CrossMark

Discovering more precise process models from event logs by filtering out chaotic activities

Niek Tax[1] · Natalia Sidorova[1] · Wil M. P. van der Aalst[2]

Abstract *Process Discovery* is concerned with the automatic generation of a process model that describes a business process from execution data of that business process. Real life event logs can contain *chaotic activities*. These activities are independent of the state of the process and can, therefore, happen at rather arbitrary points in time. We show that the presence of such *chaotic activities* in an event log heavily impacts the quality of the process models that can be discovered with process discovery techniques. The current modus operandi for filtering activities from event logs is to simply filter out infrequent activities. We show that frequency-based filtering of activities does not solve the problems that are caused by chaotic activities. Moreover, we propose a novel technique to filter out chaotic activities from event logs. We evaluate this technique on a collection of seventeen real-life event logs that originate from both the business process management domain and the smart home environment domain. As demonstrated, the developed activity filtering methods enable the discovery of process models that are more behaviorally specific compared to process models that are discovered using standard frequency-based filtering.

✉ Niek Tax
n.tax@tue.nl

Natalia Sidorova
n.sidorova@tue.nl

Wil M. P. van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] Department of Mathematics and Computer Science, Eindhoven University of Technology,
P.O. Box 513, 5600MB Eindhoven, The Netherlands

[2] Lehrstuhl für Informatik 9 / Process and Data Science, RWTH Aachen, D-52056 Aachen, Germany

⚛ Springer

1 Introduction

Process Mining (van der Aalst 2016) is a scientific discipline that bridges the gap between process analytics and data analysis and focuses on the analysis of event data logged during the execution of a business process. Events contain information on what was done, by whom, for whom, where, when, etc. Such event data is often readily available from information systems such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), or Business Process Management (BPM) systems. *Process discovery*, which plays a prominent role in process mining, is the task of automatically generating a process model that accurately describes a business process based on such event data. Many process discovery techniques have been developed over the last decade (e.g. Buijs et al. 2012; Buijs et al. 2009; Günther and van der Aalst 2007; Herbst 2000; Leemans et al. 2013b; Solé and Carmona 2013; van Zelst et al. 2015), producing process models in various forms, such as Petri nets (Murata 1989), process trees (Buijs et al. 2012), and Business Process Model and Notation (BPMN) models (Object Management Group 2011).

Figure 1b shows an example process model from van der Aalst (2016) that describes a compensation request process. The process model consists of eight process steps (called activities): *(A) register request, (B) examine thoroughly, (C) examine casually, (D) check ticket, (E) decide, (F) re-initiate request, (G) pay compensation*, and *(H) reject request*. Figure 1a shows a small example event log consisting of six execution trails of the process model. The Inductive Miner (Leemans et al. 2013b) process discovery algorithm provides the guarantee that it can re-discover the process model from an event log given that all pairs of activities that can directly follow each other in the process are present in the event log, i.e., the log is *directly-follows complete*. Since the log in Fig. 1a is directly-follows complete, applying the Inductive Miner to this log results in the process model in Fig. 1b, which generated the log.

However, the presence of activities that can occur spontaneously at any point in the process execution, which we will call *chaotic activities*, substantially impacts the quality of the resulting process models obtained with process discovery techniques. Figure 2a contains the event log obtained from the one in Fig. 1a by adding activity *(X) the customer calls* at random points, since customers can call the call center multiple times at any point in time during the execution of the process. Figure 2a shows the resulting process model discovered by the Inductive Miner (Leemans et al. 2013b) from the event log of Fig. 2a. The process model discovered from the "clean" example log without activity X (Fig. 1a) was very simple, interpretable, and accurate with respect to the behavior allowed in the process. In contrast, the process model discovered from the log containing X (Fig. 2b) is very
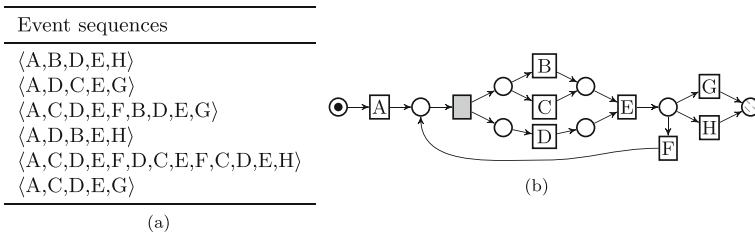

Fig. 1 **a** Event log with A=register request, B=examine thoroughly, C=examine casually, D=check ticket, E=decide, F=re-initiate request, G=pay compensation, H=reject request, and **b** the Petri net mined from this log with the Inductive Miner (Leemans et al. 2013b)

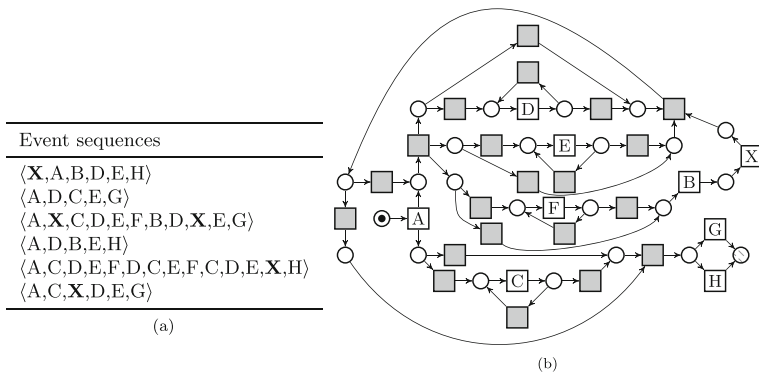| Event sequences |
| --- |
| ⟨**X**,A,B,D,E,H⟩ |
| ⟨A,D,C,E,G⟩ |
| ⟨A,**X**,C,D,E,F,B,D,**X**,E,G⟩ |
| ⟨A,D,B,E,H⟩ |
| ⟨A,C,D,E,F,D,C,E,F,C,D,E,**X**,H⟩ |
| ⟨A,C,**X**,D,E,G⟩ |

(a)

(b)

Fig. 2 **a** The event log from Fig. 1a with an added chaotic activity X, and **b** the Petri net mined from this log with the Inductive Miner (Leemans et al. 2013b)

complex, hard to interpret, and it overgeneralizes by allowing for too much behavior that is not possible in the process. We consider X to be a so-called *chaotic activity* because it does not have a clear position in the process model and it complicates the discovery of the rest of the process. The reason for the decline in the quality of process models discovered from logs with chaotic activities is that the directly follows relations, which many process discovery algorithms operate on, are affected by chaotic activities. Examples of such process discovery algorithms include the Inductive Miner (Leemans et al. 2013a), the Heuristics Miner (Weijters and Ribeiro 2011), and Fodina (Vanden Broucke and De Weerdt 2017). In a sequence of activities ⟨. . . , A, C, . . .⟩, where A was directly followed by C, the addition of a chaotic activity X can turn the sequence into ⟨. . . , A, X, C, . . .⟩, thereby obfuscating the directly-follows relation between activities A and C.

In this paper, we show that existing approaches do not solve the problem of chaotic activities and we present a technique to handle the issue. This paper is structured as follows: in Section 2 we introduce basic concepts used throughout the paper. In Section 3 we propose an approach to filter out chaotic activities. In Section 4 we evaluate our technique using synthetic data where we artificially insert chaotic activities and check whether the filtering techniques can filter out the inserted chaotic activities. Additionally, Section 4 proposes a methodology to evaluate activity filtering techniques in a real-life setting where there is no ground truth knowledge on which activities are truly chaotic, and motivates this methodology by showing that its results are consistent with the synthetic evaluation on the synthetic datasets. In Section 5 the results on a collection of seventeen real-life event logs are discussed. In Section 6 we discuss how the activity filtering techniques can be used in a toggle-based approach for human-in-the-loop process discovery. In Section 7 we discuss related techniques in the domains of process discovery and the filtering of event logs. Section 8 concludes this paper and discusses several directions for future work.

2 Preliminaries

In this section, we introduce concepts and notation throughout this paper.

$X = \{a_1, a_2, \ldots, a_n\}$ denotes a finite set. $\mathcal{P}(X)$ denotes the power set of X, i.e., the set of all possible subsets of X. $X \backslash Y$ denotes the set of elements that are in set X but not in set Y, e.g., $\{a, b, c\} \backslash \{a, c\} = \{b\}$. X^* denotes the set of all sequences over a set X and

$\sigma = \langle a_1, a_2, \ldots, a_n \rangle$ denotes a sequence of length n, with $\sigma(i) = a_i$ and $\langle \rangle$ the empty sequence. $\sigma\!\restriction_X$ is the projection of σ on X, e.g. $\langle a, b, c, a, b, c \rangle\!\restriction_{\{a,c\}} = \langle a, c, a, c \rangle$. $\sigma_1 \cdot \sigma_2$ denotes the concatenation of sequences σ_1 and σ_2, e.g., $\langle a, b, c \rangle \cdot \langle d, e \rangle = \langle a, b, c, d, e \rangle$.

A partial function $f \in X \nrightarrow Y$ with domain $dom(f)$ can be lifted to sequences over X using the following recursive definition: (1) $f(\langle \rangle) = \langle \rangle$; (2) for any $\sigma \in X^*$ and $x \in X$:

$$f(\sigma \cdot \langle x \rangle) = \begin{cases} f(\sigma) & \text{if } x \notin dom(f), \\ f(\sigma) \cdot \langle f(x) \rangle & \text{if } x \in dom(f). \end{cases}$$

A multiset (or bag) over X is a function $B : X \to \mathbb{N}$ which we write as $[a_1^{w_1}, a_2^{w_2}, \ldots, a_n^{w_n}]$, where for $1 \leq i \leq n$ we have $a_i \in X$ and $w_i \in \mathbb{N}^+$. The set of all multisets over X is denoted $\mathcal{B}(X)$.

In the context of process mining, we assume the set of all *process activities* Σ to be given. Event logs consist of sequences of events where each event represents a process activity.

Definition 1 (**Event, Trace, and Event Log**) An *event* e in an event log is the occurrence of an activity $e \in \Sigma$. We call a (non-empty) sequence of events $\sigma \in \Sigma^+$ a *trace*. An *event log* $L \in \mathcal{B}(\Sigma^+)$ is a multiset of traces.

$L = [\langle a, b, c \rangle^2, \langle b, a, c \rangle^3]$ is an example event log over process activities $\Sigma = \{a, b, c\}$, consisting of 2 occurrences of trace $\langle a, b, c \rangle$ and three occurrences of trace $\langle b, a, c \rangle$. *Activities*(L) denotes the set of process activities Σ that occur in L, e.g., *Activities*$(L) = \{a, b, c\}$. #(a, L) denotes the number of occurrences of activity a in log L, e.g., #$(a, L) = 5$.

A process model notation that is frequently used in the area of process mining is the Petri net. Petri nets can be automatically transformed into process model notations that are commonly used in business environments, such as BPMN and BPEL (Lohmann et al. 2009). A Petri net is a directed bipartite graph consisting of places (depicted as circles) and transitions (depicted as rectangles), connected by arcs. A transition describes an activity, while places represent the enabling conditions of transitions. Labels of transitions indicate the type of activity that they represent. Unlabeled transitions (τ-transitions) represent invisible transitions (depicted as gray rectangles), which are only used for routing purposes and are not recorded in the event log.

Definition 2 (**Labeled Petri net**) A *labeled Petri net* $N = \langle P, T, F, \ell \rangle$ is a tuple where P is a finite set of places, T is a finite set of transitions such that $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, called the flow relation, and $\ell : T \nrightarrow \Sigma$ is a partial labeling function that assigns a label to a transition, or leaves it unlabeled (the τ-transitions).

We write $\bullet n$ and $n \bullet$ for the input and output nodes of $n \in P \cup T$ (according to F). A state of a Petri net is defined by its *marking* $m \in \mathcal{B}(P)$ being a multiset of places. A marking is graphically denoted by putting $m(p)$ tokens on each place $p \in P$. State changes occur through transition firings. A transition t is enabled (can fire) in a given marking m if each input place $p \in \bullet t$ contains at least one token. Once t fires, one token is removed from each input place $p \in \bullet t$ and one token is added to each output place $p' \in t\bullet$, leading to a new marking $m' = m - \bullet t + t\bullet$.

A firing of a transition t leading from marking m to marking m' is denoted as step $m \xrightarrow{t} m'$. Steps are lifted to sequences of firing enabled transitions, written $m \xrightarrow{\gamma} m'$ and $\gamma \in T^*$ is a *firing sequence*.

Defining an *initial* and a set of *final* markings allows defining the *language* accepted by a Petri net as a set of finite sequences of activities.

Definition 3 (**Accepting Petri Net**) An *accepting Petri net* is a triplet $APN = (N, m_0, MF)$, where N is a labeled Petri net, $m_0 \in \mathcal{B}(P)$ is its initial marking, and $MF \subseteq \mathcal{B}(P)$ is its set of possible final markings. A sequence $\sigma \in \Sigma^*$ is a *trace* of an accepting Petri net APN if there exists a firing sequence $m_0 \overset{\gamma}{\longrightarrow} m_f$ such that $m_f \in MF$, $\gamma \in T^*$ and $\ell(\gamma) = \sigma$.

In the Petri nets that are shown in this paper, places that belong to the initial marking contain a token and places belonging to a final marking contain a bottom right label f_i with i a final marking identifier or are simply marked as ◎ in case of a single final marking.

