The connection between process complexity of event sequences and models discovered by process mining

Adriano Augusto [a],[1], Jan Mendling [b],[*],[1], Maxim Vidgof [c],[1], Bastian Wurm [c],[1]

[a] *The University of Melbourne, 3010 Parkville, VIC, Australia*
[b] *Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany*
[c] *Wirtschaftsuniversität Wien, Welthandelsplatz 1, 1020 Vienna, Austria*

ARTICLE INFO

ABSTRACT

Process mining is a research area focusing on the design of algorithms that can automatically provide insights into business processes. Among the most popular algorithms are those for automated process discovery, which have the ultimate goal to generate a process model that summarizes the behavior recorded in an event log. Past research had the aim to improve process discovery algorithms irrespective of the characteristics of the input log. In this paper, we take a step back and investigate the connection between measures capturing characteristics of the input event log and the quality of the discovered process models. To this end, we review the state-of-the-art process complexity measures, propose a new process complexity measure based on graph entropy, and analyze this set of complexity measures on an extensive collection of event logs and corresponding automatically discovered process models. Our analysis shows that many process complexity measures correlate with the quality of the discovered process models, demonstrating the potential of using complexity measures as predictors of process model quality. This finding is important for process mining research, as it highlights that not only algorithms, but also connections between input data and output quality should be studied.

1. Introduction

Recent years have seen a drastic increase in the availability of event sequence data and corresponding techniques for analyzing business processes, healthcare pathways, or software development routines [46,28,16]. Process mining is a research area focusing on the design of techniques that can automatically provide insights into business processes by analyzing historic process execution data, known as event logs [45,15]. In process mining research, various algorithms have been developed for automated process discovery. A recent study found a rich spectrum of 35 distinct groups of such algorithms scattered over more than 80 studies [6]. Much of this research on automated process discovery is motivated by the ambition to improve process discovery outputs, in terms of high precision and recall, while producing models that are simple and easy to understand [3].

So far, this stream of research on improving process discovery algorithms has been largely driven by the implicit assumption that a better algorithm would generate a better process model, no matter the characteristics of the input event log. In fact, there are good reasons to question this narrow focus. First, research on computer experiments highlights that studying the effect of input data characteristics on output is an important objective in many research areas [39,27]. Second, research on classifiers demonstrates the benefits of selecting algorithms based on characteristics of the input data [20,36,35]. Third, in various application domains, establishing a solid understanding of how input characteristics influences output has led to fundamental algorithmic innovations [38]. For these reasons, Kriegel et al. recommend factoring in the variation of input parameters over meaningful ranges when comparing algorithms [22].

In this paper, we revisit the output quality of automated process discovery algorithms in light of these arguments. More specifically, we investigate the empirical connections between measures capturing process complexity in terms of process behavior recorded in an event log and the quality of the process models discovered from that event log, as well as which of these process complexity measures can serve as a suitable predictor of process discovery quality. To this end, we first review process complexity measures defined in prior research studies. We analyze their characteristics and categorize them according to what perspective of process complexity they capture. Then, noting that each measure relates to a different perspective, we propose a new measure of process complexity based on graph entropy, which can exhaustively capture process complexity from multiple perspectives. Lastly, we analyze the process complexity measures using a prototypical implementation and an evaluation over an extensive set of event logs and their corresponding automatically discovered process models. Our analysis shows that many process complexity measures (including our novel measure) correlate with the quality of the discovered process models. Our findings demonstrate the potential of using process complexity measures as predictors for the quality of process models discovered with state-of-the-art process discovery algorithms. Such a result is important for process mining research, as it highlights that not only algorithms, but also connections between input data and output quality should be studied.

The remainder of the paper is structured as follows. Section 2 summarizes prior research on measuring process complexity and related studies. Section 3 presents our process complexity measure, how it is calculated, and which properties it satisfies. Section 4 presents our evaluation and the main findings of this study. Section 5 concludes the paper and draws ideas for future work.

2. Background and related work

In this section, we contextualize our study by discussing related work with a focus on the quality of discovered process models, automated process discovery algorithms, and process complexity.

2.1. Quality of discovered process models

Process discovery is the task that encompasses the understanding of a business process behavior and the representation of that behavior in the form of a process model [15]. Process mining research has developed various algorithms for automated process discovery. The algorithms analyze the information recorded in an input event log (i.e., the process execution data capturing the process behavior) and automatically generate a process model as an output. Several measures have been defined for assessing the quality of a discovered process model [45]. Precision, fitness, and simplicity are the most frequently used ones.

Prior research studies have proposed several implementations of precision, fitness, and simplicity measures [41,34,10]. Let us assume that an event log is given, containing four sequences of events: $\langle a,b,c,d,f,e\rangle, \langle a,c,b,d,f,e\rangle, \langle a,c,b,d,e\rangle$ and $\langle a,b,c,d,e\rangle$. Let us also assume that three different automated process discovery algorithms have used this event log to construct the three Petri net process models shown in Fig. 1.

Precision measures to which extent the process behavior (in terms of sequences of events) captured by the process model can be found in the original event log. It ranges between 0 and 1, where a value of 1 means that the process model can only generate sequences of events that are also contained in the event log. Model (c) is precise in this sense, Model (a) is less precise as it permits repetitions of the event f, and Model (b) is the least precise allowing any repetition and combination of the events b,c,d,f, as long as a is the first and e the last event.

Fitness measures to which extent the behavior contained in the event log can be reproduced by the process model. Also fitness ranges between 0 and 1, where a value of 1 means that all the sequences of events contained in the event log can be reproduced by the process model. We observe that all Models (a), (b) and (c) have a fitness of 1, since all of them can reproduce the four sequences of events: $\langle a,b,c,d,f,e\rangle, \langle a,c,b,d,f,e\rangle, \langle a,c,b,d,e\rangle$ and $\langle a,b,c,d,e\rangle$.

Simplicity captures how easily a process model can be understood by a human. A model that is not simple is called *complex*. Several complexity measures have been proposed in the literature [24]. Some of them take into account the structure of a process model (e.g., model size), others the behavioral variability of a process model (e.g., control flow complexity). In our example in Fig. 1, it is apparent that Model (c) is structurally more complex than Models (a) and (b), since it has more nodes
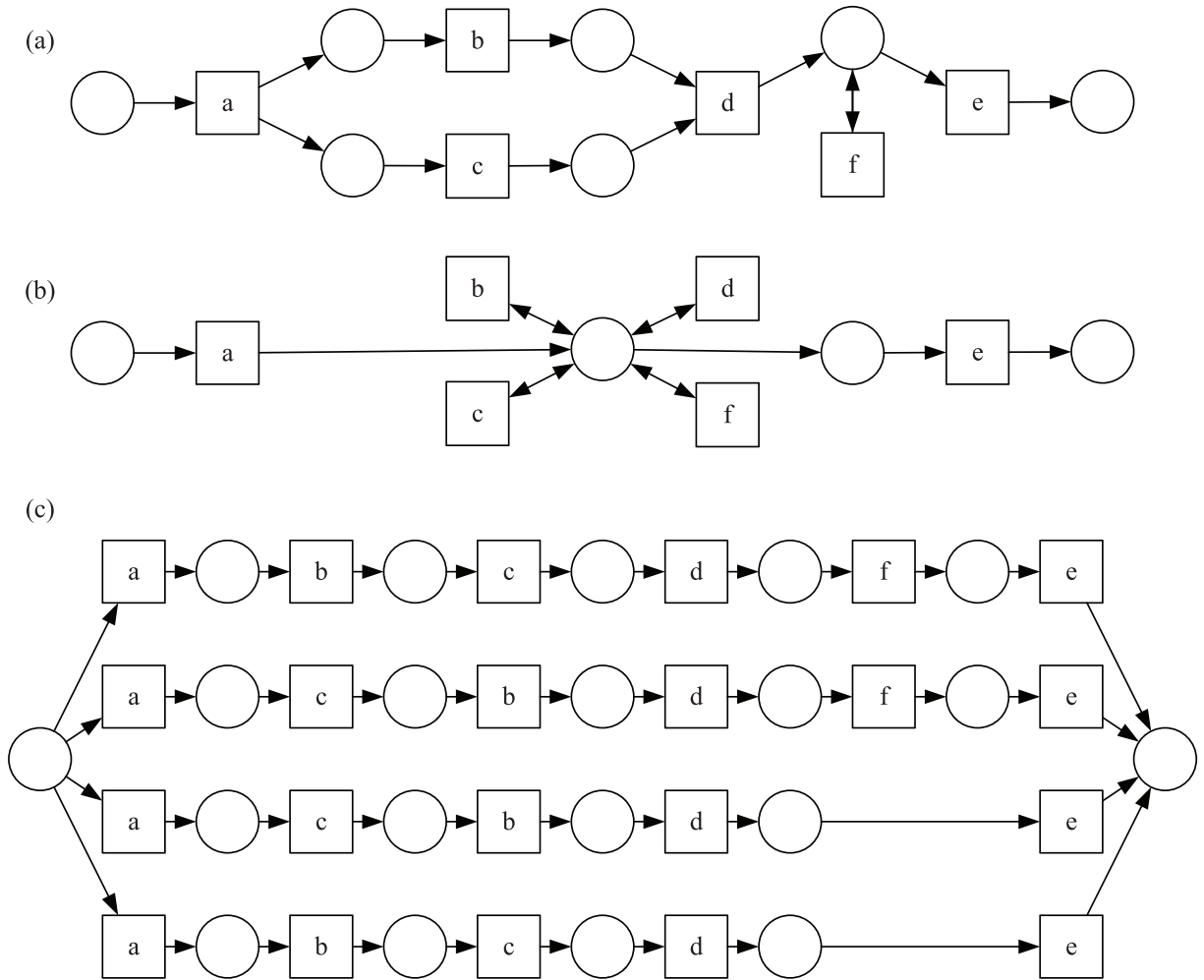
Fig. 1. Considering the event log: $\langle a, b, c, d, f, e \rangle$, $\langle a, c, b, d, f, e \rangle$, $\langle a, c, b, d, e \rangle$, $\langle a, b, c, d, e \rangle$; three different algorithms may respectively discover the (Petri net) process models (a), (b) and (c).

and edges. On the other hand, Model (c) is behaviorally simpler than Model (a) and (b), since it allows for less variation. Accordingly, its control flow complexity is lower.

2.2. Automated process discovery algorithms

Over the past decade, more than 80 research papers have proposed new algorithms for automated process discovery [6]. Often, the process models they produce significantly differ in terms of how they trade off precision, fitness, and simplicity [6]. The latest benchmark study [6] compares and evaluates seven of the most effective state-of-the-art algorithms, namely, α$ [18] (A$), Inductive Miner [23] (IM), Evolutionary Tree Miner [9] (ETM), Fodina [50] (FO), Structured Heuristic Miner 6.0 [5] (SHM), Split Miner [3] (SM), Hybrid ILP Miner [49] (HILP). Although there was no algorithm found to clearly dominate the others, IM, ETM, and SM turned out to be most reliable in terms of discovering fitting, precise, and simple process models across the whole benchmark dataset of 24 real-world event logs. However, they also showed a substantial variance of performance. HILP and A$ often discovered unsound models (i.e., containing behavioral errors, such as deadlocks) and rarely produced highly fitting, precise, or simple process models. While FO and SHM often produced accurate models, these were usually highly complex and difficult to interpret.

Even though the study by Augusto et al. [6] does not discuss this aspect, their results suggest a connection between the input event log features and the quality of the automatically discovered process models. Also other works have tried to select the most suitable discovery algorithm based on event log characteristics [36,35], but without studying the connection between log complexity and model quality. For this reason, we hypothesize that characteristics of the event log influence the quality of discovered process models.

2.3. Process complexity as a factor of process discovery quality

It is a challenge for discovery algorithms to generate process models that are easy to understand. Often, the generated models are overly complex. These complex models are called "spaghetti models". Van der Aalst emphasizes that "spaghetti-like structures are not caused by the discovery algorithm but by the variability of the process" [43]. In line with this observation, several proposals have been made for pre-processing event logs independent of the discovery algorithm applied. These proposals build on clustering, supervised sequence labeling, sequential patterns, or text matching [11,40,14]. They highlight the potential to improve process discovery outputs by modifying characteristics of the event log data as an input.

