B}_P \implies prec(L_2,P) \geq prec(L_1,P)$.

5. An alternative set of properties of precision measures is proposed in [5]. These latter properties largely overlap with those in [4].

Precision Measure		Axioms				
Name	**Label**	**1**	**2**	**3**	**4**	**5**
Set Difference	*SD*	√	?	×	√	√
Advanced Behavioral Appropriateness	*ABA*	×	?	×	√	?
Negative Events	*NE*	×	×	?	?	?
Alignment-based ETC (one-align)	*ETC*	×	×	×	×	×
Projected Conformance Checking	*PCC*	?	×	?	?	×
Anti-alignment	*AA*	?	?	?	?	×

TABLE 2: Precision axioms fulfiled by existing precision measures (according to [4]).

Tax et al. [4] showed that none of the existing precision measures fulfils all the axioms. Table 2 shows which precision measures fulfil which axioms according to [4].

3 K^{th}-ORDER MARKOVIAN ABSTRACTION

A k[th]-order Markovian abstraction (M[k]-abstraction) is a graph composed of a set of states (S) and a set of edges ($E \subseteq S \times S$). In an M[k]-abstraction, every state $s \in S$ represents a (sub)trace of at most length k, e.g. $s = \langle b,c,d\rangle$, and every state of an M[k]-abstraction is unique, i.e. there are no two states representing the same (sub)trace. Two states $s_1, s_2 \in S$ are connected via an edge $e = (s_1, s_2) \in E$ iff s_1 and s_2 satisfy the following three properties: i) the first activity of the (sub)trace represented by s_1 can occur before the (sub)trace represented by s_2, ii) the last activity of the (sub)trace represented by s_2 can occur after the (sub)trace represented by s_1, and iii) the two (sub)traces represented by s_1 and s_2 overlap with the exception of their first and last activity, e.g. $e = (\langle b,c,d\rangle, \langle c,d,e\rangle)$. An M[k]-abstraction is defined w.r.t. a given order k, which defines the length of the (sub)traces encoded in the states. An M[k]-abstraction contains a special state (denoted as $-$) representing the source and sink of the M[k]-abstraction. Intuitively, every state represents either a *trace* of length less than or equal to k or a *subtrace* of length k, whilst every edge represents an existing *subtrace* of length $k+1$ or a *trace* of length less than or equal to $k+1$. Thus, M[k]-abstraction captures how all the traces of the input behavior evolve in chunks of length k. The definitions below show the construction of an M[k]-abstraction from a given $\mathscr{B}_X$.

Definition 7. **[k[th]-order Markovian Abstraction]** Given a set of traces $\mathscr{B}_X$, the k-order Markovian Abstraction is the graph $M_X^k = (S,E)$ where S is the set of states and $E \subseteq S \times S$ is the set of edges, such that

- $S = \{-\} \cup \{\tau \,:\, \tau \in \mathscr{B}_X \wedge |\tau| \leq k\} \cup \{\tau^{i \to j} \,:\, \tau \in \mathscr{B}_X \wedge |\tau| > k \wedge \left|\tau^{i \to j}\right| = k\}$
- $E = \{(-,\tau) \,:\, \tau \in S \wedge |\tau| \leq k\} \cup \{(\tau,-) \,:\, \tau \in S \wedge |\tau| \leq k\} \cup \{(-,\tau) \,:\, \exists \widehat{\tau} \in \mathscr{B}_X \text{ s.t. } \tau = \widehat{\tau}^{\mathbf{1} \to k}\} \cup \{(\tau,-) : \exists \widehat{\tau} \in \mathscr{B}_X \text{ s.t. } \tau = \widehat{\tau}^{(|\widehat{\tau}|-k+1) \to |\widehat{\tau}|}\} \cup \{(\tau',\tau'') \,:\, \tau',\tau'' \in S \wedge \tau' \oplus \tau''_{|\tau''|} = \tau'_1 \oplus \tau'' \wedge \exists \widehat{\tau} \in \mathscr{B}_X \text{ s.t. } \tau'_1 \oplus \tau'' \subseteq \widehat{\tau}\}$ [6]

A fundamental property of Markovian abstractions (valid for any order k) is the following.

6. The operator $\oplus$ is the *concatenation* operator.

Theorem 1. **[Equality and Containment Inheritance]** Given two sets of traces $\mathscr{B}_X$ and $\mathscr{B}_Y$, and their respective M^k-abstractions $M^k_X = (S_X, E_X)$ and $M^k_Y = (S_Y, E_Y)$, any equality or containment relation between $\mathscr{B}_X$ and $\mathscr{B}_Y$ is inherited by E_X and E_Y. Formally, if $\mathscr{B}_X = \mathscr{B}_Y$ then $E_X = E_Y$, or if $\mathscr{B}_X \subset \mathscr{B}_Y$ then $E_X \subseteq E_Y$.

Proof 1. (Sketch) This follows by construction. Specifically, every edge $e \in E_X$ represents either a subtrace $\tau^{x \to y} : \tau \in \mathscr{B}_X \ \wedge \ |\tau^{x \to y}| = k+1$, or a trace $\tau : \tau \in \mathscr{B}_X \ \wedge \ |\tau| < k+1$. It follows that from the same sets of traces the corresponding M^k-abstractions contain the same sets of edges.

It should be noted that nothing can be said about traces in $\mathscr{B}_Y \setminus \mathscr{B}_X$, i.e. adding new traces to $\mathscr{B}_X$ does not imply that new edges are added to E_X. As a result the relation $\mathscr{B}_X \subset \mathscr{B}_Y$ only guarantees $E_X \subseteq E_Y$.

Moreover, an M^1-abstraction is equivalent to a *directly-follows graph* (a well-known behavioral abstraction used as starting point by many process discovery approaches [17], [18], [19], [20]). Instead, when k approaches infinity then M$^\infty$-abstraction is equivalent to listing all the traces. The order of a Markovian abstraction, i.e. k, allows us to play with the level of behavioral approximation. For example, let us consider the event log L^* as in Tab. 3, and the Process-X (P_x) in Fig. 3c. Their respective M^1-abstractions: $M^1_{L^*}$ and $M^1_{P_x}$ are shown in Fig. 4d and 4c. We can observe that $M^1_{L^*} = M^1_{P_x}$, though $\mathscr{B}_{P_x}$ is infinite whilst $\mathscr{B}_{L^*}$ is not.

Traces
$\langle a,a,b \rangle$
$\langle a,b,b \rangle$
$\langle a,b,a,b,a,b \rangle$

TABLE 3: Log L^*.

This is an example of how the M^1-abstraction can over-approximate the original behavior. Increasing k reduces the over-approximation, thus allowing us to detect more behavioral differences, e.g. increasing k to 2, the differences between L^* and P_x emerge as shown in Figs. 5d and 5c. We note that for k equal to the length of the longest trace in the log, the Markovian abstraction of the log is exact. A similar statement cannot be made for the Markovian abstraction of the model, since the longest trace of a model may be infinite.

3.1 Generating the M^k-abstraction of an Event Log

Given an event log, it is always possible to generate its M^k-abstraction in polynomial time, since the log behavior is a finite set of traces. Algorithm 1 shows how we build the M^k-abstraction abstraction given as inputs: an event log L and the order k. First, we create the set of states (S) and the sets of edges (E) of the M^k-abstraction (lines 1 and 2). Then, we initialize the source/sink state $s_0 = -$. Finally, we iterate over all the traces recorded in the log as follows. For each trace τ with length less or equal to k, we add τ to S, and two new edges (s_0, τ) and (τ, s_0), lines 7 to 9. For each trace τ with length greater than k, we read the trace using a sliding window of size k, moving this latter one activity forward per iteration (lines 15 to 18). The sliding window (in Algorithm 1, depicted by the state s_w) is initialised as the prefix of τ, and the edge (s_0, s_w) is added to the set E (lines 11 to 13). Then, at each iteration, s_w slides over τ, adding a new state to the set S and a new edge to the set E. When s_w becomes the suffix of τ, the iteration is over and a final edge is added to the set E (line 19).

Time Complexity. Algorithm 1 iterates over each activity of each trace of the input event log L. Consequently, the time complexity is polynomial on the total number of activities recorded in the event log, $O(\sum_{\tau \in L} |\tau|)$.

Algorithm 1: Calculating M^k-abstraction of an Event Log

Input : Event Log L
Input : Order k
Result: M^k-abstraction M^k_L

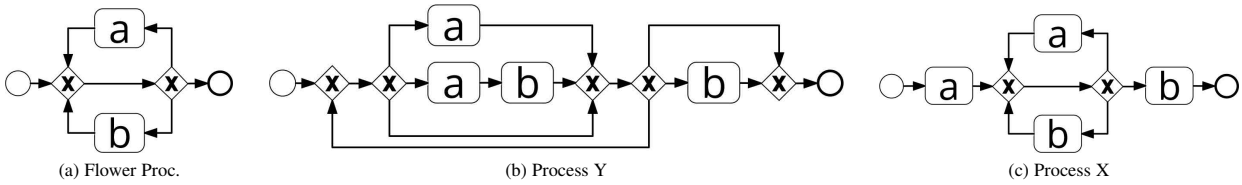
1 Set $S \leftarrow \varnothing$;
2 Set $E \leftarrow \varnothing$;
3 State $s_0 \leftarrow -$;
4 add s_0 to S;
5 **for** $\tau \in L$ **do**
6 **if** $|\tau| \leq k$ **then**
7 add τ to S;
8 add (s_0, τ) to E;
9 add (τ, s_0) to E;
10 **else**
11 State $s_w \leftarrow \tau^{1 \to k}$;
12 add s_w to S;
13 add (s_0, s_w) to E;
14 **for** $i \in [1, |\tau| - k]$ **do**
15 State $s_x \leftarrow s_w$;
16 $s_w \leftarrow \tau^{(1+i) \to (k+i)}$;
17 add s_w to S;
18 add (s_x, s_w) to E;
19 add (s_w, s_0) to E;
20 $M^k_L \leftarrow (S, E)$;
21 **return** M^k_L

3.2 Generating the M^k-abstraction of a Process

Given a process P, its behavior $\mathscr{B}_P$ can be finite or infinite. While in theory, for processes with finite behavior we could use Algorithm 1 this will not work for processes with infinite behaviour. To address this problem, we represent $\mathscr{B}_P$ as a behavioral automaton (Definition 8), and we replay this latter to generate the M^k-abstraction of the process.

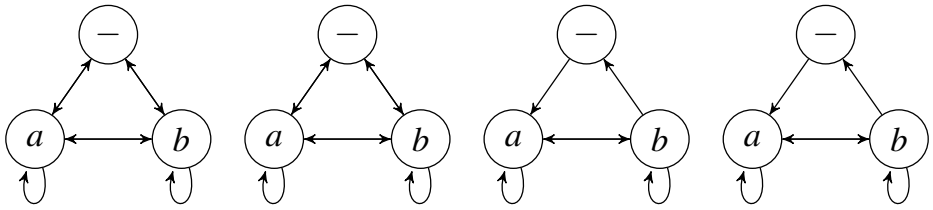
Definition 8. **[Behavioral Automaton of a Process]** Given a process P, its behavioral automaton is a graph $R = (N, A)$, where N is the set of nodes and $A \subseteq N \times N \times \Omega$ is the set of arcs. A node $n \in N$ represents a Process Execution State (PES), whilst an arc $(n_1, n_2, t) \in A$ represents the possibility to move from PES n_1 to the PES n_2 via the execution of activity t. Furthermore, we define $n_0 \in N$ as the initial PES, such that: $\forall (n_1, n_2, t) \in A \Rightarrow n_2 \neq n_0$.