The *language* $\mathfrak{L}(APN)$ is the set of all its traces, i.e., $\mathfrak{L}(APN) = \{l(\gamma) | \gamma \in T^* \wedge \exists_{m_f \in MF} m_0 \overset{\gamma}{\longrightarrow} m_f\}$, which can be of infinite size when APN contains loops. While we define the language for accepting Petri nets, in theory, $\mathfrak{L}(M)$ can be defined for any process model M with formal semantics. We denote the universe of process models as $\mathcal{M}$. For each $M \in \mathcal{M}$, $\mathfrak{L}(M) \subseteq \Sigma^+$ is defined.

A process discovery method is a function $PD : \mathcal{B}(\Sigma^+) \to \mathcal{M}$ that provides a process model for a given event log. The goal is to discover a process model that is a good description of the process from which the event log was obtained, i.e., it should allow for all the behavior that was observed in the event log (called *fitness*) while it should not allow for too much behavior that was not seen in the event log (called *precision*). For an event log L, $\tilde{L} = \{\sigma \in \Sigma^+ | L(\sigma) > 0\}$ is the *trace set* of L. For example, for log $L = [\langle a, b, c \rangle^2, \langle b, a, c \rangle^3]$, $\tilde{L} = \{\langle a, b, c \rangle \langle b, a, c \rangle\}$. For an event log L and a process model M, we say that L is *fitting* on M if $\tilde{L} \subseteq \mathfrak{L}(M)$. *Precision* is related to the behavior that is allowed by a model M that was not observed in the event log L, i.e., $\mathfrak{L}(M) \backslash \tilde{L}$.

3 Information-theoretic approaches to activity filtering

We consider a *chaotic activity* to be an activity for which the probability to occur does not change (or changes little) as an effect of occurrences of other activities and moreover, the occurrence of a chaotic activity does not change (or changes little) the probabilities to occur for other activities, i.e., they are not part of the process flow. More formally, consider a business process that is described by some process model M, with Σ some set of non-chaotic business activities that are modeled in M. Now consider a set of chaotic activities Σ^c with $\Sigma \cap \Sigma^c = \emptyset$, i.e., the probability of occurrence of the activities in Σ^c do not impact nor are impacted by the occurrence of other activities. Let M' be the process model that consists of M and additionally contains the chaotic activities Σ^c without constraints. If M is modeled as a Petri net, then M' contains one additional labeled transition t for each activity in Σ^c, with $\bullet t = t \bullet = \emptyset$. For example, let M be Fig. 1b and $\Sigma^c = \{X\}$, then Fig. 3 shows M'. Let L' be an event log that is obtained by executing the business process while also observing the activities in Σ^c, i.e., playing out model M'. Process discovery algorithms generally make some assumption about the degree of *completeness* of the event log. For example, both the Inductive Miner (Leemans et al. 2013a) and the α-miner (van der Aalst et al. 2004) assume the log to be *directly-follows-complete*, i.e., each pair of activities in the process that can possibly directly-follow each other is assumed to at least once directly follow each other in the log. When chaotic activities are present in the log, it becomes very hard for such completeness assumptions to be met, as many observed traces from the process are needed to observe all possible occurrences of an activity that is unconstrained in when it can occur.

Fig. 3 The model of Fig. 1b
with an added chaotic activity X
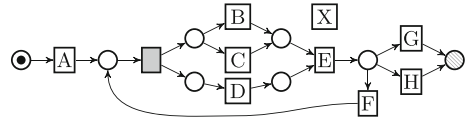
In this section, we propose a technique to detect chaotic activities in event logs and to
filter them out from those event logs. We extend the function $\#(a, L)$ to the function $\#(\sigma, L)$
to count the number of occurrence of a sequence σ, in L:

$$\#(\sigma, L) = \sum_{\sigma' \in L} |\{0 \leq i \leq |\sigma'| - |\sigma| \mid \forall_{1 \leq j \leq |\sigma|} \sigma'(i+j) = \sigma(j)\}|.$$

The *directly-follows ratio*, denoted $dfr(a, b, L)$, represents the ratio of the events of
activity a that are directly followed by an event of activity b in event log L, i.e.,
$dfr(a, b, L) = \frac{\#(\langle a, b \rangle, L)}{\#(a, L)}$.

Likewise, the *directly-precedes ratio*, denoted $dpr(a, b, L)$, represents the ratio of the
events of activity a that are directly preceded by an event of activity b in event log L, i.e.,
$dpr(a, b, L) = \frac{\#(\langle b, a \rangle, L)}{\#(a, L)}$.

$L^\rfloor$ contains the traces of event log L appended with an *artificial end event* that we
represent with $\rfloor$. For each $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ in log L, log $L^\rfloor$ contains a trace $\sigma^\rfloor =
\langle e_1, e_2, \ldots, e_n, \rfloor \rangle$. Likewise, $L^\lfloor$ contains the traces of event log L prepended with an *arti-
ficial start event* $\lfloor$, i.e., for each $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ in log L, log $L^\lfloor$ contains a trace
$\sigma^\lfloor = \langle \lfloor, e_1, \ldots, e_n \rangle$. The artificial *start* and *end* events allow us to define the ratio of start
events of an activity, e.g., $dfr(a, \rfloor, L^\rfloor)$ and $dpr(a, \lfloor, L^\lfloor)$ represent the ratio of events of
activity a that respectively occur at the end of a trace and at the beginning of a trace.

Assuming an arbitrary but consistent order over the set of process activities $Activities(L)$,
$dfr(a, L)$ represents the vector of values $dfr(a, b, L^\rfloor)$ for all $b \in Activities(L) \cup \{\rfloor\}$ and
$dpr(a, L)$ represents the vector of values $dpr(a, b, L^\lfloor)$ for all $b \in Activities(L) \cup \{\lfloor\}$. From
a probabilistic point of view, we can regard $dfr(a, L)$ and $dpr(a, L)$ as the empirical esti-
mates of the categorical distributions over respectively the activities directly prior to a and
directly after a, where the empirical estimates are based on $\#(a, L)$ trials.

3.1 Direct entropy-based activity filtering

We define the entropy of an activity in an event log L based on its directly-follows ratio
vector and the directly-precedes ratio vector by using the usual definition of function for
the categorical probability distribution: $H(X) = -\sum_{x \in X} x \log_2(x)$. We define the entropy
of activity $a \in Activities(L)$ in log L as: $H(a, L) = H(dfr(a, L)) + H(dpr(a, L))$. In case
there are zero probability values in the directly follows or directly precedes vectors, i.e.,
$0 \in dfr(a, L) \vee 0 \in dpr(a, L)$, then the value of the corresponding summand $0 \log_2(0)$ is
taken as 0, which is consistent with the limit $\lim_{p \to 0+} p \log_2(p) = 0$.

For example, let event log $L = [\langle a, b, c, x \rangle^{10}, \langle a, b, x, c \rangle^{10}, \langle a, x, b, c \rangle^{10}]$, then
$dfr(a, L) = \langle 0, \frac{20}{30}, 0, \frac{10}{30}, 0 \rangle$, using the arbitrary but consistent ordering $\langle a, b, c, x, \rfloor \rangle$, indi-
cating that 20 out of 30 events of activity a are followed by b and 10 out of 30 by x. Likewise
$dpr(a, L) = \langle 0, 0, 0, 0, 1 \rangle$, using the arbitrary but consistent ordering $\langle a, b, c, x, \lfloor \rangle$, indicat-
ing that all events of activity a are preceded by $\lfloor$. This leads to $H(dfr(a, L)) = 0.918$,
$H(dpr(a, L)) = 0$, and $H(a, L) = 0.918$. Furthermore, $H(b, L) = 1.837$, $H(c, L) =$

1.837, and $H(x, L) = 3.170$, showing that activity x has the highest entropy of the probability distributions for preceding and succeeding activities. We conjecture that activities that are chaotic and behave randomly to a high degree have high values of $H(a, L)$.

Algorithm 1 An activity filtering approach based on entropy.

Input: event log L
Output: list of event logs Q
       Initialisation :
1: $L' = L$
2: $Q = \langle L' \rangle$
       Main Procedure:
3: **while** $|Activities(L')| > 2$ **do**
4:     $acts = Activities(L')$
5:     $a = \arg\max_{a \in acts} H(a, L')$
6:     $L' = L' \lceil_{acts \setminus \{a'\}}$
7:     $Q = Q \cdot \langle L' \rangle$
8: **end while**
9: **return** Q

Algorithm 1 describes a greedy approach to iteratively filter the most randomly behaving (chaotic) activity from the event log. The algorithm takes an event log L as input and produces a list of event logs, such that the first element of the list contains a version of L with one activity filtered out, and each following element of the list has one additional activity filtered out compared to the previous element.

In the example event log L, Algorithm 1 starts by filtering out activity x, followed by activity b or c. The algorithm stops when there are two activities left in the event log. The reason not to filter any more activities past this point is closely related to the aim of process discovery: uncovering relations between activities. From an event log with less than two activities no relations between activities can be discovered.

3.2 The entropy of infrequent activities and laplace smoothing

We defined entropy of the activities in an event log L is based on the directly-follows ratios *dfr* and the directly-precedes ratios *dpr* of the activities in L. The empirical estimates of the categorical distributions $dfr(a, L)$ and $dpr(a, L)$ become unreliable for small values of $\#(a, L)$. In the extreme case, when $\#(a, L)=1$, $dfr(a, L)$ assigns an estimate of 1 to the activity that the single activity a in L happens to be preceded by and contains a probability of 0 for the other activities. Likewise, when $\#(a, L)=1$, $dpr(a, L)$ assigns value 1 to one activity and value 0 to all others. Therefore, $\#(a, L)=1$ leads to $H(dfr(a, L))=0$ and $H(dfr(a, L))=0$. This shows an undesirable consequence of Algorithm 1, infrequent activities are unlikely to be filtered out. In the extreme case, the activities that occur only once, which are the last in line activities to be filtered out. This effect is undesired, as very infrequent activities should not be the primary focus of the process model discovered from an event log.

We aim to mitigate this effect by applying Laplace smoothing (Zhai and Lafferty 2004) to the empirical estimate of the categorical distributions over the preceding and succeeding activities. Therefore, we define a *smoothed version* of the directly-follows and directly-precedes ratios, $dfr^s(a, b, L) = \frac{\alpha + \#(\langle a,b \rangle, L)}{\alpha(|Activities(L)|+1) + \#(a, L)}$, with smoothing parameter $\alpha \in \mathbb{R}_{\geq 0}$.

The value of $dfr^s(a, b, L)$ will always be between the empirical estimate $dfr(a, b, L)$ and the uniform probability $\frac{1}{|Activities(L)|+1}$, depending on the value α. Similar to dfr and dpr, $dfr^s(a, L)$ represents the vector of values $dfr^s(a, b, L^\rfloor)$ for all $b \in Activities(L) \cup \{\rfloor\}$ and $dpr^s(a, L)$ represents the vector of values $dpr^s(a, b, L^\lfloor)$ for all $b \in Activities(L) \cup \{\lfloor\}$. From a Bayesian point of view, Laplace smoothing corresponds to the expected value of the posterior distribution that consists of the categorical distribution given by $dfr(a, L)$ and a Dirichlet distributed prior that assigns equal probability to each of the possible number of next activities $|Activities(L)| + 1$ (including $\rfloor$). Parameter α indicates the weight that is assigned to the prior belief w.r.t. the evidence that is found in the data. An alternative definition of the entropy of log L, based on the smoothed distributions over the preceding and succeeding activities, is as follows: $H^s(a, L) = H(dfr^s(a, L)) + H(dpr^s(a, L))$. The smoothed direct entropy-based activity filter is similar to Algorithm 1, where function H in line 5 of the algorithm is replaced by H^s. Function $H(a, L)$ starts from the assumption that an activity is non-chaotic unless we see sufficient evidence in the data for its chaoticness, function $H^s(a, L)$ in contrast starts from the assumption that is is chaotic, unless we see evidence sufficient evidence in the data for its non-chaoticness.

Categorical distribution $dfr(a, L)$ consists of $|Activities(L)| + 1$, therefore, the maximum entropy of an activity decreases as more activities get filtered out of the event log. The keep the values of $H^s(a, L)$ comparable between iterations of the filtering algorithm, we propose to gradually increase the weight of the prior by setting weight parameter α to $\frac{1}{|Activities(L)|}$.

3.3 Indirect entropy-based activity filtering

An alternative approach to the method proposed in Algorithm 1 is to filter out activities such that the *other* activities in the log become less chaotic. We define the total entropy of an event log L as the sum of the entropies of the activities in the log, i.e., $H(L) = \sum_{a \in Activities(L)} H(a, L)$.

Algorithm 2 describes a greedy approach that iteratively filters out the activity that results in the lowest total log entropy. We call this approach the *indirect entropy-based activity filter*, as opposed to the *direct entropy-based activity filter* (Algorithm 1), which selects the to-be-filtered activity directly based on the activity entropy, instead of based on the total log entropy after removal.

Algorithm 2 An indirect activity filtering approach based on entropy.