To assess the empirical connection between log complexity and process discovery quality, we require a specific measure that can adequately quantify process log complexity. Several evaluations of process mining algorithms have reported basic measures. There have been studies that focus on process model complexity [24,25,37,26]. The few studies that consider process complexity more explicitly stem from computer science, organization science, and management science. The corresponding measures are categorized in Table 1 and described next.

The first category includes *size* measures. Various properties of an event log can be easily counted including the number of *events*, *sequences*, and *event types*, the minimum, maximum, and average *sequence length*, and the average and minimum *time difference* between two events (proposed by Günther [17, Ch.3]).

The second category contains measures related to the *variation* of the process behavior recorded in the event log. Several of these measures take the transition matrix derived from the directly-follows relations observed in the event log as a starting point. Pentland [31] proposes the calculation of complexity as the number of *acyclic paths* implied by the transition matrix derived from the event log. Hærem et al. [19] use a slight variation based on what they call the number of *ties*, which in essence is the count of directly-follows relations observed in the event log. Also, Pentland's proposal [30] of measuring the number of operations of compressing the event log using the *Lempel–Ziv* algorithm is a variation measure. Finally, the (absolute and relative) number of *distinct sequences* [45] and the average number of *distinct events per sequence* [17] also provide an indication of variation.

The third category refers to *distance* measures. Several distance notions have been defined. Günther [17] proposes the notion of *affinity*, which is based on the overlapping directly-follows relations of two event sequences. His proposed complexity measure, namely *average affinity*, is calculated as the mean of affinity over all pairs of event sequences [17, Ch.3]. This measure is closely related to the one proposed by Pentland [30] called *deviation from random* of the transition matrix. Furthermore, Pentland also proposes a second distance measure based on the average *edit distance* between event sequences based on notions of classical optimal matching [12].

Each of these measures has its limitations and blind spots. We can easily identify cases where one measure indicates a difference while other measures are unaffected. We make the following observations on the relationship between two event logs L_1 and L_2:

Observation O1: Assume that L_1 and L_2 have the same size measures of events, event types and sequences. Let us consider the corner case where all the sequences of events recorded in L_1 are the same, while the sequences of events recorded in L_2 are all different. While size measures would not be able to capture any difference, variation and distance measures would detect them.

Observation O2: Assume that L_1 and L_2 have the same variation measures in terms of the number of unique sequences. Still, L_1 and L_2 can be strikingly different in terms of size measures if the same sequences are repeated or not, and in terms of distance measures if the variants are very similar in one log and very different in the other.

Observation O3: Assume that L_1 and L_2 have the same distance measures with each pair of sequences having, e.g., an edit distance of 1 on average. If L_1 includes each sequence of L_2 twice, the size measures will be substantially different. Also,

Table 1
Complexity measures for business processes based on event logs.

Category	Measure	Label	Ref.
Size	Number of Events	magnitude	[17]
	Number of Event Types	variety	[17]
	Number of Sequences	support	[17]
	Minimum, Average, Maximum Sequence Length	TL-min, TL-avg, TL-max	[45]
	Average Time Difference between Consecutive Events	(time) granularity	[17]
Variation	Number of Acyclic Paths in Transition Matrix	LOD	[31]
	Number of Ties in Transition Matrix	t-comp	[19]
	Lempel–Ziv Complexity	LZ	[30]
	Number and Percentage of Unique Sequences	DT(#), DT(%)	[45]
	Average Distinct Events per Sequence	structure	[17]
Distance	Average Affinity	affinity	[17]
	Deviation from Random	dev-random	[30]
	Average Edit Distance	avg-dist	[30]

the variation can be quite different if L_1 includes many sequences that are the same, while a few are very different, as compared to L_2 where all sequences have rather little distance.

These observations clearly show that there is no unique process complexity measure capable of capturing information regarding size, variation, and distance at once. In the next section, we propose a new measure that addresses this challenge.

3. Process complexity based on graph entropy

In this section, we propose a novel measurement of process complexity based on graph entropy. More generally, entropy has been used for assessing the behavior of process representations and corresponding logs at the language level in [34,33]. This means that the measurements do not account for the distribution of variants of an event log. Graph entropy is particularly suited as an underlying concept because it can capture size, variation, and distance in an integral way. To this end, we have to map an event log to a graph structure that acknowledges equivalences between sequences without introducing abstractions. In Section 3.1, we define the notion of an extended prefix automaton. Section 3.2 defines graph entropy for extended prefix automata, proves monotonicity, and relates the proposed measure to Observations 1–3.

3.1. Event logs as prefix automata

In this section, we define the extended prefix automaton for an event log. The concept of a prefix automaton was introduced by Munoz-Gama and Carmona in [29] based on concepts described by Van der Aalst et al. in [44]. The concept of a prefix automaton is particularly suited for our purpose of describing the complexity of an event log. Prefix automata describe sequences without loss of information and abstraction. They account for equivalent prefixes but do not introduce complexity that is not present in the event log. Fig. 2 illustrates the idea of a prefix automaton for the event log with the four sequences we discussed above. We observe that overlapping prefixes of the sequences lead to joint paths in the prefix automaton. All paths originate from the *root*. There are as many variants in this event log as there are nodes on the right-hand side without successors. These are four in our case.

We revisit basic notions of events, event sequences, and event logs upon which we will define the construction of the extended prefix automaton.

Definition 1 (*Event, Event Sequence, Event Log [45]*). Let E be a set of unique event identifiers. For each event $e \in E$, we define four attributes:

- An *activity* $\in E \to A$ where A is the set of activities. *activity*(e) is an activity type that an event e refers to.
- A timestamp $ts \in E \to TS$ where TS is the set of timestamps. $ts(e)$ is the time when e occurred.
- A case identifier $case \in E \to CID$ where CID is the set of unique case identifiers. $case(e)$ is the case that the event e is related to.
- A predecessor $pred \in E \to E \cup \bot$ maps each event to a preceding event of the same case if such an event exists or to $\bot$ otherwise. $pred(e)$ is a predecessor of e if they share the same case identifier $case(pred(e)) = case(e)$, if

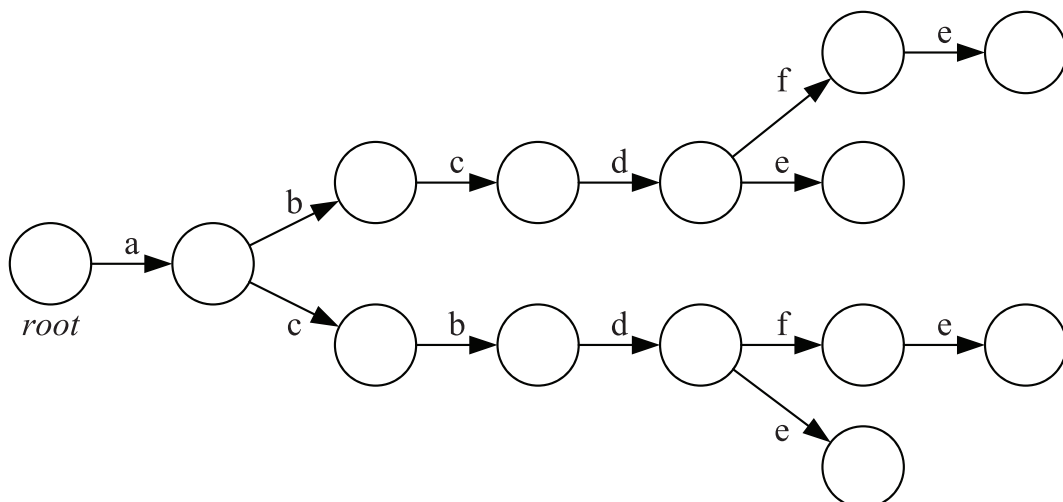**Fig. 2.** Prefix automaton derived from the previously discussed event log $\langle a,b,c,d,f,e \rangle, \langle a,c,b,d,f,e \rangle, \langle a,c,b,d,e \rangle, \langle a,b,c,d,e \rangle$.
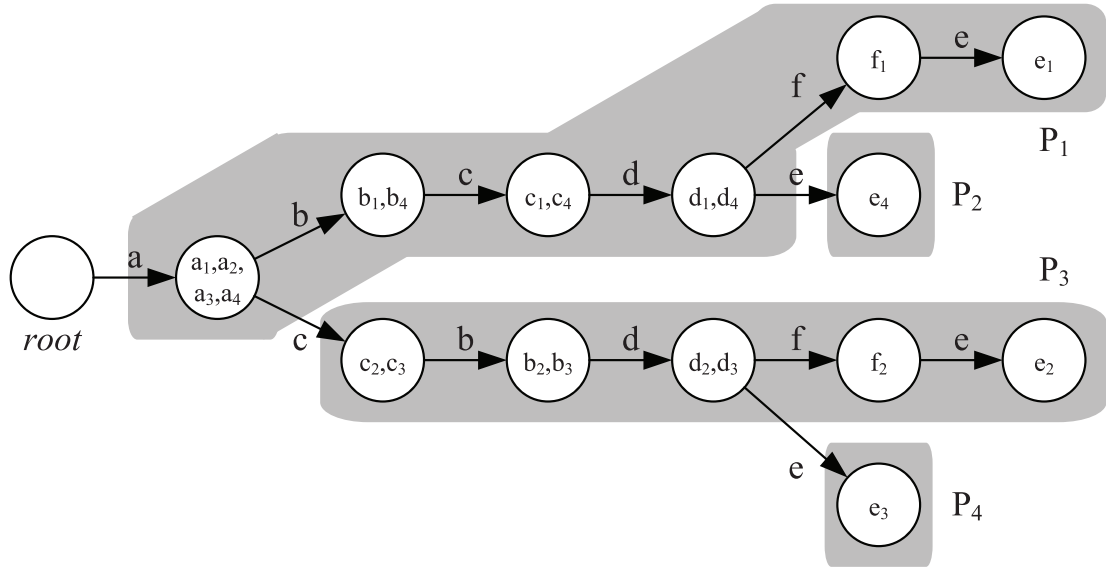
Fig. 3. Extended prefix automaton with partitions derived from the event log $\langle a,b,c,d,f,e\rangle, \langle a,c,b,d,f,e\rangle, \langle a,c,b,d,e\rangle, \langle a,b,c,d,e\rangle$.

$ts(pred(e)) < ts(e) \wedge \nexists\ e' \in E : [case(e') = case(e) \wedge ts(pred(e)) < ts(e') < ts(e)]$. $pred(e) = \perp$ if $\nexists\ e' \in E : [case(e') = case(e) \wedge ts(e') < ts(e)]$.

A plain event log $L_{plain} \in E^*$ is a finite sequence of events (with events potentially relating to different cases). A plain event log is ordered by event timestamps and not by cases.

The connection between an event log L and a corresponding plain event log L_{plain} is trivial. L_{plain} is a concatenation of all $e \in \bigcup_{\sigma \in L} \sigma$. In the opposite direction, a trace $\sigma_{cid} = \langle e_1, \ldots, e_n\rangle | e_i \in E \wedge case(e_i) = cid \wedge ts(e_i) < ts(e_{i+1})$ for all $1 \leqslant i \leqslant n - 1$, and a log $L = \bigcup_{e \in E} \sigma_{case(e)}$.

[29] defines a prefix automaton $PA = (S, T, A, s_0)$ with S being a set of states, A the set of activities, $T \subseteq S \times A \times S$ set of transitions and s_0 the initial state. Based on this, we introduce the concept of an extended prefix automaton. Compared to the prefix automaton TS in [29], our automaton is extended in two ways. First, we define the seq function that maps each state $s \in S$ to a set of events $seq(s) \subseteq E$ having the same prefix as the state itself. Second, we define a partitioning function C that splits the extended prefix automaton EPA into $0 \leqslant k \leqslant |L|$ partitions. Note that $|L|$ here refers to the number of traces in an event log L. We discuss partitioning in more detail below.