Algorithm 2 shows the steps required to generate the M^k-abstraction for a given process P. First, we initialize the set S and the set E of the M^k-abstraction, and we add the source/sink state $s_0 = -$ to S. Then, we generate the behavioral automaton of the process (R), we retrieve its initial state (n_0), and we initialize all the necessary data structures to perform its replay as follows (lines 5 to 16). We create a queue Q and we add n_0 to it. This queue stores the PESs that have to be explored. We create a map $\mathscr{V}$, to store for each PES the set of (sub)traces whose execution led to the PES and progression needs to be explored (all these sets are initialized as empty, see line 13). We create a map $\mathscr{X}$, to store for each PES the set of (sub)traces whose execution led to the PES and progression have been explored (all these sets are initialized as empty, see line 14). Finally, we add an empty (sub)trace ($\langle\rangle$) to the set of (sub)traces whose execution led to n_0 and progression



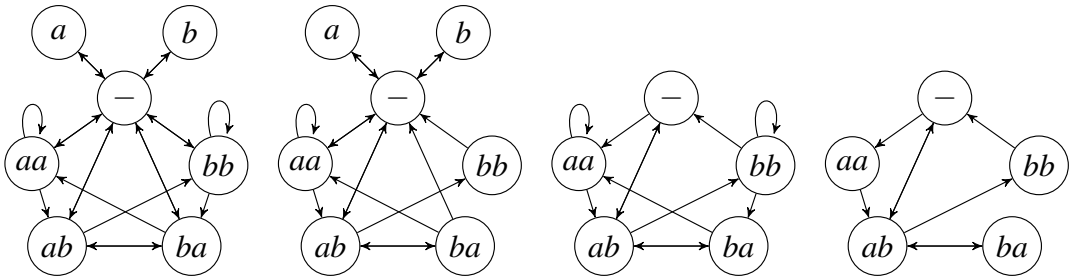
(a) Flower Proc. (b) Process Y (c) Process X

Fig. 3: Examples of processes in the BPMN language.



(a) $1-\frac{2}{8}=0.75$ (b) $1-\frac{2}{8}=0.75$ (c) $1-\frac{0}{6}=1.00$ (d) -

Fig. 4: From left to right: the M^1-abstraction of the Flower Process, Process-Y, Process-X and the event log L^*. The respective labels report the value of their MAP^1.



(a) $1-\frac{12}{20}=0.40$ (b) $1-\frac{8}{16}=0.50$ (c) $1-\frac{4}{12}=0.66$ (d) -

Fig. 5: From left to right, the M^2-abstraction of the Flower Process, Process-Y, Process-X and the event log L^*. The respective labels report the value of their MAP^2.

needs to be explored (lines 17 and18), this empty (sub)trace will be the starting point of the automaton replay.

To explore the automaton behavior we pull the first element of the queue ($\widehat{n}$), we retrieve its set of (sub)traces to explore ($V_{\widehat{n}}$), and its set of outgoing arcs (i.e. all the arcs of the automaton (n_1,n_2,t) s.t. $n_1=\widehat{n}$), see lines 20 to 22. For each (sub)trace $\widehat{\tau}$ whose execution led to $\widehat{n}$ and whose progression needs to be explored, we perform the following operations. We make a copy of $\widehat{\tau}$ (τ, line 24). If $\widehat{n}$ has no outgoing arcs ($O=\varnothing$), τ is either a full-trace (if its length is less than k, line 26) or a trace suffix, [7] accordingly, we add to E the edges (τ,s_0) and (s_0,τ) (this latter only in case of τ full-trace).

If $O\neq\varnothing$, for each outgoing arc $(\widehat{n},n_t,t)$, we execute the following operations. We update τ appending the activity t (line 32), this represents one of the possible progressions of the (sub)trace $\widehat{\tau}$. At this point three possible situations may occur: i) τ length is less than k, this means we did not observe enough activities to add τ as a state in S; ii) τ length is exactly k, this means τ is a prefix as it reached the length k after we appended the activity t, as a consequence we add τ to S and the edge (s_0,τ) to E (lines 33 to 35); iii) τ length is greater than k, and consequently we add $\tau^{2\to(k+1)}$ to S and $\left(\tau^{1\to k},\tau^{2\to(k+1)}\right)$ to E (lines 37 to 39). Once S and E have been updated, we retrieve the set of (sub)traces explored from n_t (X_{n_t}), if the set does not contain $\tau^{2\to(k+1)}$, we add this latter to the set of (sub)traces to explore from n_t (V_{n_t}), and we add n_t to Q, if this latter does not already contain it (lines 40 to 44).

We repeat these steps for all the outgoing arcs of $\widehat{n}$, then, we update $V_{\widehat{n}}$ and $X_{\widehat{n}}$, moving the (sub)trace $\widehat{\tau}$ from $V_{\widehat{n}}$ to $X_{\widehat{n}}$ (lines 45 to 47). We iterate over these steps until Q is empty, after which M_P^k is completed.

Time Complexity. Algorithm 2 iterates on all the PES of the automaton, up to the number of incoming arcs of a PES (this is $2^{|\Omega|}$ in a behavioral automaton). At each iterations two nested loops are executed. The outer one, on the outgoing arcs of the PES, the inner one, on the (sub)traces to explore from the PES. Since in the worst case, each PES can have a number of outgoing arcs equal to the number of activities executable ($|\Omega|$), and the number of (sub)traces to explore from each PES is capped by the number of combinations of length k over the alphabet Ω, i.e. $k^{|\Omega|}$. The time complexity of Algorithm 2 is $O\left(2^{|\Omega|}\cdot|\Omega|\cdot k^{|\Omega|}\right)=O\left(|\Omega|\cdot 2k^{|\Omega|}\right)$.

7. I.e. no more activities can be executed because $\widehat{n}$ is a final PES, being $O=\varnothing$.

Algorithm 2: Calculating M^k-abstraction of a Process

Input : Process P
Input : Order k
Result: M^k-abstraction M_P^k

Set $S \leftarrow \varnothing$;
Set $E \leftarrow \varnothing$;
State $s_0 \leftarrow -$;
add s_0 to S;
Graph $R \leftarrow generateAutomaton(P)$;
Set $N \leftarrow getPESs(R)$;
Node $n_0 \leftarrow getInititalPES(R)$;
Queue $Q \leftarrow \varnothing$;
add n_0 to Q;
Map $\mathscr{V} \leftarrow \varnothing$;
Map $\mathscr{X} \leftarrow \varnothing$;
for $n \in N$ **do**
    Set $V_n \leftarrow \varnothing$;
    Set $X_n \leftarrow \varnothing$;
    put (n, V_n) in $\mathscr{V}$;
    put (n, X_n) in $\mathscr{X}$;
Set $V_{n_0} \leftarrow getMapValue(\mathscr{V}, n_0)$;
add $\langle\rangle$ to V_{n_0};
while $Q \neq \varnothing$ **do**
    $\widehat{n} \leftarrow poll(Q)$;
    $V_{\widehat{n}} \leftarrow getMapValue(\mathscr{V}, \widehat{n})$;
    Set $O \leftarrow getOutgoingArcs(R, \widehat{n})$;
    for $\widehat{\tau} \in V_{\widehat{n}}$ **do**
        Subtrace $\tau \leftarrow \widehat{\tau}$;
        if $O = \varnothing$ **then**
            if $|\tau| < k$ **then**
                add τ to S;
                add (s_0, τ) to E;
            add (τ, s_0) to E;
        else
            for $(\widehat{n}, n_t, t) \in O$ **do**
                $\tau \leftarrow \tau \oplus t$;
                if $|\tau| = k$ **then**
                    add τ to S;
                    add (s_0, τ) to E;
                else
                    if $|\tau| > k$ **then**
                        add $\tau^{2\to(k+1)}$ to S;
                        add $\left(\tau^{1\to k}, \tau^{2\to(k+1)}\right)$ to E;
                $X_{n_t} \leftarrow getMapValue(\mathscr{X}, n_t)$;
                if $\tau^{2\to(k+1)} \notin X_{n_t}$ **then**
                    $V_{n_t} \leftarrow getMapValue(\mathscr{V}, n_t)$;
                    add $\tau^{2\to(k+1)}$ to V_{n_t};
                    if $n_t \notin Q$ **then** add n_t to Q;
        remove $\widehat{\tau}$ from $V_{\widehat{n}}$;
        $X_{\widehat{n}} \leftarrow getMapValue(\mathscr{X}, \widehat{n})$;
        put $\widehat{\tau}$ in $X_{\widehat{n}}$;
$M_P^k \leftarrow (S, E)$;
return M_P^k

4 Comparing Markovian Abstractions

In this section, we introduce our accuracy measures, the Markovian Abstraction-based fitness and precision. Both measures rely on the comparison of the process and the event log M^k-abstractions. Being these latter graphs, it is possible to compare them applying either a structural comparator (e.g. a graph edit distance) or a simulation-based comparator. Being the latter computational expensive, we decided to opt for the former and to compare the M^k-abstractions using a weighted edge-based graph matching algorithm.

Definition 9. **[Weighted Edge-based Graph Matching Algorithm (GMA)]** A Weighted Edge-based Graph Matching Algorithm (GMA) is an algorithm that receives as input two graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$, and outputs a mapping function $\mathscr{I}_C : E_1 \to (E_2 \cup \{\varepsilon\})$. The function $\mathscr{I}_C$ maps pairs of edges matched by the GMA or, if no mapping was found, the edges in E_1 are mapped to ε, i.e., $\forall e_1, e_2 \in E_1 : \mathscr{I}_C(e_1) = \mathscr{I}_C(e_2) \Rightarrow (e_1 = e_2) \vee (\mathscr{I}_C(e_1) = \varepsilon \wedge \mathscr{I}_C(e_2) = \varepsilon)$. A GMA is characterised by an underlying cost function $C : E_1 \times (E_2 \cup \{\varepsilon\}) \to [0,1]$, s.t. $\forall e_1 \in E_1$ and $\forall e_2 \in E_2 \Longrightarrow C(e_1, e_2) \in [0,1]$ and $\forall e_1 \in E_1 \Longrightarrow C(e_1, \varepsilon) = 1$. Hereinafter, we refer to any GMA as its mapping function $\mathscr{I}_C$.

The GMA implemented in our measures is the Hungarian Algorithm [21], [22], which matches edges of E_1 to edges of E_2 to minimize the total cost of the matching, i.e. $\Sigma_{e_1 \in E_1} C(e_1, \mathscr{I}_C(e_1))$ is minimum. Furthermore, the time complexity of the Hungarian Algorithm is polynomial [23], excluding time complexity for computing the matching costs, i.e. $\forall e_1 \in E_1$ and $\forall e_2 \in E_2$, $C(e_1, e_2)$ is known. We note that, the underlying cost function plays a relevant role both in the time complexity and the accuracy of the comparison. In this paper, we propose two alternative cost functions. The first is a boolean cost function, $C_b : E_1 \times (E_2 \cup \{\varepsilon\}) \to \{0,1\}$, s.t. $C_b(e_1, e_2) = 0 \Longleftrightarrow e_1 = e_2$ otherwise $C_b(e_1, e_2) = 1$. The time complexity of C_b is constant: $O(1)$. The second cost function is the Levenshtein distance [24] normalised on $[0,1]$, we refer to it with the symbol C_l. We remind that in an M^k-abstraction each edge represents a (sub)trace of length $k+1$ (see Definition 7), therefore, we can compute the Levenshtein distance between the (sub)traces represented by two edges. The time complexity of C_l is $O\left(k^2\right)$ [25]. However, the difference between C_b and C_l is not only on their time complexity, the latter is more robust to noise, whilst the former is very strict.

In the following, we show how to compute the Markovian Abstraction-based fitness and precision, and the properties they fulfil. For the remaining part of the section, let L_x be a log, P_x be a process model, and $M_{L_x}^k = (S_{L_x}, E_{L_x})$ and $M_{P_x}^k = (S_{P_x}, E_{P_x})$ be the M^k-abstractions of the log and the model, respectively.

4.1 Markovian Abstraction-based Fitness

Given the GMA $\mathscr{I}_{C_b}$, an event log L and a process P as inputs, we compute the k$^{\text{th}}$-order Markovian abstraction-based fitness (hereby MAF^k) applying Equation 1.

$$MAF^k(L,P) = 1 - \frac{\Sigma_{e \in E_L} C_b(e, \mathscr{I}_{C_b}(e)) \cdot F_e}{\Sigma_{e \in E_L} F_e} \tag{1}$$

Where F_e represents the frequency of the edge $e \in E_L$. Computing F_e is trivial while calculating the M^k-abstraction of the event log.[8] Furthermore, we decided to adopt C_b as underlying cost

8. This does not influence the time complexity of Algorithm 1.

function for our Markovian-abstraction Fitness because choosing C_l would not guarantee the fulfilment of the Proposition-5 for fitness measures.

4.2 Proofs of the 7-Propositions of Fitness

We now turn our attention to show that our Markovian abrastraction-based fitness measure fulfils the propositions presented in Section 2.