Input: event log L
Output: list of event logs Q

       Initialisation :

1: $L' = L$
2: $Q = \langle L' \rangle$

       Main Procedure:

3: **while** $|Activities(L')| > 2$ **do**
4:     $acts = Activities(L')$
5:     $a' = \arg\min_{a \in acts} H(L' \lceil_{acts \setminus \{a\}})$
6:     $L' = L' \lceil_{acts \setminus \{a'\}}$
7:     $Q = Q \cdot \langle L' \rangle$
8: **end while**
9: **return** Q

3.4 An indirect entropy-based activity filter with laplace smoothing

Just like the direct entropy-based activity filter, the indirect entropy-based activity filter is sensitive to infrequent activities. To deal with this problem, the ideas of the indirect entropy-based activity filtering method and Laplace smoothing can be combined, using the following definition for smoothed log entropy:

$$H^s(L) = \sum_{a \in Activities(L)} H^s(a, L).$$

The algorithm for indirect entropy-based activity filtering with Laplace smoothing is identical to Algorithm 2, in which function H in line 5 is replaced by function H^s.

4 Evaluation using synthetic data

In this section we evaluate the activity filtering techniques using synthetic data. Figure 4 gives an overview of the evaluation methodology. First, as step **(1)**, we generate a synthetic event log from a process model such that we know that all activities of this model are non-chaotic. We take well-known process models introduced by Maruster et al. (2006), which respectively consist of 12 and 22 activities and are commonly referred to as the Maruster A12, A22 models. The Maruster A12 and A22 models are shown respectively in Figs. 5a and 6a. We generated 25 traces by simulation from Maruster A12 to form log L_{A12} and generated 400 traces from Maruster A22 to form log L_{A22}. Then, in step **(2)**, we artificially insert activities that we position at random positions in the log. Since we chose the positions in the log of those activities randomly, we assume those activities to be *chaotic*. We vary the number (k) of randomly-positioned activities that we insert, to assess how well the chaotic activity filtering techniques are able to deal with different numbers of randomly-positioned activities in the event log. Furthermore, we vary the frequency of the randomly-positioned activities that we insert, where we distinguish between three types of randomly-positioned activities:

Frequent randomly-positioned activities the number of events inserted for all k randomly-positioned activities is $\max_{a \in Activities(L)} \#(a, L)$.
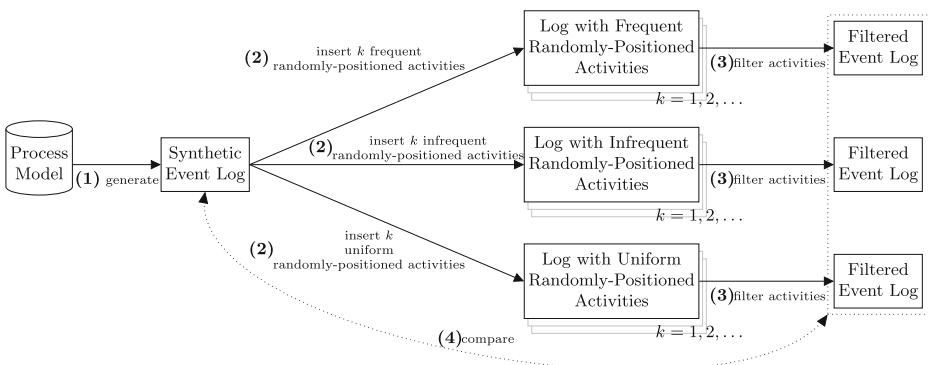

Fig. 4 An overview of the proposed evaluation methodology on synthetic data
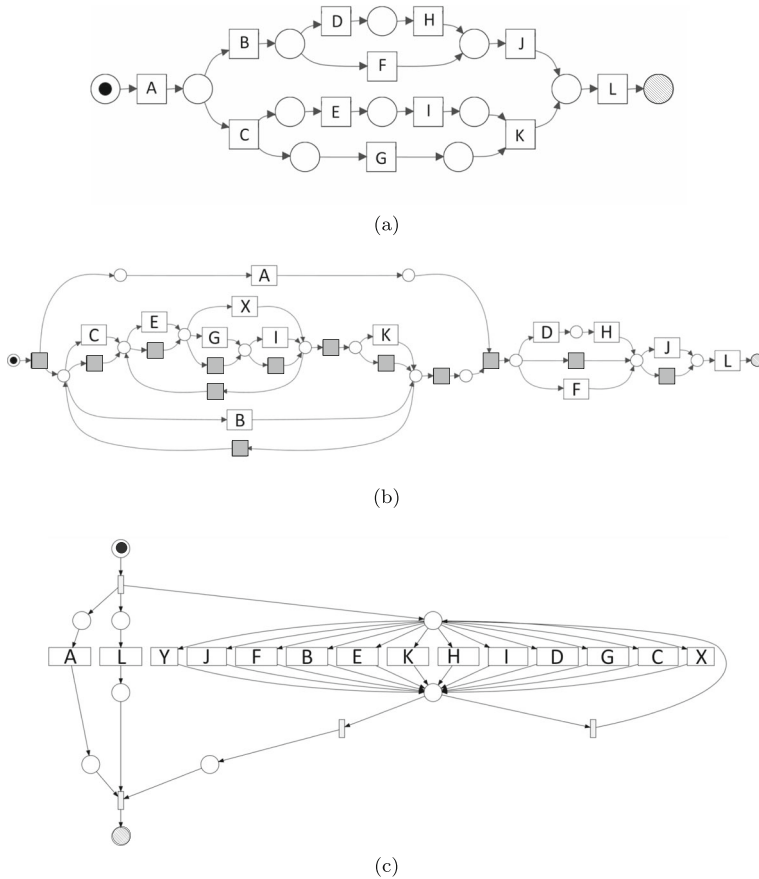
(a)

(b)

(c)

Fig. 5 **a** The synthetic process model Maruster A12, from which we generate an event log L, consisting of 25 traces, from which the process model can be rediscovered with the Inductive Miner (Leemans et al. 2013a), **b** the process model discovered by the Inductive Mining when we insert one uniform randomly-positioned activity X to L_{A12}, and *(c)* the process model discovered by the Inductive Miner after inserting a second randomly-positioned activity Y to L_{A12}

Infrequent randomly-positioned activities   the number of events inserted for all k randomly-positioned activities is $\min_{a \in Activities(L)} \#(a, L)$.

Uniform randomly-positioned activities   for each of the k inserted randomly-positioned activities the frequency is chosen at randomly from a uniform probability distribution with minimum value $\min_{a \in Activities(L)} \#(a, L)$ and maximum value $\max_{a \in Activities(L)} \#(a, L)$.

In step **(3)** we filter out all the inserted randomly-positioned activities from the event log, by removing activities one-by-one using the activity filtering approaches, until all k artificially inserted activities have been removed again. We then count how many of the activities that were originally in the process model we also removed during this procedure (step **(4)**). Using this approach, we compare the direct entropy-based activity filtering approach (with and without Laplace smoothing) with the indirect entropy-based activity filtering approach (with and without Laplace smoothing). Furthermore, we compare those activity filtering
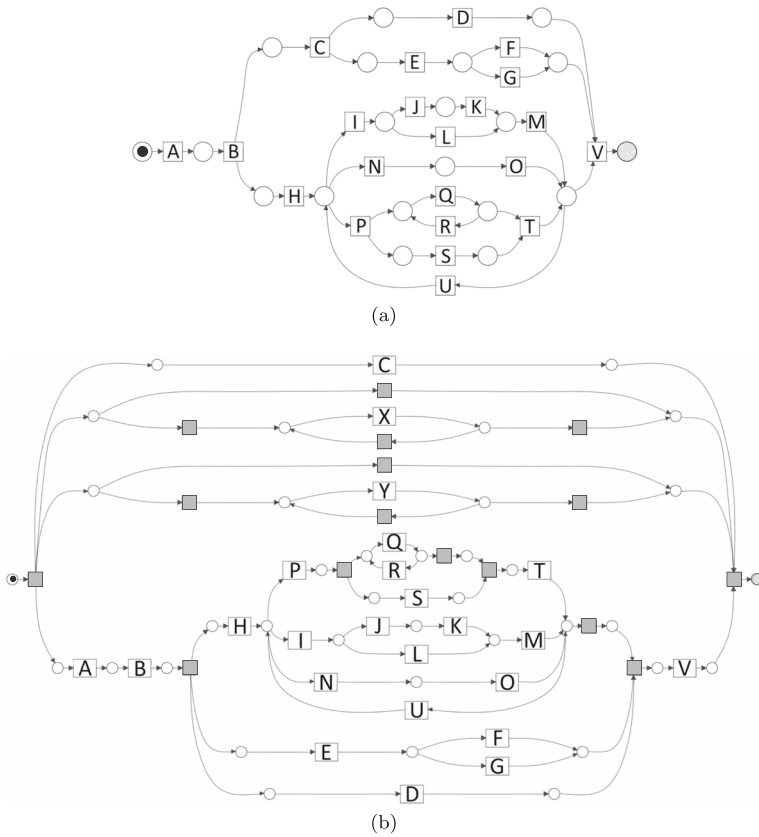
Fig. 6 **a** The synthetic process model Maruster A22, from which we generate an event log L_{A22}, consisting of 400 traces, from which the process model is re-discoverable with the Inductive Miner (Leemans et al. 2013a), and **b** the process model discovered by the Inductive Miner after inserting a uniform randomly-positioned activity X to L_{A22}

techniques with activity filtering techniques that are based on the frequency of activities, such as filtering out the activities starting from the least frequent activity (least-frequent-first), or starting from the most frequent activity (most-frequent-first). Frequency-based activity filtering techniques are the current default approach for filtering activities from event logs.

The original process models A12 and A22 can be rediscovered from generated event logs L_{A12} and L_{A22} with the Inductive Miner (Leemans et al. 2013a) when there are no added randomly-positioned activities. Figure 5b shows the process model discovered by the Inductive Miner (Leemans et al. 2013a) after inserting one uniform randomly-positioned activity, activity X, into L_{A12}. The insertion of activity X causes the Inductive Miner to create a model that overgeneralizes the behavior of the event log, as indicated by many silent transitions in the process model that allow activities to be skipped. Adding a second uniform randomly-positioned activity Y to L_{A12} results in the Inductive Miner discovering a process model (shown in Fig. 5c) that overgeneralizes even further, allowing for almost all sequences over the set of activities. Figure 6b shows the process model discovered by the

Inductive Miner after inserting two uniform randomly-paced activities (X and Y) into L_{A22}. The addition of X and Y has the effect that activity C is no longer positioned at the correct place in the process model, but it is instead put in parallel to the whole process, making the process model overly general, as it wrongly allows for activity C to occur before A and B, or after D, E, F, and G. Figures 5b, c and 6b further motivate the need for filtering out chaotic activities.

Frequent randomly-positioned activities will impact the quality of process models discovered with process discovery to a higher degree than infrequent randomly-positioned activities. Each randomly-positioned activity that is inserted at a random position in the event log is placed in-between two existing events in that log (or at the start or end of the trace). By inserting randomly-positioned activity X in-between two events of activities A and C respectively, the directly-follows relation between activities A and C gets weakened. Therefore, the impact of randomly-positioned activity X is proportional to its frequency $\#(X, L)$.

4.1 Results

Table 1 reports the number of activities that were originally part of the synthetic process models A12 and A22 that were wrongly filtered out from L_{A12} and L_{A22} as an effect of removing all inserted randomly-positioned activities from these logs. If this number is 12 for Maruster A12 or 22 for Maruster A22 this indicates that all activities of the original process model needed to be filtered out before the activity filtering technique was able to remove all inserted chaotic activities. The results show that the direct filtering approach can perfectly distinguish between actual activities from the process and artificial chaotic activities for up to 32 uniform randomly-positioned activities inserted activities to L_{A12}, up to 64 frequent randomly-positioned activities, and up to 16 infrequent randomly-positioned activities. Infrequent randomly-positioned activities are the hardest type of randomly-positioned activities to correctly filter out, as their infrequency can have the effect that the probability distributions over their surrounding activities can *by chance* have low entropy. Using Laplace smoothing with $\alpha = \frac{1}{|Activities(L)|}$ mitigates this effect, but does not completely solve it: the number of incorrectly removed activities drops from 12 to 0 as an effect of Laplace smoothing for 32 added randomly-positioned activities, and from 12 to 6 for 64 added randomly-positioned activities. The indirect activity filter starts making errors already at lower numbers of added randomly-positioned activities than the direct activity filter; however, it is more stable to errors for higher numbers of added randomly-positioned activities, i.e., fewer activities get incorrectly removed for 64 and 128 added randomly-positioned activities. In contrast to direct activity filtering, Laplace smoothing does not seem to reduce the number of wrongly removed activities for indirect activity filtering. In fact, surprisingly, the number of incorrectly removed activities even increased from 6 to 10 as an effect of using Laplace smoothing for 128 infrequent randomly-positioned activities added to L_{A12}. The direct and indirect filtering approaches, both with and without Laplace smoothing, outperform the currently widely used approach of filtering out infrequent activities from the event log (least-frequent-first filtering). Furthermore, a second frequency-based activity filtering technique is included in the evaluation in which the most-frequent activities are removed from the event log (most-frequent first filtering). Both Frequency-based filtering approaches are not able to filter out the randomly-positioned activities inserted to L_{A12} and L_{A22}, even for small numbers of added randomly-positioned activities.