Definition 2 (*Extended prefix automaton*). $EPA = (S_+, T, A, C, seq, root)$

- $S_+ = S \cup \{root\}$ is a set of states
- A is a set of activities
- $T \subseteq S_+ \times A \times S$ is a set of transitions. Note that the *root* has no incoming transitions.
- $C \in S_+ \to \mathbb{N}_0 \cup \{\perp\}$ a partitioning function, defining for each state $s \in S_+$ the partition to which it belongs. $C(s)$ refers to a partition the state s belongs to. A state can only belong to one partition. We write $C(root) = \perp$ because the root node does not belong to any partition.
- $seq \in S \to \wp(E)$ maps each state to events having the same prefix as the state. Note that every event in the log L (or L_{plain}) corresponds to one and only one state: $\forall e \in L\ \exists s \in S : e \in seq(s)$ and $\forall s, s' \in S : s \neq s' \Rightarrow seq(s) \cap seq(s') = \varnothing$.
- $root \in S_+$ is the entry state of the automaton and corresponds to an empty prefix. (same as s_0 in [29])

Although the concept of accepting states is not mentioned in [29] and mentioned only informally in [44], we want to state explicitly here that in an extended prefix automaton all states are accepting states.[1] This means that every trace $\sigma \in L$ can be replayed (*recall* = 1), but the extended prefix automaton can also produce shorter sequences that were not observed in the log L (*precision* $\leqslant$ 1). Note that this is no harm to our ambition of obtaining a representation of the event log with 100% precision and recall when we store observed event sequences using the extended prefix automation.

---

[1] It is an alternative to introduce sink nodes similar to the root node as the only accepting states. We do not consider this alternative here, because it does not allow the incremental construction of the prefix automaton while events of running cases are continuously added.

Consider our previous example sequences and Fig. 3. The example illustrates how events of the event log are mapped to states with the help of the function $seq(s)$. All sequences start with activity a, so one state s_1^1 suffices to capture all observed behavior up to that point. The second event in sequences 1 and 4 is b, which is captured by the state s_2^1 such that $C(s_2^1) = 1$ and $seq(s_2^1) = \{b_1, b_4\}$. In sequences 2 and 3, the second event is c, which requires the extended prefix automaton to branch and introduce another state s_2^3 with $seq(s_2^3) = \{c_2, c_3\}$ in a new partition P_3. Each time such branching occurs, a new partition is introduced, and the final number of partitions equals the number of observed process variants.

For its construction, we use the corresponding plain event log L_{plain}. Algorithm 1 shows how the extended prefix automaton is constructed. The algorithm iterates over the events in the log L_{plain} and uses the variables $last_{AT}$, $pred_{AT}$, $current_{AT}$, and $current_c$. The variable $last_{AT}$ is a mapping used to store the latest corresponding activity type that is added to the automaton for every case ID. While it is possible to search for it at every iteration, such a mapping increases the efficiency of the algorithm. $pred_{AT}$ is used to store the activity type of the current event's predecessor, while $current_{AT}$ stores the activity type of the event in question. $current_c$ stores the partition number of the current activity type.

Algorithm 1: Constructing the Extended Prefix Automaton

Input: Plain event log L_{plain}.

Result: Extended Prefix Automaton EPA

1   Build $EPA = (S_+, T, A, C, seq, root)$ such that
     $S_+ = \{root\}, T = A = seq = \emptyset, C(root) = 0$;

2   $last_{AT} \leftarrow \emptyset$;

3   **forall** $event\ e \in L_{plain}$ **do**

4   **if** $pred(e) \neq \perp$ **then**

5          $pred_{AT} \leftarrow last_{AT}(case(e))$;

6   **else**

7          $pred_{AT} \leftarrow root$;

8   **end**

9      $current_{AT} \leftarrow \emptyset$;

10   **if** $\exists s | (pred_{AT}, activity(e), s) \in T$ **then**

11         $current_{AT} \leftarrow s$;

12   **else**

13   **if** $\exists a, s | (pred_{AT}, a, s) \in T$ **then**

14             $current_c \leftarrow max(C) + 1$;

15   **else**

16   **if** $pred_{AT} = root$ **then**

17                 $current_c \leftarrow 1$;

18   **else**

19                 $current_c \leftarrow C(pred_{AT})$;

20   **end**

21   **end**

22   Create new state s;

23         $S_+ \leftarrow S_+ \cup \{s\}$;

24         $T \leftarrow T \cup (pred_{AT}, activity(e), s)$;

25         $A \leftarrow A \cup \{activity(e)\}$;

26         $C \leftarrow C \cup (s, current_c)$;

27         $current_{AT} \leftarrow s$;

28   **end**

29      $seq(current_{AT}) \leftarrow seq(current_{AT}) \cup e$;

30      $last_{AT}(case(e)) \leftarrow current_{AT}$;

31   **end**

Lines 1 and 2 initialize variables. From Line 3 on, we iterate over the complete set of events of the plain event log. Lines 4–8 are concerned with identifying if the current event continues an already stored case or starts a new one at the root. Lines 10–12 map the resulting state to $current_{AT}$ for the case that a transition from a preceding state using the activity type of the

current event already exists. If not, Lines 13—20 and Line 26 show how partitioning is performed. As a new state is created, it is immediately assigned to a partition and this assignment does not change. As Lines 14, 17, and 19 show, there are three possible cases of how a state can be assigned to a new partition:

1. The preceding state $pred_{AT}$ already has some outgoing transitions. However, the $activity(e)$ is not among them. This means that a new path towards the next activity is introduced in $pred_{AT}$. This new path and the new states in this path are made a new partition and the partition count is incremented.
2. The preceding state is *root* and it has no outgoing transitions, which means e is the first event in L_{plain}. In this case, the new state defines the first partition in the extended prefix automaton. Note that this case does not differ from the previous one conceptually, but requires a slightly different implementation.
3. The preceding state has no outgoing transitions but is not a root state. Since the $pred_{AT}$ has no outgoing transitions, the new activity does not add any path and thus the new state should belong to the same partition as its predecessor.

We assume that all look-up functions in this algorithm can be implemented with computational complexity in $\mathcal{O}(k)$ with a constant k. Then, iterating over the set of events E drives the complexity of this calculation. The complexity of calculating the extended prefix automaton is accordingly $\mathcal{O}(E)$.

3.2. Graph entropy of extended prefix automata

Entropy is an appropriate concept for defining complexity measures due to its properties of monotonicity [34]. Some applications of entropy have been developed by Polyvyanyy et al. for defining precision and recall measures for conformance checking at the language level, in which eigenvalues of process models and event logs are calculated iteratively with polynomial complexity [34,33,32]. Here, we highlight the opportunity to use entropy as an underlying concept for calculating the process complexity of an event log based on extended prefix automata in linear time.

More specifically, we define the four entropy measures: *variant entropy* and *sequence entropy*, as well as their corresponding normalized versions. We build on the measures proposed by Dehmer et al. [13] who define graph entropy based on a partitioning of the graph into X_i partitions.

$$|X| \cdot log(|X|) - \sum_{i=1}^{k}|X_i| \cdot log(|X_i|) \tag{1}$$

We calculate *variant entropy* based on the extended prefix automaton by only considering its structure and not the number of events associated with each state. We apply the formula with $X = S$ (this is S_+ without the *root*). Note that we do not consider the set of transitions T, because every state has exactly one incoming transition. We obtain:

$$E_v = |S| \cdot log(|S|) - \sum_{i=1}^{max(C)}|\{s \in S|C(s) = i\}| \cdot log(|\{s \in S|C(s) = i\}|) \tag{2}$$

The measure of *variant entropy* measures the complexity of an event log based on the structure of the extended prefix automaton. It is possible to obtain the same valuead for two automata with a different number of states. For this reason, we introduce a *normalized variant entropy* with a range of $[0, 1]$. It is calculated as follows:

$$\bar{E}_v = \frac{E_v}{(|S|) \cdot log(|S|)} \tag{3}$$

Both versions of the *variant entropy* measure are based on the number of *states* in a partition in (2). This is an abstraction. Each state $s \in S$ is associated with a non-empty set of events, whose cases share the same prefix $seq(s) \neq \varnothing$. Each event $e \in L$ belongs to one and only one such set. This implies that every event can also be assigned to one and only one partition in the extended prefix automaton. In this way, we obtain a measure that reflects the frequencies of events and corresponding prefixes. Thus, we calculate *sequence entropy* based on the number of *events* in a partition by extending (2). For the sake of readability, we define:

$$seq(S) = \bigcup_{s \in S} seq(s) \tag{4}$$

and

$$seq_i(S) = \bigcup_{s \in S|C(s)=i} seq(s) \tag{5}$$

Then, we can define *sequence entropy* in the following way:

$$E_s = |seq(S)| \cdot log(|seq(S)|) - \sum_{i=1}^{max(C)}|seq_i(S)| \cdot log(|seq_i(S)|) \tag{6}$$

Same as *variant entropy*, the absolute *sequence entropy* measure depends on the number of states in the extended prefix automaton, but additionally on the number of events associated with each state. The same idea of normalization can be applied to *sequence entropy* resulting in the *normalized sequence entropy*:

$$\bar{E}_s = \frac{E_s}{|seq(S)| \cdot log(|seq(S)|)} \tag{7}$$

Lemma 1. *The sequence entropy measure is monotonous with respect to an increasing number of events.*
   This property makes sequence entropy particularly suitable for measuring process complexity based on event logs.

Proof. In order to prove monotonicity of the sequence entropy measure, we have to show that adding one event increases this measure. To this end, we have to show that the following equation holds for $|seq(S_2)| = |seq(S_1)| + 1$.

$$|seq(S_2)| \cdot log(|seq(S_2)|) - \sum_{i=1}^{max(C_2)} |seq_i(S_2)| \cdot log(|seq_i(S_2)|) > \tag{8}$$

$$|seq(S_1)| \cdot log(|seq(S_1)|) - \sum_{i=1}^{max(C_1)} |seq_i(S_1)| \cdot log(|seq_i(S_1)|) \tag{9}$$

We observe a corner case. If $max(C_2) = max(C_1) = 1$, then each summation equals the preceding term, such that both the left-hand and the right-hand side of the equation yield zero.
   We rearrange the equation by bringing the sums onto the right-hand side.

$$|seq(S_2)| \cdot log(|seq(S_2)|) - |seq(S_1)| \cdot log(|seq(S_1)|) > \tag{10}$$

$$\sum_{i=1}^{max(C_2)} |seq_i(S_2)| \cdot log(|seq_i(S_2)|) - \tag{11}$$

$$\sum_{i=1}^{max(C_1)} |seq_i(S_1)| \cdot log(|seq_i(S_1)|) \tag{12}$$

Now we can distinguish two cases. If $max(C_2) > max(C_1)$, then there must be one new partition that includes only one event. As a result, the right-hand side becomes $1 \cdot log(1) = 0$, such that the formula holds true because $|seq(S_2)| \cdot log(|seq(S_2)|)$ is larger than $|seq(S_1)| \cdot log(|seq(S_1)|)$ due to each of its factors being larger, respectively.
   Let us consider the alternative case that $max(C_2) = max(C_1)$. In this case, there exists an index x where there is a difference between $seq_i(S_2)$ and $seq_i(S_1)$. We write

$$|seq(S_2)| \cdot log(|seq(S_2)|) - |seq(S_1)| \cdot log(|seq(S_1)|) > \tag{13}$$

$$|seq(S_{x2})| \cdot log(|seq(S_{x2})|) - |seq(S_{x1})| \cdot log(|seq(S_{x1})|) \tag{14}$$

Furthermore, we can assume that there is a natural number $m < S_1$, such that we can write

$$|seq(S_2)| \cdot log(|seq(S_2)|) - |seq(S_1)| \cdot log(|seq(S_1)|) > \tag{15}$$

$$(|seq(S_2)| - m) \cdot log(|seq(S_2)| - m) - (|seq(S_1)| - m) \cdot log(|seq(S_1)| - m) \tag{16}$$