Proposition-1. $MAF^k(L,P)$ is a deterministic function. Given a log L and a process P, The construction of M_L^k and M_P^k is fully deterministic for $\mathscr{B}_P$ and $\mathscr{B}_L$ (see Definition 7). Furthermore, being the graph matching algorithm $\mathscr{I}_{C_b}$ deterministic, and being $MAF^k(L,P)$ function of E_L, E_P and $\mathscr{I}_{C_b}$ (see Equation 1), it follows that $MAF^k(L,P)$ is also deterministic with codomain $[0,1]$ by definition.

Proposition-2. Given two process models P_1,P_2 and a log L, if $\mathscr{B}_{P_1} = \mathscr{B}_{P_2} \implies MAF^k(L,P_1) = MAF^k(L,P_2)$. From Theorem 1 the following relation holds: $E_{P_1} = E_{P_2}$. Being $MAF^k(L,P)$ function of E_L, E_P and $\mathscr{I}_{C_b}$ (see Equation 1), it follows straightforward $MAF^k(L,P_1) = MAF^k(L,P_2)$.

Proposition-3. Given two process models P_1,P_2 and a log L, if $\mathscr{B}_{P_1} \subseteq \mathscr{B}_{P_2} \implies MAF^k(L,P_2) \geq MAF^k(L,P_1)$. From Theorem 1 the following relation holds: $E_{P_1} \subseteq E_{P_2}$.
Then, we distinguish two possible cases:

1. if $E_{P_1} = E_{P_2}$, then $MAF^k(L,P_1) = MAF^k(L,P_2)$ follows straightforward (see Proposition-2 proof and Equation 1).
2. if $E_{P_1} \subset E_{P_2}$, then the GMA would find matchings for either the same or a larger number of edges when applied on M_L^k and $M_{P_2}^k$ than when applied on M_L^k and $M_{P_1}^k$. Thus, a smaller or equal number of edges will be mapped to ε in the case of $MAF^k(L,P_2)$, not increasing the total matching cost $\Sigma_{e\in E_L} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e$, and guaranteeing $MAF^k(L,P_2) \geq MAF^k(L,P_1)$.

Proposition-4. Given a process model P and two event logs L_1,L_2, if $\mathscr{B}_{L_2} \subseteq \mathscr{B}_P \implies MAF^k(L_1 \cup L_2,P) \geq MAF^k(L_1,P)$. Let $L_3 = L_1 \cup L_2$, $E_{L_3} = E_{L_2} \cup E_{L_1}$, and $E_d = E_{L_3} \setminus E_{L_1}$, it follows that $\forall e \in E_d \implies e \in E_{L_2} \setminus E_{L_1}$. Being $\mathscr{B}_{L_2} \subseteq \mathscr{B}_P$, $\forall e_2 \in E_{L_2} \exists \widehat{e} \in E_P$ s.t. $e_2 = \widehat{e}$ (see Theorem 1). Consequently, the relation $E_d \subseteq E_P$ holds. Taking into account this latter result, we prove that $MAF^k(L_3,P) \geq MAF^k(L_1,P)$.

$$1 - \frac{\Sigma_{e\in E_{L_3}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e}{\Sigma_{e\in E_{L_3}} F_e} \geq 1 - \frac{\Sigma_{e\in E_{L_1}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e}{\Sigma_{e\in E_{L_1}} F_e}$$

We rewrite the relation.

$$\frac{\Sigma_{e\in E_{L_1}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e + \Sigma_{e\in E_d} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e}{\Sigma_{e\in E_{L_1}\cup E_d} F_e} \leq \frac{\Sigma_{e\in E_{L_1}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e}{\Sigma_{e\in E_{L_1}} F_e}$$

We note that, $\Sigma_{e\in E_d} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e = 0$, because $\forall e \in E_d \implies e \in E_{L_2}$ and $\exists \widehat{e} \in E_P$ s.t. $e = \widehat{e}$ and $C_b(e,\mathscr{I}_{C_b}(e)) = 0$. Removing the zero value from the relation, we obtain:

$$\frac{\Sigma_{e\in E_{L_1}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e}{\Sigma_{e\in E_{L_1}\cup E_d} F_e} \leq \frac{\Sigma_{e\in E_{L_1}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e}{\Sigma_{e\in E_{L_1}} F_e}$$

This latter is always true, because $\Sigma_{e\in E_{L_1}} F_e < \Sigma_{e\in E_{L_1}\cup E_d} F_e$.

Proposition-5. Given a process model P and two event logs L_1,L_2, if $\mathscr{B}_{L_2} \cap \mathscr{B}_P = \varnothing \implies MAF^k(L_1,P) \geq MAF^k(L_1 \cup L_2,P)$. Let $L_3 = L_1 \cup L_2$ and $E_d = E_{L_3} \setminus E_{L_1}$, two scenarios can materialise:

1. $E_d \cap E_P = \varnothing$, this case occurs when none of the (sub)traces of length $k+1$ in $\mathscr{B}_{L_2}$ can be found among the (sub)traces in $\mathscr{B}_P$. It follows that $\forall e \in E_d$, $C_b(e,\mathscr{I}_{C_b}(e)) = 1$, because no matching edges can be found in the set E_P. Consequently, $\Sigma_{e\in E_{L_1}\cup E_d} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e$ will be greater than $\Sigma_{e\in E_{L_1}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e$, and $MAF^k(L_1,P) > MAF^k(L_1 \cup L_2,P)$.
2. $E_d \cap E_P \neq \varnothing$, this case occurs when there exists at least one (sub)trace of length $k+1$ in $\mathscr{B}_{L_2}$ that can be found among the (sub)traces in $\mathscr{B}_P$. in this case we cannot prove $MAF^k(L_1,P) \geq MAF^k(L_1 \cup L_2,P)$ for any k, but only for $k > k^*$, where k^* is the length of the longest (sub)trace in $\mathscr{B}_{L_2}$ that can be found among the (sub)traces in $\mathscr{B}_P$. Indeed, choosing $k > k^*$, we would fall in the first scenario. In the worst case, k^* is the length of the longest trace in L_2.

Proposition-6. Given a process model P and an event log L, let $n \in \mathbb{N}$ and $L^n = \bigcup_n L \implies MAF^k(L,P) = MAF^k(L^n,P)$. The factor n alters the frequency of all the traces in L, but not its behavior, i.e. $\mathscr{B}_L = \mathscr{B}_{L^n}$. Being the M^k-abstraction construction only function of the behavior $M_L^k = M_{L^n}^k$, i.e. $S_L = S_{L^n}$ and $E_L = E_{L^n}$. However, MAF^k is function of the frequencies of the M^k-abstraction edges, and these frequencies will be affected by a factor n, s.t. $\forall e \in E_L$ and $\widehat{e} \in E_{L^n} \mid \widehat{e} = e \implies F_{\widehat{e}} = F_e \cdot n$. Nevertheless, n will not alter the value of MAF^k, as shown below.

$$MAF^k(L^n,P) = \frac{\Sigma_{e\in E_{L_1}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e \cdot n}{\Sigma_{e\in E_L} F_e \cdot n} = \frac{n\cdot\Sigma_{e\in E_{L_1}} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e}{n\cdot\Sigma_{e\in E_L} F_e} = MAF^k(L,P)$$

It follows $MAF^k(L^n,P) = MAF^k(L,P)$.

Proposition-7. Given a process model P and a log L, if $\mathscr{B}_L \subseteq \mathscr{B}_P \implies MAF^k(L,P) = 1$. From Theorem 1 the following relation holds: $E_L \subseteq E_P$, it follows that $\forall e \in E_L \implies C_b(e,\mathscr{I}_{C_b}(e)) = 0$, because for any e in E_L there exists an equal matching edge in E_P. Consequently, $\Sigma_{e\in E_L} C_b(e,\mathscr{I}_{C_b}(e))\cdot F_e = 0$ and $MAF^k(L,P) = 1$.

4.3 Markovian Abstraction-based Precision

Given the GMA $\mathscr{I}_{C_l}$, an event log L and a process P as inputs, we compute the k$^{\text{th}}$-order Markovian abstraction-based precision (hereby MAP^k) applying Equation 2.

$$MAP^k(L,P) = 1 - \frac{\Sigma_{e\in E_P} C_l(e,\mathscr{I}_{C_l}(e))}{|E_P|} \tag{2}$$

Recollecting the BPMN models shown in Figure 3 and their respective Markovian abstractions (for $k=1$ and $k=2$, Figures 4a-4c and 5a-5c). We can observe that by increasing k the quality of the behavioral abstractions increases, capturing more details. Consequently, our MAP^k outputs a finer result. Note that, our proposed precision measure fulfils the properties of an ordinal scale. Specifically, given an event log $\mathscr{L}$ and a k, MAP^k induces an order over the possible process models that fit log $\mathscr{L}$. This

property is desirable given that the purpose of a precision measure is to allow us to compare two different process models in terms of their extra behavior.

4.4 Proofs of the 5-Axioms of Precision

We now turn our attention to show that our Markovian abrastraction-based precision measure fulfils the axioms presented in Section 2.

Axiom-1. $MAP^k(L,P)$ is a deterministic function. Given a log L and a process P, The construction of M_L^k and M_P^k is fully deterministic for $\mathscr{B}_P$ and $\mathscr{B}_L$ (see Definition 7). Furthermore, being the graph matching algorithm $\mathscr{I}_{C_l}$ deterministic, and being $MAP^k(L,P)$ function of E_L, E_P and $\mathscr{I}_{C_l}$ (see Equation 2), it follows that $MAP^k(L,P)$ is also deterministic with codomain $[0,1]$ by definition.

Axiom-2. Given two processes P_1,P_2 and an event log L, s.t. $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subseteq \mathscr{B}_{P_2}$, then $MAP^k(L,P_1) \geq MAP^k(L,P_2)$. First, the following relation holds, $E_L \subseteq E_{P_1} \subseteq E_{P_2}$ (see Theorem 1). Then, we distinguish two possible cases:

1. if $E_{P_1} = E_{P_2}$, then it follows straightforward $MAP^k(L,P_1) = MAP^k(L,P_2)$ (see Axiom-1 proof and Equation 2).
2. if $E_{P_1} \subset E_{P_2}$, then $E_L \subset E_{P_2} \wedge \ (|E_{P_2}| - |E_{P_1}|) > 0$. In this case, we show that $MAP^k(L,P_2) - MAP^k(L,P_1) < 0$ is always true, as follows.

$$1 - \frac{\Sigma_{e_2 \in E_{P_2}} C_l(e_2, \mathscr{I}_{C_l}(e_2))}{|E_{P_2}|} - \left(1 - \frac{\Sigma_{e_1 \in E_{P_1}} C_l(e_1, \mathscr{I}_{C_l}(e_1))}{|E_{P_1}|}\right) = \frac{\Sigma_{e_1 \in E_{P_1}} C_l(e_1, \mathscr{I}_{C_l}(e_1))}{|E_{P_1}|} - \frac{\Sigma_{e_2 \in E_{P_2}} C_l(e_2, \mathscr{I}_{C_l}(e_2))}{|E_{P_2}|} < 0$$

For each edge e_1 that can be found both in E_{P_1} and E_L, the cost $C_l(e_1, \mathscr{I}_{C_l}(e_1))$ is 0, being $\mathscr{I}_{C_l}(e_1) = e_1$. Instead, for each edge e_1 that can be found in E_{P_1} but not in E_L, the cost $C_l(e_1, \mathscr{I}_{C_l}(e_1))$ is 1, being $\mathscr{I}_{C_l}(e_1) = \varepsilon$. It follows that the total cost of matching E_{P_1} over $_L$ is $\Sigma_{e_1 \in E_{P_1}} C_l(e_1, \mathscr{I}_{C_l}(e_1)) = |E_{P_1}| - |E_L|$. A similar reasoning can be done for the matching of E_{P_2} over $_L$. Indeed, $\forall e_2 \in E_{P_2} \cap E_L \implies C_l(e_2, \mathscr{I}_{C_l}(e_2)) = 0$ and $\forall e_2 \in E_{P_2} \setminus E_L \implies C_l(e_2, \mathscr{I}_{C_l}(e_2)) = C_l(e_2, \varepsilon) = 1$, therefore $\Sigma_{e_2 \in E_{P_2}} C_l(e_2, \mathscr{I}_{C_l}(e_2)) = |E_{P_2}| - |E_L|$.
Applying these results to the above inequality, it turns into the following:

$$\frac{|E_{P_1}| - |E_L|}{|E_{P_1}|} - \frac{|E_{P_2}| - |E_L|}{|E_{P_2}|} = \frac{|E_L|\,(|E_{P_1}| - |E_{P_2}|)}{|E_{P_1}|\,|E_{P_2}|} < 0$$

This latter is always true, since the starting hypothesis of this second case is $(|E_{P_1}| - |E_{P_2}|) < 0$.