Table 1 The number of incorrectly filtered activities per filtering approach on L_{A12} and L_{A22} with k added Uniform (U) / Frequent (F) /Infrequent (I) chaotic activities

Approach	1			2			4			8			16			32			64			128		
	U	F	I	U	F	I	U	F	I	U	F	I	U	F	I	U	F	I	U	F	I	U	F	I
Maruster A12 (Number of inserted randomly-positioned activities →)																								
Direct	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	12	10	1	12
Direct ($\alpha=\frac{1}{A}$)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	6	6	2	12
Indirect	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	2	0	1	3	1	6
Indirect ($\alpha=\frac{1}{A}$)	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	2	0	1	2	1	10
Least-frequent-first	9	12	0	11	12	0	6	12	0	11	12	0	11	12	0	12	12	0	12	12	0	12	12	0
Most-frequent-first	11	0	12	3	0	12	7	0	12	10	0	12	12	0	12	12	0	12	12	0	12	12	0	12
Maruster A22 (Number of inserted randomly-positioned activities →)																								
Direct	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	5
Direct ($\alpha=\frac{1}{A}$)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	5
Indirect	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1
Indirect ($\alpha=\frac{1}{A}$)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	1	0	1
Least-frequent-first	16	22	0	17	22	0	6	22	0	21	22	0	19	22	0	22	22	0	22	22	0	22	22	0
Most-frequent-first	7	0	22	8	0	22	19	0	22	17	0	22	19	0	22	22	0	22	22	0	22	22	0	22

4.2 An evaluation methodology for event data without ground truth information

In a real-life data evaluation that we perform in the following section, there is no ground truth knowledge on which activities of the process are chaotic. This motivates a more indirect evaluation in which we evaluate the quality of the process model discovered from the event log after filtering out activities with the proposed activity filtering techniques. In this section we propose a methodology for evaluation of activity filtering techniques by assessing the quality of discovered process models, we apply this evaluation methodology to the Maruster A12 and Maruster A22 event logs, and we discuss the agreement between the findings of Table 1 and the quality of the discovered process models.

There are several ways to quantify the quality of a process model for an event log. Ideally, a process model M should allow for all behavior that was observed in the event log L, i.e., $\tilde{L} \setminus \mathfrak{L}(M)$ should be as small as possible, preferably empty. The *fitness* quality dimension covers this. Furthermore, model M should not allow for too much additional behavior that was *not* seen in the event log, i.e., $\mathfrak{L}(M) \setminus \tilde{L}$ should be as small as possible. This aspect is called *precision*. For each process model that we discovered, we measure *fitness* and *precision* with respect to the filtered log. Fitness is measured using the alignment-based fitness measure (Adriansyah et al. 2011) and we measure precision using negative event precision (Vanden Broucke et al. 2013). Based on the fitness and precision results we also calculate *F-score* (De Weerdt et al. 2011), i.e., the harmonic mean between fitness and precision.

Precision is likely to increase by filtering out one or more activities from an event log *independently of which activities are removed from the log*, as a result of two factors. First, precision measures express $\mathfrak{L}(M) \setminus \tilde{L}$ in terms of the number of activities that are enabled at certain points in the process, w.r.t. the number of activities seen that were actually observed at these points in the process. With the log and model containing fewer activities after filtering, the number of enabled activities is likely to decrease as well. Secondly, activity filtering leads to log L' that contains less behavior than original log L (i.e., $\tilde{L}'$ is smaller than $\tilde{L}$), this makes it easier for process discovery methods to discover a process model with less behavior. These two factors make precision values between event logs with different numbers of activities filtered out incomparable. The degree to which the behavior of filtered log L' decreases w.r.t. an unfiltered log L depends on the activities that are filtered out: when very chaotic activities are filtered from L the behavior decreases much more than when very structured activities are filtered from L. One effect of this is that too much behavior in a process model affects the precision of that model more for the log from which the non-chaotic activities are filtered out than for the log from which the chaotic activities are filtered out.

To measure the behavior allowed by the process model independent of which activities are filtered from the event log is to determine the average number of enabled activities when replaying the traces of the log on the model. To deal with traces of the event log that do not fit the behavior of the process model, we calculate *alignments* (Adriansyah et al. 2011) between log and model. Alignments are a function $\Gamma^m : \mathcal{M} \times \Sigma^+ \rightarrow \mathcal{B}(P)^+$ that map each trace from the event log to a sequence of markings $\langle m_0, \ldots, m_f \rangle$ that are reached to replay that trace on the model, with m_0 the initial marking and $m_f \in MF$, such that for each two consecutive markings $\langle m_i, m_{i+1} \rangle$ there exists a transition $t \in T$ such that $m_{i+1} = m_i - \bullet t + t \bullet$. Furthermore, alignments also provide a function $\Gamma^t : \mathcal{M} \times \Sigma^+ \rightarrow T^+$ that provides the sequence of transitions $\langle t_0, \ldots t_n \rangle$ that matches the changes in the sequence of markings, i.e., $m_1 = m_0 - \bullet t_0 + t_0 \bullet$, etc. For each trace $\sigma \in \Sigma^+$ that fits a process model $N \in \mathcal{M}$ the alignment $l(\Gamma^t(N, \sigma)) = \sigma$. For unfitting traces $\sigma \in \Sigma^+$, the alignment is such that $l(\Gamma^t(N, \sigma))$ is as close as possible to l according to some cost function. We

refer to Adriansyah et al. (2011) for a more exhaustive introduction of alignments. Let $\overline{\Gamma^t}$ denote the sequence consisting of only the visible transitions in Γ^t, and let $\overline{\Gamma^m}$ correspondingly denote the sequence of markings prior to each firing of a visible transition. Given a marking $m \in \mathcal{B}(P)$ we define the nondeterminism of that marking to be the number of reachable visible transitions that can be fired as first next visible transition from m, i.e., $nondeterminism(m) = |\{a \in \Sigma | m \xrightarrow{\gamma} m_i \wedge t \in \gamma \wedge l(t) = a \wedge \forall_{\gamma_i \in \gamma} \gamma_i \in dom(l) \implies \gamma_i = t\}|$. We define the nondeterminism of a model $N \in \mathcal{M}$ given a trace $\sigma \in \Sigma^+$ as the average nondeterminism of the markings $\overline{\Gamma^m}(N, \sigma)$ and define the nondeterminism for a model N and a log L as the average nondeterminism over the traces of L.

Figure 7 shows the F-scores measured for different percentages of activities filtered out from the Maruster L_{A12} log with different numbers of uniform chaotic activities added. Note that the line stops when further removal of activities does not lead to further improvement in F-score. Note that on the original event log with 0 chaotic activities added the F-score on the original log is already 1.0, resulting in no lines being drawn. With one chaotic activity added, the least-frequent-first filter needs to remove 75% of the activities before it ends up with F-score 1, which can be explained by the fact that 9 out of 12 non-chaotic needed to be removed in order with the least-frequent-first filter to remove all uniform chaotic activities, as shown in Table 1. All entropy-based activity filtering techniques remove the chaotic activity in the first filtering step, immediately leading to an F-score of 1.0. Up until 8 added chaotic activities there is no difference between the entropy-based activity filtering techniques in terms of F-score of the resulting process models, which is consistent with the fact that all these filtering techniques were found to filter without errors for these number of inserted chaotic activities in Table 1. For 16 and 32 activities, the direct filtering methods outperform the indirect filtering methods, consistent with the fact that the indirect approach made one filtering error according to the ground truth for these numbers of added chaotic activities. Note that the least-frequent-first filter is outperformed by the entropy-based filtering methods in terms of F-score of the discovered models, as would be expected given the filtering results according to the ground truth.

Figure 8 shows the results in terms of nondeterminism measured for different percentages of activities filtered out from the Maruster L_{A12} log with various numbers of uniform
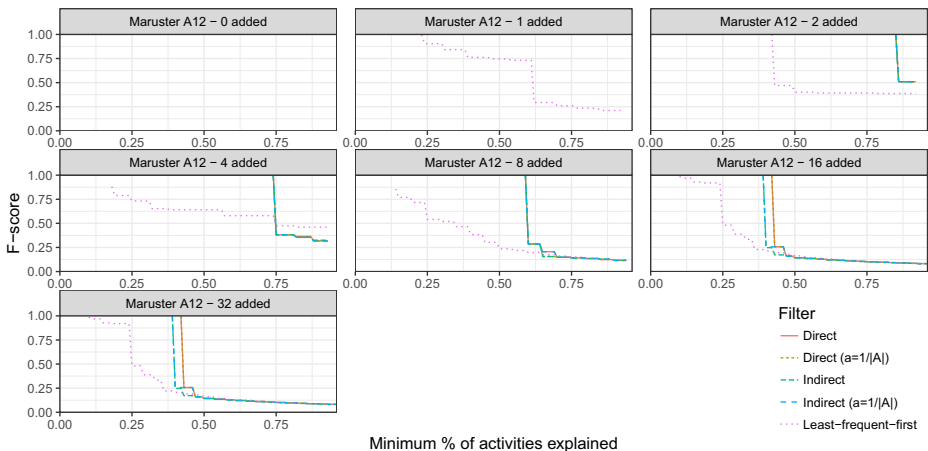

Fig. 7 F-score on the log generated from the Maruster A12 model with inserted artificial chaotic activities
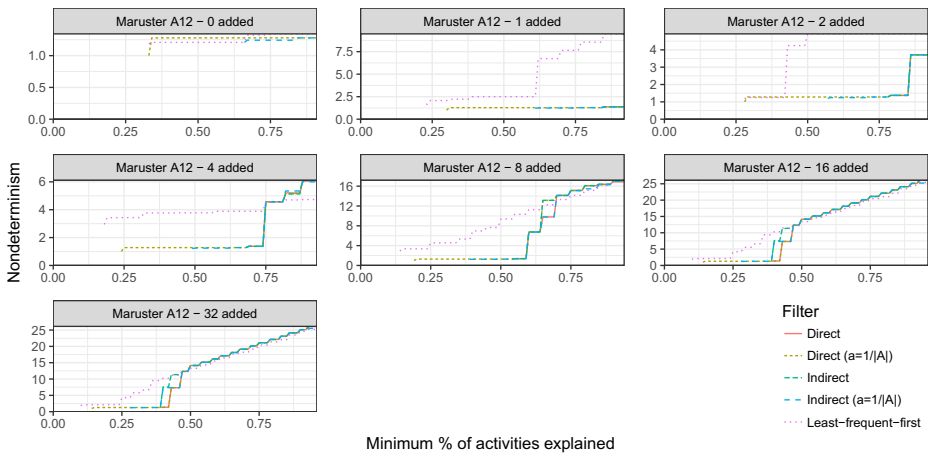
Fig. 8 Nondeterminism on the log generated from the Maruster A12 model with inserted artificial chaotic activities

chaotic activities added. The results show very clearly that when filtering out a number of activities that is identical to the number of added chaotic activities (this corresponds to 92% for one added activity, 86% for two added activities, 75% for 4 added activities, 60% for 8 added activities, 43% for 16 added activities, and 27% for 32 added activities), the nondeterminism reaches a value of 1.5, which is the nondeterminism value of the model discovered from the original log without added chaotic activities. The least-frequent-first filter, however, leads to process models where many activities are enabled on average, therefore overgeneralizing the process behavior, as an effect of filtering out nonchaotic activities instead of the added chaotic activities.

5 Evaluation using real life data

For the experiments on real-life event logs we do not artificially insert chaotic activities to event logs, but instead filter directly on the activities that are present in these logs. Whether these logs contain chaotic activities that impact process discovery results is not known upfront. Therefore, we apply different activity filtering techniques to these logs and use them to filter out a varying number of activities, after which we assess the quality of the process model that is discovered from these filtered logs. Table 2 gives an overview of the real-life event logs that we use in the experiment. In total, we include five event logs from the business domain. Furthermore, we include twelve event logs that contain events of human behavior, recorded in smart home environments or through wearable devices. Mining process model descriptions of daily life is a novel application of process mining that has recently gained popularity (Dimaggio et al. 2016; Leotta et al. 2015; Sztyler et al. 2015; Tax et al. 2017). Furthermore, human behavior event data are often challenging for process discovery because of the presence of highly chaotic activities, like *going to the toilet*. We perform the experiments with activity filtering techniques on real-life data with RapidProM (van der Aalst et al. 2017), which is an extension that adds process mining capabilities to the RapidMiner platform for repeatable scientific workflows.

Table 2 An overview of the event logs used in the experiments

Name	Category	# traces	# events	# activities
BPI'12 (Van Dongen 2012)	Business	13087	164506	23
BPI'12 resource 10939 (Tax et al. 2016)	Business	49	1682	14
Environmental permit (Buijs 2014)	Business	1434	8577	27
SEPSIS (Mannhardt 2016)	Business	1050	15214	16
Traffic Fine (De Leoni and Mannhardt 2015)	Business	150370	561470	11
Bruno (Bruno et al. 2013)	Human behavior	57	553	14
CHAD 1600010 (McCurdy et al. 2000)	Human behavior	26	238	10
MIT A (Tapia et al. 2004)	Human behavior	16	2772	27
MIT B (Tapia et al. 2004)	Human behavior	17	1962	20
Ordonez A (Ordónez et al. 2013)	Human behavior	15	409	12
van Kasteren (van Kasteren et al. 2008)	Human behavior	23	220	7
Cook hh102 labour (Cook et al. 2013)	Human behavior	18	576	18
Cook hh102 weekend (Cook et al. 2013)	Human behavior	18	210	18
Cook hh104 labour (Cook et al. 2013)	Human behavior	43	2100	19
Cook hh104 weekend (Cook et al. 2013)	Human behavior	18	864	19
Cook hh110 labour (Cook et al. 2013)	Human behavior	21	695	17
Cook hh110 weekend (Cook et al. 2013)	Human behavior	6	184	14

For each event log, we apply seven different activity filtering techniques for comparison: 1) direct entropy filter without Laplace smoothing, 2) direct entropy filter with Laplace smoothing ($\alpha = \frac{1}{|Activities(L)|}$), 3) indirect entropy filter without Laplace smoothing, 4) indirect entropy filter with Laplace smoothing ($\alpha = \frac{1}{|Activities(L)|}$), 5) least-frequent-first filtering, 6) most-frequent-first filtering, 7) filtering the activities from the log in a random order. Recall that the activity filtering procedure stops at the point where all but two activities are filtered from the event log because process models that contain just one activity do not communicate any information regarding the relations between activities. For each event log and for each activity filtering approach we discover a process model after each filtering step (i.e., after each removal of an activity). The process discovery step is performed with two process discovery approaches: the Inductive Miner (Leemans et al. 2013a), and the Inductive Miner infrequent (20%) (Leemans et al. 2013b).