We bring all terms to the left-hand side, yielding

$$|seq(S_2)| \cdot log(|seq(S_2)|) \tag{17}$$

$$- (|seq(S_2)| - m) \cdot log(|seq(S_2)| - m) \tag{18}$$

$$- |seq(S_1)| \cdot log(|seq(S_1)|) \tag{19}$$

$$+ (|seq(S_1)| - m) \cdot log(|seq(S_1)| - m) \tag{20}$$

$$> 0 \tag{21}$$

Now, let us consider that we can write $S_2 = S_1 \cdot f = S_1 + 1$, such that $f = \frac{S_1+1}{S_1}$. Then, we obtain

$$f \cdot |seq(S_1)| \cdot log(f \cdot |seq(S_1)|) \tag{22}$$

$$- (f \cdot |seq(S_1)| - m) \cdot log(f \cdot |seq(S_1)| - m) \tag{23}$$

$$- |seq(S_1)| \cdot log(|seq(S_1)|) \tag{24}$$

$$+ (|seq(S_1)| - m) \cdot log(|seq(S_1)| - m) \tag{25}$$

$$> 0 \tag{26}$$

We pull f out of the logarithms and reorder the terms to obtain

$$f \cdot |seq(S_1)| \cdot log(f)|) \tag{27}$$
$$- (f \cdot |seq(S_1)| - m) \cdot log(f) \tag{28}$$
$$+ f \cdot |seq(S_1)| \cdot log(|seq(S_1)|) \tag{29}$$
$$- |seq(S_1)| \cdot log(|seq(S_1)|) \tag{30}$$
$$- (f \cdot |seq(S_1)| - m) \cdot log(|seq(S_1)| - m) \tag{31}$$
$$+ (|seq(S_1)| - m) \cdot log(|seq(S_1)| - m) \tag{32}$$
$$> 0 \tag{33}$$

These can be pulled together to obtain

$$f \cdot log(f) \cdot (|seq(S_1)| - |seq(S_1)| + m) \tag{34}$$
$$+ (f - 1) \cdot |seq(S_1)| \cdot log(|seq(S_1)|) \tag{35}$$
$$- (f - 1) \cdot (|seq(S_1)| - m) \cdot log(|seq(S_1)| - m) \tag{36}$$
$$> 0 \tag{37}$$

This can be simplified and rewritten to

$$f \cdot log(f) \cdot m \tag{38}$$
$$+ (f - 1) \cdot |seq(S_1)| \cdot log(|seq(S_1)|) \tag{39}$$
$$> (f - 1) \cdot (|seq(S_1)| - m) \cdot log(|seq(S_1)| - m) \tag{40}$$

We can replace the product on the right-hand side of the inequality with a term that is larger by replacing factors that are greater one with larger factors. In turn, we replace $|seq(S_1)| - m$ with $|seq(S_1)|$, such that the terms from lines (39) and (40) become equal. We then obtain

$$f \cdot log(f) \cdot m > 0 \tag{41}$$

This equation holds true because the three factors in line (41) are greater than zero due to properties of f and m. $\square$

Let us revisit **Observations 1–3** and focus on the sequence entropy measure. **Observation 1** states that we would want to distinguish two event logs L_1 and L_2 that have the same amount of events $n = |E|$ and sequences $m = |\{\sigma| \sigma \in L\}|$. If all sequences of L_1 are identical, we obtain one partition. Based on the corresponding extended prefix automaton, we obtain $E_s = |seq(S)| \cdot log(|seq(S)|) - \sum_{i=1}^{1} |seq_i(S)| \cdot log(|seq_i(S)|) = 0$. If all sequences are fully different in L_2 (and let us assume them to be of equal length l), then we obtain $E_s = m \cdot l \cdot log(m \cdot l) - \sum_{i=1}^{m} l \cdot log(l) = m \cdot l \cdot log(m \cdot l) - m \cdot l \cdot log(l) = m \cdot l \cdot log(m) > 0$. This means that these cases can be distinguished.

Observation 2 states event logs L_1 and L_2 can have the same number of variants, but different size. Let us assume that L_1 is a duplication of L_2 such that $|L_1| = 2 \cdot |L_2|$. Then, the amount of variants remains the same: $max(C_1) = max(C_2)$. For evaluating the impact on the entropy measure, we calculate the difference $E_s(L_1) - E_s(L_2)$. We assume S refers to L_2, which yields the following equation: $2 \cdot |S| \cdot log(2 \cdot |S|) - \sum_{i=1}^{max(C)} 2 \cdot |seq_i(S)| \cdot log(2 \cdot |seq_i(S)|) - |S| \cdot log(|S|) + \sum_{i=1}^{max(C)} |seq_i(S)| \cdot log(|seq_i(S)|)$, which is equal to $2 \cdot |S| \cdot log(2) + |S| \cdot log(|S|) - \sum_{i=1}^{max(C)} 2 \cdot |seq_i(S)| \cdot log(2) + |seq_i(S)| \cdot log(|seq_i(S)|)$. This is the same as $E_s + 2 \cdot |S| \cdot log(2) - \sum_{i=1}^{max(C)} 2 \cdot |seq_i(S)| \cdot log(2)$ and greater than zero if E_s is greater than one.

Observation 3 states that two logs L_1 and L_2 with the same edit distance between cases can differ in size and in their number of variants. Regarding size, we already demonstrated that duplicating the event log increases entropy if there is more than one partition. If we assume a case of a constant number of states and events in the extended prefix automaton, then it suffices to show that two times the amount of events in one partition ($2 \cdot |seq_i(S)|$) yields a higher entropy than two partitions with $|seq_i(S)|$ events. For these two cases, we have summands in the summation as follows: $(2 \cdot |seq_i(S)|) \cdot log(2 \cdot |seq_i(S)|)$ for less and larger partitions and $2 \cdot (|seq_i(S)| \cdot log(|seq_i(S)|))$ for double the amount of partitions with half the number of events. It is easy to see that

$$(2 \cdot |seq_i(S)|) \cdot log(2 \cdot |seq_i(S)|) > 2 \cdot (|seq_i(S)| \cdot log(|seq_i(S)|)) \tag{42}$$

because

$$2 \cdot |seq_i(S)| \cdot 2 + 2 \cdot |seq_i(S)| \cdot log(|seq_i(S)|) > 2 \cdot (|seq_i(S)| \cdot log(|seq_i(S)|)) \tag{43}$$

yields

$$2 \cdot |seq_i(S)| \cdot 2 > 0 \tag{44}$$

To summarize, we observe that our sequence entropy yields a process complexity measure for event logs that is monotonously growing with events being added no matter where in the event log. In comparison to the other measures presented in Section 2, we observe that the critical Observations 1–3 are well addressed by our entropy-based measure.

4. Evaluation

We have implemented our graph entropy-based process complexity measures as a Python application[2], which also computes all log complexity measures discussed in Section 2. The application receives as input an event log (either in CSV or XES format) and calculates all complexity measures or a user-selected subset of them. Henceforth, for simplicity, we will refer to these measures as *log complexity* measures to distinguish them from the complexity that is associated with process models. We focus our evaluation and the analysis of the different complexity measures on the following two research questions:

RQ1. How does log complexity affect the quality of automatically discovered process models?
RQ2. What log complexity measures could be used as a proxy to predict the quality of automatically discovered process models?

In the following, we describe in detail the analysis that we conducted towards answering the two research questions.

4.1. Dataset and setup

For our experiments, we first selected the collection of 24 event logs that Augusto et al. used for their review [6]. We extended this collection with eight event logs of the Business Process Intelligence Challenges 2019 [47] (three out of eight) and 2020 [48] (five out of eight). Hence, the collection of event logs we use for our experiments includes a total of 32 event logs (20 of which are publicly available and 12 private). To the best of our knowledge, this is the largest collection of real-world event logs used in a process mining study so far, representing an increase of 33.3% from the former largest collection [6]. The public subset of the collection of event logs previously used by Augusto et al. [6] is available for download from the 4TU Research Data Centre [7]. The event logs we used in our experiments record the executions of business processes from a variety of domains, including healthcare, finance, government, and IT service management. They demonstrate an heterogeneous degree of complexity across the different complexity measures. Tables 3–5 show the complexity of each event log, reporting our novel graph entropy-based complexity measures (Table 3) as well as the complexity measures from prior research (Tables 4 and 5). The labels of the state-of-the-art log complexity measures are reported in Table 1 in Section 2, while the labels of our measures are reported in Table 2.

Table 2
Process complexity measure labels.

Process complexity measure	Label
Variant Entropy	var-e
Sequence Entropy	seq-e
Normalised Variant Entropy	nvar-e
Normalised Sequence Entropy	nseq-e

The values[3] shown in Tables 3–5 highlight the heterogeneous nature of the event logs in the collection, with the vast majority of the log complexity measures varying over a large range of values (e.g. *DT(%)* covers the range 0.01% to 97.5%).

Given our interest in understanding how log complexity measures relate to the quality of automatically discovered process models, the second component of our evaluation dataset includes the process models automatically discovered from the 32 event logs using the three most reliable algorithms to date, according to [6]: the Evolutionary Tree Miner (ETM) [9], the Inductive Miner - Infrequent Behavior Variant (IM) [23], and the Split Miner (SM) [3].

Table 6 shows the quality of the process models automatically discovered by ETM, IM, and SM, covering both accuracy (i.e., fitness and precision) and complexity of the process models. The corresponding measures are: *fitness*, *precision*, *F-score* (of fitness and precision), *size*, and *control flow complexity* (CFC). We recall that *fitness* quantifies the amount of behavior contained in the event log that the process model is able to replay. The *precision* measure quantifies the amount of behavior allowed by the process model that can be found in the event log. The F-score of fitness and precision is the product of the two measurements divided by their sum and multiplied by two. Over the past decade, several fitness and precision measures have been proposed [41], each of them suffering from different limitations (from approximation to low scalability). The results shown in Table 6 report the alignment-based fitness and precision measures proposed by Adriansyah et al. [2,1]. The choice of measures was guided by the goal to maintain consistency with the latest benchmark results [6]. The *alignment-based fitness* [2] is calculated as one minus the normalized sum of the minimal alignment cost between each trace in the event log and the closest corresponding trace that can be replayed by the process model. The *alignment-based precision* [1], instead, builds a prefix-automaton of the event log and then replays the process model on top of it, assessing the number of times that the process behavior diverges from the behavior of the prefix-automation. Finally, the *size* of a process model

---

[2] Sources available at https://github.com/MaxVidgof/process-complexity
[3] A dash '-' represents an exception during the calculation of the measure, e.g. due to out-of-memory exception or a timeout (which we set at 12 h).

Table 3

Complexity of the evaluation dataset - Part 1.

log	var-e	seq-e	nvar-e	nseq-e	LOD	granularity
BPIC12	474928.8	1384057.4	0.708	0.423	8.2	0.26
BPIC13cp	3502.3	18231.6	0.705	0.311	2.4	2502013.36
BPIC13inc	88677.4	294092.5	0.718	0.405	2.6	1540.64
BPIC14	1114386.4	2490701.5	0.772	0.526	5.5	139.90
BPIC15f1	20385.6	90549.4	0.652	0.419	24.0	95.79
BPIC15f2	50042.2	120443.3	0.640	0.483	**36.1**	761.23
BPIC15f3	78199.7	231332.5	0.690	0.494	31.8	63.11
BPIC15f4	34433.4	131444.0	0.664	0.434	34.1	0.01
BPIC15f5	39903.1	134870.6	0.661	0.436	30.8	**0.00**
BPIC17	505341.9	3443057.9	0.777	0.358	14.4	**0.00**
BPIC19c1	1152607.5	1776714.2	0.598	0.499	3.7	434462.49
BPIC19c2	56603.5	2475331.5	0.799	0.183	4.3	347055.36
BPIC19c3	1194.1	11332.1	0.633	0.264	2.9	13745.14
BPIC20a	1649.7	101733.0	0.696	0.165	5.2	11984.67
BPIC20b	30722.0	273919.7	0.758	0.339	10.7	1746.48
BPIC20c	96618.4	413564.3	0.734	0.420	10.8	2573.42
BPIC20d	5488.8	56758.6	0.724	0.317	8.5	236.00
BPIC20e	1322.8	73131.6	0.704	0.189	5.2	3163.74
RTFMP	1892.3	831983.5	0.769	0.112	3.7	2174191.14
SEPSIS	40624.5	76528.7	0.696	0.522	9.1	5.56
PRT1	36311.9	236957.6	0.770	0.280	4.5	7689.03
PRT2	255222.9	302150.0	0.638	**0.608**	7.9	0.01
PRT3	31408.0	64170.9	0.656	0.491	8.4	**0.00**
PRT4	305234.0	1035205.9	0.818	0.518	7.1	738.05
PRT5	**4.5**	**2050.0**	**0.270**	**0.055**	6.0	590.90
PRT6	4079.4	19076.6	0.750	0.365	7.5	665.25
PRT7	3013.1	54130.7	0.745	0.341	8.2	0.00
PRT8	38225.6	44186.7	0.547	0.534	6.1	474.93
PRT9	21210.3	3628862.6	0.734	0.139	2.2	154278.43
PRT10	949.6	214831.2	0.805	0.242	**1.6**	**17685299.33**
PRT11	**1207005.2**	**8588051.4**	0.651	0.281	9.0	**0.00**
PRT12	54394.4	541576.6	**0.852**	0.276	4.3	132703.06

(in BPMN format[4]) is equal to the number of nodes composing the process model, while the *CFC* of a (BPMN) process model calculates the amount of branching induced by its split gateways [24].