Axiom-3. Given two processes P_1,P_2 and an event log L, s.t. $\mathscr{B}_L \subseteq \mathscr{B}_{P_1} \subset \mathscr{B}_{P_2} = \Gamma_\Omega$ then $MAP^k(L,P_1) > MAP^k(L,P_2)$. For any $k \in \mathbb{N}$, the relation $MAP^k(L,P_1) \geq MAP^k(L,P_2)$ holds for Axiom-2. The case $MAP^k(L,P_1) = MAP^k(L,P_2)$ occurs when $M_{P_2}^k$ over-approximates the behavior of P_2, i.e. $\mathscr{B}_{P_1} \subset \mathscr{B}_{P_2}$ and $E_{P_1} = E_{P_2}$. Nevertheless, for any $\mathscr{B}_{P_1}$ there always exists a k^* s.t. $E_{P_1} \subset E_{P_2}$. This is true since being $\mathscr{B}_{P_1}$ strictly contained in $\mathscr{B}_{P_2}$, there exists a trace $\widehat{\tau} \in \mathscr{B}_{P_2}$ s.t. $\widehat{\tau} \notin \mathscr{B}_{P_1}$. Choosing $k^* = |\widehat{\tau}|$, the $M_{P_2}^{k^*}$ would produce an edge $\widehat{e} = (-,\widehat{\tau}) \in E_{P_2}$ s.t. $\widehat{e} \notin E_{P_1}$ because $\widehat{\tau} \notin \mathscr{B}_{P_1}$ (see also Definition 7).[9] Consequently, for any $k \geq k^*$, we have $E_{P_1} \subset E_{P_2}$ and $MAP^k(L,P_1) > MAP^k(L,P_2)$ holds, being this latter the case 2 of Axiom-2.

Axiom-4. Given two processes P_1,P_2 and an event log L, s.t. $\mathscr{B}_{P_1} = \mathscr{B}_{P_2}$ then $MAP^k(L,P_1) = MAP^k(L,P_2)$. If $\mathscr{B}_{P_1} = \mathscr{B}_{P_2}$, then $E_{P_1} = E_{P_2}$ (see Theorem 1). It follows straightforward that $MAP^k(L,P_1) = MAP^k(L,P_2)$ (see proof Axiom-1 and Equation 2).

Axiom-5. Given two event logs L_1,L_2 and a process P, s.t. $\mathscr{B}_{L_1} \subseteq \mathscr{B}_{L_2} \subseteq \mathscr{B}_P$, then $MAP^k(L_2,P) \geq MAP^k(L_1,P)$. Consider the two following cases:

1. if $\mathscr{B}_{L_1} = \mathscr{B}_{L_2}$, then $E_{L_1} = E_{L_2}$ (see Theorem 1). It follows $MAP^k(L_2,P) = MAP^k(L_1,P)$, because $MAP^k(L,P)$ is a deterministic function of E_L, E_P and $\mathscr{I}_{C_l}$ (see Axiom-1 proof and Equation 2).
2. if $\mathscr{B}_{L_1} \subset \mathscr{B}_{L_2}$, then $E_{L_1} \subseteq E_{L_2}$ (see Theorem 1). In this case, the graph matching algorithm would find matchings for either the same number or a larger number of edges between M_P^k and $M_{L_2}^k$, than between M_P^k and $M_{L_1}^k$ (this follows from $E_{L_1} \subseteq E_{L_2}$). Thus, a smaller or equal number of edges will be mapped to ε in the case of $MAP^k(L_2,P)$ not decreasing the value for the precision, i.e., $MAP^k(L_2,P) \geq MAP^k(L_1,P)$.

In Axiom-3 we showed that there exists a specific value of k, namely k^*, for which $MAP^{k^*}(L_x,P_x)$ satisfies Axiom-3 and we identified such value being $k^* = |\widehat{\tau}|$, where $\widehat{\tau}$ can be any trace of the set difference $\Gamma_\Omega \setminus \mathscr{B}_{P_x}$. In the following, we show how to identify the minimum value of k^* such that all the 5-Axioms are satified. To identify the lowest value of k^*, we have to consider the traces $\widehat{\tau} \in \Gamma_\Omega$ such that does not exists a $\tau \in \mathscr{B}_{P_x}$ where $\widehat{\tau} \subseteq \tau$. If a trace $\widehat{\tau} \in \Gamma_\Omega$ that is not a sub-trace of any other trace of the process model behavior ($\mathscr{B}_{P_x}$) is found, by setting $k^* = |\widehat{\tau}|$ would mean that in the M^{k^*}-abstraction of Γ_Ω there will be a state $\widehat{s} = \widehat{\tau}$ and an edge $(-,\widehat{\tau})$ that are not captured by the M^{k^*}-abstraction of $\mathscr{B}_{P_x}$. This difference will allow us to distinguish the process P_x from the flower model (i.e. the model having a behavior equal to Γ_Ω), satisfying in this way the Axiom-3. At this point, considering the set of the lengths of all the subtraces not contained in any trace of $\mathscr{B}_{P_x}$, $Z = \{|\widehat{\tau}| \ : \ \widehat{\tau} \in \Gamma_\Omega \ \wedge \ \nexists\, \tau \in \mathscr{B}_{P_x} \mid \widehat{\tau} \subseteq \tau\}$, we can set the lower-bound of $k^* \geq min(Z)$.

Note that the value of k^* is equal to 2 for any process model with at least one activity that cannot be executed twice in a row. If we have an activity $\widehat{t}$ that cannot be executed twice in a row, it means that $\left|\langle \widehat{t},\widehat{t} \rangle\right| \in Z$ and thus we can set $k^* = 2$. In practice, $k^* = 2$ satisfies all the 5-Axioms in real-life cases, since it is very common to find process models that have the above topological characteristic.

5 Evaluation

In this section, we report on a two-pronged evaluation we performed to assess the following two objectives: i) comparing our measures to the state-of-the-art fitness and precision measures; and ii) analysing the role of the order k.

To do so, we implemented the Markovian Abstraction-based Fitness and Precision (MAF^k and MAP^k) in a standalone open-source tool[10] and used it to carry out a qualitative evaluation on synthetic data and a quantitative evaluation on real-life data.[11]

9. Formally, $\exists \widehat{\tau} \in \mathscr{B}_{P_2} \setminus \mathscr{B}_{P_1}$, s.t. for $k^* = |\widehat{\tau}| \implies \exists (-,\widehat{\tau}) \in E_{P_2} \setminus E_{P_1}$.

10. Available at http://apromore.org/platform/tools

11. The public data used in the experiments can be found at http://doi.org/10.6084/M9.FIGSHARE.7397006.V1

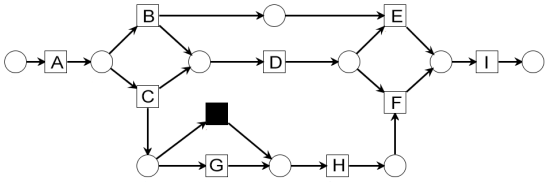


Fig. 6: Original model [16].

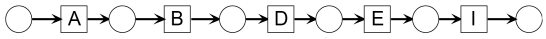


Fig. 7: Single trace model [16].

All the experiments were executed on an Intel Core i5-6200U @2.30GHz with 16GB RAM running Windows 10 Pro (64-bit) and JVM 8 with 12GB RAM (8GB Stack and 4GB Heap). For each measurement we set a timeout of two hours.

5.1 Qualitative evaluation dataset

In a previous study, van Dongen et al. [16] showed that their anti-alignment precision was able to generate more intuitive rankings of model-log pairs than existing state-of-the-art precision measures using a synthetic dataset of model-log pairs. Table 1 (Section 2) shows the synthetic event log used in [16], while Figure 6 shows the "original model" that was used in that paper to derive eight process model variants. The set of models includes a *single trace* model capturing the most frequent trace, Fig. 7; a model incorporating all *separate traces*, Fig. 8; a *flower model* of all activities in the log, Fig. 9; a model with activities G and H in parallel (*Opt. G || Opt. H*, see Fig. 10); one with G and H in self-loop (*↻G, ↻H*, Fig. 11); a model with D in self-loop (*↻D*, Fig. 12); a model with all activities in parallel (*All parallel*, Fig. 13); and a model where all activities are in round robin (*Round robin*, Fig. 14).

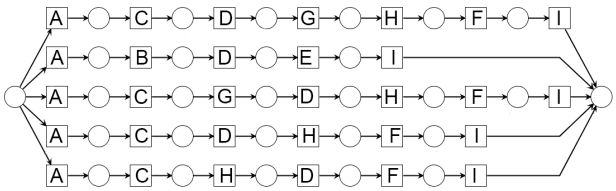


Fig. 8: Separate traces model [16].

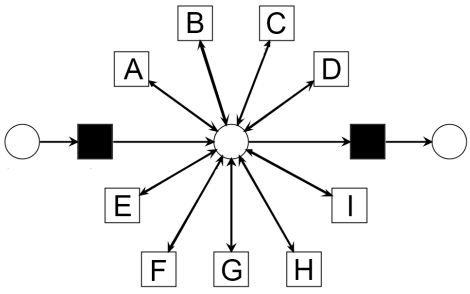


Fig. 9: Flower model [16].

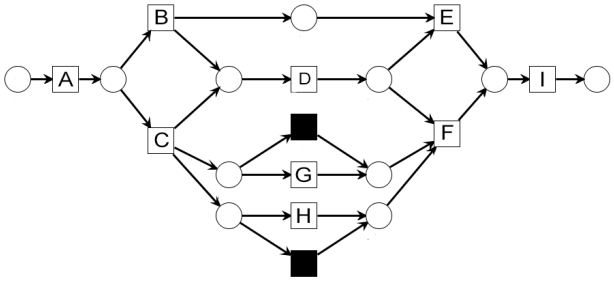


Fig. 10: Opt. G || Opt. H model [16].

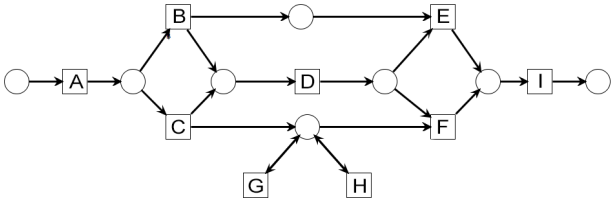


Fig. 11: ↻G, ↻H model [16].

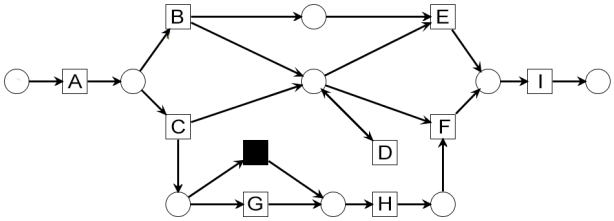


Fig. 12: ↻D model [16].

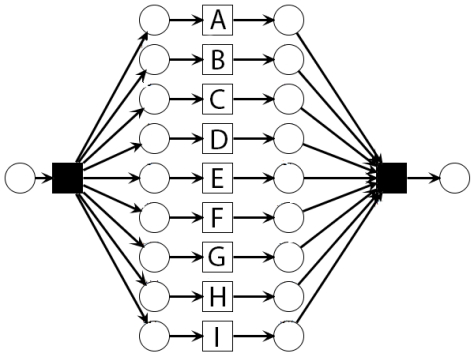


Fig. 13: All parallel model [16].

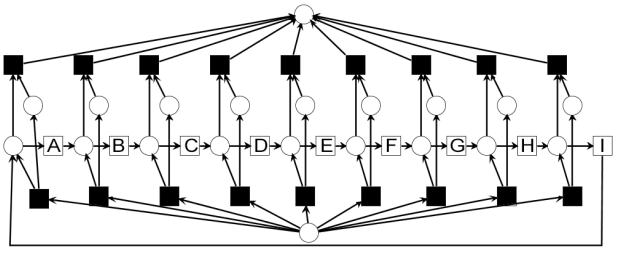


Fig. 14: Round robin model [16].