5.1 Results on business process event logs

Figure 9 shows the F-score of the process models discovered with the Inductive Miner (Leemans et al. 2013a) and the Inductive Miner with infrequent behavior filtering (Leemans et al. 2013b) (20% filtering) on the five business event logs for different percentages of activities filtered out and different activity filtering techniques. The figure shows an increasing trend in F-score for all event logs when more activities are filtered from the event log. Furthermore, the line for the least-frequent-first filtering approach is below the lines of the entropy-based filtering techniques for most of the percentages of activities removed on most event logs, which shows that entropy-based filtering enables the discovery of models with higher F-score compared to simply filtering out infrequent activities. There are a few exceptions where filtering out infrequent activities outperforms the entropy-based techniques,
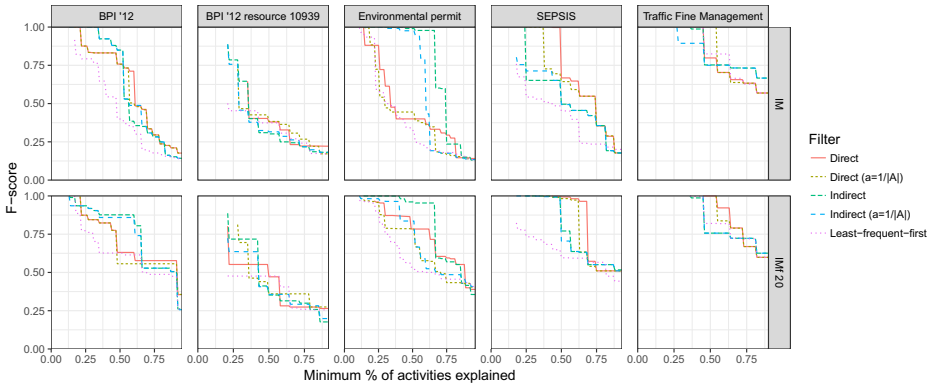
Fig. 9 F-score on business logs dependent on the minimum share of activities remaining

e.g., the Inductive Miner on the BPI '12 resource 10939 event log (around 40% of activities explained) and the traffic fines event log (around 55% of activities explained). It differs between event logs which of the entropy-based techniques performs best: for the environmental permit log the indirect filter without Laplace smoothing almost dominates the other techniques while for the SEPSIS log the direct filter without Laplace smoothing outperforms the other techniques. Generally, it seems that the use of Laplace smoothing harms F-score, as most parts of the lines of indirect filtering with Laplace smoothing are below the lines of the indirect approach without Laplace smoothing, and similar for the direct approach with and without Laplace smoothing. However, the detrimental effect of Laplace smoothing does not seem to be large, and in some cases, the usage of Laplace smoothing in filtering increases the F-score of the discovered models.

Figure 10 shows the nondeterminism of the process models as a function of the minimum percentage of activities. The green dashed line indicates the nondeterminism of the flower model, i.e., the process model that allows for all behavior over the activities. The lines stop when further removal of activities does not lead to further improvement of nondeterminism. It is clear that the filtering mechanism of the Inductive Miner helps to discover process models that are more behaviorally constrained, as the nondeterminism values are
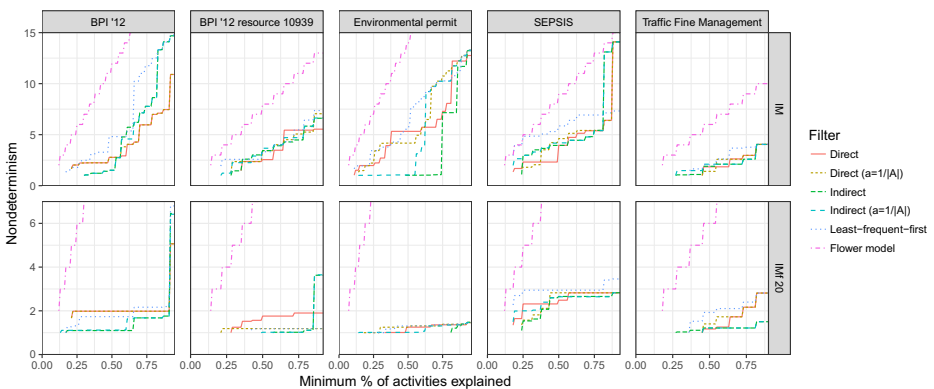

Fig. 10 Nondeterminism on business behavior logs dependent on the minimum share of the activities remaining

lower for the Inductive Miner infrequent 20% compared to the Inductive Miner without fil-
tering. However, the results show even when already using the 20% frequency filter of the
Inductive Miner infrequent, the chaotic activity filter can lead to an additional reduction of
nondeterminism. Furthermore, the results on the environmental permit log and the SEPSIS
log show that filtering several chaotic activities from the event log also enables the discov-
ery of a model with low nondeterminism using the Inductive Miner without filtering. Which
of the activity filtering approaches works best seems to be dependent on the event log: the
indirect entropy-based filter leads to the models with the lowest nondeterminism on the traf-
fic fine event log, the environmental permit event log, while the direct entropy-based filter
works better for some percentages of remaining activities for the SEPSIS log and the BPI
'12 resource 10939 log.

Figures 11 and 12 show the fitness and precision values for the business process event
logs at the filtering step that leads to the highest F-score while describing at least 75% of
the activities of the original log. In addition to the filtering techniques shown in Fig. 9 it
also shows the frequency-based activity filter where the *most* frequent activities are filtered
out first, and a random baseline is shown which iteratively picks a random activity from the
event log to filter out. The error bar for the random activity filter indicates one *standard error
of the mean (SEM)* based on eight repetitions of applying the filter. The black dotted hori-
zontal lines indicate the fitness and precision values of the process models discovered from
the original event log without filtering any activities. Note that the fitness values are only
shown for the Inductive Miner infrequent 20% (Leemans et al. 2013b) because the Induc-
tive Miner without infrequent behavior filter (Leemans et al. 2013a) provides the formal
guarantee that the fitness of the discovered model is 1. Figure 11 shows that generally, the
differences in fitness between the models discovered from the filtered logs are very minor,
and very close to the fitness of the unfiltered log (i.e., the dotted line). Figure 12, however,
shows that the entropy-based filtering approaches outperform filtering out activities based
on frequency and filtering out random activities from the event log. The F-scores of the dis-
covered process models is determined mostly by the precision of the models because the
activity filtering impacts precision more than it impacts fitness. One exception is the BPI
'12 resource 10939 log (Tax et al. 2016), where the fitness decreases to below 0.75 as a
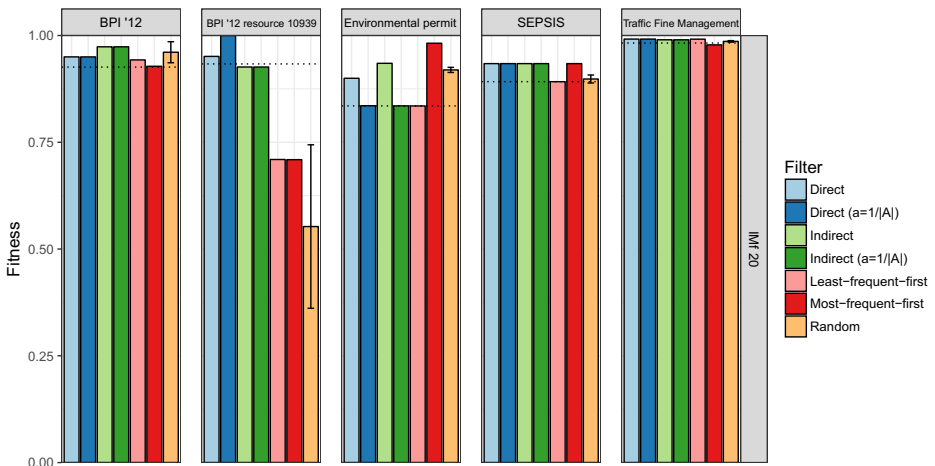

Fig. 11 Fitness on business logs with least 75% of the activities remaining
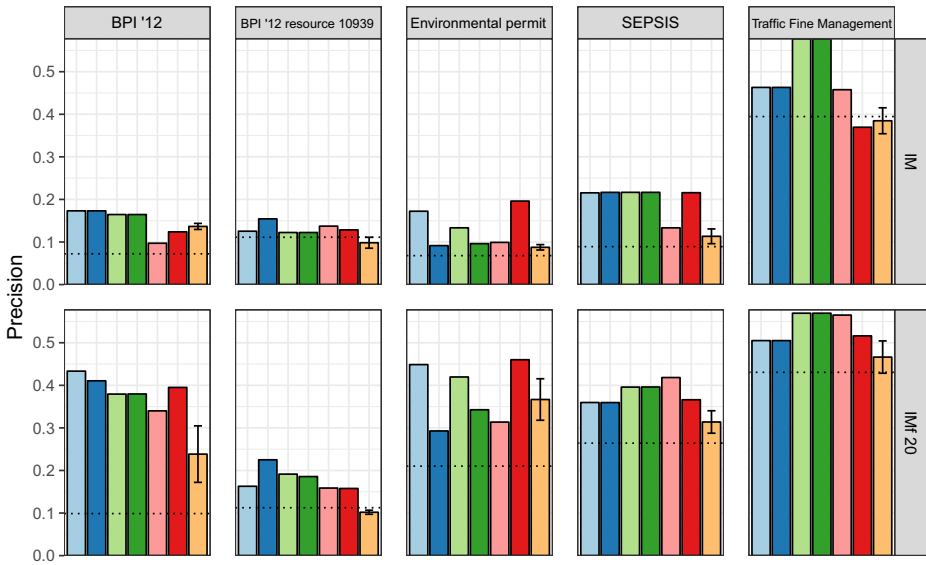
Fig. 12 Precision on business logs with least 75% of the activities remaining

result of applying one of the two frequency-based filters, while the precision increase as an effect of applying the filter is only minor.

5.2 Results on human behavior event logs

Figure 13 shows the maximum F-score for different human behavior event logs as a function of the minimum percentage of activities that are remaining in the log. Again, the general pattern is that the F-score of the discovered process model decreases when the minimum percentage of events explained increases, as the process discovery task gets easier for smaller numbers of activities. The figure shows that filtering infrequent activities from the event log is dominated in terms of F-score by the entropy-based filtering techniques. Like on the business process event logs, there are mixed results on which of the four configurations of the
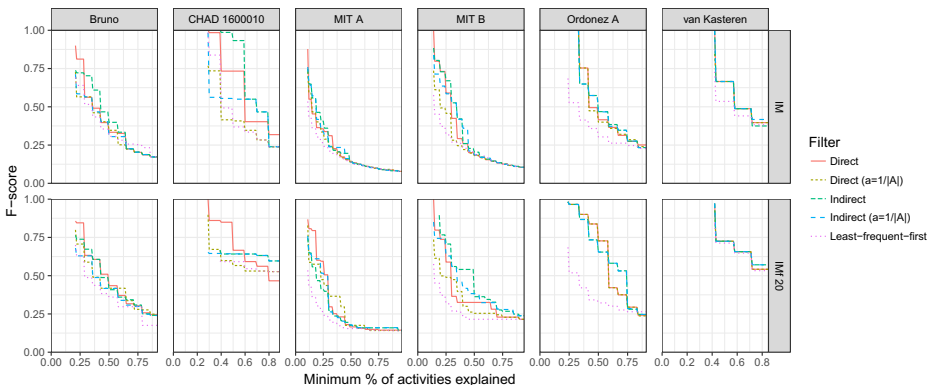

Fig. 13 F-score on human behavior logs dependent on the minimum share of activities

entropy-based filtering technique leads to the highest F-score: on the CHAD event log the indirect activity filter outperforms the direct activity filter when using the Inductive Miner infrequent 20%; however, the direct activity filter leads to higher F-score for the Inductive Miner when filtering more than 50% of the activities.

Figure 14 shows the nondeterminism results for the human behavior event logs. It is noticeable that the nondeterminism values of the process models that are discovered when filtering very few activities are much closer to the flower model compared to what we have seen before for the business process event logs. This is caused by human behavior event logs having much more variability in behavior compared to execution data from business processes, resulting in a much harder process discovery task. After filtering several chaotic activities, the nondeterminism drops significantly to ranges comparable to nondeterminism values seen for logs from the business process domain. This shows that the problem of chaotic activities is much more prominent in human behavior event logs than in business process event logs. The entropy-based activity filtering approaches lead to more deterministic process models compared to filtering out infrequent activities. Two clear examples of this are the MIT B log and the Ordonez A log, on which filtering out infrequent activities after several filtering steps results in a flower model (i.e., nondeterminism is identical to that of the flower model), while entropy-based activity filters enable the discovery of a model with nondeterminism close to one (i.e., very close to a sequential model) while at the same time keeping 75% of the activities in the event log.