Starting from the log complexity measurements and the process model quality measurements, we calculated the Pearson correlation and the Kendall correlation [21] for each pair of corresponding measurement series (log complexity, process model quality) and assessed their statistical significance. While the Pearson correlation focuses on the likelihood of an existing linear relation between two measurement series (e.g., between the *magnitude* of the event logs and the *fitness* of the models discovered by IM), the Kendall correlation tells us whether two measurement series exhibit the same rank. The Kendall correlation is particularly useful when two measurement series do not exhibit the same trend (e.g., a linear or an exponential relation), yet they rank the objects of the measurements identically or similarly (to a certain degree, assessed via statistical significance). In our context, the Kendall correlation allows us to understand if a log complexity measure calculated for a set of logs ranks the logs identically or similarly to the rank yielded by a process model quality measure. Furthermore, we conduct a regression analysis to study the potential to predict process model complexity based on log complexity.

We leverage the results of the correlation analysis to collect evidence that allowed us to answer RQ1. Then, we select a subset of the log complexity measures that correlated the most with the quality measures of automatically discovered process models, and we explore if and how they could be used to estimate apriori the quality of the automatically discovered process models (answering RQ2). Finally, we compare the time performance of our novel entropy-based log complexity measures against the existing log complexity measures. All the experiments were run on an Intel Core i7-8565U@1.80 GHz with 32 GB RAM running Windows 10 Pro (64-bit) and Python 3.8.5 with no RAM limitation.

4.2. Results of correlation analysis

Tables 7 and 8, respectively, show the Kendall and Pearson correlations values of each pair of log complexity measure and process model quality measure (for a total of 300 pairs). We highlight in bold and underline the correlation values that yield a p-value less than or equal to 0.01 and a p-value less than or equal to 0.05. Both thresholds are traditionally used to identify statistical significance.

We note that both the Pearson and Kendall correlations yield similar significance values, identifying 67 (Pearson) and 78 (Kendall) pairs of correlation measures with a p-value below 0.01. The rate of overlap between the identified pairs is 75% (54

Table 4
Complexity of the evaluation dataset - Part 2.

log	structure	affinity	dev-random	avg-dist.	LZ	t-comp	magnitude
BPIC12	0.783	0.258	0.755	22.1	41378	173248	262200
BPIC13cp	0.375	0.560	0.545	2.7	912	1715	6660
BPIC13inc	0.313	0.644	0.577	6.7	6527	28280	65533
BPIC14	0.580	**0.142**	0.842	8.3	62106	224591	369485
BPIC15f1	0.975	0.229	0.880	20.6	3790	10367	21656
BPIC15f2	0.971	0.208	**0.903**	29.8	4496	17820	24678
BPIC15f3	0.951	0.216	0.895	23.3	7413	28553	43786
BPIC15f4	0.964	0.243	0.894	24.7	4798	16502	29403
BPIC15f5	0.972	0.224	0.894	26.6	4731	18789	30030
BPIC17	0.904	0.689	0.734	13.8	84060	**283587**	714198
BPIC19c1	0.240	0.411	0.481	23.8	18367	221868	283407
BPIC19c2	0.344	0.394	0.647	2.3	53117	21295	979942
BPIC19c3	0.375	0.466	0.580	3.5	**593**	625	5038
BPIC20a	0.865	0.541	0.617	1.3	7947	734	56437
BPIC20b	0.830	0.406	0.770	5.1	13401	9582	72151
BPIC20c	0.787	0.303	0.800	8.2	16519	25074	86581
BPIC20d	0.806	0.395	0.740	3.4	3932	1937	18246
BPIC20e	0.878	0.515	0.622	1.4	5596	618	36796
RTFMP	0.421	0.367	0.602	8.6	38212	1166	561470
SEPSIS	0.551	0.256	0.787	11.8	3115	12956	15214
PRT1	0.543	0.566	0.647	2.4	12388	13419	75353
PRT2	**0.025**	0.294	0.821	36.5	8826	45864	46282
PRT3	0.742	0.246	0.763	3.2	1541	13650	13720
PRT4	0.694	0.253	0.795	5.2	32584	68432	166282
PRT5	0.806	**0.714**	0.636	**0.7**	904	**12**	**4434**
PRT6	0.728	0.399	0.747	2.9	1471	1691	6011
PRT7	0.775	0.323	0.762	2.5	4183	1168	16353
PRT8	0.859	0.155	0.831	**51.7**	2846	8950	9086
PRT9	0.063	0.482	**0.437**	2.0	141857	7278	1808706
PRT10	0.798	0.142	0.626	1.6	6976	356	78864
PRT11	**0.982**	–	0.839	18.2	**165919**	278322	**2099835**
PRT12	0.355	0.236	0.730	2.3	28888	17397	163224

pairs). As expected, we see that the Kendall correlation is looser than the Pearson correlation. If we consider pairs that correlate with a p-value at 0.05, Kendall identifies 119 pairs while Pearson identifies 99 (17% less), with an overlap of 83% (90 pairs). Table 9 shows the overlap of statistical significance and the corresponding p-values. A pair of measures that correlate negatively is identified with the dash symbol between brackets. To reinforce the statistical significance, in the following, we refer only to the results reported in Table 9. The correlation *summary* reported in Table 9 can be used to analyze which log complexity measures affect the process model quality measures the most and vice versa. This analysis allows us to answer RQ1.

Focusing on the quality measures of the automatically discovered process models by ETM (columns 2 to 6, Table 9), we notice that only one of the log complexity measures correlates with the precision and the CFC of ETM's process models (with a p-value at 0.05). This highlights that both the precision and the CFC of the process models discovered by ETM are not affected by the complexity of the input log. A similar observation can be made for the fitness of the process models discovered by IM. By design, IM always strives to discover a highly fitting process model [23]. Consequently, the overall complexity of the input log does not influence the fitness of IM: no correlation exists between the log complexity measures and the fitness of IM's process models. Finally, also the CFC of the models discovered by SM appears to be resistant to the log complexity, correlating with only three complexity measures: *affinity*, *avg-dist*, and *TL-avg*.

Next, we look at the process model quality measures that correlate the most with the log complexity measures. We find that the fitness and size of the models discovered by ETM correlate both with 9 log complexity measures; the precision, F-score, size, and CFC of the models discovered by IM correlate with 8 or 9 log complexity measures; and the fitness, F-score, and size of the models discovered by SM correlate with 7, 11, and 7 log complexity measures, respectively. We note that when the log complexity measures correlate with the process model accuracy measures (fitness, precision, and their F-score), they almost always (48 out of 52 times) show a negative correlation. There is only one positive correlation recorded for *affinity*, which measures the overlapping directly-follows relations observed in two event sequences. This shows that some features of the process behavior – captured via the log complexity measures – may have a negative impact on the accuracy of the automatically discovered process models.

Turning our attention to the log complexity measures, we note that 40% of them (8 out of 20) do not correlate with any process model quality measure. These log complexity measures are *seq-e*, *nvar-e*, *granularity*, *lempel–ziv*, *magnitude*, *support*, *DT (#)*, and *TL-min*. On the other hand, although, there is no log complexity measure that consistently affects all the quality measures of all the automatically discovered process models, we can identify a subset that correlates with the majority. This subset includes *nseq-e*, *affinity*, *dev-random*, *variety*, *DT (%)*, *avg-dist*, and *TL-avg*. The latter two correlate with 12 out of 15

Table 5
Complexity of the evaluation dataset - Part 3.

log	support	variety	DT (%)	DT (#)	TL-min	TL-avg	TL-max
BPIC12	13087	24	33.4	4371	3	20	175
BPIC13cp	1487	**4**	12.3	183	**1**	4	35
BPIC13inc	7554	**4**	20.0	1511	**1**	9	123
BPIC14	41353	9	36.1	**14928**	3	9	167
BPIC15f1	902	70	32.7	295	5	24	50
BPIC15f2	681	82	61.7	420	4	36	63
BPIC15f3	1369	62	60.3	826	4	32	54
BPIC15f4	860	65	52.4	451	5	34	54
BPIC15f5	975	74	45.7	446	4	31	61
BPIC17	21861	18	40.1	8766	11	33	113
BPIC19c1	15129	5	20.9	3159	**1**	19	794
BPIC19c2	220810	8	1.2	2706	**1**	4	179
BPIC19c3	1027	**4**	10.1	104	2	5	19
BPIC20a	10500	17	0.9	99	**1**	5	24
BPIC20b	6449	34	11.7	753	3	11	7
BPIC20c	7065	51	20.9	1478	3	12	90
BPIC20d	2099	29	9.6	202	**1**	9	21
BPIC20e	6886	19	1.3	89	**1**	5	20
RTFMP	150370	11	0.2	301	2	4	20
SEPSIS	1050	16	80.6	846	3	14	185
PRT1	12720	9	8.1	1030	2	5	64
PRT2	1182	9	97.5	1152	**12**	39	276
PRT3	1600	15	19.9	318	6	8	9
PRT4	20000	11	29.7	5940	6	8	36
PRT5	739	6	0.10	**1**	6	6	**6**
PRT6	744	9	22.4	167	7	8	21
PRT7	2000	13	6.4	128	8	8	11
PRT8	**225**	55	**99.9**	225	2	**40**	350
PRT9	**787657**	8	**0.01**	79	**1**	2	58
PRT10	43514	19	**0.01**	4	**1**	**1**	15
PRT11	174842	**310**	3.0	5245	2	12	**804**
PRT12	37345	20	7.5	2801	**1**	4	27

quality measures, which makes them the log complexity measures affecting the most the quality of the discovered process models. It is worth noting that while the total number of distinct traces (DT(#)) does not correlate with any process model quality measure, the percentage of distinct traces (DT(%)) correlates with more than 50% of them (8 out of 15).

Considering only the measures that correlate the most and selecting only the correlations with a p-value of 0.01, we are left with the results shown in Table 10.

4.3. Results of regression analysis

We now want to investigate to what extent log complexity measures could be used to predict a process model quality measure. To this end, for each log complexity measure that correlates with a process model quality measure in Table 10, we assessed the *residual errors* (minimum, median, and maximum) and the coefficient of determination (R^2) of the corresponding linear regression model, where the log complexity measure is the *predictor variable* and the process model quality measure is the *outcome variable*. The linear regression models that exhibit the highest R^2 should be preferred, since R^2 is defined as $1 - \frac{SS_r}{SS_t}$, where SS_r is *the residual sum of squares* and SS_t is *the total sum of squares*. In general, the higher R^2 the more accurate is the linear regression model. The results of this analysis are reported in Table 11. Looking at the process model accuracy measures (fitness$_{ETM}$, precision$_{IM}$, F-score$_{IM}$, fitness$_{SM}$, and F-score$_{SM}$), we note that the best complexity measures to predict them are *avg-dist* (for ETM and IM) and *nseq-e* (for SM). This shows that the accuracy of the process models discovered by ETM, IM, and SM is closely affected by the variation of process behavior recorded in the event logs, rather than the absolute amount of process behavior. Therefore, automatically discovering a process model is likely to be more challenging when the event log records a small amount of process behavior that varies greatly than when the event log records a huge amount of process behavior that varies little. Looking at the model complexity measures Size$_{ETM}$, CFC$_{ETM}$, CFC$_{IM}$, and Size$_{SM}$, one log complexity measure seems to highly affect them all, which is *variety*.