To evaluate the MAF^k fitness measures, we generated from each of the above models, an event log containing the whole behavior of the process model. In doing so, we capped the number of times each cycle in the model was executed to two iterations each. We held out the *Flower* model because we were unable to generate its corresponding event log, given that this model has almost one million unique traces if each cycle is executed at most twice. This led to a total of eight model-log pairs.

Below, we evaluate the proposed fitness and precision measures using the above model-log pairs.

5.2 Qualitative evaluation of MAF^k

Using the original model (Figure 6), we measured the fitness of the eight logs synthetized from the process model variants. We compared our fitness measure MAF^k to the fitness measures discussed in Section 2, namely: token-based fitness (*Token*), alignment-based fitness (*Alignment*), and PCC fitness, with projections size equal to 10 (PCC_{10}). We chose a size of 10 for PCC because this was the highest value for which we were able to obtain more than 50% of measurements.

Table 4 gives the numerical results of the fitness measurements. We note that some of the fitness values returned by the *Token* measure are counter-intuitive. For example, for the log generated from the original model, this measure scores a value different than 1 despite this model fully fits this log. As such, this result brakes Proposition-7 of fitness. Other counter-intuitive values are returned for the logs generated from the *separate traces* and the *all parallel* models, respectively 0.951 and 0.556. The former should also be 1 as per Proposition-7, whilst the latter, given that none of the log traces are contained in the original model behavior, should return a value as close as possible to 0.

Alignment and PCC_{10} also return high values of fitness for the *all parallel* log, respectively 0.464 and 1. These results are clearly counter-intuitive if we consider that the *all parallel* log contains more than 350 thousand different traces and none of them is contained in the original model behavior. A similar reasoning can be done for the fitness values of *Token* and *Alignment* on the log generated from the *round robin* model. This latter log contains no traces of the original process behavior, yet both measurements are very high (respectively 0.655 and 0.501).

As for our fitness measure MAF^k, we note that the numeric values for all the logs generated from the acyclic models stabilize at k equal to 4, whilst the values for the logs generated from the three cyclic models (1: $\circlearrowleft G$, $\circlearrowleft H$, 2: $\circlearrowleft D$ and 3: *Round robin*) tend to 0 as we increase k. This is expected as the higher the k, the larger the behavior captured in the M^k-abstractions, and consequently the more precise the measurement. In line with Proposition-7, MAF^k returns a fitness value of 1 for all the logs fully fitting the original model (first three rows of Table 4).

In contrast with the state of the art, our measure is able to identify the large difference between the behavior of the original model and that recorded in the *all parallel* log, as it returns a value close to 0 already for k equal to 2, similarly to the *round robin* log. The only measure that returns a value close to 0 for the *round robin* log is PCC_{10}, besides our measure.

Since each fitness measure assesses the fitness of a log over a model in a different way, it is useful to analyse the ranking yielded by each measure for the set of logs in question. This is shown in Table 5. *Alignment* and MAF^k agree on the most intuitive ranking (having exactly the same ranking with k between 4 and 6). They assign the highest rank to the fully fitting logs, followed by the partially fitting logs with acyclic behavior (i.e. *Opt. G* || *Opt. H*), then by the logs containing cyclic behavior ($\circlearrowleft G$, $\circlearrowleft H$, and $\circlearrowleft D$) and finally by the two completely unfitting logs (*all parallel* and *round robin*). These latter two logs are ranked equally only by MAF^7, which is able to identify that none of the them contains any behavior of the original model.

5.3 Qualitative evaluation of MAP^k

Using the original log (as in Table 1), we measured the precision of the nine model variants (as shown in Fig. 6 to 14), and compared our precision measure MAP^k with the state of the art discussed in Section 2, which includes the same precision measures evaluated by van Dongen et al. [16]). These are: traces set difference precision (*SD*), alignment-based ETC precision (ETC_a), negative events precision (*NE*), projected conformance checking with its projections size equal to 2 (PCC_2), and anti-alignment precision (*AA*). We chose size 2 for PCC since this is the highest value for which we were able to obtain more than 50% of the measurements. We left out the advanced behavioral appropriateness (*ABA*) as it is not defined for some of the models in this dataset. We limited the order k to 7, because it is the length of the longest trace in the log. Setting an order greater than 7 would only (further) penalise the cyclic behavior of the models.

Table 6 reports the results of our qualitative evaluation.[12] To discuss these results, we use two precision measures as a reference, as these have been advocated as possible ground truths of precision, though none of them satisfies the axioms in [4]. The first one is *AA*. This measure has been shown [16] to be intuitively more accurate than other precision measures. The second one is *SD*, as it closely operationalizes the definition of precision by capturing the exact percentage of model behavior that cannot be found in the log. As discussed in Section 2 though, this measure can only be computed for acyclic models, and uses by design a value of zero for cyclic models.

From the results in Table 6, we can observe that MAP^1 does not penalise enough the extra behavior of some models, such as the *original* model, which cannot be distinguished from the *single trace* and the *separate traces* models (all have a precision of 1). Also, the values of MAP^1 are far away from those of both *AA* and *SD* (with the exception of the simplest models, i.e. *single trace* and *separate traces*). As we increase k, MAP^k tends to get closer to *AA* and to *SD*, barring a few exceptions. In particular, the more the cyclic behavior allowed by a model, the quicker MAP^k tends to zero. In this respect, let us consider the cyclic models in our datasets: i) the *flower* model, ii) the $\circlearrowleft G$, $\circlearrowleft H$ model (Fig. 11), iii) the $\circlearrowleft D$ model (Fig. 12), and iv) the *round robin* (Fig. 14). The value of our precision measure immediately drops to zero in the *flower* model ($k = 3$) because this latter model allows the greatest amount of cyclic behavior, due to all the possible combinations of activities being permitted. This is consistent with both *SD* and *AA*.

On the contrary, MAP^k tends to zero slower in the *round robin* model because this model is very strict on the order in which activities can be executed, despite having infinite behavior. In fact, it only allows the sequence $\langle A,B,C,D,F,G,H,I\rangle$ to be executed, with the starting activity and the number of repetitions being variable. This is taken into account by our measure, since even with $k = 7$ we do not reach a value of zero for this model, as

12. Some values differ from those in [16] as we used each measure's latest implementation.

Log	*Token*	*Alignment*	PCC_{10}	MAP^1	MAF^2	MAF^3	MAF^4	MAF^5	MAF^6	MAF^7
Original model	0.960	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Single trace	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Separate traces	0.951	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Opt. G || Opt. H	0.920	0.956	0.500	0.889	0.679	0.563	0.500	0.500	0.500	0.500
↻G, ↻H	0.880	0.918	0.231	0.889	0.633	0.409	0.259	0.167	0.143	0.200
↻D	0.781	0.873	0.171	0.889	0.633	0.419	0.250	0.138	0.094	0.109
All parallel	0.556	0.464	1.000	0.222	0.038	0.006	0.001	< 0.001	< 0.001	0.000
Round robin	0.655	0.501	0.000	0.444	0.000	0.000	0.000	0.000	0.000	0.000

TABLE 4: Values of the fitness measures over the synthetic dataset.

Log	*Token*	*Alignment*	PCC_{10}	MAP^1	MAF^2	MAF^3	MAF^4	MAF^5	MAF^6	MAF^7
Original model	7	6	5	6	6	6	6	6	6	6
Single trace	8	6	5	6	6	6	6	6	6	6
Separate traces	6	6	5	6	6	6	6	6	6	6
Opt. G || Opt. H	5	5	4	2	5	5	5	5	5	5
↻G, ↻H	4	4	3	2	3	3	4	4	4	4
↻D	3	3	2	2	3	4	3	3	3	3
All parallel	1	2	5	1	2	2	2	2	2	1
Round robin	2	1	1	5	1	1	1	1	1	1

TABLE 5: Model ranking induced by the fitness measures over the synthetic dataset.

Process Variant	*Model Traces (#)*	*SD*	ETC_a	*NE*	PCC_2	*AA*	MAP^1	MAP^2	MAP^3	MAP^4	MAP^5	MAP^6	MAP^7
Original model	6	0.833	0.900	0.995	1.000	0.871	1.000	0.895	0.833	0.786	0.778	0.833	0.833
Single trace	1	1.000	1.000	0.893	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Separate traces	5	1.000	1.000	0.985	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Flower model	986,410	0.000	0.153	0.117	0.509	0.000	0.176	0.021	0.002	t/o	t/o	t/o	t/o
Opt. G || Opt. H	12	0.417	0.682	0.950	0.991	0.800	0.889	0.607	0.469	0.393	0.389	0.417	0.417
↻G, ↻H	362	0.000	0.719	0.874	0.889	0.588	0.800	0.370	0.134	0.041	0.011	0.003	0.001
↻D	118	0.000	0.738	0.720	0.937	0.523	0.889	0.515	0.273	0.128	0.055	0.028	0.021
All parallel	362,880	0.000	0.289	0.158	0.591	0.033	0.222	0.034	0.005	0.005	t/o	t/o	t/o
Round robin	27	0.000	0.579	0.194	1.000	0.000	0.600	0.467	0.425	0.340	0.267	0.200	0.213

TABLE 6: Values of the precision measures over the synthetic dataset.

Process Variant	*SD*	ETC_a	*NE*	PCC_2	*AA*	MAP^1	MAP^2	MAP^3	MAP^4	MAP^5	MAP^6	MAP^7
Original model	7	7	9	6	7	7	7	7	7	7	7	7
Single trace	8	8	6	6	8	7	8	8	8	8	8	8
Separate traces	8	8	8	6	8	7	8	8	8	8	8	8
Flower model	1	1	1	1	1	1	1	1	1	1	1	1
Opt. G || Opt. H	6	3	7	5	6	5	6	6	6	6	6	6
↻G, ↻H	1	5	5	3	5	4	3	3	3	3	3	3
↻D	1	6	4	4	4	5	5	4	4	4	4	4
All parallel	1	2	2	2	3	2	2	2	2	2	2	2
Round robin	1	4	3	6	1	3	4	5	5	5	5	5

TABLE 7: Model rankings induced by the precision measures over the synthetic dataset.

opposed to *SD* and *AA*. This allows us to discriminate the *round robin* model from other models with very large behavior such as the *flower* model. This is not possible with *SD* and *AA*, because both models have a precision of zero in these two measures.

As for the other two cyclic models in our dataset, MAP^k tends to zero with speeds between those of the *flower* model and the *round robin* model, with the ↻*G,* ↻*H* model dropping faster than the ↻*D*, due to the former allowing more cyclic behavior than the latter. Similar considerations as above apply to these two models. Even for $k = 7$, their precision does not reach zero, allowing us to distinguish the precision of these two models. While in *SD* the precision of these two models is set to zero by design, in *AA* these two models have a precision well above zero, with the ↻*G,* ↻*H* model having a higher precision than the ↻*D* model (0.588 vs. 0.523). This is counter-intuitive, since the former model allows more behavior not permitted by the log (in terms of number of different traces) than the latter model does. In addition, *AA* penalizes more the *round robin* model, despite this model having less behavior than the two models with self-loop activities.

Altogether, these precision results show that the higher the k, the more the behavioral differences our measure can catch and penalise. Furthermore, in line with the time complexity analysis of MAP^k, we can see that when k increases, the execution time increases faster for those models allowing a huge amount of behavior, timing out for $k > 3$ and $k > 4$ on the *flower* and *all parallel* models (respectively).

In terms of ranking (see Table 7), our measure is the most consistent with the ranking of the models yielded by both *SD* (for acyclic models) and *AA* (for all models), than all other measures. As discussed above, the only differences are in the swapping of the order of the two models with self loops, and in the order of the *round robin* model. Note that given that both the round robin and the flower model have a value of zero in *AA*, the next model in the ranking (*all parallel*) is assigned a rank of 3 instead of 2 in MAP^k. This is just the way the ranking is computed and is not really indicative of a ranking inconsistency between the two measures. Another observation is that the ranking yielded by our precision measure remains the same for $k > 2$. This indicates that as we increase k, while the extent of behavioral differences we can identify and penalize increases, this is not achieved at the price of changing the ranking of the models.