Figure 15 shows the precision values for the human behavior logs for the filtering step that leads to the highest F-score while describing at least 50% of the activities of the original log. Similarly to what we have seen in the nondeterminism graph, removing random activities from the log and removing infrequent activities from the log results in smaller precision increases compared to the entropy-based activity filters. Furthermore, it is noticeable that removing frequent activities from the log works quite well to improve the precision of models discovered from the human behavior application domain. The reason for this is that some of the chaotic activities that are present in many of those event logs, including *going to the toilet* and *getting a drink*, also happen to be frequent. On the van Kasteren event log the indirect activity filter with Laplace smoothing leads to the largest increase in precision when mining a model with at least 50% of the activities (from 0.324 to 0.732 with the Inductive Miner infrequent 20%).
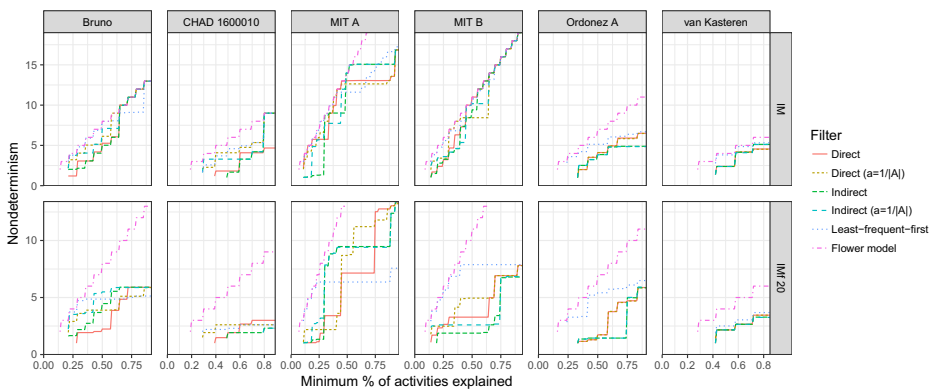

Fig. 14 Nondeterminism on human behavior logs dependent on the minimum share of the activities remaining
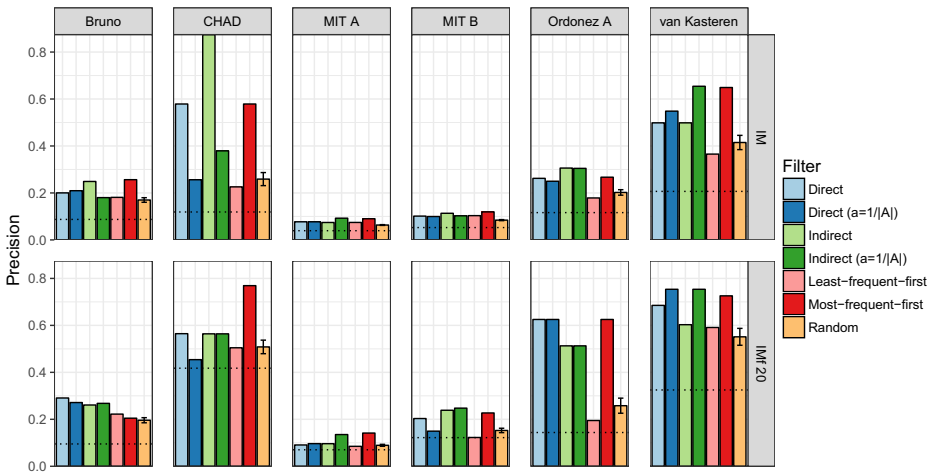
Fig. 15 Precision on human behavior logs with at least 50% of the activities

Table 3 shows in which order activities are filtered from the van Kasteren event log by 1) the indirect entropy-based activity filter with Laplace smoothing and 2) the least-frequent-first filter. It shows that the entropy-based filter filters *use toilet* as the first activity, which from domain knowledge we know to be a chaotic activity, as people generally just go to the toilet whenever they need to, regardless of which other activities they have just performed. For the infrequent activity filter *use toilet* would be the last choice of the activities to filter out, because it is the most frequent activity in the van Kasteren event log.

Figure 16a and b show the corresponding process models discovered with the Inductive Miner infrequent 20% from the logs filtered with the indirect activity filter with Laplace smoothing and the infrequent activity filter respectively. The process model discovered after filtering three activities with the Indirect entropy-based activity filter with Laplace smoothing is very specific on the behavior that it described: after *going to bed*, either the logging ends, or *prepare breakfast* occurs next, followed by *taking a shower*. After taking a shower, there is a possibility to either *go to bed* again or to *prepare dinner* before *going to bed*.

Table 3 Left: the order in which activities are filtered using the direct activity filter with Laplace smoothing ($\alpha = \frac{1}{|Activities(L)|}$) on the van Kasteren log

Order	Filtered activity (indirect entropy-based filter with Laplace smoothing)	Filtered activity (least-frequent-first filter)
1	Use toilet	Prepare dinner
2	Get drink	Get drink
3	Leave house	Prepare breakfast
4	Take shower	Take shower
5	Go to bed	Go to bed
6	Prepare breakfast	Leave house
7	Prepare dinner	Use toilet

Right: the order in which the activities are filtered using the least-frequent-first filter
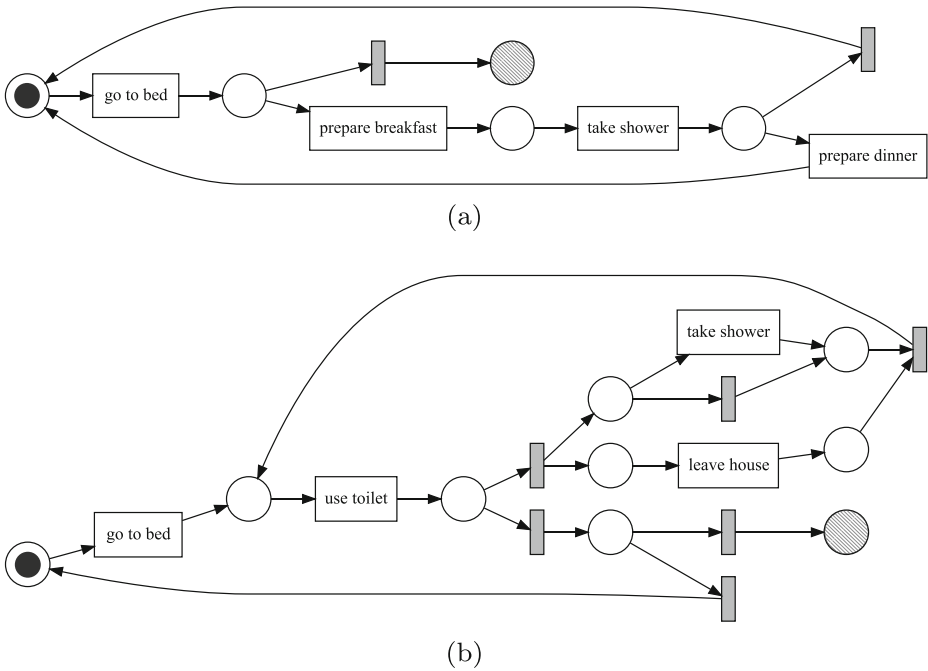
(a)

(b)

Fig. 16 **a** The model discovered with Inductive Miner infrequent 20% on the Van Kasteren log after filtering all but four activities with the indirect approach with Laplace smoothing, and **b** the model discovered from the same log with the same miner when filtering all but four activities when filtering out the least frequent activities

The process model discovered after filtering three activities with the infrequent activity filter allows for many more traces: it starts with *go to bed* followed by *use toilet*, after which any of the activities *go to bed*, *take shower*, and *leave house* can occur as next event or the logging can end. Furthermore, the activities *leave house* and *take shower* can occur in any order, and *take shower* can also be skipped.

Figure 17 shows the results on F-score for the human behavior event logs by Cook et al. (2013). The results on the Cook event logs are in-line with the results on the human behavior event logs, however, on these event logs, it is even more clear that filtering out infrequent activities leads to suboptimal process models in terms of F-score. Which of the filtering approaches results in the optimal process model in terms of F-score is very dependent on the event log and the minimum number of activities to be remained after filtering: each of the four configurations of the entropy-based filtering approach is optimal for at least one combination of log and minimum percentage of activities explained.

Figure 18 shows the results in terms of nondeterminism for the same event logs. Filtering infrequent activities at high percentages of activities explained has much lower nondeterminism compared to the flower model, while further left on the graph, after filtering out more activities, the nondeterminism of filtering out infrequent activities gets closer to the flower model. This shows that filtering out infrequent activities can even be harmful to the quality of the obtained process discovery result. The nondeterminism values obtained with the four configurations of the entropy-based filtering approach are generally close to each other, where the optimal configuration is dependent on the log and the number of filtered activities.
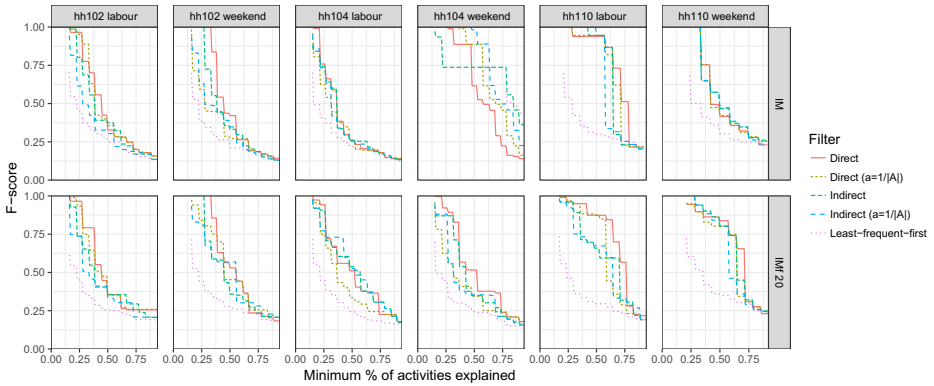
Fig. 17 F-score on cook's human behavior logs dependent on the minimum share of the activities remaining

5.3 Aggregated analysis over all event logs

We have observed in Figs. 10, 14, and 18 that the entropy-based activity filtering techniques perform differently on different datasets and for different numbers of activities filtered. To evaluate the overall performance of activity filtering techniques, we use the number of other filtering techniques that it can beat over all the seventeen event logs of Table 2. This metric, known as *winning number*, is commonly used for evaluation in the Information Retrieval (IR) field (Qin et al. 2010; Tax et al. 2015). Formally, winning number is defined as

$$W_i^x = \sum_{j=1}^{17} \sum_{k=1}^{7} \mathbb{1}_{\{N_i^x(j) < N_k^x(j)\}}$$

where j is the index of an event log, i and k are indices of activity filtering techniques, $N_i^x(j)$ is the performance of the i-th algorithm on the j-th event log in terms of
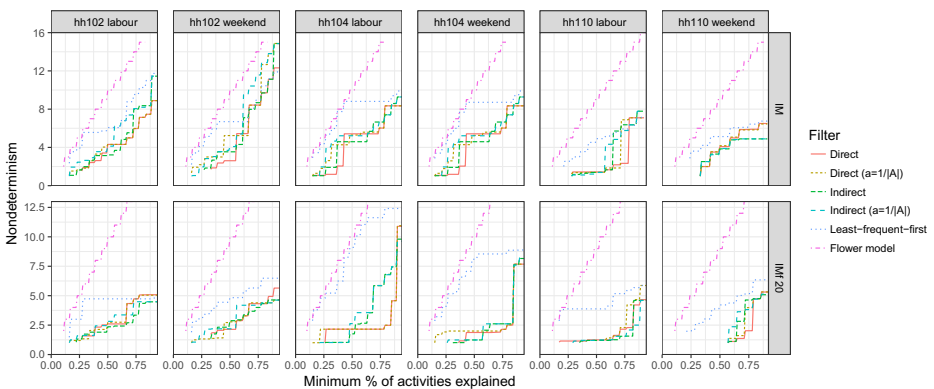

Fig. 18 Nondeterminism on cook's human behavior logs dependent on the minimum share of the activities remaining

nondeterminism where each least $x\%$ of activities are explained and $\mathbb{1}_{\{N_i^x(j) < N_k^x(j)\}}$ is the indicator function

$$\mathbb{1}_{\{N_i^x(j) < N_k^x(j)\}} = \begin{cases} 1, & \text{if } N_i^x(j) < N_k^x(j), \\ 0, & \text{otherwise.} \end{cases}$$

We define $\overline{W}_i^x = \frac{W_i^x}{17}$ as the average number of other activity filtering techniques that are outperformed by filtering technique i at the point where at least $x\%$ of activities are explained.

Figure 19 shows the average winning number $\overline{W}_i^x$ for different values of x and for the seven different activity filtering techniques. We observe that for higher ratios of activities explained the differences between filtering techniques are smaller than for lower numbers of activities explained. Intuitively this can be explained by the fact that for lower ratios of activities explained more activities have been filtered out from the log. Therefore the effect of the filtering techniques is more clearly visible. The figure shows that, up until $+-74\%$ of activities explained, the indirect entropy-based activity filtering technique leads to the most deterministic process models averaged over all event logs included in the experiment, where it outperforms between 4 and 4.5 other filtering techniques. Between $+-75\%$ and $+-87.5\%$ the indirect entropy-based activity filtering technique with Laplace smoothing results in the highest average winning number, although the difference with the indirect entropy-based filtering technique seems negligible. Filtering out random activities from the event log outperforms none of the 6 other activities filtering techniques for the most of the graph, indicating that frequency-based filtering clearly outperforms filtering random activities.