It is possible to generate more accurate regression models by using *uncorrelated* multi-predictors or non-linear regression models (e.g., generalized additive models). Ideally, having an accurate regression model to estimate the quality of the automatically discovered process models could save considerable time for process analysts, especially if an optimization is required during process discovery [6,8]. In fact, one could select the automated process discovery algorithm to be used based on the results of the regression model predictions. However, the design of such an accurate regression model and its systematic evaluation would deserve a separate study, a very different dataset (e.g., a large collection of event logs, even artificial), and a different type of evaluation, as seen in previous studies [26]. We also note that seminal studies in that area have been

Table 6
Quality measurements of the process models automatically discovered by ETM, IM, and SM.

Log	ETM					IM					SM				
	Fit.	Prec.	F-score	Size	CFC	Fit.	Prec.	F-score	Size	CFC	Fit.	Prec.	F-score	Size	CFC
BPIC12	0.33	0.98	0.49	69	10	0.98	0.50	0.66	59	37	0.96	0.81	0.88	51	41
BPIC13cp	0.99	0.76	0.86	11	17	**0.82**	**1.00**	0.90	**9**	4	0.99	0.94	0.96	11	6
BPIC13inc	0.84	0.80	0.82	28	24	0.92	0.56	0.70	13	7	0.98	0.92	0.95	12	8
BPIC14	0.68	0.94	0.79	22	15	0.89	0.65	0.75	31	18	0.77	0.93	0.84	20	14
BPIC15f1	0.57	0.89	0.69	73	21	0.97	0.57	0.71	164	108	0.90	0.90	0.90	111	45
BPIC15f2	0.62	0.90	0.73	78	19	0.93	0.56	0.70	193	123	0.77	0.91	0.83	126	45
BPIC15f3	0.66	0.88	0.75	78	26	0.95	0.55	0.70	159	108	0.79	0.93	0.85	94	33
BPIC15f4	0.66	0.95	0.78	74	17	0.96	0.59	0.73	162	111	**0.73**	0.90	0.81	100	35
BPIC15f5	0.58	0.89	0.70	82	26	0.94	**0.18**	**0.30**	134	95	0.79	0.94	0.86	106	34
BPIC17	0.72	**1.00**	0.84	31	5	0.98	0.70	0.82	35	20	0.96	0.85	0.90	31	18
BPIC19c1	0.30	1.00	0.46	33	4	0.97	0.37	0.54	19	11	0.85	**0.61**	**0.71**	17	9
BPIC19c2	0.87	0.99	0.93	25	13	0.93	0.69	0.79	20	9	0.92	1.00	0.96	28	19
BPIC19c3	0.78	0.77	0.78	21	9	**1.00**	0.77	0.87	13	7	0.89	0.98	0.93	12	6
BPIC20a	0.88	**0.43**	0.58	23	10	0.95	0.86	0.90	34	19	0.97	0.78	0.87	33	16
BPIC20b	0.77	0.55	0.64	55	25	0.88	0.55	0.68	88	49	0.96	0.84	0.90	70	43
BPIC20c	0.67	0.92	0.77	68	32	0.74	0.32	0.44	100	62	0.97	0.78	0.86	146	113
BPIC20d	0.80	0.98	0.88	44	19	0.91	0.44	0.60	66	37	0.95	0.86	0.90	58	31
BPIC20e	0.89	0.88	0.88	35	16	0.92	0.87	0.90	38	21	**1.00**	0.87	0.93	37	19
RTFMP	0.79	0.98	0.87	46	33	0.99	0.70	0.82	34	20	**1.00**	0.97	0.98	22	17
SEPSIS	0.71	0.84	0.77	30	15	0.99	0.45	0.62	50	32	0.76	0.86	0.81	33	23
PRT1	0.99	0.81	0.89	23	12	0.90	0.67	0.77	20	9	0.98	0.99	0.98	29	18
PRT2	0.57	0.94	0.71	**86**	21	-	-	-	45	33	0.81	0.74	0.77	40	30
PRT3	0.98	0.86	0.92	51	37	0.98	0.68	0.80	37	20	0.83	0.91	0.87	32	17
PRT4	0.84	0.85	0.84	64	28	0.93	0.75	0.83	27	13	0.87	0.99	0.93	34	21
PRT5	**1.00**	**1.00**	**1.00**	**10**	**1**	**1.00**	**1.00**	**1.00**	10	**1**	**1.00**	**1.00**	**1.00**	**10**	**1**
PRT6	0.98	0.80	0.88	41	16	0.99	0.82	0.90	23	10	0.94	**1.00**	0.97	16	5
PRT7	0.90	0.81	0.85	60	29	**1.00**	0.73	0.84	29	13	0.91	**1.00**	0.95	30	11
PRT8	0.35	0.88	0.50	75	12	0.98	0.33	0.49	111	92	0.97	0.67	0.79	406	488
PRT9	0.75	0.49	0.59	27	13	0.90	0.61	0.73	28	16	0.92	**1.00**	0.96	30	20
PRT10	**1.00**	0.63	0.77	61	**45**	0.96	0.79	0.87	41	29	0.97	0.97	0.97	79	68
PRT11	**0.10**	**1.00**	**0.18**	21	3	-	-	-	**549**	**365**	-	-	-	**712**	**609**
PRT12	0.63	**1.00**	0.77	21	8	**1.00**	0.77	0.87	32	25	0.96	0.99	0.97	97	84

Table 7
Kendall correlation values. Bold and underlined values respectively highlight correlations at $p < 0.01$ and $p < 0.05$.

Log Comp. Measure	ETM					IM					SM				
	Fit.	Prec.	F-score	Size	CFC	Fit.	Prec.	F-score	Size	CFC	Fit.	Prec.	F-score	Size	CFC
var-e	**−0.41**	<u>0.34</u>	−0.16	0.16	−0.08	−0.17	**−0.40**	**−0.40**	0.07	0.12	<u>−0.29</u>	−0.23	**−0.37**	0.11	0.16
seq-e	−0.25	0.24	−0.16	0.00	−0.01	<u>−0.26</u>	−0.18	−0.19	−0.01	0.03	−0.10	−0.02	−0.06	0.03	0.16
nvar-e	0.24	−0.03	0.19	−0.16	0.13	−0.18	0.22	0.20	−0.17	−0.15	0.10	0.21	**0.38**	−0.07	0.05
nseq-e	**−0.39**	0.10	−0.25	**0.37**	0.13	−0.04	**−0.43**	**−0.43**	<u>0.28</u>	<u>0.30</u>	**−0.48**	<u>−0.29</u>	**−0.61**	<u>0.26</u>	0.16
LOD	<u>−0.31</u>	0.20	−0.09	**0.46**	0.15	0.00	<u>−0.32</u>	<u>−0.29</u>	**0.59**	**0.56**	**−0.37**	−0.19	**−0.43**	**0.45**	<u>0.32</u>
granularity	0.19	−0.17	−0.02	<u>−0.32</u>	−0.10	−0.18	0.22	0.19	<u>−0.30</u>	<u>−0.27</u>	<u>0.32</u>	0.09	<u>0.32</u>	−0.19	−0.07
structure	−0.17	0.08	−0.12	**0.46**	0.17	−0.01	−0.10	−0.09	**0.60**	**0.54**	−0.06	−0.23	<u>−0.27</u>	**0.49**	<u>0.31</u>
affinity	**0.34**	−0.10	<u>0.26</u>	**−0.52**	<u>−0.29</u>	−0.04	<u>0.26</u>	0.25	**−0.46**	**−0.49**	**0.42**	0.05	**0.34**	**−0.49**	**−0.47**
dev-random	<u>−0.32</u>	0.16	−0.12	**0.56**	0.24	−0.03	**−0.34**	<u>−0.32</u>	**0.60**	**0.59**	**−0.44**	−0.11	**−0.44**	**0.50**	**0.39**
avg-dist.	**−0.62**	0.25	**−0.37**	**0.50**	0.08	−0.03	**−0.55**	**−0.55**	**0.40**	**0.43**	**−0.39**	**−0.36**	**−0.55**	<u>0.31</u>	<u>0.27</u>
LZ	−0.17	0.20	−0.14	−0.03	−0.03	−0.25	−0.11	−0.12	−0.02	0.02	−0.02	−0.04	0.03	0.01	0.14
t-comp	**−0.41**	<u>0.33</u>	−0.13	0.17	−0.03	−0.17	**−0.36**	**−0.36**	0.10	0.15	<u>−0.32</u>	−0.18	**−0.34**	0.11	0.14
magnitude	−0.16	0.20	−0.13	−0.02	0.00	−0.25	−0.11	−0.12	−0.05	−0.02	−0.02	−0.03	0.04	0.00	0.15
support	0.07	0.04	0.02	−0.24	0.00	−0.15	0.08	0.06	−0.24	−0.21	0.16	0.08	0.24	−0.17	−0.01
variety	**−0.39**	0.13	<u>−0.29</u>	**0.61**	0.23	−0.05	<u>−0.31</u>	<u>−0.29</u>	**0.87**	**0.86**	−0.19	<u>−0.31</u>	**−0.37**	**0.81**	**0.63**
DT (%)	**−0.37**	0.17	−0.18	**0.36**	0.02	0.08	**−0.33**	<u>−0.33</u>	<u>0.30</u>	<u>0.32</u>	**−0.38**	−0.19	**−0.43**	0.24	0.18
DT (#)	<u>−0.30</u>	**0.35**	−0.05	0.06	−0.06	−0.13	<u>−0.30</u>	<u>−0.30</u>	0.03	0.07	−0.24	−0.18	−0.25	0.03	0.10
TL-min	−0.08	0.10	0.12	0.27	0.16	0.27	−0.01	0.02	0.22	0.18	−0.26	0.10	−0.09	0.08	−0.03
TL-avg	**−0.50**	0.23	<u>−0.30</u>	**0.48**	−0.01	0.00	**−0.48**	**−0.45**	**0.47**	**0.46**	**−0.35**	**−0.40**	**−0.61**	0.37	<u>0.26</u>
TL-max	**−0.41**	0.25	<u>−0.28</u>	0.09	−0.25	−0.18	**−0.41**	**−0.42**	0.04	0.09	−0.21	<u>−0.28</u>	**−0.39**	0.11	0.16

Table 8

Pearson correlation values. Bold and underlined values respectively highlight correlations at $p < 0.01$ and $p < 0.05$.