5.4 Quantitative evaluation dataset

In our second evaluation, we used two datasets for a total of 20 logs. The first dataset is the collection of real-life logs publicly available from the 4TU Centre for Research Data.[13] Out of this collection, we retained twelve logs related to business processes, as opposed to e.g. software development processes. These include the *BPI Challenge* (BPIC) logs (2012-17), the *Road Traffic Fines Management Process* (RTFMP) log, and the *SEPSIS* log. These logs record executions of business processes from a variety of domains, e.g. healthcare, finance, government and IT service management. In seven logs (BPIC14, the BPIC15 collection, and BPIC17), we applied the filtering technique proposed in [26] to remove infrequent behavior. This was required otherwise most of the fitness and precision measures would throw an out-of-memory exception. The second dataset is composed of eight proprietary logs sourced from several companies in the education, insurance, IT service management and IP management domains.

Table 8 reports the characteristics of both datasets. For each of these 20 logs, we automatically discovered three process models using three state-of-the-art automated process discovery methods [2]: Split Miner [18] (SM), Inductive Miner [27] (IM), and Structured Heuristics Miner [28] (SHM), totalling 60 log-model pairs that we used for our quantitative evaluation.

5.5 Quantitative evaluation of MAF^k

We measured our MAF^k on each of the 60 log-model pairs, varying k in the range 2–5. For our comparison, we retained the *Alignment* fitness and PCC_2 fitness. For PCC we used a projections size of 2 as this was the highest value for which we were able to obtain more than 50% of the measurements. We held out the *Token* fitness since it does not scale to real-life models.

Table 9 and 10 show respectively the numerical values of each measurement and the fitness ranking of the models discovered by SM, IM and SHM for each log. [14] As a baseline for our comparison we used *Alignment*, since it is to date the most-scalable and widely-accepted fitness measure for automated process discovery in real-life settings [2].

Also in this evaluation, we can observe that the values of MAF^k reduce as we increase the order k. This is by design, since the higher the k the more are the behavioral details captured by the measure, and the more are the mismatches identified by our measure. Furthermore, we recall that our MAF^k is very strict since it uses a boolean function to penalise behavioral mismatches.

In Table 9, we can see that for the models discovered by SM from the logs PRT1 and PRT6, the values of MAF^k quickly drop when moving from $k = 3$ to $k = 4$. The results of *Alignment* and MAF^2 for the model of SM discovered from the log PRT9, are also interesting: *Alignment* returns a value of 0.915, suggesting an almost fully-fitting model, while our MAF^k returns a low value of fitness (0.197) already at $k = 2$. This can be linked to the fact that *Alignment* is sometimes too accommodative leading to counter-intuitive results, as shown in the qualitative evaluation.

If we look at the ranking yielded by the measures, MAF^k agrees with *Alignment* more than 30% of the times at $k = 3$ and $k = 5$, and more than 25% of the times at $k = 2$ and $k = 4$. Instead, PCC_2 agrees with *Alignment* only 10% of the times. Increasing k for our MAP^k does not alter frequently the ranking yielded: indeed 50% of the times the ranking was stable already at $k = 2$ and 75% at $k = 3$.

Finally, Tables 11 and 12 report the time performance of MAF^k, *Alignment* and PCC_2.[15] We separated the results by public and proprietary logs to allow the reproducibility of the experiments for the set of public logs. We note that PCC_2 underperforms *Alignment*, despite this measure was designed to be more efficient. Instead, MAF^k is significantly faster than *Alignment* in the majority of the log-model pairs, especially when the input model has a reasonable state space as that produced by SM and SHM (i.e. when flower-like constructs are absent). On the other hand, by increasing k the performance of MAF^k reduces sharply for flower-like models, as those produced by IM.

For the sake of measuring the time performance of *Alignment* we also considered the alternative implementation reported in [29]. However, this latter implementation consistently performed worse than the standard implementation for the majority of log-model pairs used in our evaluation.

5.6 Quantitative evaluation of MAP^k

Similarly, we assessed our MAP^k against each real-life log-model pair while varying the order k in the range 2–5. Unfortunately, we were not able to use any of the reference measures used in the qualitative evaluation, because *SD* does not work for cyclic models (all models discovered by IM were cyclic) while *AA* does not scale to real-life models [16]. Furthermore, we excluded *NE* precision, since we were not able to perform the measurements for more than 50% of the log-model pairs within the two-hour timeout. Thus, we resorted to ETC_a and PCC_2 as two baselines.

Table 13 shows the results of the quantitative evaluation. In line with the former evaluation, the value of MAP^k decreases when k increases. However, being the behavior of the real-life models more complex than that of the artificial models, for some logs (e.g. the BPIC15 logs), it was not possible to compute MAP^4 and MAP^5 for the models discovered by IM. This was due to scalability issues,[16] as the models discovered by IM exhibit flower-like behavior (with more than 50 distinct activities per flower construct), which is already identified by MAP^2 and MAP^3, whose values are very low for these models. We recall that by design, for small values of k, MAP^k compares small chunks of the model behavior to small chunks of the log behavior. Thus, low values of MAP^2 and MAP^3 already indicate poorly-precise models.

ETC_a and MAP^5 agree on the precision ranking 50% of the times (cf. Table 14), whilst ETC_a and PCC_2 agree on the 30% of the rankings. Also, in-line with the former evaluation, ETC_a is tolerant to infinite model behavior, regardless of the type of such behavior. An example that illustrates this flaw is the SEPSIS log. The models discovered by IM and SM are shown in Fig. 15 and 16. We observe that more than the 80% of the activities in the IM model are skippable and over 60% of them are inside a long loop, resembling a flower construct with some constraints, e.g. the first activity is always the same. Instead, the model discovered by SM, even if cyclic, does not allow many variants of behavior. Consequently, for the IM model, the value of MAP^k drastically drops when increasing k from 2 to 3, whilst it remains 1 for the SM model. In contrast, ETC_a gives a precision of 0.445 for IM, which is counter-intuitive considering its flower-like structure.

13. https://data.4tu.nl/repository/collection:event_logs_real

14. A "–" symbol is used to report a failed measurement due to either a time-out or an exception.

15. Highlighted in bold the best scores, underlined the second best scores.

16. The allocated RAM was not enough to complete the measurements.

Log		BPIC12	$BPIC13_{cp}$	$BPIC13_{inc}$	$BPIC14_f$	$BPIC15_{1f}$	$BPIC15_{2f}$	$BPIC15_{3f}$	$BPIC15_{4f}$	$BPIC15_{5f}$	$BPIC17_f$
Total Traces		13,087	1,487	7,554	41,353	902	681	1,369	860	975	21,861
Dist. Traces		33.4	12.3	20	36.1	32.7	61.7	60.3	52.4	45.7	40.1
Total Events		262,200	6,660	65,533	369,485	21,656	24,678	43,786	29,403	30,030	714,198
Dist. Events		36	7	13	9	70	82	62	65	74	41
Tr. length	**(min)**	3	1	1	3	5	4	4	5	4	11
	(avg)	20	4	9	9	24	36	32	34	31	33
	(max)	175	35	123	167	50	63	54	54	61	113

Log		RTFMP	SEPSIS	PRT1	PRT2	PRT3	PRT4	PRT6	PRT7	PRT9	PRT10
Total Traces		150,370	1,050	12,720	1,182	1,600	20,000	744	2,000	787,657	43,514
Dist. Traces		0.2	80.6	8.1	97.5	19.9	29.7	22.4	6.4	0.01	0.01
Total Events		561,470	15,214	75,353	46,282	13,720	166,282	6,011	16,353	1,808,706	78,864
Dist. Events		11	16	9	9	15	11	9	13	8	19
Tr. length	**(min)**	2	3	2	12	6	6	7	8	1	1
	(avg)	4	14	5	39	8	8	8	8	2	1
	(max)	2	185	64	276	9	36	21	11	58	15

TABLE 8: Descriptive statistics of the real-life logs (public and proprietary).

Log	BPIC12			$BPIC13_{cp}$			$BPIC13_{inc}$			$BPIC14_f$			$BPIC15_{1f}$		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
Alignment	0.970	0.990	-	0.989	0.820	0.940	0.977	0.920	0.910	0.767	0.890	-	0.900	0.970	-
PCC_2	0.043	0.043	0.687	0.480	0.308	0.696	0.000	0.035	0.000	0.407	0.106	0.261	0.598	0.266	0.000
MAF^2	0.400	0.840	0.896	0.636	0.364	0.636	0.667	0.667	0.583	0.441	0.971	1.000	0.758	0.960	0.992
MAF^3	0.308	0.834	0.715	0.565	0.217	0.478	0.548	0.613	0.419	0.338	0.796	0.972	0.614	0.949	0.990
MAF^4	0.267	0.834	0.416	0.568	0.091	0.364	0.429	0.586	0.271	0.212	0.626	0.903	0.510	0.934	0.983
MAF^5	0.257	0.370	0.197	0.537	0.061	0.256	0.356	0.557	0.188	0.154	0.465	0.788	0.434	0.535	0.901

Log	$BPIC15_{2f}$			$BPIC15_{3f}$			$BPIC15_{4f}$			$BPIC15_{5f}$			$BPIC17_f$		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
Alignment	0.773	0.948	0.981	0.780	0.950	0.950	0.731	0.955	0.991	0.791	0.937	1.000	0.962	0.979	0.954
PCC_2	0.015	0.087	0.013	0.523	0.024	0.016	0.000	0.000	0.007	0.005	0.301	0.016	0.244	0.466	0.013
MAF^2	0.538	0.853	0.919	0.413	0.894	0.894	0.543	0.921	0.974	0.591	0.922	0.987	0.935	0.935	0.903
MAF^3	0.335	0.793	0.885	0.224	0.804	0.811	0.329	0.842	0.959	0.371	0.874	0.982	0.923	0.885	0.769
MAF^4	0.226	0.688	0.855	0.147	0.643	0.734	0.230	0.558	0.938	0.263	0.347	0.981	0.904	0.843	0.675
MAF^5	0.168	0.307	0.825	0.105	0.290	0.678	0.169	0.372	0.916	0.191	0.135	0.982	0.909	0.818	0.606

Log	RTFMP			SEPSIS			PRT1			PRT2			PRT3		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
Alignment	1.000	0.980	0.980	0.763	0.991	0.920	0.977	0.902	0.883	0.811	-	-	0.824	0.975	1.000
PCC_2	0.000	0.062	0.757	0.000	0.129	0.296	0.276	0.225	0.883	0.000	0.000	0.802	0.264	0.420	0.056
MAF^2	0.257	0.843	0.957	0.226	0.930	0.757	0.459	0.730	0.811	0.291	0.987	0.747	0.581	0.977	0.977
MAF^3	0.187	0.743	0.930	0.155	0.953	0.678	0.318	0.418	0.691	0.111	0.880	0.637	0.291	0.882	0.976
MAF^4	0.151	0.655	0.889	0.164	0.962	0.654	0.258	0.154	0.585	0.071	0.803	0.640	0.185	0.807	0.982
MAF^5	0.125	0.576	0.882	0.187	0.647	0.631	0.223	0.035	0.502	0.077	0.354	0.684	0.131	0.778	0.985

Log	PRT4			PRT6			PRT7			PRT9			PRT10		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
Alignment	0.833	0.927	1.000	0.943	0.989	1.000	0.910	1.000	1.000	0.915	0.900	0.964	0.969	0.964	-
PCC_2	0.082	0.309	0.318	0.632	0.320	0.503	0.426	0.385	0.533	0.780	0.011	0.990	0.327	0.623	-
MAF^2	0.541	0.892	1.000	0.545	1.000	1.000	0.605	1.000	1.000	0.197	0.818	0.833	0.471	0.966	-
MAF^3	0.284	0.622	1.000	0.220	0.898	1.000	0.243	1.000	1.000	0.106	0.596	0.652	0.374	0.925	-
MAF^4	0.154	0.318	1.000	0.082	0.776	1.000	0.161	1.000	1.000	0.074	0.312	0.476	0.343	0.934	-
MAF^5	0.084	0.118	1.000	0.044	0.716	1.000	0.141	1.000	1.000	0.055	0.127	0.349	0.343	0.852	-

TABLE 9: Comparison of fitness measures over the 20 real-life logs.

As discussed in Section 3, $k = 2$ is sufficient to satisfy all the 5-Axioms in practice. However, as we also observe from the results of this second experiment for precision, higher values of k lead to finer results for MAP^k. In fact, the notable drops of value from $k = 2$ to $k = 3$ (e.g. in SEPSIS, BPIC17$_f$ and PRT9), confirm that the 5-Axioms are a necessary but not sufficient condition for a reliable precision measure [4].