To investigate to what degree the order in which activities are removed from the logs differs between the activity filtering techniques we calculate Kendall's tau (τ_b) rank correlation for each log between the activity filtering techniques in a pairwise way. Table 4 shows the rank correlation values found between the activity filters, averaged over the 17 event logs. The indirect activity filter with Laplace smoothing and the indirect activity filter without Laplace smoothing generate orderings over the activities of a log that are strongly correlated. Between the direct activity filter without Laplace smoothing and the direct activity filter without Laplace smoothing there is only a weak correlation. All the other activity filtering techniques are uncorrelated or very weakly correlated. Using the Kendall τ_b statistic, we apply a tau test for each pair of activity filtering techniques on each event log to test
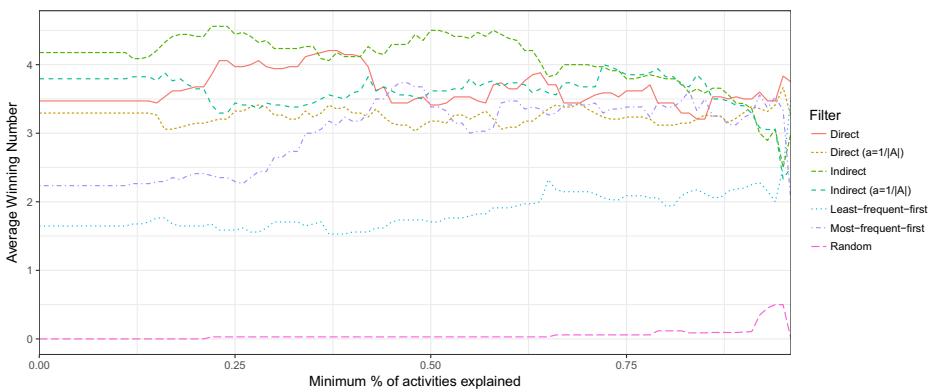

Fig. 19 The average winning number for the seven activity filtering techniques dependent on the minimum ratio of activities explained, averaged over the 17 event logs used in the experiment

Table 4 Kendall τ_b rank correlation between five activity filtering methods, mean and standard deviation over the 17 event logs

	Direct	Direct ($\alpha=\frac{1}{|A|}$)	Indirect	Indirect ($\alpha=\frac{1}{|A|}$)	Least-frequent-first
Direct	1.0	0.2956	0.0829	0.1408	0.0504
Direct ($\alpha=\frac{1}{|A|}$)	0.2956	1.0	0.0698	0.0536	0.1454
Indirect	0.0829	0.0698	1.0	0.6852	−0.0275
Indirect ($\alpha=\frac{1}{|A|}$)	0.1408	0.0536	0.6852	1.0	−0.0392
Least-frequent-first	0.0504	0.1454	−0.0275	−0.0392	1.0

the null hypothesis that the two orderings in which activities are filtered by the two activity filtering techniques are uncorrelated, using a significance level $\alpha = 0.05$.

For each pair of activity filtering techniques Table 5 shows the number of event logs for which the null hypothesis was rejected, i.e., the number of event logs for which the order in which activities are filtered is statistically correlated. The indirect activity filters with and without Laplace smoothing create correlated orderings of activities for all seventeen event logs. For all other pairs of activity filtering techniques the orderings in which activities are filtered are only correlated with for low numbers of event logs.

6 Entropy-based toggles for process discovery

In the previous section we have shown that all four configurations of the entropy-based activity filtering technique lead to more deterministic process models compared to simply filtering out infrequent activities. However, the differences in determinism of the process models that are discovered after applying any of the four configurations are small and dependent on the event log to which they are applied. Furthermore, all four configurations of the activity filtering technique simply impose an ordering over the activities, but do not specify at which step the filtering should be stopped. Additionally, the proposed filtering technique ignores the semantics of activities: activities that are chaotic may still be relevant for the process. Leaving them out of the process model to discover will harm the usefulness of the discovered process model.

To address the three issues we propose to use the filtering technique as a sorting technique over the activities in combination with toggles that interactively allow the process analyst to "disable" (filter out) or "enable" activities, and then rediscover and visualize the process model according to the new settings. This approach is similar to the Inductive Visual Miner

Table 5 Number of event logs for which we can reject the null hypothesis that the orderings of activities returned by activity filters are uncorrelated, according to the tau test

	Direct	Direct ($\alpha=\frac{1}{|A|}$)	Indirect	Indirect ($\alpha=\frac{1}{|A|}$)	Least-frequent-first
Direct	17	5	1	2	0
Direct ($\alpha=\frac{1}{|A|}$)	5	17	1	1	3
Indirect	1	1	17	17	3
Indirect ($\alpha=\frac{1}{|A|}$)	2	1	17	17	3
Least-frequent-first	0	3	3	3	17

(Leemans et al. 2014), an interactive implementation of the Inductive Miner (Leemans et al. 2013b) algorithm which allows the process analyst to filter the event log interactively using a slider-based approach. The Inductive visual miner contains two sliders: with one slider activities can be filtered using the least-frequent-first filter, where the user can control how many activities are filtered out by moving the slider up and down. We propose to replace this slider with a sorted list of activities and toggles, as this allows the process analyst to override the ordering of the activities that is determined by the activity filtering technique with domain knowledge. Figure 20 shows a mockup of the proposed way to use the activity filter. Activities are by default sorted using the chaotic activity filter, showing the entropy to indicate the assessed degree of chaoticness of each activity. Based on this information, the process analyst can choose to rely on the filtering technique and filter out the top of the list or to override this list with domain knowledge. Furthermore, other activity filtering techniques, such as the least-frequent-first filter, can be included as an additional column on which the activities of the process can be sorted. This allows the process analyst to control how many activities, and which activities, are filtered out of the process model, and thereby also empowers the user to prevent the removal of semantically important activities that should not be removed. Furthermore, this approach allows the process analyst to explore himself which of the filtering techniques leads to the most useful process model from the event log that he is analyzing.

7 Related work

Existing work on filtering of event logs in process mining focuses either at removing logging mistakes (called *noise*) with the aim to prevent those logging mistakes to propagate to a discovered process model, or alternatively focus at removing *infrequent* behavior in order to discover a process model that models the mainstream behavior in the log. With regard to noise, real-life events logs often contain all sorts of such data quality issues (Suriadi et al. 2017), include incorrectly logged events, events that are logged in the wrong order, and events that took place without being logged. Many event log filtering techniques have been proposed to address the problem of noise (Conforti et al. 2017; Lu et al. 2015; Fani Sani et al. 2017; Ghionna et al. 2008; Cheng and Kumar 2015). Note that there is a conceptual
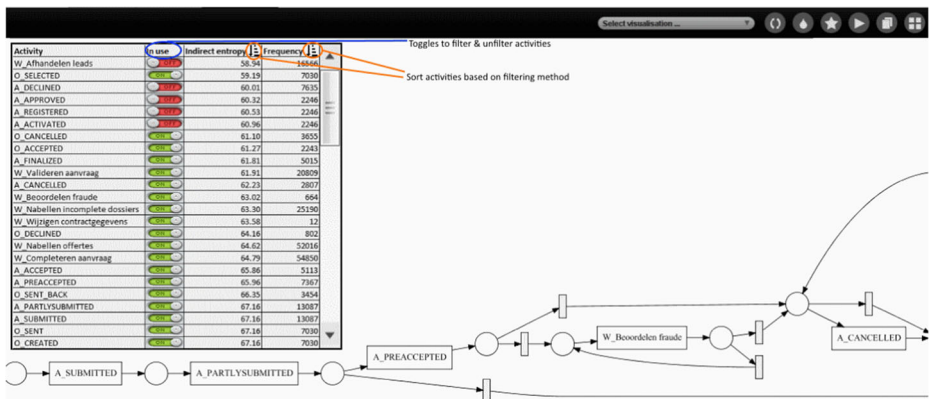

Fig. 20 A mockup of the proposed way to use the activity filters in an interactive setting

difference between *chaotic activities* and noise: where noise finds its origin in mistakes related to logging, events from chaotic activities are in fact correctly logged, but are still undesired because they represent an activity that is logged by the system even though it not part of the main process flow. Chaotic activities are also clearly distinct from filtering of infrequent behavior, as chaotic activities can be frequent.

Existing filtering techniques in the process mining field can be classified into four categories: 1) event filtering techniques, 2) process discovery techniques that have an integrated filtering mechanism build in, 3) trace filtering techniques, and 4) activity filtering techniques. We use these categories to discuss and structure related work.

7.1 Event filtering

Conforti et al. (2017) recently proposed a technique to filter out outlier events from an event log. The technique starts by building a prefix automaton of the event log, which is minimal in terms of the number of arcs in the automaton, using an Integer Linear Programming (ILP) solver. Infrequent arcs are removed from the minimal prefix automaton, and finally, the events belonging to removed arcs are filtered out from the event log.

Lu et al. (2015) advocate the use of *event mappings* (Lu et al. 2014) to distinguish between events that are part of the mainstream behavior of a process and outlier events. Event mappings compute similar behavior and dissimilar behavior between each two executions of the process as a mapping: the similar behavior is formed by all pairs of events that are mapped to each other, whereas events that are not mapped are dissimilar behavior. Fani Sani et al. (2017) proposes the use of sequential pattern mining techniques to distinguish between events that are part of the mainstream behavior and outlier events.

All three of the event filtering techniques listed above aim filter out outlier events from the event log, while keeping the mainstream behavior. Event filtering techniques model the frequently occurring contexts of activities and filter out the contexts of activities that occur infrequently in the log. For example, consider an activity B such that 98% of its occurrences are in context $\langle \ldots, A, B, C, \ldots \rangle$, with the remaining 2% of the events of activity B are in context $\langle \ldots, D, B, E, \ldots \rangle$, then the B events that occur between D and E will be filtered out by event filtering techniques. Note that our filtering technique is orthogonal to event filtering: it would consider activity B to be nonchaotic and would not filter out anything. However, when a log L contains a chaotic activity X, then event filtering techniques are not able to remove all events of this chaotic activity. One of the contexts of X will by chance be more frequent than other contexts, i.e., for some activity A, it will hold that $\forall B \in Activities(L) : \#(\langle A, X \rangle, L) > \#(\langle B, X \rangle, L)$, even though $\langle A, X \rangle$ might only be slightly more frequent. This will result in X events after a B being removed, while the X events after an A remain in the log. Applying a process discovery technique to this filtered log will then result in a process model where activity X is misleadingly positioned after activity A, while in fact X can happen anywhere in the process. The activity filtering technique presented in this paper will instead detect that activity X is chaotic, and completely remove it from the event log, preventing the misleading effect of event filtering.

7.2 Process discovery techniques with integrated filtering

Several process discovery algorithms offer integrated filtering mechanisms as part of the approach. The α-Miner (van der Aalst et al. 2004) is one of the early foundational techniques for process discovery, which starts by inferring *causal*, *exclusive* and *parallel* relations

between pairs of activities, which are converted into a Petri net in a second step of the algorithm. In later work, Maruster et al. (2006) explored the use of supervised learning techniques to extract those causal, exclusive and parallel relations from the event log, allowing the α-Miner to disregard "noisy" events that would have heavily impacted those relations in the original approach.

The Inductive Miner (IM) (Leemans et al. 2013a) is a process discovery algorithm which first discovers a *directly-follows graph* from the event logs, where activities are connected that directly follow each other in the log, from which in a second step a process model is discovered. The directly-follows relations are affected by the presence of a chaotic activity X: sequence $\langle \ldots, A, X, C, \ldots \rangle$ leads to false directly-follows relations between A and X and between X and C, while the directly-follows relation between A and C is obfuscated by X. The Inductive Miner infrequent (IMf) (Leemans et al. 2013b) is an extension of the IM where infrequent directly-follows relations are filtered out from the set of directly-follows relations that are used to generate to process models. The filtering mechanism of IMf can help to filter out the directly-follows relations between A and X and between X and C, but it does not help to recover the obfuscated directly-follows relation between A and C. Instead, the activity filtering technique presented in this paper filters out the chaotic activity X, leading to sequence $\langle \ldots, A, X, C, \ldots \rangle$ being transformed into $\langle \ldots, A, C, \ldots \rangle$, thereby recovering the directly follows relation between A and C.

The Heuristics Miner (Weijters and Ribeiro 2011) and the Fodina algorithm (Vanden Broucke and De Weerdt 2017), in addition to the directly-follows relation, defines an *eventually-follows relation* between activities and allows the process analyst to filter out infrequent directly-follows and eventually follows relations. Two activities A and B are in an eventually-follows relation when A is eventually followed by B, before the next appearance of A or B. The eventually-follows relation, unlike the directly-follows relation, is not impacted by the presence of chaotic activities. The Heuristic Miner (Weijters and Ribeiro 2011) and Fodina (Vanden Broucke and De Weerdt 2017) both include filtering methods for the directly-follows and eventually-follows relations that are similar in nature to the filtering mechanism that is used in the Inductive Miner infrequent (Leemans et al. 2013b). However, the use of sequential orderings and parallel constructs in the mining approaches of the Heuristic Miner (De Weerdt et al. 2011) and Fodina (Vanden Broucke and De Weerdt 2017) is based on the directly-follows relations only, with the eventually follows relations being used for the mining of long-term dependencies. Furthermore, in contrast to the Inductive Miner, the process models discovered with the Heuristic Miner (Weijters and Ribeiro 2011) or Fodina (Vanden Broucke and De Weerdt 2017) can be unsound, i.e., the can contain deadlocks.