Log Comp.	ETM					IM					SM				
Measure	Fit.	Prec.	F-score	Size	CFC	Fit.	Prec.	F-score	Size	CFC	Fit.	Prec.	F-score	Size	CFC
var-e	**−0.48**	0.33	−0.34	−0.13	−0.30	−0.02	−0.21	−0.19	−0.18	−0.17	−0.27	−0.42	**−0.50**	−0.17	−0.10
seq-e	−0.21	0.07	−0.17	−0.27	−0.31	−0.07	−0.05	−0.01	−0.27	−0.28	−0.02	−0.01	−0.02	−0.23	−0.14
nvar-e	0.13	−0.17	0.05	0.03	0.36	−0.24	−0.01	0.04	−0.14	−0.16	0.04	0.23	0.21	−0.19	−0.18
nseq-e	**−0.52**	0.26	−0.32	**0.53**	0.17	−0.06	**−0.59**	**−0.56**	0.45	**0.47**	**−0.64**	−0.41	**−0.71**	0.37	0.28
LOD	−0.32	0.22	−0.09	**0.69**	0.13	0.03	−0.39	−0.37	**0.90**	**0.88**	**−0.65**	−0.02	−0.42	0.29	−0.01
granularity	0.27	−0.26	0.03	0.08	**0.50**	0.02	0.20	0.19	−0.11	−0.08	0.19	0.15	0.23	0.01	0.03
structure	−0.07	0.10	0.02	**0.65**	0.23	0.08	−0.15	−0.15	**0.65**	**0.63**	−0.19	−0.17	−0.25	0.39	0.20
affinity	0.43	−0.15	0.30	**−0.68**	**−0.46**	−0.08	0.46	0.41	**−0.54**	**−0.57**	**0.56**	0.09	0.43	**−0.49**	−0.37
dev-random	−0.30	0.33	0.00	**0.71**	0.24	0.04	−0.41	−0.37	**0.76**	**0.75**	**−0.55**	−0.04	−0.38	**0.47**	0.26
avg-dist.	**−0.79**	0.31	**−0.58**	**0.64**	−0.11	0.16	**−0.65**	**−0.63**	**0.69**	**0.75**	−0.39	**−0.56**	**−0.71**	**0.76**	**0.66**
LZ	−0.13	−0.10	−0.17	−0.24	−0.22	−0.11	0.00	0.04	−0.24	−0.24	0.06	0.14	0.14	−0.19	−0.11
t-comp	−0.46	0.37	−0.28	−0.09	−0.36	0.05	−0.18	−0.14	−0.15	−0.14	−0.16	−0.38	−0.40	−0.16	−0.11
magnitude	−0.05	−0.14	−0.13	−0.26	−0.19	−0.08	0.00	0.04	−0.25	−0.25	0.09	0.17	0.19	−0.19	−0.12
support	0.04	−0.36	−0.16	−0.21	−0.07	−0.12	0.02	0.04	−0.19	−0.18	0.08	0.26	0.25	−0.13	−0.08
variety	**−0.47**	0.15	−0.29	**0.81**	0.21	−0.10	**−0.56**	**−0.56**	**0.98**	**0.98**	−0.44	−0.23	−0.46	**0.64**	0.36
DT (%)	**−0.48**	0.23	−0.27	**0.49**	−0.06	0.22	−0.45	−0.41	**0.58**	**0.63**	**−0.55**	−0.26	**−0.56**	**0.60**	**0.52**
DT (#)	−0.22	0.32	−0.02	−0.18	−0.22	−0.09	−0.02	0.00	−0.19	−0.19	−0.22	−0.06	−0.18	−0.17	−0.10
TL-min	0.10	0.25	0.29	0.26	0.03	0.31	0.09	0.14	0.14	0.11	−0.22	0.18	0.01	−0.07	−0.14
TL-avg	**−0.66**	0.34	−0.40	**0.66**	−0.12	0.15	**−0.58**	**−0.55**	**0.78**	**0.81**	**−0.48**	−0.45	**−0.66**	**0.62**	0.44
TL-max	**−0.65**	0.29	**−0.56**	−0.03	−0.37	0.08	−0.46	−0.44	−0.06	−0.02	−0.16	**−0.71**	**−0.67**	0.19	0.27

Table 9

Overlapping of Pearson and Kendall statically significant correlated measures (at .01 or .05), negative correlation is highlighted with the symbol (-).

Log Comp.	ETM					IM					SM				
Measure	Fit.	Prec.	F-score	Size	CFC	Fit.	Prec.	F-score	Size	CFC	Fit.	Prec.	F-score	Size	CFC
var-e	.01(-)												.01(-)		
seq-e															
nvar-e															
nseq-e	.01(-)			.01			.01(-)	.01(-)	.05	.05	.01(-)	.05(-)	.01(-)	.05	
LOD				.01			.05(-)	.05(-)	.01	.01	.01(-)		.05(-)		
granularity															
structure				.01					.01	.01				.05	
affinity	.05			.01(-)	.05(-)		.05		.01(-)	.01(-)	.01		.05	.01(-)	.05(-)
dev-random				.01			.05(-)	.05(-)	.01	.01	.01(-)		.05(-)	.01	
avg-dist.	.01(-)		.01(-)	.01			.01(-)	.01(-)	.01	.01	.05(-)	.01(-)	.01(-)	.05	.05
LZ															
t-comp	.05(-)	.05											.05(-)		
magnitude															
support															
variety	.01(-)			.01			.05(-)	.05(-)	.01	.01			.05(-)	.01	
DT (%)	.01(-)			.01			.05(-)	.05(-)	.05	.05	.01(-)		.01(-)		
DT (#)															
TL-min															
TL-avg	.01(-)		.05(-)	.01			.01(-)	.01(-)	.01	.01	.01(-)	.05(-)	.01(-)	.01	.05
TL-max	.01(-)		.05(-)				.05(-)	.05(-)				.05(-)	.01(-)		

Table 10

Log complexity measures and process model quality measures that correlate the most, with emphasis on correlations with $p < 0.01$.

Log Comp.	ETM		IM				SM		
Measure	Fit.	Size	Prec.	F-score	Size	CFC	Fit.	F-score	Size
nseq-e	.01(-)	.01	.01(-)	.01(-)			.01(-)	.01(-)	
affinity		.01(-)			.01(-)	.01(-)	.01		.01(-)
dev-random		.01			.01	.01	.01(-)		.01
avg-dist.	.01(-)	.01	.01(-)	.01(-)	.01	.01		.01(-)	
variety	.01(-)	.01			.01	.01			.01
DT (%)	.01(-)	.01					.01(-)	.01(-)	
TL-avg	.01(-)	.01	.01(-)	.01(-)	.01	.01	.01(-)	.01(-)	.01

Table 11
Results of the regression analysis.

Outcome Variable	Predictor Variable	Residual Errors			Predictor		
		min	median	max	coeff.	p-value	R^2
	nseq-e	−0.368	**0.005**	0.333	−0.747	0.003	0.275
	avg-dist.	**−0.284**	0.014	**0.135**	−0.013	**0.000**	**0.632**
ETM Fitness	variety	−0.533	0.040	0.223	−0.004	0.008	0.223
	DT (%)	−0.530	0.022	0.229	−0.003	0.007	0.234
	TL-avg	−0.398	0.024	0.180	−0.011	**0.000**	0.435
	nseq-e	−38.617	**1.017**	29.276	87.706	0.003	0.282
	affinity	−44.328	1.087	**23.366**	−93.835	**0.000**	0.463
	dev-random	−38.982	2.470	27.335	126.467	**0.000**	0.507
ETM Size	avg-dist.	−28.108	2.151	26.790	1.212	**0.000**	0.410
	variety	**−20.198**	−2.628	29.969	0.767	**0.000**	**0.658**
	DT (%)	−39.073	−1.568	31.732	0.408	0.006	0.239
	TL-avg	−38.768	1.606	31.796	1.268	**0.000**	0.432
	nseq-e	−0.383	0.030	0.331	−0.841	0.001	0.351
IM Precision	avg-dist.	−0.344	0.041	**0.280**	−0.011	**0.000**	**0.423**
	TL-avg	**−0.330**	**0.012**	0.292	−0.010	0.001	0.331
	nseq-e	−0.387	0.042	0.195	−0.636	0.001	0.316
IM F-score	avg-dist.	−0.318	0.047	**0.177**	−0.008	**0.000**	**0.401**
	TL-avg	**−0.316**	**0.028**	0.219	−0.007	0.002	0.300
	affinity	−67.971	−7.348	105.713	−177.023	0.002	0.295
	dev-random	−67.632	2.120	75.020	317.187	**0.000**	0.570
IM Size	avg-dist.	−80.604	−8.423	74.846	3.090	**0.000**	0.477
	variety	**−31.157**	0.109	20.143	2.192	**0.000**	**0.963**
	TL-avg	−93.232	−4.092	67.909	3.571	**0.000**	0.614
	affinity	−48.798	−4.414	64.821	−130.587	0.001	0.327
	dev-random	−46.816	**0.205**	44.763	220.020	**0.000**	0.558
IM CFC	avg-dist.	−57.225	−6.559	47.295	2.353	**0.000**	0.562
	variety	**−16.869**	−1.123	**14.600**	1.539	**0.000**	**0.965**
	TL-avg	−67.689	−3.600	43.697	2.598	**0.000**	0.661
	nseq-e	−0.147	0.010	0.132	−0.387	**0.000**	**0.405**
	affinity	−0.144	**0.008**	0.125	0.284	0.001	0.313
SM Fitness	dev-random	−0.141	0.023	**0.102**	−0.361	0.002	0.305
	DT (%)	**−0.127**	0.013	0.194	−0.002	0.001	0.308
	TL-avg	−0.155	0.018	0.151	−0.003	0.007	0.231
	nseq-e	**−0.137**	−0.002	0.089	−0.357	**0.000**	**0.511**
SM F-score	avg-dist.	**−0.137**	0.014	0.071	−0.004	**0.000**	0.497
	DT (%)	−0.221	**0.001**	**0.070**	−0.001	0.001	0.315
	TL-avg	−0.170	0.015	0.074	−0.004	**0.000**	0.438
	affinity	−93.412	**−0.441**	295.530	−226.293	0.006	0.243
SM Size	dev-random	−77.411	−11.069	311.668	279.926	0.008	0.224
	variety	**−54.125**	−9.066	284.199	2.017	**0.000**	**0.411**
	TL-avg	−108.224	−9.111	**239.024**	3.965	**0.000**	0.381

conducted by Riberio et al. [36,35], however, neither taking into account the log complexity measures nor the novel automated process discovery algorithms designed in the past five years (including, Fodina [50] and Split Miner [3]). Furthermore, Riberio et al. propose a black-box prediction system, which does not focus on the connection between log complexity measures and the quality of the discovered process model.

4.4. Results of computational performance

Finally, we compared the computational performance of all the log complexity measures, assessing their average, maximum, minimum, and median execution times over the 32 event logs. The results are shown in Table 12. We did not report the execution times of derived measures (e.g., the percentage of distinct traces, which is derived from the total number of distinct traces and support), or of ratios (e.g., our normalized graph entropy measures). We note that all the complexity measures have a median execution time below one second, with the exception of *affinity* with a median execution time of 9.6 seconds. Indeed, *affinity* was the slowest measure to be calculated, followed by our graph entropy-based complexity measures and *avg-dist*. However, while *affinity* has an average execution time well above a minute, our measures and *avg-dist* have an average execution time in the order of seconds, which is reasonable given that these measures are not designed to work in a real-time context.

4.5. Threats to validity

The findings reported in this study should be interpreted taking into account the size and variety of the 32 event logs of our dataset. Although the event logs are a good approximation of event logs that one may find in real-world scenarios, both

Table 12
Execution times (in seconds) of the complexity measures.

	var-e	seq-e	LOD	granularity	structure	affinity	dev-random	
AVG	22.1	21.7	0.2	0.2	0.6	148.4	0.8	
MAX	397.5	384.2	2.2	1.8	5.8	2020.1	10.7	
MIN	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	
MEDIAN	0.1	0.1	< 0.1	< 0.1	0.2	9.6	0.1	
	avg-dist.	LZ	t-comp	magnitude	support	variety	DT (%)	TL-min
AVG	18.1	0.6	< 0.1	< 0.1	< 0.1	0.2	0.2	0.1
MAX	176.0	8.4	< 0.1	< 0.1	< 0.1	2.0	1.8	0.9
MIN	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
MEDIAN	0.6	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1

in terms of complexity and variety of domains, they cannot possibly synthesize all possible business processes observable in an industrial setting. It is interesting to note that the original submission of this study's manuscript included only 24 event logs [7] and that the additional 8 event logs were added following the reviewers' suggestions. Nonetheless, the changes in the statistical analysis were minor. For instance, the total number of pairs of measures that correlated according to both Pearson and Kendall correlations increased from 84 to 90 (see Table 9). Even more importantly, the seven log complexity measures that correlated the most did not change (Table 10), nor did the most robust predictors (Table 11).

Another threat to the validity of our evaluation is the selection of the quality measures of automatically discovered process models, which in the past few years have been under scrutiny [42,41] in the process mining research community. However, these research studies showed that no existing precision measure is ideal, while newly designed ones are either approximate [4] or computationally inefficient [32,34]. Although one may argue that one precision measure is better than another, we note that the choice of the precision measure does not affect the results much. The findings are reliable and accurate in light of the quality measures we used, which remain the most popular as of today.

5. Conclusions

With this paper, we provide two major contributions to measuring process complexity and to assessing process mining algorithms. First, we analyzed existing measures for process complexity that are based on event logs. Each of these measures emphasizes different complexity criteria including size, variation, and distance. We defined new measures of process complexity based on graph entropy, which capture all three concerns of process complexity by adhering to monotonicity. Second, we evaluated the identified set of process complexity measures, including our novel measures, using a benchmark collection of event logs and their corresponding automatically discovered process models. The goal of our evaluation was to investigate which empirical connections hold between the process complexity measures and the quality of discovered models. Our results show that many process complexity measures (including our novel measure) correlate with the quality of the discovered process models and that it is possible to use process complexity measures as predictors for the quality of process models discovered with state-of-the-art process discovery algorithms.