Finally, Tables 15 and 16 report the time performance of MAP^k, PCC_2 and ETC_a.[15] Similarly to the quantitative evaluation of fitness, we separated the results by public and proprietary logs to allow the reproducibility of the experiments for the public logs. The results are consistent with those obtained for fitness: PCC_2 is the slowest measure while MAP^k is overall the fastest measure, especially for those models that do not exhibit flower-like structures (SM and SHM), while ETC_a outperforms our measure when the state space of the model is very large (in the case of models discovered by IM, for high values of k).

5.7 The role of k

As expected, the results of both qualitative and quantitative evaluation show that the order k directly influences how our Markovian fitness and precision measures penalise behavioral mismatches between log and model, and ultimately how they rank process models by accuracy. Furthermore, k plays an important role on the time performance of our proposed measures. Indeed, while for low k's MAF^k and MAP^k scale well to large real-life logs, outperforming the state of the art, for high k's, the time performance in some cases deteriorates even dramatically. However, we showed that in practice a low value of k does approximate well the fitness and precision ranking that would be obtained with hight k values, over a set of automatically discovered process models.

We remind that $k+1$ defines the size of the subtraces in the log and in the process model that we compare with each other to detect mismatches. Bearing this in mind, we advise that whenever possible k should be set to the length of the longest trace recorded

Log	BPIC12			$BPIC13_{cp}$			$BPIC13_{inc}$			$BPIC14_{f}$			$BPIC15_{1f}$		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
Alignment	2	3	-	3	1	2	3	2	1	2	3	-	2	3	-
PCC_2	1	1	2	2	1	3	1	2	1	3	1	2	3	2	1
MAF^2	1	2	3	3	1	3	3	3	2	1	2	3	1	2	3
MAF^3	1	3	2	3	1	2	2	3	1	1	2	3	1	2	3
MAF^4	1	3	2	2	1	2	2	3	1	1	2	3	1	2	3
MAF^5	2	3	1	3	1	2	2	3	1	1	2	3	1	2	3

Log	$BPIC15_{2f}$			$BPIC15_{3f}$			$BPIC15_{4f}$			$BPIC15_{5f}$			$BPIC17_{f}$		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
Alignment	1	2	3	2	3	3	1	2	3	1	2	3	2	3	1
PCC_2	2	3	1	3	2	1	1	1	2	1	3	2	2	3	1
MAF^2	1	2	3	2	3	3	1	2	3	1	2	3	3	3	1
MAF^3	1	2	3	1	2	3	1	2	3	1	2	3	3	2	1
MAF^4	1	2	3	1	2	3	1	2	3	1	2	3	3	2	1
MAF^5	1	2	3	1	2	3	1	2	3	2	1	3	3	2	1

Log	RTFMP			SEPSIS			PRT1			PRT2			PRT3		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
Alignment	3	2	2	1	2	3	3	2	1	-	-	-	1	2	3
PCC_2	1	2	3	1	2	3	2	1	3	1	1	2	2	3	1
MAF^2	1	2	3	1	3	2	1	2	3	1	3	2	1	3	3
MAF^3	1	2	3	1	3	2	1	2	3	1	3	2	1	2	3
MAF^4	1	2	3	1	3	2	2	1	3	1	3	2	1	2	3
MAF^5	1	2	3	1	3	2	2	1	3	1	2	3	1	2	3

Log	PRT4			PRT6			PRT7			PRT9			PRT10		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
Alignment	1	2	3	1	2	3	2	3	3	2	1	3	3	2	-
PCC_2	1	2	3	3	1	2	2	1	3	2	1	3	1	2	-
MAF^2	1	2	3	2	3	3	1	3	3	1	2	3	2	3	-
MAF^3	1	2	3	1	2	3	1	3	3	1	2	3	2	3	-
MAF^4	1	2	3	1	2	3	1	3	3	1	2	3	2	3	-
MAF^5	1	2	3	1	2	3	1	3	3	1	2	3	2	3	-

TABLE 10: Models ranking yielded by fitness measures over the 20 real-life logs.

	Split Miner				Inductive Miner				Struct. Heuristics Miner			
Precision	**avg**	**max**	**min**	**total**	**avg**	**max**	**min**	**total**	**avg**	**max**	**min**	**total**
Alignment	30.1	217.8	0.5	361.0	33.7	155.2	0.5	404.8	37.2	174.4	0.7	335.0
PCC_2	77.1	685.1	0.1	925.4	138.9	1344.9	0.1	1667.1	152.1	1378.8	0.1	1825.1
MAP^2	0.2	2.0	**>0.1**	2.7	**0.3**	**1.1**	**>0.1**	**4.2**	**0.9**	**4.6**	**>0.1**	**10.7**
MAP^3	**0.1**	**0.3**	**>0.1**	**0.7**	108.2	890.9	**>0.1**	1298.1	4.2	27.9	**>0.1**	50.4
MAP^4	**0.1**	**0.3**	**>0.1**	0.9	1386.2	4383.9	**>0.1**	16634.9	14.1	84.7	**>0.1**	169.4
MAP^5	**0.1**	0.6	**>0.1**	1.3	1612.4	5365.6	**>0.1**	19348.6	39.1	214.4	**>0.1**	469.2

TABLE 11: Time performance (in seconds) of fitness measures using the twelve public logs.

	Split Miner				Inductive Miner				Struct. Heuristics Miner			
Precision	**avg**	**max**	**min**	**total**	**avg**	**max**	**min**	**total**	**avg**	**max**	**min**	**total**
Alignment	12.4	40.6	0.4	98.9	5.9	29.9	0.4	41.6	144.2	813.2	0.5	865.1
PCC_2	154.2	1226.2	**>0.1**	1233.3	257.0	2048.9	**>0.1**	2056.3	287.3	1960.2	**>0.1**	2010.8
MAP^2	0.2	1.0	**>0.1**	1.3	**2.1**	**15.8**	**>0.1**	**17.2**	**0.7**	**2.1**	**>0.1**	**4.8**
MAP^3	**0.1**	**0.3**	**>0.1**	**0.6**	8.7	66.2	**>0.1**	69.4	1.5	6.0	**>0.1**	10.4
MAP^4	**0.1**	0.4	**>0.1**	**0.6**	80.7	597.0	**>0.1**	645.2	12.5	52.6	**>0.1**	87.8
MAP^5	**0.1**	0.5	**>0.1**	0.7	352.8	2792.9	**>0.1**	2822.6	21.8	90.5	**>0.1**	152.7

TABLE 12: Time performance (in seconds) of fitness measures using the eight proprietary logs.

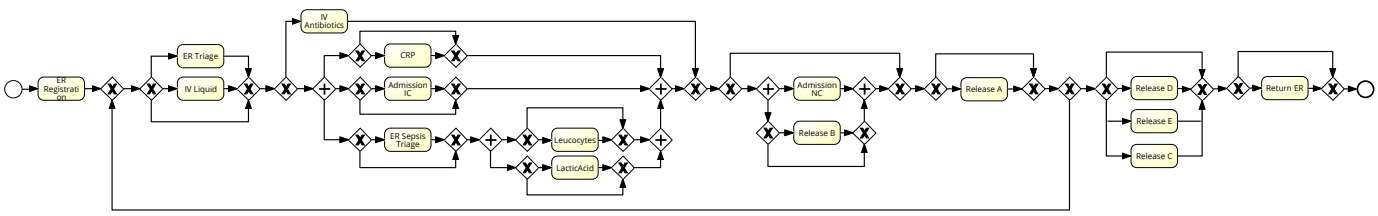


Fig. 15: Model discovered by IM from the SEPSIS log.

Log	BPIC12			$BPIC13_{cp}$			$BPIC13_{inc}$			$BPIC14_{f}$			$BPIC15_{1f}$		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
ETC_a	0.762	0.502	-	0.974	1.000	0.992	0.979	0.558	0.978	0.673	0.646	-	0.880	0.566	-
PCC_2	1.000	0.902	0.919	0.935	0.347	0.703	1.000	0.588	0.938	0.990	0.744	0.991	0.995	0.885	1.000
MAP^2	1.000	0.399	0.316	1.000	1.000	0.708	1.000	1.000	1.000	1.000	0.786	0.500	1.000	0.144	0.054
MAP^3	0.826	0.083	0.073	0.952	0.944	0.682	0.965	0.955	0.976	1.000	0.701	0.274	0.969	0.019	0.016
MAP^4	0.640	0.013	0.027	0.931	0.917	0.638	0.919	0.898	0.911	1.000	0.656	0.164	0.932	-	-
MAP^5	0.362	-	0.015	0.907	0.925	0.626	0.893	0.828	0.883	1.000	0.631	0.162	0.896	-	-

Log	$BPIC15_{2f}$			$BPIC15_{3f}$			$BPIC15_{4f}$			$BPIC15_{5f}$			$BPIC17_{f}$		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
ETC_a	0.901	0.556	0.594	0.939	0.554	0.671	0.910	0.585	0.642	0.943	0.179	0.687	0.846	0.699	0.620
PCC_2	1.000	0.912	1.000	0.979	0.974	1.000	1.000	1.000	1.000	0.998	0.913	1.000	0.939	0.600	0.911
MAP^2	1.000	0.160	0.995	1.000	0.196	1.000	1.000	0.121	1.000	1.000	0.092	1.000	1.000	0.843	0.347
MAP^3	0.972	0.022	0.835	0.986	0.032	0.787	0.963	0.015	0.789	0.964	0.006	0.817	0.705	0.566	0.151
MAP^4	0.930	-	0.588	0.967	-	0.502	0.920	-	0.531	0.917	-	0.577	0.482	0.370	0.069
MAP^5	0.878	-	0.356	0.939	-	0.275	0.877	-	0.321	0.858	-	0.366	0.340	0.245	0.034

Log	RTFMP			SEPSIS			PRT1			PRT2			PRT3		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
ETC_a	1.000	0.700	0.952	0.859	0.445	0.419	0.985	0.673	0.768	0.737	-	-	0.914	0.680	0.828
PCC_2	1.000	0.831	0.793	1.000	0.745	0.689	0.970	0.607	0.793	1.000	0.788	0.617	0.945	0.716	0.996
MAP^2	1.000	0.939	0.745	1.000	0.603	0.521	1.000	0.868	0.842	1.000	0.975	1.000	1.000	0.913	1.000
MAP^3	0.955	0.492	0.309	0.954	0.210	0.193	0.966	0.729	0.655	0.995	0.960	0.992	0.907	0.878	0.954
MAP^4	0.862	0.182	0.087	0.895	0.048	0.062	0.939	0.677	0.467	0.995	0.631	0.895	0.839	0.763	0.631
MAP^5	0.756	0.070	0.025	0.831	-	-	0.917	0.658	0.309	0.979	0.442	0.410	0.775	0.560	0.293

Log	PRT4			PRT6			PRT7			PRT9			PRT10		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
ETC_a	0.995	0.753	0.865	1.000	0.822	0.908	0.999	0.726	0.998	0.999	0.611	0.982	0.972	0.790	-
PCC_2	0.955	0.705	0.822	0.800	0.694	0.832	0.857	0.743	0.857	0.927	0.658	0.738	1.000	0.804	-
MAP^2	1.000	0.917	1.000	1.000	0.957	1.000	1.000	0.927	1.000	1.000	0.991	0.958	1.000	0.387	-
MAP^3	1.000	0.979	1.000	1.000	0.873	0.983	1.000	0.920	0.972	0.971	0.526	0.542	0.814	0.061	-
MAP^4	1.000	0.977	1.000	1.000	0.830	0.950	1.000	0.699	0.725	0.913	0.207	0.208	0.639	0.007	-
MAP^5	0.998	0.982	0.968	1.000	0.574	0.632	1.000	0.543	0.645	0.853	0.076	0.064	0.435	-	-

TABLE 13: Comparison of precision measures over 20 real-life logs.