The ILP-miner (van der Werf et al. 2009) is a process discovery algorithm where a set of behavioral constraints over activities is discovered for each prefix (called the *prefix-closure*) of the event log, based on which a process model is discovered that satisfies these constraints using *Integer Linear Programming* (ILP). van Zelst et al. (2015) proposed a filtering technique for the ILP-miner where the prefix closure of the event log is filtered prior to solving the ILP problem by removing infrequently observed prefixes. It is easy to see that a chaotic activity X affect the prefix-closure that is discovered from the event log: given log consisting of two traces $\langle A, X, C \rangle$ and $\langle X, A, C \rangle$, activity X causes the prefixes closures of the two traces to have no overlap in states, while without activity X the two traces are identical. This makes the filtering method of the prefix-closure proposed by van Zelst et al. (2015) less effective, as frequent prefixes randomly get distributed over several infrequent prefixes when chaotic activities are present. Instead, the chaotic activity filtering technique presented in this paper would remove chaotic activity X, leading to traces

$\langle A, X, C \rangle$ and $\langle X, A, C \rangle$ becoming identical after filtering, therefore leading to a simpler process model while still describing the behavior of the event log accurately.

The Fuzzy Miner (Günther and van der Aalst 2007) is a process discovery algorithm that aims at mining models from flexible processes, and it discovers a process model without formal semantics. The Fuzzy Miner discovers this graph by extracting the eventually follows relation from the event log, which is not affected by chaotic activities. Similar to the Heuristics Miner (Weijters and Ribeiro 2011) and Fodina (Vanden Broucke and De Weerdt 2017) the Fuzzy Miner allows to filter out infrequent eventually-follows relations between activities. In practice, the lack of formal semantics of the Fuzzy Miner models hinders the usability of the models, as the models are not precise on what behavior is allowed in the process under analysis.

7.3 Trace filtering

Ghionna et al. (2008) proposed a technique to identify outlier traces from the event log that consists of two steps: 1) mining frequent patterns from the event log, and 2) applying MCL clustering (Van Dongen 2008) on the traces, where the similarity measure for traces is defined on the number of patterns that jointly characterize the execution of the traces. Traces that are not assigned to a cluster by the MCL clustering algorithm are considered to be outlier traces and are filtered from the event log.

Cheng and Kumar (2015) propose a supervised approach to filter out noisy traces from an event logs, where they assume a *marked sub-log* on which they assume a process worker to have manually inspected and labeled traces into *clean* and *noisy* traces. Additionally, there is an *unmarked sub-log* for which it is unknown which traces are noisy. The use the PRISM rule-induction algorithm (Cendrowska 1987) to extract classification rules to differentiate between clean and noisy traces by training on the marked sub-log, and then apply these classification rules to identify and filter out noisy traces from the unmarked sub-log.

It is easy to see that trace filtering techniques address a fundamentally different problem than chaotic activity filtering: in the event log shown in Fig. 2b there are only two traces that do not contain an instance of chaotic activity X, therefore, even if a trace filtering technique would be able to perfectly filter out traces that contain a chaotic event, the number of remaining traces will become too small to mine a fitting and precise process model when the chaotic activity is frequent.

7.4 Activity filtering

The modus operandi for filtering activities is to simply filter out infrequent activities from the event log. The plugin *'Filter Log using Simple Heuristics'* in the ProM process mining toolkit (Van Dongen et al. 2005) offers tool support for this type of filtering. The Inductive Visual Miner (Leemans et al. 2014) is an interactive process discovery tool that implements the Inductive Miner (Leemans et al. 2013b) process discovery algorithm in an interactive way: the process analyst can filter the event log using sliders and is then shown the process model that is discovered from this filtered log. One of the available sliders in the Inductive Visual Miner offers the same frequency-based activity filtering functionality. The working assumption behind filtering out infrequent activities is that when there are just a few occurrences of an activity, there is probably not enough evidence to establish their relation to other activities to model their behavior. However, as we have shown in this paper, for frequent but chaotic activities, while they are frequent enough to establish their relation to other activities, complicate the process discovery task by lowering directly-follows counts

between other activities in the event log. The activity filtering technique presented in this paper is able to filter out chaotic activities, thereby reconstructing the directly-follows relations between the non-chaotic activities of the event log, at the expense of losing the chaotic activities.

8 Conclusion & future work

In this paper, we have shown the possible detrimental effect of the presence of chaotic activities in event logs on the quality of process models produced by process discovery techniques. We have shown through synthetic experiments that frequency-based techniques for filtering activities from event logs, which is currently the *modus operandi* for activity filtering in the process mining field, do not necessarily handle chaotic activities well. As shown, chaotic activities can be frequent or infrequent. We have proposed four novel techniques for filtering chaotic from event logs, which find their roots in information theory and Bayesian statistics. Through experiments on seventeen real-life datasets, we have shown that all four proposed activity filtering techniques outperform frequency-based filtering on real data. The indirect entropy-based activity filter has been found to be the best performing activity filter overall averaged over all datasets used in the experiments; however, the performance of the four proposed activity filtering techniques is highly dependent on the characteristics of the event log.

Because the performance of the filtering techniques was found to be log-dependent, we propose the use the activity filtering techniques in a slider-based approach where the user can filter activities interactively and directly see the process model discovered from the filtered event log. Ultimately, only the user can decide which activities to include. In future work, we aim to construct a hybrid activity filtering technique that combines the four techniques proposed in this paper by using supervised learning techniques from the data mining field to predict the effect of removing a particular activity.

References

Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P. (2011). Conformance checking using cost-based fitness analysis. In *Proceedings of the 15 IEEE international enterprise distributed object computing conference (EDOC)* (pp. 55–64). IEEE.

Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T., Zaccaria, R. (2013). Analysis of human behavior recognition algorithms based on acceleration data. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 1602–1607). IEEE.

Buijs, J.C.A.M. (2014). Receipt phase of an environmental permit application process (WABO), CoSeLoG project. https://doi.org/10.4121/uuid:a07386a5-7be3-4367-9535-70bc9e77dbe6.

Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P. (2012). A genetic algorithm for discovering process trees. In *Proceedings of the 2012 IEEE congress on evolutionary computation* (pp. 1–8). IEEE.

Cendrowska, J. (1987). PRISM: an algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, *27*(4), 349–370.

Cheng, H.J., & Kumar, A. (2015). Process mining on noisy logscan log sanitization help to improve performance? *Decision Support Systems*, *79*, 138–149.

Conforti, R., La Rosa, M., ter Hofstede, A.H.M. (2017). Filtering out infrequent behavior from business process event logs. *IEEE Transactions on Knowledge and Data Engineering*, *29*(2), 300–314.

Cook, D.J., Crandall, A.S., Thomas, B.L., Krishnan, N.C. (2013). CASAS: a smart home in a box. *Computer*, *46*(7), 62–69.

De Leoni, M., & Mannhardt, F. (2015). Road traffic fine management process. https://doi.org/10.4121/uuid: 270fd440-1057-4fb9-89a9-b699b47990f5.

De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B. (2011). A robust F-measure for evaluating discovered process models. In *Proceedings of the IEEE symposium on computational intelligence and data mining (CIDM)* (pp. 148–155). IEEE.

Dimaggio, M., Leotta, F., Mecella, M., Sora, D. (2016). Process-based habit mining: experiments and techniques. In: *Proceedings of the international IEEE conference on ubiquitous intelligence & computing* (pp. 145–152). IEEE.

Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P. (2017). Improving process discovery results by filtering outliers using conditional behavioural probabilities. In *Proceedings of the international workshop on business process intelligence*. Springer.

Ghionna, L., Greco, G., Guzzo, A., Pontieri, L. (2008). Outlier detection techniques for process mining applications. In *International symposium on methodologies for intelligent systems* (pp. 150–159). Springer.

Goedertier, S., Martens, D., Vanthienen, J., Baesens, B. (2009). Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, *10*, 1305–1340.

Günther, C.W., & van der Aalst, W.M.P. (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *International conference on business process management* (pp. 328–343). Springer.

Herbst, J. (2000). A machine learning approach to workflow management. In *European conference on machine learning* (pp. 183–194). Springer.

Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P. (2013a). Discovering block-structured process models from event logs - a constructive approach. In *International conference on applications and theory of petri nets and concurrency* (pp. 311–329). Springer.

Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P. (2013b). Discovering block-structured process models from event logs containing infrequent behaviour. In *International conference on business process management* (pp. 66–78). Springer.

Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P. (2014). Process and deviation exploration with inductive visual miner. In *Proceedings of the BPM Demo Track* (Vol. 1295, p. 46). CEUR-WS.org.

Leotta, F., Mecella, M., Mendling, J. (2015). Applying process mining to smart spaces: perspectives and research challenges. In *International conference on advanced information systems engineering* (pp. 298–304). Springer.

Lohmann, N., Verbeek, E., Dijkman, R. (2009). Petri net transformations for business processes–a survey. In *Transactions on petri nets and other models of concurrency II* (pp. 46–63). Springer.

Lu, X., Fahland, D., van der Aalst, W.M.P. (2014). Conformance checking based on partially ordered event data. In *International conference on business process management* (pp. 75–88). Springer.

Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P. (2015). Detecting deviating behaviors without models. In *Proceedings of the international workshop on business process intelligence* (pp. 126–139). Springer.

Mannhardt, F. (2016). Sepsis cases - event log. https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35 f063a460.

Maruster, L., Weijters, A.J.M.M., Aalst, W.M.P., Bosch, A. (2006). A rule-based approach for process discovery: dealing with noise and imbalance in process logs. *Data Mining & Knowledge Discovery*, *13*(1), 67–87.

McCurdy, T., Glen, G., Smith, L., Lakkadi, Y. (2000). The national exposure research laboratory's consolidated human activity database. *Journal of Exposure Analysis and Environmental Epidemiology*, *10*(6), 566–578.

Murata, T. (1989). Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580.

Object Management Group (2011). Notation (BPMN) version 2.0. OMG Specification.

Ordónez, F.J., de Toledo, P., Sanchis, A. (2013). Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors*, *13*(5), 5460–5477.

Qin, T., Liu, T.Y., Xu, J., Li, H. (2010). LETOR: a benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, *13*(4), 346–374.

Solé, M., & Carmona, J. (2013). Region-based foldings in process discovery. *IEEE Transactions on Knowledge and Data Engineering*, *25*(1), 192–205.

Suriadi, S., Andrews, R., ter Hofstede, A.H.M., Wynn, M.T. (2017). Event log imperfection patterns for process mining: towards a systematic approach to cleaning event logs. *Information Systems*, *64*, 132–150.

Sztyler, T., Völker, J., Carmona Vargas, J., Meier, O., Stuckenschmidt, H. (2015). Discovery of personal processes from labeled sensor data: an application of process mining to personalized health care. In *Proceedings of the international workshop on algorithms & theories for the analysis of event data, CEUR-WS.org* (pp. 31–46).

Tapia, E.M., Intille, S.S., Larson, K. (2004). Activity recognition in the home using simple and ubiquitous sensors. In *International conference on pervasive computing* (pp. 158–175). Springer.

Tax, N., Bockting, S., Hiemstra, D. (2015). A cross-benchmark comparison of 87 learning to rank methods. *Information Processing & Management*, *51*(6), 757–772.

Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.P. (2016). Mining local process models. *Journal of Innovation in Digital Ecosystems*, *3*(2), 183–196.

Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.P. (2017). Mining process model descriptions of daily life through event abstraction. In *Intelligent systems and applications* (p To appear). Springer.

Van Dongen, S. (2008). Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications*, *30*(1), 121–141.

Van Dongen, B. (2012). BPI challenge 2012. https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.

Van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., Van Der Aalst, W.M.P. (2005). The ProM framework: a new era in process mining tool support. In *International conference on application and theory of petri nets* (pp. 444–454). Springer.

van der Aalst, W.M.P. (2016). Process mining: data science in action. Springer.

van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L. (2004). Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, *16*(9), 1128–1142.

van der Aalst, W.M.P., Bolt, A., van Zelst, S.J. (2017). RapidProM: mine your processes and not just your data. In Hofmann, M., & Klinkenberg, R. (Eds.) *RapidMiner: data mining use cases and business analytics applications* (p To Appear): Chapman & Hall/CRC Data Mining and Knowledge Discovery Series.

van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A. (2009). Process discovery using integer linear programming. *Fundamenta Informaticae*, *94*(3), 387–412.

van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B. (2008). Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on ubiquitous computing* (pp. 1–9). ACM.

van Zelst, S.J., van Dongen, B.F., van der Aalst, W.M.P. (2015). Avoiding over-fitting in ILP-based process discovery. In *International conference on business process management* (pp. 163–171). Springer International Publishing.

Vanden Broucke, S.K.L.M., De Weerdt, J., Vanthienen, J., Baesens, B. (2013). Determining process model precision and generalization with weighted artificial negative events. *IEEE Transactions on Knowledge and Data Engineering*.

Vanden Broucke, S.K.L.M., & De Weerdt, J. (2017). Fodina: a robust and flexible heuristic process discovery technique. *Decision Support Systems*.

Weijters, A.J.M.M., & Ribeiro, J.TS. (2011). Flexible heuristics miner (FHM). In *Proceedings of the IEEE symposium on computational intelligence and data mining (CIDM)* (pp. 310–317). IEEE.

Zhai, C., & Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, *22*(2), 179–214.