The findings we reported in this paper are important for process mining research, as they highlight that not only algorithms but also empirical connections between input data complexity and output quality should be investigated. Our results demonstrate the potential to examine the concept of process complexity and its corresponding measures in connection with automated process discovery and there are various opportunities to extend this approach to related research problems. Additional aspects of event log data, such as data complexity, could be used to study connections with further output parameters, such as the process model's usefulness perceived by analysts.

CRediT authorship contribution statement

Adriano Augusto: Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Supervision. **Jan Mendling:** Conceptualization, Investigation, Formal analysis, Validation, Writing - original draft, Writing - review & editing. **Maxim Vidgof:** Conceptualization, Investigation, Software, Validation, Writing - original draft, Writing - review & editing. **Bastian Wurm:** Conceptualization, Investigation, Writing - original draft, Writing - review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

References

[1] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. van Dongen, W. van der Aalst, Measuring precision of modeled behavior, ISeB 13 (1) (2015).
[2] Arya Adriansyah, Boudewijn F van Dongen, and Wil MP van der Aalst. Conformance checking using cost-based fitness analysis. In IEEE International on Enterprise Distributed Object Computing Conference (EDOC), pages 55–64. IEEE, 2011..
[3] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, A. Polyvyanyy, Split miner: automated discovery of accurate and simple business process models from event logs, Knowledge and Information Systems (2018).
[4] A. Augusto, R. Conforti, A. Armas-Cervantes, M. Dumas, and M. La Rosa. Measuring fitness and precision of automatically discovered process models: A principled and scalable approach. IEEE Transactions on Knowledge and Data Engineering, 2020, early access. doi: 10.1109/TKDE.2020.3003258..
[5] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Giorgio Bruno, Automated discovery of structured process models from event logs: The discover-and-structure approach, Data & Knowledge Engineering 117 (2018) 373–392.
[6] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, Andrea Marrella, Massimo Mecella, and Allar Soo. Automated discovery of process models from event logs: Review and benchmark. IEEE Transactions on Knowledge and Data Engineering, 31 (4): 686–705, 2019..
[7] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, Andrea Marrella, Massimo Mecella, and Allar Soo. Evaluation data-set of the paper titled: Automated discovery of process models from event logs: Review and benchmark, 2019. doi: 10.4121/uuid: adc42403-9a38-48dc-9f0a-a0a49bfb6371..
[8] Adriano Augusto, Marlon Dumas, Marcello La Rosa, Metaheuristic optimization for automated business process discovery, in: International Conf. on Business Process Management, Springer, 2019, pp. 268–285.
[9] J. Buijs, B. van Dongen, W. van der Aalst, On the role of fitness, precision, generalization and simplicity in process discovery, in: CoopIS, Springer, 2012.
[10] Alessio Cecconi, Giuseppe De Giacomo, Claudio Di Ciccio, Fabrizio Maria Maggi, and Jan Mendling. Measuring the interestingness of temporal logic behavioral specifications in process mining. Information Systems, page 101920, 2021..
[11] Hsin-Jung Cheng, Akhil Kumar, Process mining on noisy logs–can log sanitization help to improve performance?, Decision Support Systems 79 (2015) 138–149
[12] Benjamin Cornwell, Social sequence analysis: Methods and applications, volume 37, Cambridge University Press, 2015.
[13] Matthias Dehmer, Abbe Mowshowitz, A history of graph entropy measures, Information Sciences 181 (1) (2011) 57–78.
[14] Kiarash Diba, Kimon Batoulis, Matthias Weidlich, Mathias Weske, Extraction, correlation, and abstraction of event data for process mining, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 10 (3) (2020) e1346.
[15] Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A. Reijers, Fundamentals of Business Process Management, Second Edition., Springer, 2018, ISBN 978-3-662-56508-7.
[16] Kenneth T Goh, Brian T Pentland, From actions to paths to patterning: Toward a dynamic theory of patterning in routines, Academy of Management Journal 62 (6) (2019) 1901–1929.
[17] C.W. Günther. Process mining in flexible environments. PhD thesis, Technische Universiteit Eindhoven, 2009..
[18] Qinlong Guo, Lijie Wen, Jianmin Wang, Zhiqiang Yan, and S Yu Philip. Mining invisible tasks in non-free-choice constructs. In International Conference on Business Process Management, pages 109–125. Springer, 2015..
[19] Thorvald Hærem, Brian T Pentland, Kent D Miller, Task complexity: Extending a core concept, Academy of Management Review 40 (3) (2015) 446–460.
[20] A Jordan et al. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Advances in neural information processing systems, 14 (2002): 841, 2002..
[21] Maurice George Kendall, Rank correlation methods, Griffin, 1948.
[22] Hans-Peter Kriegel, Erich Schubert, Arthur Zimek, The (black) art of runtime evaluation: Are we comparing algorithms or implementations?, Knowledge and Information Systems 52 (2) (2017) 341–378
[23] S. Leemans, D. Fahland, W. van der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: BPM Workshops, Springer, 2014.
[24] Jan Mendling, Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, Lecture Notes in Business Information Processing, volume 6, Springer, 2008.
[25] Jan Mendling, Hajo A Reijers, and Wil MP van der Aalst. Seven process modeling guidelines (7pmg). Information and Software Technology, 52 (2): 127–136, 2010..
[26] Jan Mendling, Laura Sánchez-González, Félix García, Marcello La Rosa, Thresholds for error probability measures of business process models, Journal of Systems and Software 85 (5) (2012) 1188–1197.
[27] Jan Mendling, Benoît Depaire, and Henrik Leopold. Theory and practice of algorithm engineering. arXiv preprint arXiv:2107.10675, 2021..
[28] Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, Ben Shneiderman, Temporal event sequence simplification, IEEE transactions on visualization and computer graphics 19 (12) (2013) 2227–2236.
[29] Jorge Munoz-Gama and Josep Carmona. A fresh look at precision in process conformance. In Richard Hull, Jan Mendling, and Stefan Tai, editors, Business Process Management - 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13–16, 2010. Proceedings, volume 6336 of Lecture Notes in Computer Science, pages 211–226. Springer, 2010..
[30] Brian T Pentland, Conceptualizing and measuring variety in the execution of organizational work processes, Management Science 49 (7) (2003) 857–870.
[31] Brian T. Pentland, Peng Liu, Waldemar Kremser, Thorvald Hærem, The dynamics of drift in digitized processes, MIS Quarterly 44 (1) (2020).
[32] A. Polyvyanyy, A. Kalenkova, Monotone conformance checking for partially matching designed and observed processes, in: 2019 International Conference on Process Mining (ICPM), 2019, pp. 81–88, https://doi.org/10.1109/ICPM.2019.00022.
[33] Artem Polyvyanyy, Hanan Alkhammash, Claudio Di Ciccio, Luciano García-Bañuelos, Anna Kalenkova, Sander JJ Leemans, Jan Mendling, Alistair Moffat, and Matthias Weidlich. Entropia: a family of entropy-based conformance checking measures for process mining. arXiv preprint arXiv:2008.09558, 2020..
[34] Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio, Jan Mendling, Monotone precision and recall measures for comparing executions and specifications of dynamic systems, ACM Transactions on Software Engineering and Methodology (TOSEM) 29 (3) (2020) 1–41.
[35] Joel Ribeiro and Josep Carmona Vargas. A method for assessing parameter impact on control-flow discovery algorithms. In Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data: Brussels, Belgium, June 22–23, 2015, pages 83–96. CEUR-WS. org, 2015..
[36] Joel Ribeiro, Josep Carmona, Mustafa Mısır, Michele Sebag, in: A recommender system for process discovery In International Conference on Business Process Management, Springer, 2014, pp. 67–83.
[37] Laura Sánchez-González, Félix García, Francisco Ruiz, Jan Mendling, Quality indicators for business process models from a gateway complexity perspective, Information and Software Technology 54 (11) (2012) 1159–1174.
[38] Peter Sanders, Algorithm engineering–an attempt at a definition, in: Efficient Algorithms, Springer, 2009, pp. 321–340.
[39] Thomas J Santner, Brian J Williams, William I Notz, Brain J Williams, The design and analysis of computer experiments, volume 1, 2003, Springer.

[40] Suriadi Suriadi, Robert Andrews, Arthur HM ter Hofstede, Moe Thandar Wynn, Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs, Information Systems 64 (2017) 132–150.
[41] Anja F Syring, Niek Tax, and Wil MP van der Aalst. Evaluating conformance measures in process mining using conformance propositions. Petri Nets and Other Models of Concurrency XIV LNCS 11790 (2019) 192.
[42] N. Tax, X. Lu, N. Sidorova, D. Fahland, W. van der Aalst, The imprecisions of precision measures in process mining, Information Processing Letters 135 (2018).
[43] Wil Van Der Aalst, Process mining, Communications of the ACM 55 (8) (2012) 76–83.
[44] Wil M.P. van der Aalst, Vladimir A. Rubin, H.M.W. Verbeek, Boudewijn F. van Dongen, Ekkart Kindler, Christian W. Günther, Process mining: a two-step approach to balance between underfitting and overfitting, Software and Systems Modeling 9 (1) (2010) 87–111.
[45] M.P. Wil, van der Aalst, Data Science in Action, Second Edition. Springer, Process Mining, 2016.
[46] Wil MP Van Der Aalst, Hajo A Reijers, Anton JMM Weijters, Boudewijn F van Dongen, AK Alves De Medeiros, Minseok Song, and HMW Verbeek. Business process mining: An industrial application, Information Systems 32 (5) (2007) 713–732.
[47] Boudewijn van Dongen, BPI Challenge (2019) 2019, URL: https://data.4tu.nl/articles/dataset/BPI_Challenge_2019/12715853/1.
[48] Boudewijn van Dongen, BPI Challenge (2020) 2020, URL: https://data.4tu.nl/collections/BPI_Challenge_2020/5065541/1.
[49] Sebastiaan J. van Zelst, Boudewijn F. van Dongen, Wil M.P. van der Aalst, ILP-Based Process Discovery Using Hybrid Regions, in: International Workshop on Algorithms & Theories for the Analysis of Event Data, ATAED 2015, volume 1371 of CEUR Workshop Proceedings CEUR-WS.org, 2015, pp. 47–61.
[50] Seppe KLM vanden Broucke and Jochen De Weerdt. Fodina: a robust and flexible heuristic process discovery technique. Decision Support Systems, 100: 109–118, 2017..

Adriano Augusto is an Associate Lecturer at the University of Melbourne, where he coordinates the Business Process Management subject. He completed his joint Ph.D. at the University of Tartu and the University of Melbourne, in 2020, by defending a thesis focused on automated process discovery. His thesis was awarded the best BPM dissertation runner-up award at the 18th International conference on Business Process Management, 2020. His teaching and research interests include business process management, applied process mining in healthcare, automated process discovery, and robotic process automation.

Jan Mendling is the Einstein-Professor of Process Science with the Department of Computer Science at Humboldt-Universität zu Berlin, Germany. His research interests include various topics in the area of business process management and information systems. He has published more than 450 research papers and articles, among others in MIS Quarterly, ACM Transactions on Software Engineering and Methodology, IEEE Transactions on Software Engineering, Journal of the Association of Information Systems and Decision Support Systems. He is a department editor for Business and Information Systems Engineering, and a co-author of the textbooks Fundamentals of Business Process Management, Second Edition, and Wirtschaftsinformatik, 12th Edition.

Maxim Vidgof is a doctoral student with the Institute for Data, Process and Knowledge Management at Vienna University of Economics and Business (WU Wien), Austria. Maxim received his bachelor's degree in Business, Economics and Social Sciences with major in Information Systems from WU, a bachelor's degree in Economics with honors from Don State University of Technology (Rostov-on-Don, Russia) and a master's degree in Information Systems with honors from WU. His research interests are Process Mining and Robotic Process Automation. Maxim is a member of the Austrian Society for Process Management.

Bastian Wurm is a Ph.D. student with the Institute for Data, Process and Knowledge Management at Wirtschaftsuniversität Wien (WU Vienna), Austria. Bastian is interested in topics at the intersection of information systems and organization science. In particular, his research is centered around business process change and the development of holacratic organizations. His master thesis has won the Best Master Thesis Award by the University of Liechtenstein and the Young Scientist Award by the Austrian Society for Process Management. Bastian is a HICSS Doctoral Fellow.