Log	BPIC12			$BPIC13_{cp}$			$BPIC13_{inc}$			$BPIC14_{f}$			$BPIC15_{1f}$		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
ETC_a	3	2	-	1	3	2	3	1	2	3	2	-	3	2	-
PCC_2	3	1	2	3	1	2	3	1	2	2	1	3	2	1	3
MAP^2	3	2	1	3	3	2	3	3	3	3	2	1	3	2	1
MAP^3	3	2	1	3	2	1	2	1	3	3	2	1	3	2	1
MAP^4	3	1	2	3	2	1	3	1	2	3	2	1	-	-	-
MAP^5	3	-	2	2	3	1	3	1	2	3	2	1	-	-	-

Log	$BPIC15_{2f}$			$BPIC15_{3f}$			$BPIC15_{4f}$			$BPIC15_{5f}$			$BPIC17_{f}$		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
ETC_a	3	1	2	3	1	2	3	1	2	3	1	2	3	2	1
PCC_2	3	2	3	2	1	3	3	3	3	2	1	3	3	1	2
MAP^2	3	1	2	3	2	3	3	2	3	3	2	3	3	2	1
MAP^3	3	1	2	3	1	2	3	1	2	3	1	2	3	2	1
MAP^4	3	-	2	3	-	2	3	-	2	3	-	2	3	2	1
MAP^5	3	-	2	3	-	2	3	-	2	3	-	2	3	2	1

Log	RTFMP			SEPSIS			PRT1			PRT2			PRT3		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
ETC_a	3	1	2	3	2	1	3	1	2	-	-	-	3	1	2
PCC_2	3	2	1	3	2	1	3	1	2	3	2	1	2	1	3
MAP^2	3	2	1	3	2	1	3	2	1	3	2	3	3	2	3
MAP^3	3	2	1	3	2	1	3	2	1	3	1	2	2	1	3
MAP^4	3	2	1	3	1	2	3	2	1	3	1	2	3	2	1
MAP^5	3	2	1	-	-	-	3	2	1	3	2	1	3	2	1

Log	PRT4			PRT6			PRT7			PRT9			PRT10		
Miner	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**	**SM**	**IM**	**SHM**
ETC_a	3	1	2	3	1	2	3	1	2	3	1	2	3	2	-
PCC_2	3	1	2	2	1	3	3	2	3	3	1	2	3	2	-
MAP^2	3	2	3	3	2	3	3	2	3	3	2	1	3	2	-
MAP^3	3	2	3	3	1	2	3	1	2	3	1	2	3	2	-
MAP^4	3	2	3	3	1	2	3	1	2	3	1	2	3	2	-
MAP^5	3	2	1	3	1	2	3	1	2	3	2	1	-	-	-

TABLE 14: Models ranking yielded by precision measures over 20 real-life logs.

Precision	Split Miner				Inductive Miner				Struct. Heuristics Miner			
	avg	max	min	total	avg	max	min	total	avg	max	min	total
ETC_a	60.0	351.9	0.3	720.3	84.2	642.7	0.1	1009.8	34.0^+	101.4^+	0.2^+	305.9^+
PCC_2	145.8	1482.8	0.1	1749.6	157.3	1598.3	0.1	1887.3	164.8	1530.4	0.1	1977.8
MAP^2	**0.1**	0.8	**>0.1**	1.6	**0.9**	**3.3**	**>0.1**	**10.9**	**1.1**	**5.7**	**>0.1**	**12.9**
MAP^3	**0.1**	**0.4**	**>0.1**	**1.5**	323.2	2844.0	**>0.1**	3878.9	8.4	43.2	**>0.1**	100.4
MAP^4	0.2	0.6	**>0.1**	1.9	818.5^+	4377.6^+	$\mathbf{>0.1}^+$	5729.3^+	49.8	340.2	**>0.1**	547.6
MAP^5	0.7	3.2	**>0.1**	8.7	-	-	-	-	97.2^+	357.4^+	$\mathbf{>0.1}^+$	972.4^+

TABLE 15: Time performance (in seconds) of precision measures using the twelve public logs ('+' indicates a result obtained on a subset of the twelve logs, due to some of the measurements not being available).

Precision	Split Miner				Inductive Miner				Struct. Heuristics Miner			
	avg	max	min	total	avg	max	min	total	avg	max	min	total
ETC_a	16.1	106.5	0.2	129.1	16.4	99.2	0.2	114.9	74.3	350.2	0.7	520.1
PCC_2	184.8	1468.9	**>0.1**	1478.8	152.7	1213.9	**>0.1**	1221.3	242.6	1649.4	**>0.1**	1698.5
MAP^2	**0.1**	0.4	**>0.1**	0.6	**2.1**	**15.9**	**>0.1**	**16.7**	**0.7**	**2.3**	**>0.1**	**4.7**
MAP^3	**0.1**	**0.2**	**>0.1**	**0.5**	7.0	47.0	**>0.1**	55.9	2.7	12.6	**>0.1**	19.1
MAP^4	**0.1**	0.4	**>0.1**	0.7	146.2	1131.2	**>0.1**	1169.9	29.4	120.0	**>0.1**	205.9
MAP^5	0.8	5.8	**>0.1**	6.8	55.5	332.6	0.1	388.2	77.8	329.4	0.1	544.4

TABLE 16: Time performance (in seconds) of precision measures using the eight proprietary logs.

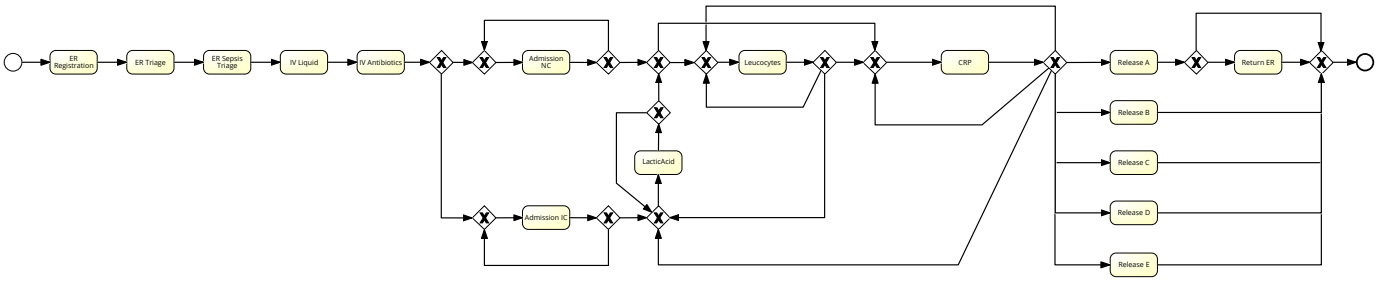


Fig. 16: Model discovered by SM from the SEPSIS log.

in the log, in order to obtain the most accurate results.

6 Conclusion

This article presented a family of fitness and precision measures to assess the accuracy of process models automatically discovered from event logs. The key idea of the proposal is to compare the k^{th}-order Markovian abstraction of a process model against that of an event log using a graph matching algorithm. We showed that the fitness measures fulfill six out of the seven fitness properties in [5] for any value of k and all seven properties for a suitably chosen value of k dependent on the log. Conversely, the precision measures fulfil four of the five precision properties in [4] for any value of k and all five properties for a value of k dependent on the log. To the best of our knowledge, these are the first fitness and precision measures that fulfil all of the above-mentioned properties.

While fulfiling the proposed properties is a desirable quality, it does not guarantee that the proposed measures provide intuitive results in practice. To validate the intuitiveness of the proposed measures, we compared the ranking they induce against those induced by existing fitness and precision measures using a collection of model-log pairs proposed in [16] (extended to cover fitness in addition to precision). For $k \geq 4$, the proposed fitness measures induce rankings that coincide with alignment-based fitness – a commonly used fitness measure. Meanwhile, for $k \geq 3$, the proposed precision measures induce rankings consistent with those of the anti-alignment precision measure, which has been previously posited as a ground-truth for precision measures.

A second evaluation using real-life logs showed that the execution times of the proposed fitness measures (for $k \leq 5$) are considerably lower than existing fitness measures, except when applied to process models that contain flower structures, in which case alignment-based fitness offers the best performance. Similarly, the execution times of the proposed precision measures (for $k \leq 5$) are considerably lower than existing precision measures, except for models that contain flower structures in which case ETC_a precision provides the best performance.

The experimental evaluation also put into evidence the scalability limitations of the proposed approach for $k > 5$, highlighting the trade-off between the ability to measure fitness and precision in a principled manner (i.e. in a way that fulfils all theoretical properties) and the imperative of scalability. Possible avenues for future work include the design of more efficient fitness and precision measures by exploring alternative behavioral abstractions (besides Markovian ones) or by combining the proposed approach with divide-and-conquer approaches, which have been recently explored in the context of alignment-based fitness [30].

Acknowledgments

This research is funded by the Australian Research Council (grant DP180102839) and the Estonian Research Council (IUT20-55).

References

[1] J. D. Weerdt, M. D. Backer, J. Vanthienen, and B. Baesens, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Information Systems*, vol. 37, no. 7, 2012.

[2] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated discovery of process models from event logs: Review and benchmark," *IEEE Transactions on Knowledge and Data Engineering (to appear)*, 2018.

[3] G. Janssenswillen, N. Donders, T. Jouck, and B. Depaire, "A comparative study of existing quality measures for process discovery," *Information Systems*, vol. 71, pp. 1–15, 2017.

[4] N. Tax, X. Lu, N. Sidorova, D. Fahland, and W. van der Aalst, "The imprecisions of precision measures in process mining," *Information Processing Letters*, vol. 135, pp. 1–8, 2018.

[5] W. M. P. van der Aalst, "Relating process models and event logs - 21 conformance propositions," in *International Workshop on Algorithms & Theories for the Analysis of Event Data*. Springer, 2018, pp. 56–74.

[6] A. Augusto, A. Armas-Cervantes, R. Conforti, M. Dumas, M. L. Rosa, and D. Reißner, "Abstract-and-compare: A family of scalable precision measures for automated process discovery," in *International Conference on Business Process Management (BPM)*. Springer, 2018, pp. 158–175.

[7] A. Rozinat and W. M. Van der Aalst, "Conformance testing: Measuring the fit and appropriateness of event logs and process models," in *International Conference on Business Process Management*. Springer, 2005, pp. 163–176.

[8] S. K. vanden Broucke, J. Munoz-Gama, J. Carmona, B. Baesens, and J. Vanthienen, "Event-based real-time decomposed conformance analysis," in *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*. Springer, 2014, pp. 345–363.

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

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

[11] G. Greco, A. Guzzo, L. Pontieri, and D. Sacca, "Discovering expressive process models by clustering log traces," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, 2006.

[12] A. Rozinat and W. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, no. 1, 2008.

[13] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A robust f-measure for evaluating discovered process models," in *IEEE Symposium on CIDM*. IEEE, 2011.

[14] J. Munoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *BPM*. Springer, 2010.

[15] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. van Dongen, and W. van der Aalst, "Measuring precision of modeled behavior," *ISeB*, vol. 13, no. 1, 2015.

[16] B. van Dongen, J. Carmona, and T. Chatain, "A unified approach for measuring precision and generalization based on anti-alignments," in *BPM*. Springer, 2016.

[17] S. Leemans, D. Fahland, and W. van der Aalst, "Discovering block-structured process models from event logs - a constructive approach," in *International Conference on Petri Nets and Other Models of Concurrency*. Springer, 2013.

[18] A. Augusto, R. Conforti, M. Dumas, and M. L. Rosa, "Split miner: Discovering accurate and simple business process models from event logs," in *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017.

[19] S. vanden Broucke and J. De Weerdt, "Fodina: a robust and flexible heuristic process discovery technique," *Decision Support Systems*, vol. 100, pp. 109–118, 2017.

[20] A. Weijters and J. Ribeiro, "Flexible heuristics miner (FHM)," in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 2011.

[21] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[22] ——, "Variants of the hungarian method for assignment problems," *Naval Research Logistics Quarterly*, vol. 3, no. 4, pp. 253–258, 1956.

[23] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[24] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[25] A. Backurs and P. Indyk, "Edit distance cannot be computed in strongly subquadratic time (unless SETH is false)," in *47th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2015, pp. 51–58.

[26] R. Conforti, M. L. Rosa, and A. ter Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 300–314, 2017.

[27] S. Leemans, D. Fahland, and W. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *BPM Workshops*. Springer, 2014.

[28] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, and G. Bruno, "Automated discovery of structured process models from event logs: The discover-and-structure approach," *Data and Knowledge Engineering*, vol. 117, pp. 373–392, 2018.

[29] B. F. van Dongen, "Efficiently computing alignments - using the extended marking equation," in *International Conference on Business Process Management (BPM)*. Springer, 2018, pp. 197–214.

[30] W. L. J. Lee, H. M. W. Verbeek, J. Munoz-Gama, W. M. P. van der Aalst, and M. Sepúlveda, "Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining," *Information Science*, vol. 466, pp. 55–91, 2018.